

Lineare Algebra in der Praxis

Numerische Aspekte

Benjamin Sambale

Leibniz Universität Hannover

12. 07. 2021

programmiert mit \LaTeX + TikZ + BEAMER

Komplexität von Multiplikationen

- Matrizenmultiplikation basiert auf Multiplikation und Addition von Zahlen.

Komplexität von Multiplikationen

- Matrizenmultiplikation basiert auf Multiplikation und Addition von Zahlen.
- Multiplikationen sind „aufwendiger“ als Additionen.

Komplexität von Multiplikationen

- Matrizenmultiplikation basiert auf Multiplikation und Addition von Zahlen.
- Multiplikationen sind „aufwendiger“ als Additionen.
- Die Schul-Multiplikation zweier n -stelliger Zahlen x, y erfordert n^2 **Ziffer-Multiplikationen**:

$$\begin{array}{r} 123 \cdot 567 = 69741 \\ \hline 861 \\ + 738 \\ + 615 \\ \hline 69741 \end{array}$$

Es geht besser

Karatsuba-Algorithmus (1962)

- 1 Teile x und y in zwei Hälften der Länge $m \approx n/2$:

$$x = x_1 10^m + x_0, \quad y = y_1 10^m + y_0, \quad x_0, y_0 < 10^m$$

Es geht besser

Karatsuba-Algorithmus (1962)

- ① Teile x und y in zwei Hälften der Länge $m \approx n/2$:

$$x = x_1 10^m + x_0, \quad y = y_1 10^m + y_0, \quad x_0, y_0 < 10^m$$

- ② Berechne rekursiv

$$z_0 := x_0 y_0, \quad z_2 := x_1 y_1, \quad z_1 := (x_1 - x_0)(y_0 - y_1) + z_0 + z_2.$$

Es geht besser

Karatsuba-Algorithmus (1962)

- ① Teile x und y in zwei Hälften der Länge $m \approx n/2$:

$$x = x_1 10^m + x_0, \quad y = y_1 10^m + y_0, \quad x_0, y_0 < 10^m$$

- ② Berechne rekursiv

$$z_0 := x_0 y_0, \quad z_2 := x_1 y_1, \quad z_1 := (x_1 - x_0)(y_0 - y_1) + z_0 + z_2.$$

- ③ Dann gilt $xy = 10^{2m} z_2 + 10^m z_1 + z_0$.

Karatsuba-Algorithmus im Test

Beispiel

Für $x = 87 = 80 + 7$ und $y = 91 = 90 + 1$ erhält man

Karatsuba-Algorithmus im Test

Beispiel

Für $x = 87 = 80 + 7$ und $y = 91 = 90 + 1$ erhält man

$$z_0 = 7 \cdot 1 = 7,$$

Karatsuba-Algorithmus im Test

Beispiel

Für $x = 87 = 80 + 7$ und $y = 91 = 90 + 1$ erhält man

$$z_0 = 7 \cdot 1 = 7,$$

$$z_2 = 8 \cdot 9 = 72,$$

Karatsuba-Algorithmus im Test

Beispiel

Für $x = 87 = 80 + 7$ und $y = 91 = 90 + 1$ erhält man

$$z_0 = 7 \cdot 1 = 7,$$

$$z_2 = 8 \cdot 9 = 72,$$

$$z_1 = (8 - 7)(1 - 9) + z_0 + z_2 = -8 + 7 + 72 = 71,$$

Karatsuba-Algorithmus im Test

Beispiel

Für $x = 87 = 80 + 7$ und $y = 91 = 90 + 1$ erhält man

$$z_0 = 7 \cdot 1 = 7,$$

$$z_2 = 8 \cdot 9 = 72,$$

$$z_1 = (8 - 7)(1 - 9) + z_0 + z_2 = -8 + 7 + 72 = 71,$$

$$xy = 7200 + 710 + 7 = 7917.$$

Analyse

Satz

Der Karatsuba-Algorithmus für n -stellige Zahlen benötigt ca.

$$n^{\log_2(3)} \approx n^{1.58} < n^2$$

Ziffer-Multiplikationen.

Analyse

Satz

Der Karatsuba-Algorithmus für n -stellige Zahlen benötigt ca.

$$n^{\log_2(3)} \approx n^{1.58} < n^2$$

Ziffer-Multiplikationen.

Beweis.

Induktion nach n :

Analyse

Satz

Der Karatsuba-Algorithmus für n -stellige Zahlen benötigt ca.

$$n^{\log_2(3)} \approx n^{1.58} < n^2$$

Ziffer-Multiplikationen.

Beweis.

Induktion nach n : Für $n = 1$ braucht man $1 = 1^{\log_2(3)}$ Ziffer-Multiplikation.

Analyse

Satz

Der Karatsuba-Algorithmus für n -stellige Zahlen benötigt ca.

$$n^{\log_2(3)} \approx n^{1.58} < n^2$$

Ziffer-Multiplikationen.

Beweis.

Induktion nach n : Für $n = 1$ braucht man $1 = 1^{\log_2(3)}$ Ziffer-Multiplikation.

Die Berechnung von z_0, z_1, z_2 für $n \geq 2$ erfordert 3 Multiplikationen m -stelliger Zahlen.

Analyse

Satz

Der Karatsuba-Algorithmus für n -stellige Zahlen benötigt ca.

$$n^{\log_2(3)} \approx n^{1.58} < n^2$$

Ziffer-Multiplikationen.

Beweis.

Induktion nach n : Für $n = 1$ braucht man $1 = 1^{\log_2(3)}$ Ziffer-Multiplikation.

Die Berechnung von z_0, z_1, z_2 für $n \geq 2$ erfordert 3 Multiplikationen m -stelliger Zahlen.

Induktiv erhält man

$$3m^{\log_2(3)} \approx 3(n/2)^{\log_2(3)} = n^{\log_2(3)}$$

Ziffer-Multiplikationen. □

Verbesserungen

- Toom-Cook-Algorithmus (1966): $n^{\log(5)/\log(3)} \approx n^{1.46}$.
Idee: Teile in mehr als zwei Teile auf.

Verbesserungen

- Toom-Cook-Algorithmus (1966): $n^{\log(5)/\log(3)} \approx n^{1.46}$.
Idee: Teile in mehr als zwei Teile auf.
- Schönhage-Strassen-Algorithmus (1971): $n \log(n) \log \log(n)$.
Idee: **Schnelle Fourier-Transformation** (FFT). Verbreiteter Standard.

Verbesserungen

- Toom-Cook-Algorithmus (1966): $n^{\log(5)/\log(3)} \approx n^{1.46}$.
Idee: Teile in mehr als zwei Teile auf.
- Schönhage-Strassen-Algorithmus (1971): $n \log(n) \log \log(n)$.
Idee: **Schnelle Fourier-Transformation** (FFT). Verbreiteter Standard.
- Fürer-Algorithmus (2007): $n \log(n) 8^{\log_2^*(n)}$. Verbesserung von Schönhage-Strassen.
Nur für extrem große Zahlen relevant.

Verbesserungen

- Toom-Cook-Algorithmus (1966): $n^{\log(5)/\log(3)} \approx n^{1.46}$.
Idee: Teile in mehr als zwei Teile auf.
- Schönhage-Strassen-Algorithmus (1971): $n \log(n) \log \log(n)$.
Idee: **Schnelle Fourier-Transformation** (FFT). Verbreiteter Standard.
- Fürer-Algorithmus (2007): $n \log(n) 8^{\log_2^*(n)}$. Verbesserung von Schönhage-Strassen.
Nur für extrem große Zahlen relevant.
- Harvey-van der Hoeven (2019): $n \log n$. Praktisch bisher irrelevant.

Matrizen-Multiplikation

- Die direkte Multiplikation zweier $n \times n$ -Matrizen A, B erfordert n^3 Zahl-Multiplikationen $a_{ij}b_{jk}$ mit $1 \leq i, j, k \leq n$.

Matrizen-Multiplikation

- Die direkte Multiplikation zweier $n \times n$ -Matrizen A, B erfordert n^3 Zahl-Multiplikationen $a_{ij}b_{jk}$ mit $1 \leq i, j, k \leq n$.
- Man kommt auch hier mit weniger aus. Für $n = 2$ berechnen wir:

$$c_1 := (a_{11} + a_{22})(b_{11} + b_{22}),$$

$$c_2 := (a_{21} + a_{22})b_{11},$$

$$c_3 := a_{11}(b_{12} - b_{22}),$$

$$c_4 := a_{22}(b_{21} - b_{11}),$$

$$c_5 := (a_{11} + a_{12})b_{22},$$

$$c_6 := (a_{21} - a_{11})(b_{11} + b_{12}),$$

$$c_7 := (a_{12} - a_{22})(b_{21} + b_{22}).$$

Matrizen-Multiplikation

- Die direkte Multiplikation zweier $n \times n$ -Matrizen A, B erfordert n^3 Zahl-Multiplikationen $a_{ij}b_{jk}$ mit $1 \leq i, j, k \leq n$.
- Man kommt auch hier mit weniger aus. Für $n = 2$ berechnen wir:

$$c_1 := (a_{11} + a_{22})(b_{11} + b_{22}),$$

$$c_2 := (a_{21} + a_{22})b_{11},$$

$$c_3 := a_{11}(b_{12} - b_{22}),$$

$$c_4 := a_{22}(b_{21} - b_{11}),$$

$$c_5 := (a_{11} + a_{12})b_{22},$$

$$c_6 := (a_{21} - a_{11})(b_{11} + b_{12}),$$

$$c_7 := (a_{12} - a_{22})(b_{21} + b_{22}).$$

- Jeder Eintrag von AB ist eine Summe/Differenz der c_i :

$$AB = \begin{pmatrix} c_1 + c_4 - c_5 + c_7 & c_3 + c_5 \\ c_2 + c_4 & c_1 - c_2 + c_3 + c_6 \end{pmatrix}$$

Analyse

- Wir haben also nur 7 anstatt $2^3 = 8$ Zahl-Multiplikationen verwendet.

Analyse

- Wir haben also nur 7 anstatt $2^3 = 8$ Zahl-Multiplikationen verwendet.
- Für $n > 2$ teilt man A und B in je vier Blöcke und iteriert (**Strassen-Algorithmus**).

Analyse

- Wir haben also nur 7 anstatt $2^3 = 8$ Zahl-Multiplikationen verwendet.
- Für $n > 2$ teilt man A und B in je vier Blöcke und iteriert (**Strassen-Algorithmus**).
- Dies erfordert ca. $n^{\log_2(7)} \approx n^{2.81}$ Zahl-Multiplikationen.

Analyse

- Wir haben also nur 7 anstatt $2^3 = 8$ Zahl-Multiplikationen verwendet.
- Für $n > 2$ teilt man A und B in je vier Blöcke und iteriert (**Strassen-Algorithmus**).
- Dies erfordert ca. $n^{\log_2(7)} \approx n^{2.81}$ Zahl-Multiplikationen.
- Aktueller Stand: $n^{2.371552}$ Zahl-Multiplikationen (Williams et al., 2024).

Das Assoziativgesetz

- Seien A, B, C Matrizen vom Format $n \times m$, $m \times k$ und $k \times l$.

Das Assoziativgesetz

- Seien A, B, C Matrizen vom Format $n \times m$, $m \times k$ und $k \times l$.
- Bekanntlich gilt das Assoziativgesetz $(AB)C = A(BC)$.

Das Assoziativgesetz

- Seien A, B, C Matrizen vom Format $n \times m$, $m \times k$ und $k \times l$.
- Bekanntlich gilt das Assoziativgesetz $(AB)C = A(BC)$.
- Aber: $(AB)C$ erfordert $nmk + nkl = nk(m + l)$ Zahl-Multiplikationen.

Das Assoziativgesetz

- Seien A, B, C Matrizen vom Format $n \times m$, $m \times k$ und $k \times l$.
- Bekanntlich gilt das Assoziativgesetz $(AB)C = A(BC)$.
- Aber: $(AB)C$ erfordert $nmk + nkl = nk(m + l)$ Zahl-Multiplikationen.
 $A(BC)$ erfordert $mk l + nml = ml(n + k)$ Zahl-Multiplikationen.

Das Assoziativgesetz

- Seien A, B, C Matrizen vom Format $n \times m$, $m \times k$ und $k \times l$.
- Bekanntlich gilt das Assoziativgesetz $(AB)C = A(BC)$.
- Aber: $(AB)C$ erfordert $nmk + nkl = nk(m + l)$ Zahl-Multiplikationen.
 $A(BC)$ erfordert $mkl + nml = ml(n + k)$ Zahl-Multiplikationen.

Fakt

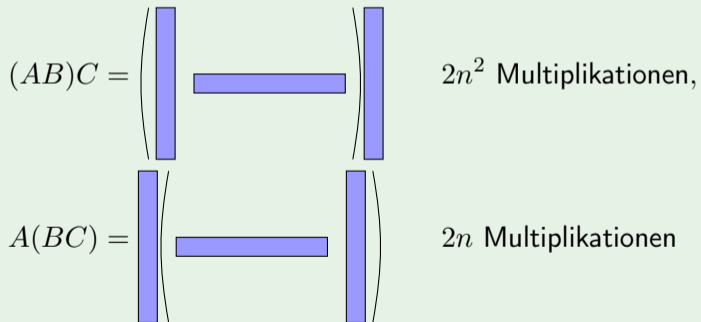
Genau dann lässt sich $(AB)C$ schneller berechnen als $A(BC)$, wenn

$$\frac{1}{m} + \frac{1}{l} < \frac{1}{n} + \frac{1}{k}.$$

Das Assoziativgesetz

Beispiel

Extremfall $n = k > 1 = m = l$:



Praktische Probleme mit Gleichungssystemen

- 1 Messabweichungen können große Auswirkungen auf die Lösung haben.

Praktische Probleme mit Gleichungssystemen

- ① Messabweichungen können große Auswirkungen auf die Lösung haben.
- ② Rundungsfehler beim Gauß-Algorithmus führen zu falschen Ergebnissen.

Praktische Probleme mit Gleichungssystemen

- 1 Messabweichungen können große Auswirkungen auf die Lösung haben.
- 2 Rundungsfehler beim Gauß-Algorithmus führen zu falschen Ergebnissen.
- 3 Überbestimmte Systeme haben keine exakte Lösung.

Kondition

Kleine Abweichungen der Eingabe bewirken große Veränderungen der Lösung:

Kondition

Kleine Abweichungen der Eingabe bewirken große Veränderungen der Lösung:

$$\begin{pmatrix} 1 & 1 \\ 1.1 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{Lösung: } x = \begin{pmatrix} -10 \\ 12 \end{pmatrix}$$

Kondition

Kleine Abweichungen der Eingabe bewirken große Veränderungen der Lösung:

$$\begin{pmatrix} 1 & 1 \\ 1.1 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{Lösung: } x = \begin{pmatrix} -10 \\ 12 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 1.01 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{Lösung: } x = \begin{pmatrix} -100 \\ 102 \end{pmatrix}$$

Kondition

Kleine Abweichungen der Eingabe bewirken große Veränderungen der Lösung:

$$\begin{pmatrix} 1 & 1 \\ 1.1 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$\text{Lösung: } x = \begin{pmatrix} -10 \\ 12 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 1.01 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$\text{Lösung: } x = \begin{pmatrix} -100 \\ 102 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

nicht lösbar!

Kondition

Kleine Abweichungen der Eingabe bewirken große Veränderungen der Lösung:

$$\begin{pmatrix} 1 & 1 \\ 1.1 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{Lösung: } x = \begin{pmatrix} -10 \\ 12 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 1.01 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{Lösung: } x = \begin{pmatrix} -100 \\ 102 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{nicht lösbar!}$$

- Grund: Die Determinante der Koeffizientenmatrix A ist (fast) 0.

Kondition

Kleine Abweichungen der Eingabe bewirken große Veränderungen der Lösung:

$$\begin{pmatrix} 1 & 1 \\ 1.1 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{Lösung: } x = \begin{pmatrix} -10 \\ 12 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 1.01 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{Lösung: } x = \begin{pmatrix} -100 \\ 102 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{nicht lösbar!}$$

- Grund: Die Determinante der Koeffizientenmatrix A ist (fast) 0.
- Folglich ist $A^{-1} = \det(A)^{-1} \tilde{A}$ sehr groß.

Kondition

Kleine Abweichungen der Eingabe bewirken große Veränderungen der Lösung:

$$\begin{pmatrix} 1 & 1 \\ 1.1 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{Lösung: } x = \begin{pmatrix} -10 \\ 12 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 1.01 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{Lösung: } x = \begin{pmatrix} -100 \\ 102 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} x = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{nicht lösbar!}$$

- Grund: Die Determinante der Koeffizientenmatrix A ist (fast) 0.
- Folglich ist $A^{-1} = \det(A)^{-1} \tilde{A}$ sehr groß.
- Man sagt, das Problem ist **schlecht konditioniert**.

Konditionszahl

- Genauer definiert man die **Konditionszahl**

$$\kappa(A) := \frac{\max\{|Ax| : |x| = 1\}}{\min\{|Ax| : |x| = 1\}} \geq 1.$$

Konditionszahl

- Genauer definiert man die **Konditionszahl**

$$\kappa(A) := \frac{\max\{|Ax| : |x| = 1\}}{\min\{|Ax| : |x| = 1\}} \geq 1.$$

- Je größer $\kappa(A)$, desto schwieriger das Problem.

Konditionszahl

- Genauer definiert man die **Konditionszahl**

$$\kappa(A) := \frac{\max\{|Ax| : |x| = 1\}}{\min\{|Ax| : |x| = 1\}} \geq 1.$$

- Je größer $\kappa(A)$, desto schwieriger das Problem.
- Um den Effekt nicht zu verschlimmern, benötigt man **stabile** Algorithmen.

Rundungsfehler

- Auf Computern lassen sich reelle Zahlen nur näherungsweise durch **Gleitkommazahlen** darstellen.

Rundungsfehler

- Auf Computern lassen sich reelle Zahlen nur näherungsweise durch **Gleitkommazahlen** darstellen.
- Selbst endliche Dezimalbrüche lassen sich nicht exakt speichern, wenn die interne Binärdarstellung unendlich ist:

$$0.1 = 2^{-4} + \dots \qquad \approx 0.0625$$

Rundungsfehler

- Auf Computern lassen sich reelle Zahlen nur näherungsweise durch **Gleitkommazahlen** darstellen.
- Selbst endliche Dezimalbrüche lassen sich nicht exakt speichern, wenn die interne Binärdarstellung unendlich ist:

$$0.1 = 2^{-4} + 2^{-5} + \dots \quad \approx 0.0938$$

Rundungsfehler

- Auf Computern lassen sich reelle Zahlen nur näherungsweise durch **Gleitkommazahlen** darstellen.
- Selbst endliche Dezimalbrüche lassen sich nicht exakt speichern, wenn die interne Binärdarstellung unendlich ist:

$$0.1 = 2^{-4} + 2^{-5} + 2^{-8} + \dots \quad \approx 0.0977$$

Rundungsfehler

- Auf Computern lassen sich reelle Zahlen nur näherungsweise durch **Gleitkommazahlen** darstellen.
- Selbst endliche Dezimalbrüche lassen sich nicht exakt speichern, wenn die interne Binärdarstellung unendlich ist:

$$0.1 = 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + \dots \approx 0.0996$$

Rundungsfehler

- Auf Computern lassen sich reelle Zahlen nur näherungsweise durch **Gleitkommazahlen** darstellen.
- Selbst endliche Dezimalbrüche lassen sich nicht exakt speichern, wenn die interne Binärdarstellung unendlich ist:

$$0.1 = 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + \dots \approx 0.0996$$

- Nehmen wir an, dass unser Datentyp nur vier Dezimalziffern + Exponent speichert:

$$10001 \rightsquigarrow 1.000e4 \quad 0.00123 \rightsquigarrow 1.23e-3.$$

Rundungsfehler

- Auf Computern lassen sich reelle Zahlen nur näherungsweise durch **Gleitkommazahlen** darstellen.
- Selbst endliche Dezimalbrüche lassen sich nicht exakt speichern, wenn die interne Binärdarstellung unendlich ist:

$$0.1 = 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + \dots \approx 0.0996$$

- Nehmen wir an, dass unser Datentyp nur vier Dezimalziffern + Exponent speichert:

$$10001 \rightsquigarrow 1.000e4 \quad 0.00123 \rightsquigarrow 1.23e-3.$$

- Subtraktion fast gleichgroßer Zahlen führt zum Verlust von Genauigkeit (**Auslöschung**):

$$1.234 - 1.233 = 1.???e-3.$$

Rundungsfehler beim Gauß-Algorithmus

Beispiel

Das Gleichungssystem

$$\begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix} x = \begin{pmatrix} 10^4 \\ 1 \end{pmatrix}$$

besitzt die eindeutige Lösung $x = (-10^4, 10^4 + 1)$.

Rundungsfehler beim Gauß-Algorithmus

Beispiel

Das Gleichungssystem

$$\begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix} x = \begin{pmatrix} 10^4 \\ 1 \end{pmatrix}$$

besitzt die eindeutige Lösung $x = (-10^4, 10^4 + 1)$.

Der Gauß-Algorithmus mit Gleitkommazahlen liefert jedoch

$$\left(\begin{array}{cc|c} 1e-4 & 1 & 1e4 \\ 1 & 1 & 1 \end{array} \right) \sim \left(\begin{array}{cc|c} 1 & 1e4 & 1e8 \\ 1 & 1 & 1 \end{array} \right) \sim \left(\begin{array}{cc|c} 1 & 1e4 & 1e8 \\ 0 & -1e4 & -1e8 \end{array} \right)$$

Rundungsfehler beim Gauß-Algorithmus

Beispiel

Das Gleichungssystem

$$\begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix} x = \begin{pmatrix} 10^4 \\ 1 \end{pmatrix}$$

besitzt die eindeutige Lösung $x = (-10^4, 10^4 + 1)$.

Der Gauß-Algorithmus mit Gleitkommazahlen liefert jedoch

$$\begin{pmatrix} 1e-4 & 1 & | & 1e4 \\ 1 & 1 & | & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 1e4 & | & 1e8 \\ 1 & 1 & | & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 1e4 & | & 1e8 \\ 0 & -1e4 & | & -1e8 \end{pmatrix}$$
$$\implies x = (0, 1e4) \quad \color{red}{\times}$$

Fazit: Der Gauß-Algorithmus in Reinform ist **instabil**.

Vorkonditionierung

- Idee: Ersetze $Ax = b$ durch $A'y = b'$, wobei $A' := SAT^{-1}$ und $b' := Sb$ mit $S, T \in O(n, \mathbb{R})$.

Vorkonditionierung

- Idee: Ersetze $Ax = b$ durch $A'y = b'$, wobei $A' := SAT^{-1}$ und $b' := Sb$ mit $S, T \in O(n, \mathbb{R})$.
- Löse $A'y = b'$ und danach $Tx = y$.

Vorkonditionierung

- Idee: Ersetze $Ax = b$ durch $A'y = b'$, wobei $A' := SAT^{-1}$ und $b' := Sb$ mit $S, T \in O(n, \mathbb{R})$.
- Löse $A'y = b'$ und danach $Tx = y$.
- Dann gilt $Ax = S^{-1}SAT^{-1}y = S^{-1}A'y = S^{-1}b' = b$.

Vorkonditionierung

- Idee: Ersetze $Ax = b$ durch $A'y = b'$, wobei $A' := SAT^{-1}$ und $b' := Sb$ mit $S, T \in O(n, \mathbb{R})$.
- Löse $A'y = b'$ und danach $Tx = y$.
- Dann gilt $Ax = S^{-1}SAT^{-1}y = S^{-1}A'y = S^{-1}b' = b$.
- Da S und T orthogonal sind, gilt $\{x : |x| = 1\} = \{T^{-1}x : |x| = 1\}$ und

$$\{|Ax| : |x| = 1\} = \{|SAT^{-1}x| : |x| = 1\} = \{|A'x| : |x| = 1\}.$$

Vorkonditionierung

- Idee: Ersetze $Ax = b$ durch $A'y = b'$, wobei $A' := SAT^{-1}$ und $b' := Sb$ mit $S, T \in O(n, \mathbb{R})$.
- Löse $A'y = b'$ und danach $Tx = y$.
- Dann gilt $Ax = S^{-1}SAT^{-1}y = S^{-1}A'y = S^{-1}b' = b$.
- Da S und T orthogonal sind, gilt $\{x : |x| = 1\} = \{T^{-1}x : |x| = 1\}$ und

$$\{|Ax| : |x| = 1\} = \{|SAT^{-1}x| : |x| = 1\} = \{|A'x| : |x| = 1\}.$$

- Dies zeigt $\kappa(A) = \kappa(A')$, d. h. das neue System ist nicht schlechter konditioniert.

Vorkonditionierung

- Idee: Ersetze $Ax = b$ durch $A'y = b'$, wobei $A' := SAT^{-1}$ und $b' := Sb$ mit $S, T \in O(n, \mathbb{R})$.
- Löse $A'y = b'$ und danach $Tx = y$.
- Dann gilt $Ax = S^{-1}SAT^{-1}y = S^{-1}A'y = S^{-1}b' = b$.
- Da S und T orthogonal sind, gilt $\{x : |x| = 1\} = \{T^{-1}x : |x| = 1\}$ und

$$\{|Ax| : |x| = 1\} = \{|SAT^{-1}x| : |x| = 1\} = \{|A'x| : |x| = 1\}.$$

- Dies zeigt $\kappa(A) = \kappa(A')$, d. h. das neue System ist nicht schlechter konditioniert.
- Wegen $\kappa(T) = 1$ ist das System $Tx = y$ **gut konditioniert**.

Vorkonditionierung

- Idee: Ersetze $Ax = b$ durch $A'y = b'$, wobei $A' := SAT^{-1}$ und $b' := Sb$ mit $S, T \in O(n, \mathbb{R})$.
- Löse $A'y = b'$ und danach $Tx = y$.
- Dann gilt $Ax = S^{-1}SAT^{-1}y = S^{-1}A'y = S^{-1}b' = b$.
- Da S und T orthogonal sind, gilt $\{x : |x| = 1\} = \{T^{-1}x : |x| = 1\}$ und

$$\{|Ax| : |x| = 1\} = \{|SAT^{-1}x| : |x| = 1\} = \{|A'x| : |x| = 1\}.$$

- Dies zeigt $\kappa(A) = \kappa(A')$, d. h. das neue System ist nicht schlechter konditioniert.
- Wegen $\kappa(T) = 1$ ist das System $Tx = y$ **gut konditioniert**.
- Wähle S und T , sodass $A'y = b'$ „leicht“ zu lösen ist.

Wahlen von S und T

Wie findet man geeignete S und T ?

Wahlen von S und T

Wie findet man geeignete S und T ?

Satz (Singulärwertzerlegung)

Für $A \in \mathbb{R}^{n \times n}$ existieren $S, T \in O(n, \mathbb{R})$, sodass SAT^{-1} eine Diagonalmatrix mit nicht-negativen Einträgen ist. Die positiven Einträge sind bis auf die Reihenfolge eindeutig bestimmt und heißen **Singulärwerte** von A .

Wahlen von S und T

Wie findet man geeignete S und T ?

Satz (Singulärwertzerlegung)

Für $A \in \mathbb{R}^{n \times n}$ existieren $S, T \in O(n, \mathbb{R})$, sodass SAT^{-1} eine Diagonalmatrix mit nicht-negativen Einträgen ist. Die positiven Einträge sind bis auf die Reihenfolge eindeutig bestimmt und heißen **Singulärwerte** von A .

- Ist $A' = SAT^{-1}$ eine Diagonalmatrix, so ist $A'y = b'$ besonders leicht zu lösen.

Wahlen von S und T

Wie findet man geeignete S und T ?

Satz (Singulärwertzerlegung)

Für $A \in \mathbb{R}^{n \times n}$ existieren $S, T \in O(n, \mathbb{R})$, sodass SAT^{-1} eine Diagonalmatrix mit nicht-negativen Einträgen ist. Die positiven Einträge sind bis auf die Reihenfolge eindeutig bestimmt und heißen **Singulärwerte** von A .

- Ist $A' = SAT^{-1}$ eine Diagonalmatrix, so ist $A'y = b'$ besonders leicht zu lösen.
- Problem: Die genaue Berechnung von S und T ist schwierig.

Pivotisierung

- Versuch: Wähle für S oder T Permutationsmatrizen.

Pivotisierung

- Versuch: Wähle für S oder T Permutationsmatrizen.
- Dies realisiert Zeilen- bzw. Spaltenvertauschungen von A .

Pivotisierung

- Versuch: Wähle für S oder T Permutationsmatrizen.
- Dies realisiert Zeilen- bzw. Spaltenvertauschungen von A .
- Man maximiert dadurch den ersten Eintrag (**Pivot-Element**) von A .

Pivotisierung

- Versuch: Wähle für S oder T Permutationsmatrizen.
- Dies realisiert Zeilen- bzw. Spaltenvertauschungen von A .
- Man maximiert dadurch den ersten Eintrag (**Pivot-Element**) von A .

Beispiel

Im vorherigen Beispiel tauschen wir Spalten (also $T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$):

$$\left(\begin{array}{cc|c} 1 & 1e-4 & 1e4 \\ 1 & 1 & 1 \end{array} \right) \sim \left(\begin{array}{cc|c} 1 & 1e-4 & 1e4 \\ 0 & 1 & -1e4 \end{array} \right)$$

Pivotisierung

- Versuch: Wähle für S oder T Permutationsmatrizen.
- Dies realisiert Zeilen- bzw. Spaltenvertauschungen von A .
- Man maximiert dadurch den ersten Eintrag (**Pivot-Element**) von A .

Beispiel

Im vorherigen Beispiel tauschen wir Spalten (also $T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$):

$$\left(\begin{array}{cc|c} 1 & 1e-4 & 1e4 \\ 1 & 1 & 1 \end{array} \right) \sim \left(\begin{array}{cc|c} 1 & 1e-4 & 1e4 \\ 0 & 1 & -1e4 \end{array} \right)$$
$$y = (1e4, -1e4),$$

Pivotisierung

- Versuch: Wähle für S oder T Permutationsmatrizen.
- Dies realisiert Zeilen- bzw. Spaltenvertauschungen von A .
- Man maximiert dadurch den ersten Eintrag (**Pivot-Element**) von A .

Beispiel

Im vorherigen Beispiel tauschen wir Spalten (also $T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$):

$$\left(\begin{array}{cc|c} 1 & 1e-4 & 1e4 \\ 1 & 1 & 1 \end{array} \right) \sim \left(\begin{array}{cc|c} 1 & 1e-4 & 1e4 \\ 0 & 1 & -1e4 \end{array} \right)$$

$$y = (1e4, -1e4),$$

$$x = (-1e4, 1e4) \approx (-10^4, 10^4 + 1) \checkmark$$

Überbestimmte Gleichungssysteme

- In der Neujahrsnacht 1801 entdeckte der Astronom Piazzi den Zwergplaneten **Ceres**.

Überbestimmte Gleichungssysteme

- In der Neujahrsnacht 1801 entdeckte der Astronom Piazzi den Zwergplaneten **Ceres**.
- Die genaue Position wurde 40 Tage aufgezeichnet bis Ceres aus dem Blickfeld verschwand.

Überbestimmte Gleichungssysteme

- In der Neujahrsnacht 1801 entdeckte der Astronom Piazzi den Zwergplaneten **Ceres**.
- Die genaue Position wurde 40 Tage aufgezeichnet bis Ceres aus dem Blickfeld verschwand.
- Die Koordinaten von Ceres beschreiben ein **überbestimmtes** Gleichungssystem $Ax = b$, wobei x die Parameter der gesuchten Bahnkurve enthält ($A \in \mathbb{R}^{n \times m}$ mit $\text{rk}(A) = m < n$).

Überbestimmte Gleichungssysteme

- In der Neujahrsnacht 1801 entdeckte der Astronom Piazzi den Zwergplaneten **Ceres**.
- Die genaue Position wurde 40 Tage aufgezeichnet bis Ceres aus dem Blickfeld verschwand.
- Die Koordinaten von Ceres beschreiben ein **überbestimmtes** Gleichungssystem $Ax = b$, wobei x die Parameter der gesuchten Bahnkurve enthält ($A \in \mathbb{R}^{n \times m}$ mit $\text{rk}(A) = m < n$).
- Wegen Messabweichungen gibt es keine exakte Lösung. Wir suchen daher \tilde{x} mit

$$|A\tilde{x} - b| = \min\{|Ax - b| : x \in \mathbb{R}^m\}.$$

Satz (Methode der kleinsten Quadrate)

Die **Normalgleichung** $A^t Ax = A^t b$ besitzt genau eine Lösung \tilde{x} und es gilt $|A\tilde{x} - b| < |Ax - b|$ für alle $x \neq \tilde{x}$.

Satz (Methode der kleinsten Quadrate)

Die **Normalgleichung** $A^t Ax = A^t b$ besitzt genau eine Lösung \tilde{x} und es gilt $|A\tilde{x} - b| < |Ax - b|$ für alle $x \neq \tilde{x}$.

Beweis.

Für $x \in \text{Ker}(A^t A)$ gilt

$$|Ax|^2 = x^t A^t Ax = 0 \implies Ax = 0 \xrightarrow{\text{rk}(A)=m} x = 0.$$

Satz (Methode der kleinsten Quadrate)

Die **Normalgleichung** $A^t Ax = A^t b$ besitzt genau eine Lösung \tilde{x} und es gilt $|A\tilde{x} - b| < |Ax - b|$ für alle $x \neq \tilde{x}$.

Beweis.

Für $x \in \text{Ker}(A^t A)$ gilt

$$|Ax|^2 = x^t A^t Ax = 0 \implies Ax = 0 \stackrel{\text{rk}(A)=m}{\implies} x = 0.$$

Dies zeigt $\text{Ker}(A^t A) = \{0\}$ und $A^t A \in \mathbb{R}^{m \times m}$ ist invertierbar.

Satz (Methode der kleinsten Quadrate)

Die **Normalgleichung** $A^t Ax = A^t b$ besitzt genau eine Lösung \tilde{x} und es gilt $|A\tilde{x} - b| < |Ax - b|$ für alle $x \neq \tilde{x}$.

Beweis.

Für $x \in \text{Ker}(A^t A)$ gilt

$$|Ax|^2 = x^t A^t Ax = 0 \implies Ax = 0 \stackrel{\text{rk}(A)=m}{\implies} x = 0.$$

Dies zeigt $\text{Ker}(A^t A) = \{0\}$ und $A^t A \in \mathbb{R}^{m \times m}$ ist invertierbar.

Also ist \tilde{x} eindeutig bestimmt.

Satz (Methode der kleinsten Quadrate)

Die **Normalgleichung** $A^t Ax = A^t b$ besitzt genau eine Lösung \tilde{x} und es gilt $|A\tilde{x} - b| < |Ax - b|$ für alle $x \neq \tilde{x}$.

Beweis.

Für $x \in \text{Ker}(A^t A)$ gilt

$$|Ax|^2 = x^t A^t Ax = 0 \implies Ax = 0 \stackrel{\text{rk}(A)=m}{\implies} x = 0.$$

Dies zeigt $\text{Ker}(A^t A) = \{0\}$ und $A^t A \in \mathbb{R}^{m \times m}$ ist invertierbar.

Also ist \tilde{x} eindeutig bestimmt. Für $y \neq 0$ gilt

$$|A(\tilde{x} + y) - b|^2 = [(A\tilde{x} - b) + Ay, (A\tilde{x} - b) + Ay]$$



Satz (Methode der kleinsten Quadrate)

Die **Normalgleichung** $A^t Ax = A^t b$ besitzt genau eine Lösung \tilde{x} und es gilt $|A\tilde{x} - b| < |Ax - b|$ für alle $x \neq \tilde{x}$.

Beweis.

Für $x \in \text{Ker}(A^t A)$ gilt

$$|Ax|^2 = x^t A^t Ax = 0 \implies Ax = 0 \stackrel{\text{rk}(A)=m}{\implies} x = 0.$$

Dies zeigt $\text{Ker}(A^t A) = \{0\}$ und $A^t A \in \mathbb{R}^{m \times m}$ ist invertierbar.

Also ist \tilde{x} eindeutig bestimmt. Für $y \neq 0$ gilt

$$\begin{aligned} |A(\tilde{x} + y) - b|^2 &= [(A\tilde{x} - b) + Ay, (A\tilde{x} - b) + Ay] \\ &= |A\tilde{x} - b|^2 + 2[Ay, A\tilde{x} - b] + |Ay|^2 \end{aligned}$$



Satz (Methode der kleinsten Quadrate)

Die **Normalgleichung** $A^t Ax = A^t b$ besitzt genau eine Lösung \tilde{x} und es gilt $|A\tilde{x} - b| < |Ax - b|$ für alle $x \neq \tilde{x}$.

Beweis.

Für $x \in \text{Ker}(A^t A)$ gilt

$$|Ax|^2 = x^t A^t Ax = 0 \implies Ax = 0 \stackrel{\text{rk}(A)=m}{\implies} x = 0.$$

Dies zeigt $\text{Ker}(A^t A) = \{0\}$ und $A^t A \in \mathbb{R}^{m \times m}$ ist invertierbar.

Also ist \tilde{x} eindeutig bestimmt. Für $y \neq 0$ gilt

$$\begin{aligned} |A(\tilde{x} + y) - b|^2 &= [(A\tilde{x} - b) + Ay, (A\tilde{x} - b) + Ay] \\ &= |A\tilde{x} - b|^2 + 2[Ay, A\tilde{x} - b] + |Ay|^2 \\ &= |A\tilde{x} - b|^2 + 2y^t A^t (A\tilde{x} - b) + |Ay|^2 \quad (= 0) \end{aligned}$$



Satz (Methode der kleinsten Quadrate)

Die **Normalgleichung** $A^t Ax = A^t b$ besitzt genau eine Lösung \tilde{x} und es gilt $|A\tilde{x} - b| < |Ax - b|$ für alle $x \neq \tilde{x}$.

Beweis.

Für $x \in \text{Ker}(A^t A)$ gilt

$$|Ax|^2 = x^t A^t Ax = 0 \implies Ax = 0 \stackrel{\text{rk}(A)=m}{\implies} x = 0.$$

Dies zeigt $\text{Ker}(A^t A) = \{0\}$ und $A^t A \in \mathbb{R}^{m \times m}$ ist invertierbar.

Also ist \tilde{x} eindeutig bestimmt. Für $y \neq 0$ gilt

$$\begin{aligned} |A(\tilde{x} + y) - b|^2 &= [(A\tilde{x} - b) + Ay, (A\tilde{x} - b) + Ay] \\ &= |A\tilde{x} - b|^2 + 2[Ay, A\tilde{x} - b] + |Ay|^2 \\ &= |A\tilde{x} - b|^2 + 2y^t A^t (A\tilde{x} - b) + |Ay|^2 \quad (= 0) \\ &= |A\tilde{x} - b|^2 + |Ay|^2 > |A\tilde{x} - b|^2. \end{aligned}$$

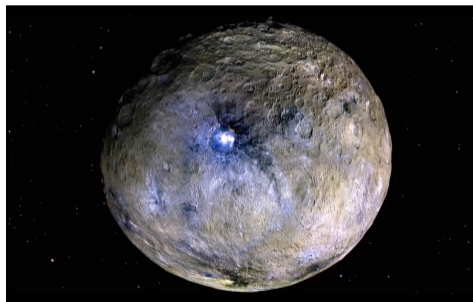


Anwendung

- Gauß hat auf diese Weise die Bahn von Ceres approximiert und die genaue Position erfolgreich vorhersagen können.

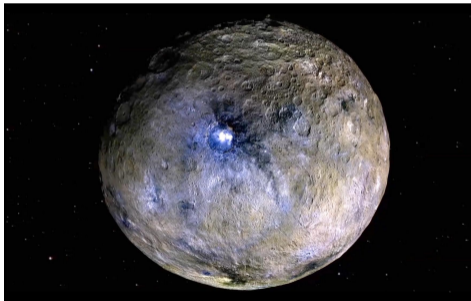
Anwendung

- Gauß hat auf diese Weise die Bahn von Ceres approximiert und die genaue Position erfolgreich vorhersagen können.
- 2015 hat die NASA-Raumsonde **Dawn** Ceres fotografiert:



Anwendung

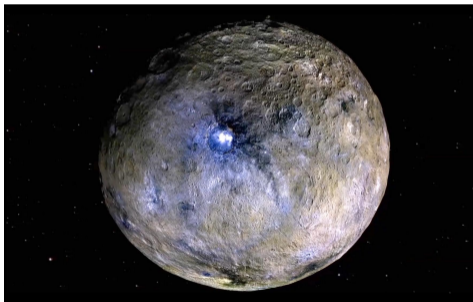
- Gauß hat auf diese Weise die Bahn von Ceres approximiert und die genaue Position erfolgreich vorhersagen können.
- 2015 hat die NASA-Raumsonde **Dawn** Ceres fotografiert:



- Die Daten wurden mit dem **Hamming-Code** übertragen.

Anwendung

- Gauß hat auf diese Weise die Bahn von Ceres approximiert und die genaue Position erfolgreich vorhersagen können.
- 2015 hat die NASA-Raumsonde **Dawn** Ceres fotografiert:



- Die Daten wurden mit dem **Hamming-Code** übertragen.
- Der helle Fleck ist der mit Salz gefüllte Meteoriten-Krater **Occator**.

Verfahren für positiv definite Matrizen

Für das System $A^t Ax = A^t b$ gibt es spezielle stabile Verfahren:

Verfahren für positiv definite Matrizen

Für das System $A^t Ax = A^t b$ gibt es spezielle stabile Verfahren:

Satz (Cholesky-Zerlegung)

Hat $A \in \mathbb{R}^{n \times m}$ vollen Rang, so existiert genau eine obere Dreiecksmatrix $R \in \mathbb{R}^{m \times m}$ mit positiven Diagonaleinträgen und $A^t A = R^t R$.

Verfahren für positiv definite Matrizen

Für das System $A^t Ax = A^t b$ gibt es spezielle stabile Verfahren:

Satz (Cholesky-Zerlegung)

Hat $A \in \mathbb{R}^{n \times m}$ vollen Rang, so existiert genau eine obere Dreiecksmatrix $R \in \mathbb{R}^{m \times m}$ mit positiven Diagonaleinträgen und $A^t A = R^t R$.

Die Matrix R lässt sich leicht rekursiv bestimmen:

$$A^t A = \begin{pmatrix} A_{m-1} & a \\ a^t & \lambda \end{pmatrix} = \begin{pmatrix} R_{m-1}^t & 0 \\ r^t & \mu \end{pmatrix} \begin{pmatrix} R_{m-1} & r \\ 0 & \mu \end{pmatrix}$$

Verfahren für positiv definite Matrizen

Für das System $A^t Ax = A^t b$ gibt es spezielle stabile Verfahren:

Satz (Cholesky-Zerlegung)

Hat $A \in \mathbb{R}^{n \times m}$ vollen Rang, so existiert genau eine obere Dreiecksmatrix $R \in \mathbb{R}^{m \times m}$ mit positiven Diagonaleinträgen und $A^t A = R^t R$.

Die Matrix R lässt sich leicht rekursiv bestimmen:

$$A^t A = \begin{pmatrix} A_{m-1} & a \\ a^t & \lambda \end{pmatrix} = \begin{pmatrix} R_{m-1}^t & 0 \\ r^t & \mu \end{pmatrix} \begin{pmatrix} R_{m-1} & r \\ 0 & \mu \end{pmatrix}$$
$$R_{m-1}^t r = a, \quad \mu = \sqrt{\lambda - r^t r}$$

Vorwärts- und Rückwärtssubstitution

Anstelle von $R^t R x = b$ löst man $R^t y = b$ durch **Vorwärtssubstitution**:

$$\begin{pmatrix} r_{11} & & & 0 \\ r_{21} & r_{22} & & \\ \vdots & \ddots & \ddots & \\ r_{m1} & \cdots & \cdots & r_{mm} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \Rightarrow \begin{cases} y_1 = \frac{b_1}{r_{11}} \\ y_2 = \frac{b_2 - r_{21}y_1}{r_{22}} \\ \vdots \end{cases}$$

Vorwärts- und Rückwärtssubstitution

Anstelle von $R^t R x = b$ löst man $R^t y = b$ durch **Vorwärtssubstitution**:

$$\begin{pmatrix} r_{11} & & & 0 \\ r_{21} & r_{22} & & \\ \vdots & \ddots & \ddots & \\ r_{m1} & \cdots & \cdots & r_{mm} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \Rightarrow \begin{cases} y_1 = \frac{b_1}{r_{11}} \\ y_2 = \frac{b_2 - r_{21}y_1}{r_{22}} \\ \vdots \end{cases}$$

und anschließend $R x = y$ durch **Rückwärtssubstitution**:

$$\begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ & \ddots & \ddots & \vdots \\ & & r_{m-1,m-1} & r_{m-1,m} \\ 0 & & & r_{mm} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \Rightarrow \begin{cases} x_m = \frac{y_m}{r_{mm}} \\ x_{m-1} = \frac{y_{m-1} - r_{m-1,m}x_m}{r_{m-1,m-1}} \\ \vdots \end{cases}$$

Splitting-Methoden

- Für große Matrizen ($n > 10^5$) ist die Cholesky-Zerlegung zu aufwendig.

Splitting-Methoden

- Für große Matrizen ($n > 10^5$) ist die Cholesky-Zerlegung zu aufwendig.
- Stattdessen benutzt man **iterative Verfahren**, die auf einer additiven Zerlegung basieren:

$$A = \begin{pmatrix} * & & 0 \\ \vdots & \ddots & \\ * & \cdots & * \end{pmatrix} + \begin{pmatrix} 0 & & * \\ & \ddots & \\ 0 & & 0 \end{pmatrix} = L + R.$$

Splitting-Methoden

- Für große Matrizen ($n > 10^5$) ist die Cholesky-Zerlegung zu aufwendig.
- Stattdessen benutzt man **iterative Verfahren**, die auf einer additiven Zerlegung basieren:

$$A = \begin{pmatrix} * & & 0 \\ \vdots & \ddots & \\ * & \cdots & * \end{pmatrix} + \begin{pmatrix} 0 & & * \\ & \ddots & \\ 0 & & 0 \end{pmatrix} = L + R.$$

- Wegen

$$Ax = b \iff Lx + Rx = b \iff x = -L^{-1}Rx + L^{-1}b$$

suchen wir Fixpunkte von $f(x) = -L^{-1}Rx + L^{-1}b$.

Schon wieder Gauß

Satz (Gauß-Seidel-Verfahren)

Ist $A = L + R \in \mathbb{R}^{n \times n}$ symmetrisch und positiv definit und $x_1 \in \mathbb{R}^n$, so konvergiert

$$x_{k+1} := -L^{-1}Rx_k + L^{-1}b \quad (k = 1, 2, \dots)$$

gegen die Lösung von $Ax = b$.

Schon wieder Gauß

Satz (Gauß-Seidel-Verfahren)

Ist $A = L + R \in \mathbb{R}^{n \times n}$ symmetrisch und positiv definit und $x_1 \in \mathbb{R}^n$, so konvergiert

$$x_{k+1} := -L^{-1}Rx_k + L^{-1}b \quad (k = 1, 2, \dots)$$

gegen die Lösung von $Ax = b$.

Der Beweis benutzt **Banachs Fixpunktsatz** aus der Analysis.

Schon wieder Gauß

Satz (Gauß-Seidel-Verfahren)

Ist $A = L + R \in \mathbb{R}^{n \times n}$ symmetrisch und positiv definit und $x_1 \in \mathbb{R}^n$, so konvergiert

$$x_{k+1} := -L^{-1}Rx_k + L^{-1}b \quad (k = 1, 2, \dots)$$

gegen die Lösung von $Ax = b$.

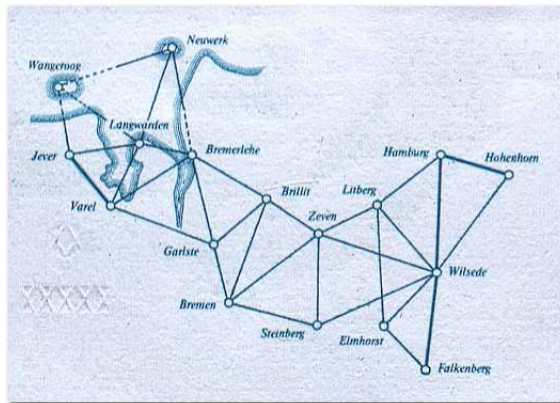
Der Beweis benutzt **Banachs Fixpunktsatz** aus der Analysis.

Zitat Gauß:

„Ich empfehle Ihnen diesen Modus zur Nachahmung. Schwerlich werden Sie je wieder direct eliminieren, wenigstens nicht, wenn Sie mehr als 2 Unbekannte haben. Das indirecte Verfahren lässt sich halb im Schlafe ausführen, oder man kann während desselben an andere Dinge denken.“

Anwendung

Gauß hat mit dieser Methode das **Königreich Hannover** vermessen. Davon zeugt eine Zeichnung auf dem 10DM-Schein:



Benchmark

Besonders schlecht konditioniert ist das Gleichungssystem $H_n x = b$ mit der **Hilbert-Matrix**

$$H_n := \left(\frac{1}{i+j-1} \right)_{i,j} = \begin{pmatrix} 1 & 1/2 & \cdots & 1/n \\ 1/2 & 1/3 & \cdots & 1/n+1 \\ \vdots & \vdots & & \vdots \\ 1/n & 1/n+1 & \cdots & 1/2n-1 \end{pmatrix}.$$

Benchmark

Besonders schlecht konditioniert ist das Gleichungssystem $H_n x = b$ mit der **Hilbert-Matrix**

$$H_n := \left(\frac{1}{i+j-1} \right)_{i,j} = \begin{pmatrix} 1 & 1/2 & \cdots & 1/n \\ 1/2 & 1/3 & \cdots & 1/n+1 \\ \vdots & \vdots & & \vdots \\ 1/n & 1/n+1 & \cdots & 1/2n-1 \end{pmatrix}.$$

Zum Beispiel ist $\det H_4 = 1/6048000$ und $\kappa(H_4) \approx 15514$.

Benchmark

Beispiel

Das System $H_4x = (1, 1, 1, 1)^t$ hat die Lösung $x = (-4, 60, -180, 140)^t$.

Benchmark

Beispiel

Das System $H_4x = (1, 1, 1, 1)^t$ hat die Lösung $x = (-4, 60, -180, 140)^t$. Gauß-Seidel liefert:

$$x_1 := (\quad 0, \quad 0, \quad 0, \quad 0 \quad)$$

Benchmark

Beispiel

Das System $H_4x = (1, 1, 1, 1)^t$ hat die Lösung $x = (-4, 60, -180, 140)^t$. Gauß-Seidel liefert:

$$\begin{aligned}x_1 &:= (0, 0, 0, 0) \\x_{10} &\approx (-1.37, -2.27, 4.34, 7.51)\end{aligned}$$

Benchmark

Beispiel

Das System $H_4x = (1, 1, 1, 1)^t$ hat die Lösung $x = (-4, 60, -180, 140)^t$. Gauß-Seidel liefert:

$$\begin{array}{lcl} x_1 & := & (\quad 0, \quad 0, \quad 0, \quad 0 \quad) \\ x_{10} & \approx & (-1.37, \quad -2.27, \quad 4.34, \quad 7.51 \quad) \\ x_{100} & \approx & (\quad 2.84, \quad -14.30, \quad -5.01, \quad 27.8 \quad) \end{array}$$

Benchmark

Beispiel

Das System $H_4x = (1, 1, 1, 1)^t$ hat die Lösung $x = (-4, 60, -180, 140)^t$. Gauß-Seidel liefert:

$$\begin{array}{lcl} x_1 & := & (\quad 0, \quad 0, \quad 0, \quad 0 \quad) \\ x_{10} & \approx & (-1.37, \quad -2.27, \quad 4.34, \quad 7.51 \quad) \\ x_{100} & \approx & (\quad 2.84, \quad -14.30, \quad -5.01, \quad 27.8 \quad) \\ x_{1000} & \approx & (-1.10, \quad 28.80, \quad -107.00, \quad 93.3 \quad) \end{array}$$

Benchmark

Beispiel

Das System $H_4x = (1, 1, 1, 1)^t$ hat die Lösung $x = (-4, 60, -180, 140)^t$. Gauß-Seidel liefert:

$$\begin{array}{lcl} x_1 & := & (\quad 0, \quad 0, \quad 0, \quad 0 \quad) \\ x_{10} & \approx & (-1.37, \quad -2.27, \quad 4.34, \quad 7.51 \quad) \\ x_{100} & \approx & (\quad 2.84, \quad -14.30, \quad -5.01, \quad 27.8 \quad) \\ x_{1000} & \approx & (-1.10, \quad 28.80, \quad -107.00, \quad 93.3 \quad) \\ x_{10000} & \approx & (-4.000, \quad 59.995, \quad -179.988, \quad 139.992 \quad) \end{array}$$

Probleme bei der Eigenwertberechnung

- 1 Das charakteristische Polynom ist schlecht konditioniert.

Probleme bei der Eigenwertberechnung

- 1 Das charakteristische Polynom ist schlecht konditioniert.
- 2 Die Nullstellen eines Polynoms sind schlecht konditioniert.

Probleme bei der Eigenwertberechnung

- 1 Das charakteristische Polynom ist schlecht konditioniert.
- 2 Die Nullstellen eines Polynoms sind schlecht konditioniert.
- 3 Es gibt keine exakte Lösungsformel für die Nullstellen eines Polynoms.

Kondition des charakteristischen Polynoms

- Das Absolutglied des charakteristischen Polynoms χ_A ist $\pm \det A$.

Kondition des charakteristischen Polynoms

- Das Absolutglied des charakteristischen Polynoms χ_A ist $\pm \det A$.
- Mit $\det A$ ist auch χ_A schlecht konditioniert:

$$\det \begin{pmatrix} 1 & 33 \\ 3 & 100 \end{pmatrix} = 1$$

$$\det \begin{pmatrix} 1.1 & 33 \\ 3 & 100 \end{pmatrix} = 11$$

Kondition des charakteristischen Polynoms

- Das Absolutglied des charakteristischen Polynoms χ_A ist $\pm \det A$.
- Mit $\det A$ ist auch χ_A schlecht konditioniert:

$$\det \begin{pmatrix} 1 & 33 \\ 3 & 100 \end{pmatrix} = 1$$

$$\det \begin{pmatrix} 1.1 & 33 \\ 3 & 100 \end{pmatrix} = 11$$

- Noch schlechter ist das Minimalpolynom μ_A konditioniert, denn jede Abweichung kann sogar $\deg \mu_A$ ändern.

Kondition der Nullstellen eines Polynoms

Ebenso sind die Nullstellen eines Polynoms schlecht konditioniert:

$$X^2 - 21X + 110$$

Nullstellen: 10, 11

$$X^2 - 21.1X + 110$$

Nullstellen: $\approx 9.41, 11.69$

$$X^2 - 20.9X + 110$$

Nullstellen: $\approx 10.45 \pm 0.89i$

Kondition der Nullstellen eines Polynoms

Ebenso sind die Nullstellen eines Polynoms schlecht konditioniert:

$$X^2 - 21X + 110$$

Nullstellen: 10, 11

$$X^2 - 21.1X + 110$$

Nullstellen: $\approx 9.41, 11.69$

$$X^2 - 20.9X + 110$$

Nullstellen: $\approx 10.45 \pm 0.89i$

Dennoch gilt:

Fakt

Die Eigenwerte einer symmetrischen Matrix sind **gut konditioniert!**

Kondition der Nullstellen eines Polynoms

Ebenso sind die Nullstellen eines Polynoms schlecht konditioniert:

$$X^2 - 21X + 110$$

Nullstellen: 10, 11

$$X^2 - 21.1X + 110$$

Nullstellen: $\approx 9.41, 11.69$

$$X^2 - 20.9X + 110$$

Nullstellen: $\approx 10.45 \pm 0.89i$

Dennoch gilt:

Fakt

Die Eigenwerte einer symmetrischen Matrix sind **gut konditioniert!**

Strategie: Bestimme die Eigenwerte direkt (ohne Polynome).

Gershgorin-Kreise

Die Lage der Eigenwerte in der komplexen Ebene lässt sich leicht eingrenzen:

Gershgorin-Kreise

Die Lage der Eigenwerte in der komplexen Ebene lässt sich leicht eingrenzen:

Satz (GERSHGORIN)

Für jeden Eigenwert $\lambda \in \mathbb{C}$ von $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ existiert ein i mit

$$|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|.$$

Gershgorin-Kreise

Die Lage der Eigenwerte in der komplexen Ebene lässt sich leicht eingrenzen:

Satz (GERSHGORIN)

Für jeden Eigenwert $\lambda \in \mathbb{C}$ von $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ existiert ein i mit

$$|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|.$$

Je kleiner die Einträge a_{ij} mit $i \neq j$ sind, desto genauer lassen sich die Eigenwerte lokalisieren.

Gershgorin-Kreise

Beweis.

Sei $x = (x_1, \dots, x_n) \in \mathbb{C}^n$ ein Eigenvektor zu λ mit

$$|x_i| = \max\{|x_1|, \dots, |x_n|\}.$$

Gershgorin-Kreise

Beweis.

Sei $x = (x_1, \dots, x_n) \in \mathbb{C}^n$ ein Eigenvektor zu λ mit

$$|x_i| = \max\{|x_1|, \dots, |x_n|\}.$$

Nach Skalierung mit x_i^{-1} gilt $x_i = 1$.

Gershgorin-Kreise

Beweis.

Sei $x = (x_1, \dots, x_n) \in \mathbb{C}^n$ ein Eigenvektor zu λ mit

$$|x_i| = \max\{|x_1|, \dots, |x_n|\}.$$

Nach Skalierung mit x_i^{-1} gilt $x_i = 1$. Aus der Dreiecksungleichung folgt

$$|\lambda - a_{ii}| = |\lambda x_i - a_{ii} x_i| = |(Ax)_i - a_{ii} x_i| = \left| \sum_{j=1}^n a_{ij} x_j - a_{ii} x_i \right|$$



Gershgorin-Kreise

Beweis.

Sei $x = (x_1, \dots, x_n) \in \mathbb{C}^n$ ein Eigenvektor zu λ mit

$$|x_i| = \max\{|x_1|, \dots, |x_n|\}.$$

Nach Skalierung mit x_i^{-1} gilt $x_i = 1$. Aus der Dreiecksungleichung folgt

$$\begin{aligned} |\lambda - a_{ii}| &= |\lambda x_i - a_{ii} x_i| = |(Ax)_i - a_{ii} x_i| = \left| \sum_{j=1}^n a_{ij} x_j - a_{ii} x_i \right| \\ &= \left| \sum_{j \neq i} a_{ij} x_j \right| \leq \sum_{j \neq i} |a_{ij}| |x_j| \leq \sum_{j \neq i} |a_{ij}|. \end{aligned}$$

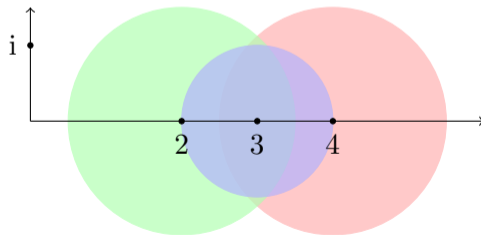


Diagonal-dominante Matrizen

Die Eigenwerte von

$$A = \begin{pmatrix} 3 & 1 & 0 \\ 0.5 & 4 & 1 \\ 1.5 & 0 & 2 \end{pmatrix}$$

liegen in folgenden Kreisen:

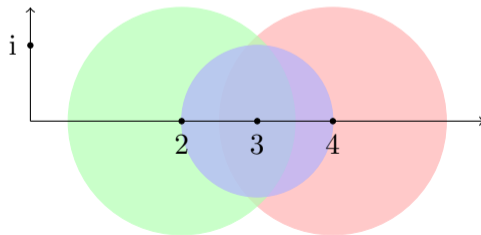


Diagonal-dominante Matrizen

Die Eigenwerte von

$$A = \begin{pmatrix} 3 & 1 & 0 \\ 0.5 & 4 & 1 \\ 1.5 & 0 & 2 \end{pmatrix}$$

liegen in folgenden Kreisen:

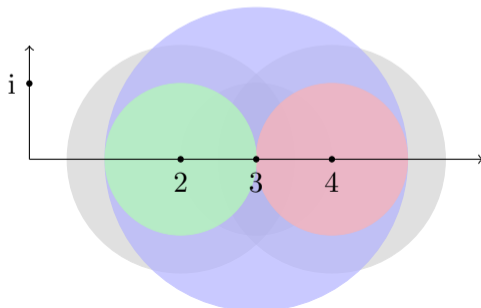


Insbesondere ist A invertierbar.

Diagonal-dominante Matrizen

$$A = \begin{pmatrix} 3 & 1 & 0 \\ 0.5 & 4 & 1 \\ 1.5 & 0 & 2 \end{pmatrix}$$

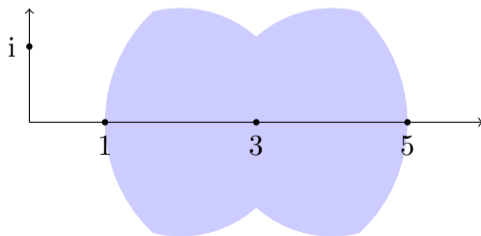
Da A und A^t die gleichen Eigenwerte besitzen, kann man auch die Spalten benutzen:



Diagonal-dominante Matrizen

$$A = \begin{pmatrix} 3 & 1 & 0 \\ 0.5 & 4 & 1 \\ 1.5 & 0 & 2 \end{pmatrix}$$

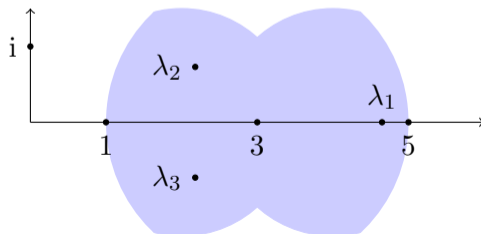
Da A und A^t die gleichen Eigenwerte besitzen, kann man auch die Spalten benutzen... und beide Mengen schneiden:



Diagonal-dominante Matrizen

$$A = \begin{pmatrix} 3 & 1 & 0 \\ 0.5 & 4 & 1 \\ 1.5 & 0 & 2 \end{pmatrix}$$

Da A und A^t die gleichen Eigenwerte besitzen, kann man auch die Spalten benutzen... und beide Mengen schneiden:



Positive Matrizen

Der **Spektralradius** von $A \in \mathbb{R}^{n \times n}$ ist definiert durch

$$\rho(A) := \max\{|\lambda| : \lambda \text{ Eigenwert von } A\}.$$

Positive Matrizen

Der **Spektralradius** von $A \in \mathbb{R}^{n \times n}$ ist definiert durch

$$\rho(A) := \max\{|\lambda| : \lambda \text{ Eigenwert von } A\}.$$

In vielen Anwendungen treten Matrizen mit lauter **positiven** Einträgen auf (z. B. Googles **PageRank-Algorithmus**).

Positive Matrizen

Der **Spektralradius** von $A \in \mathbb{R}^{n \times n}$ ist definiert durch

$$\rho(A) := \max\{|\lambda| : \lambda \text{ Eigenwert von } A\}.$$

In vielen Anwendungen treten Matrizen mit lauter **positiven** Einträgen auf (z. B. Googles **PageRank-Algorithmus**).

Satz (PERRON)

Für jede positive Matrix A ist $\rho(A)$ ein Eigenwert mit algebraischer Vielfachheit 1 und es existiert ein positiver Eigenvektor zu $\rho(A)$. Für jeden weiteren Eigenwert λ von A gilt $|\lambda| < \rho(A)$.

Potenz-Methode

Satz (MISES)

Seien $A \in \mathbb{R}^{n \times n}$ und $x_1 \in \mathbb{R}^n$ positiv. Dann konvergiert die Folge

$$x_{k+1} := \frac{Ax_k}{|Ax_k|} \quad (k = 1, 2, \dots)$$

gegen einen positiven Eigenvektor zu $\rho(A)$. Insbesondere ist

$$\rho(A) = \lim_{k \rightarrow \infty} |Ax_k|.$$

Potenz-Methode

Satz (MISES)

Seien $A \in \mathbb{R}^{n \times n}$ und $x_1 \in \mathbb{R}^n$ positiv. Dann konvergiert die Folge

$$x_{k+1} := \frac{Ax_k}{|Ax_k|} \quad (k = 1, 2, \dots)$$

gegen einen positiven Eigenvektor zu $\rho(A)$. Insbesondere ist

$$\rho(A) = \lim_{k \rightarrow \infty} |Ax_k|.$$

Das Verfahren konvergiert exponentiell (abhängig vom zweitgrößten Eigenwert).

Positiv definite Matrizen

- Zur Berechnung **aller** Eigenwerte nehmen wir zunächst an, dass A symmetrisch und positiv definit ist (alle Eigenwerte sind positiv).

Positiv definite Matrizen

- Zur Berechnung **aller** Eigenwerte nehmen wir zunächst an, dass A symmetrisch und positiv definit ist (alle Eigenwerte sind positiv).
- Nach dem Spektralsatz lässt sich A diagonalisieren. Dafür bräuchte man aber bereits die Eigenwerte.

Positiv definite Matrizen

- Zur Berechnung **aller** Eigenwerte nehmen wir zunächst an, dass A symmetrisch und positiv definit ist (alle Eigenwerte sind positiv).
- Nach dem Spektralsatz lässt sich A diagonalisieren. Dafür bräuchte man aber bereits die Eigenwerte.
- Idee: Konstruiere eine Folge von Matrizen

$$A_1 := A, \quad A_{k+1} := S_k^{-1} A_k S_k \quad (k \geq 1, S_k \in \text{GL}(n, \mathbb{R})),$$

die gegen eine Diagonalmatrix konvergiert.

Positiv definite Matrizen

- Zur Berechnung **aller** Eigenwerte nehmen wir zunächst an, dass A symmetrisch und positiv definit ist (alle Eigenwerte sind positiv).
- Nach dem Spektralsatz lässt sich A diagonalisieren. Dafür bräuchte man aber bereits die Eigenwerte.
- Idee: Konstruiere eine Folge von Matrizen

$$A_1 := A, \quad A_{k+1} := S_k^{-1} A_k S_k \quad (k \geq 1, S_k \in \text{GL}(n, \mathbb{R})),$$

die gegen eine Diagonalmatrix konvergiert.

- Da A und A_k die gleichen Eigenwerte haben, approximieren die Diagonaleinträge von A_k die Eigenwerte von A .

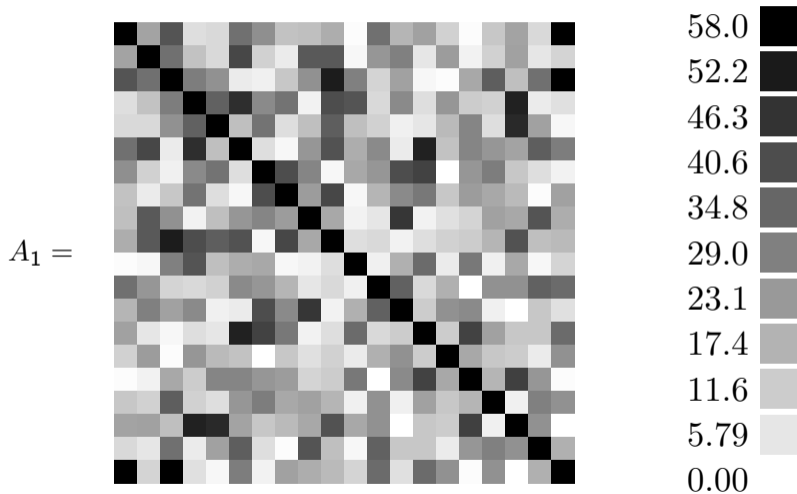
Positiv definite Matrizen

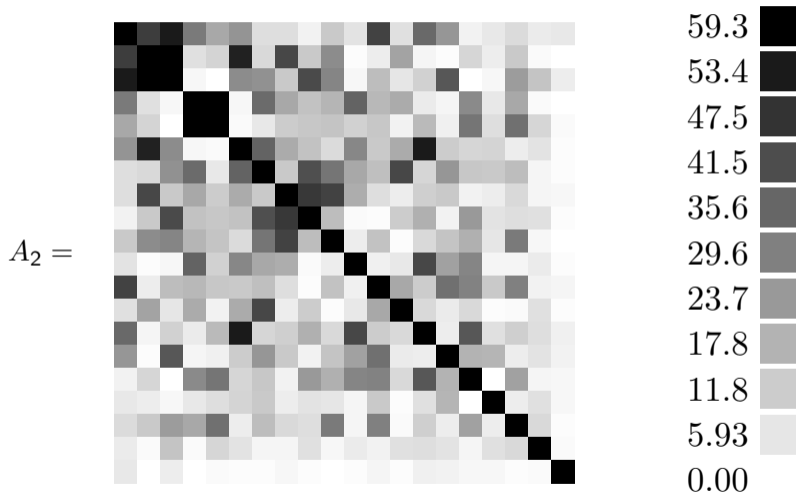
Satz (Cholesky-Verfahren)

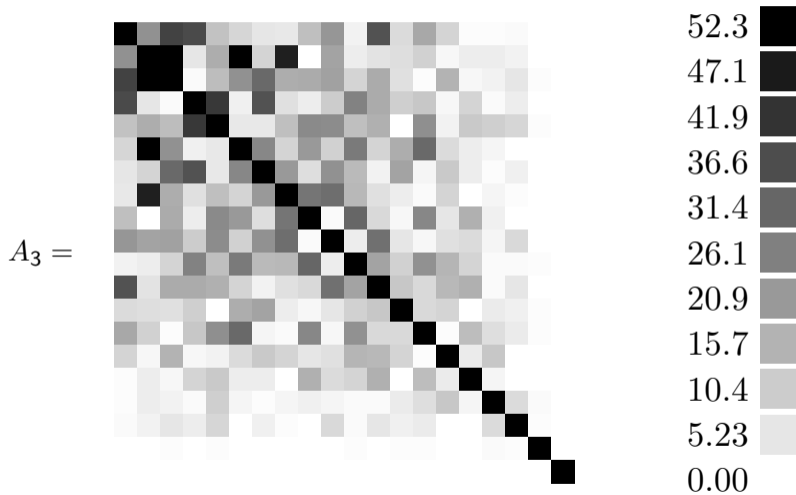
Sei $A_1 := A$ positiv definit. Für $k = 1, 2, \dots$ sei $A_k = R^t R$ die Cholesky-Zerlegung und

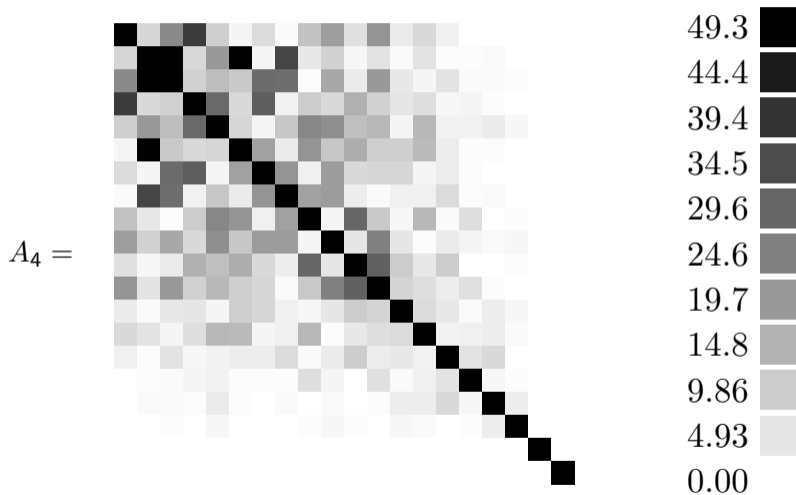
$$A_{k+1} := R^{-t} A_k R^t = R R^t.$$

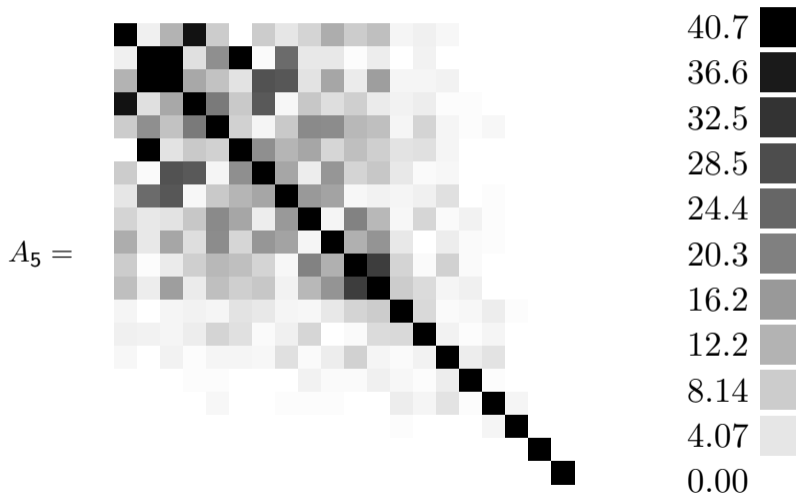
Dann konvergiert A_k gegen eine Diagonalmatrix.

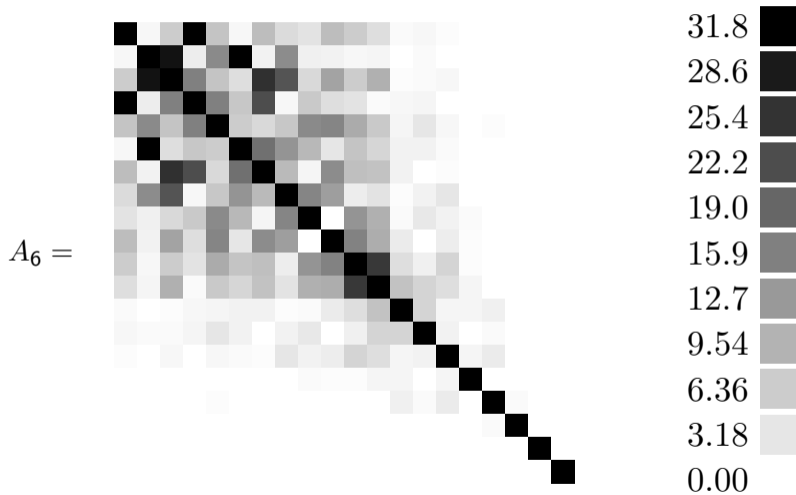
Zufallsmatrix der Größe 20×20 

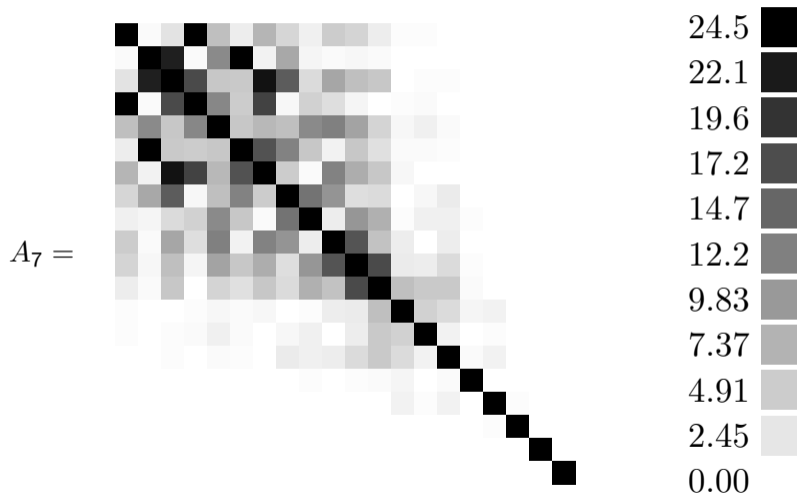
Zufallsmatrix der Größe 20×20 

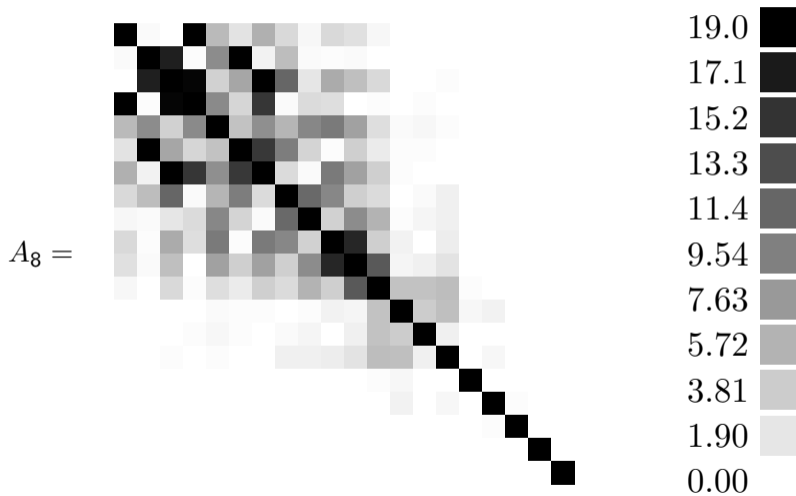
Zufallsmatrix der Größe 20×20 

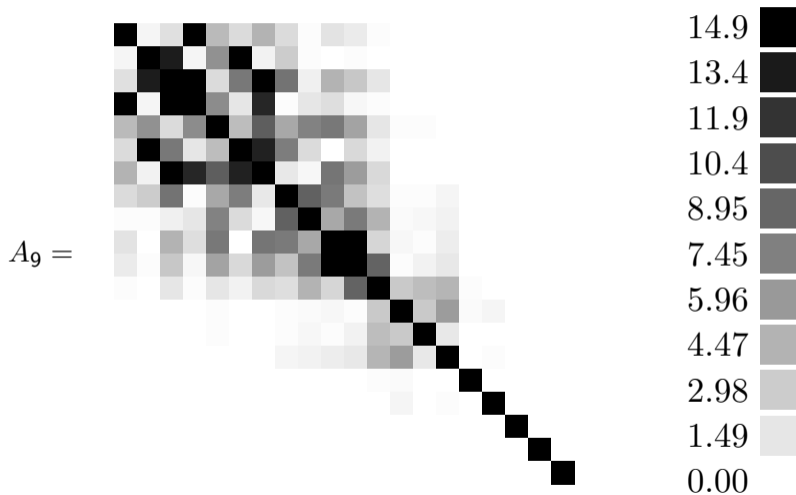
Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20 

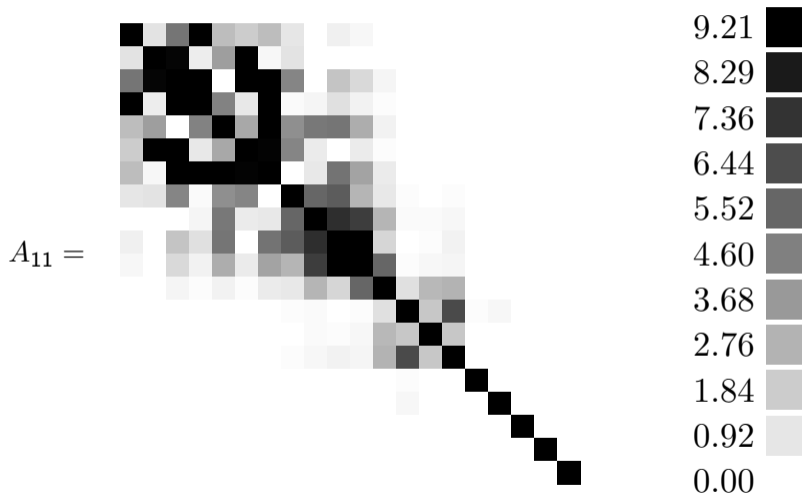
Zufallsmatrix der Größe 20×20 

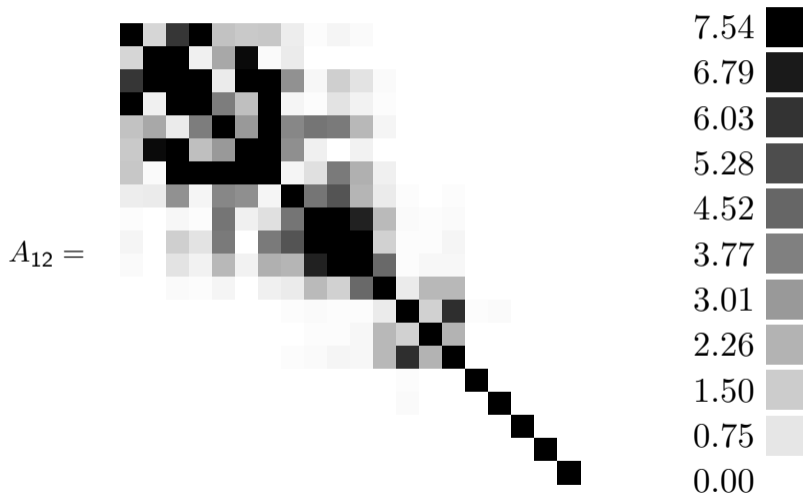
Zufallsmatrix der Größe 20×20 

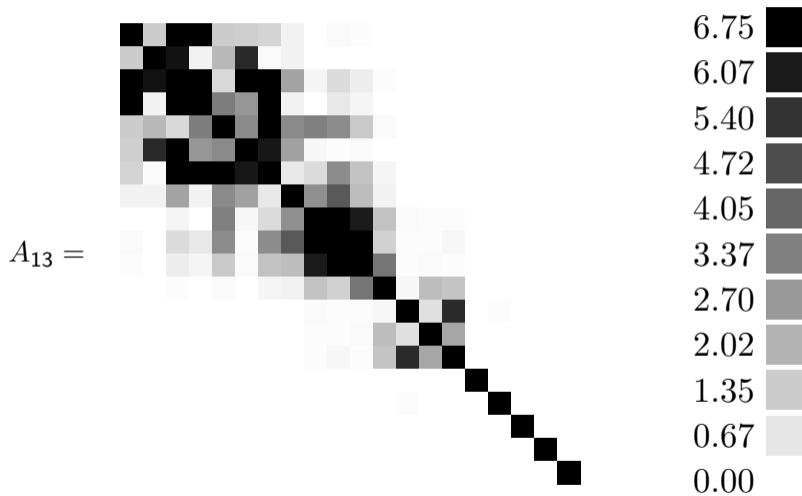
Zufallsmatrix der Größe 20×20 

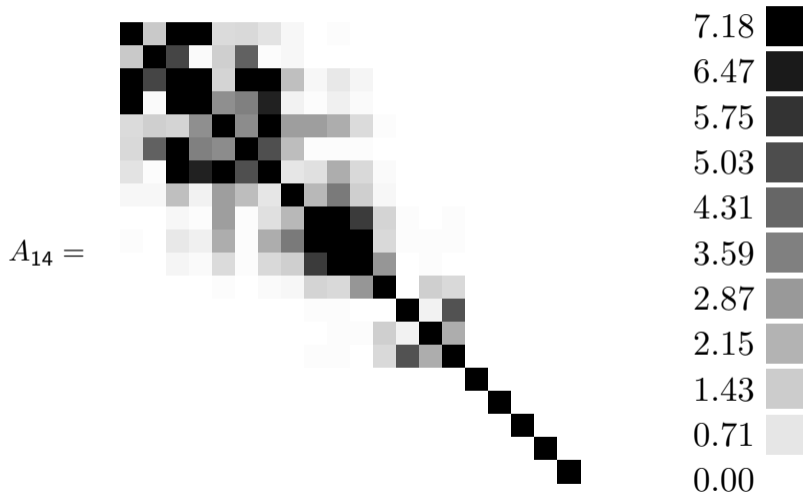
Zufallsmatrix der Größe 20×20 

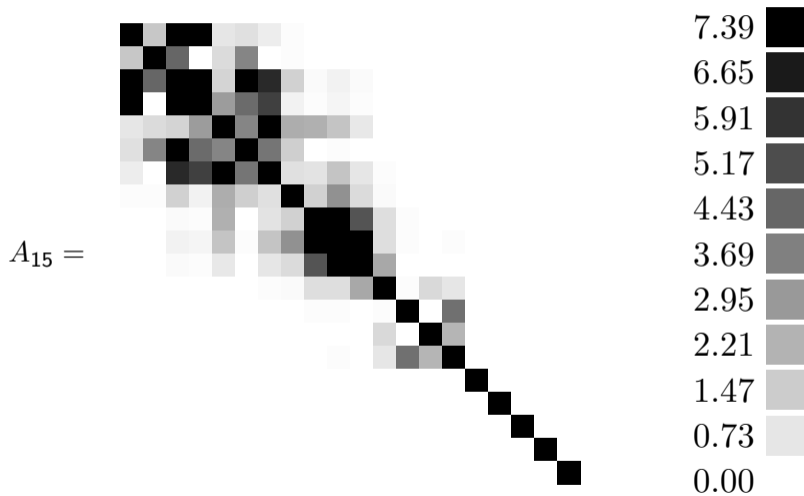
Zufallsmatrix der Größe 20×20 

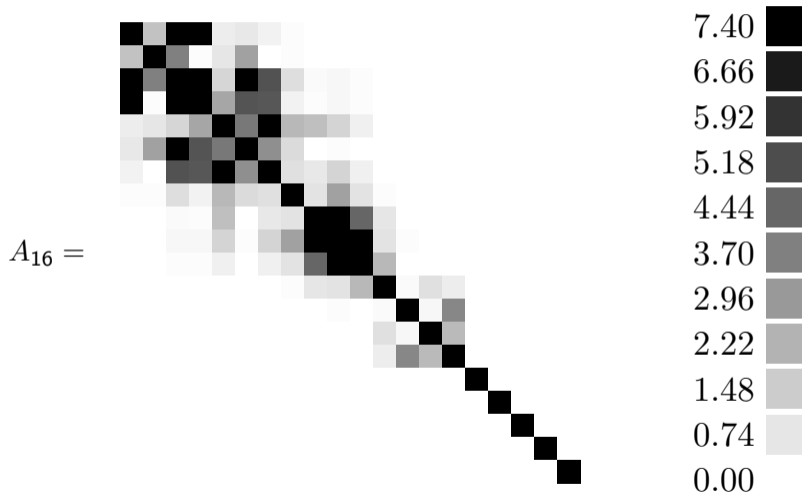
Zufallsmatrix der Größe 20×20 

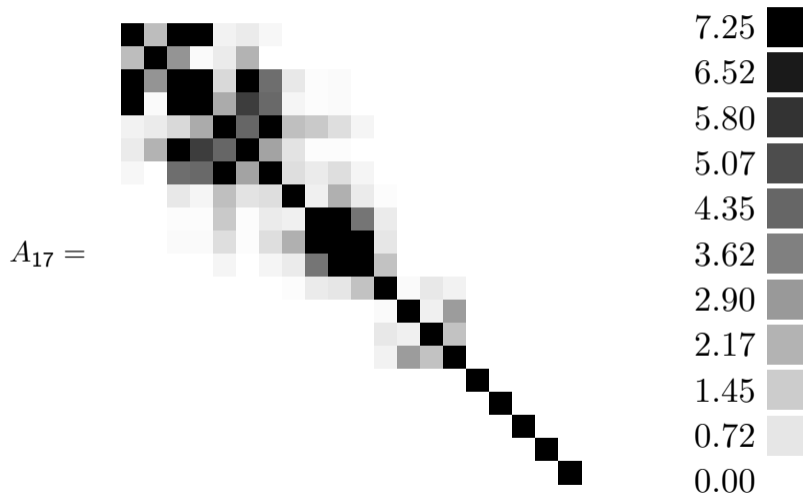
Zufallsmatrix der Größe 20×20 

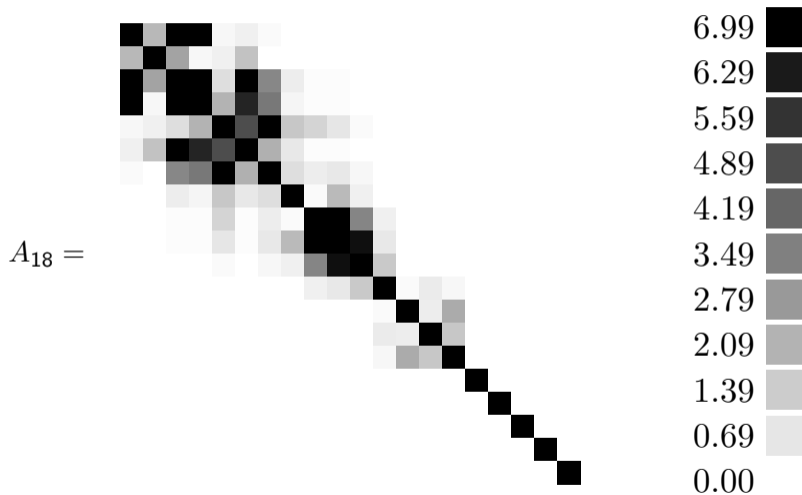
Zufallsmatrix der Größe 20×20 

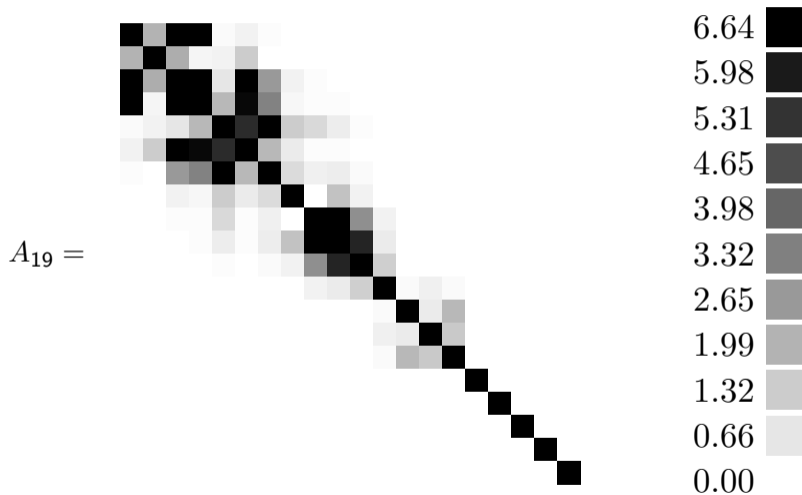
Zufallsmatrix der Größe 20×20 

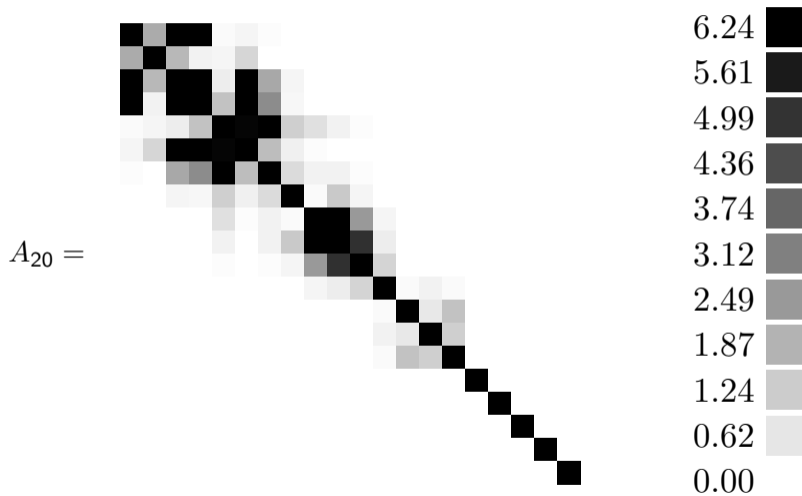
Zufallsmatrix der Größe 20×20 

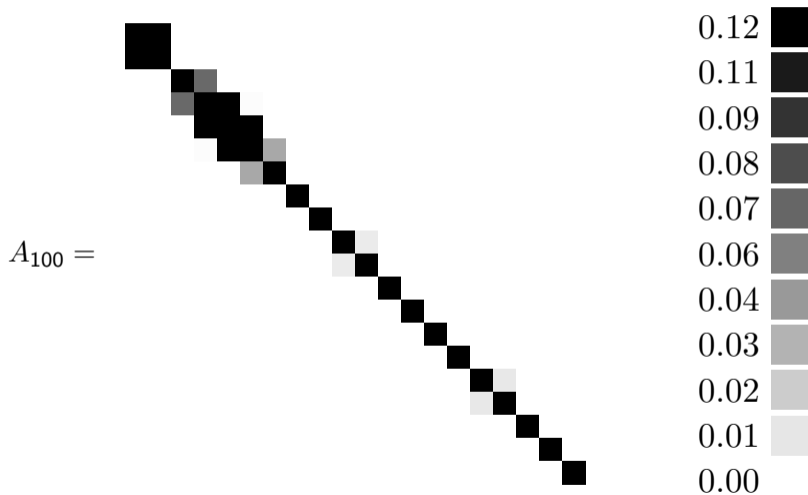
Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20 

Beliebige Matrizen

- Für beliebige Matrizen konvergiert A_k nicht unbedingt gegen eine Diagonalmatrix.

Beliebige Matrizen

- Für beliebige Matrizen konvergiert A_k nicht unbedingt gegen eine Diagonalmatrix.
- Wir streben stattdessen eine obere Dreiecksmatrix an (**Schursche Normalform** von A).

Beliebige Matrizen

- Für beliebige Matrizen konvergiert A_k nicht unbedingt gegen eine Diagonalmatrix.
- Wir streben stattdessen eine obere Dreiecksmatrix an (**Schursche Normalform** von A).
- Die Eigenwerte stehen nach wie vor auf der Hauptdiagonalen.

Beliebige Matrizen

- Für beliebige Matrizen konvergiert A_k nicht unbedingt gegen eine Diagonalmatrix.
- Wir streben stattdessen eine obere Dreiecksmatrix an (**Schursche Normalform** von A).
- Die Eigenwerte stehen nach wie vor auf der Hauptdiagonalen.
- Dafür ersetzt man die Cholesky-Zerlegung durch:

QR-Zerlegung

Satz (QR-Zerlegung)

Für $A \in \mathbb{R}^{n \times n}$ existiert $Q \in O(n, \mathbb{R})$ und eine obere Dreiecksmatrix R mit $A = QR$.

QR-Zerlegung

Satz (QR-Zerlegung)

Für $A \in \mathbb{R}^{n \times n}$ existiert $Q \in O(n, \mathbb{R})$ und eine obere Dreiecksmatrix R mit $A = QR$.

Beweis.

- Nach Gram-Schmidt existiert eine Orthonormalbasis b_1, \dots, b_n von \mathbb{R}^n , sodass die k -te Spalte von A die Form $\sum_{i=1}^k r_{ik} b_i$ hat für $k = 1, \dots, n$.

QR-Zerlegung

Satz (QR-Zerlegung)

Für $A \in \mathbb{R}^{n \times n}$ existiert $Q \in O(n, \mathbb{R})$ und eine obere Dreiecksmatrix R mit $A = QR$.

Beweis.

- Nach Gram-Schmidt existiert eine Orthonormalbasis b_1, \dots, b_n von \mathbb{R}^n , sodass die k -te Spalte von A die Form $\sum_{i=1}^k r_{ik} b_i$ hat für $k = 1, \dots, n$.
- Definiere $Q \in O(n, \mathbb{R})$ mit den Spalten b_1, \dots, b_n und $R := (r_{ij})$. □

Eigenwerte mittels QR-Zerlegung

Francis-Algorithmus

Setze $A_1 := A$. Für $k = 1, 2, \dots$ berechne QR-Zerlegung $A_k = QR$ und definiere

$$A_{k+1} := Q^t A_k Q = RQ.$$

Eigenwerte mittels QR-Zerlegung

Francis-Algorithmus

Setze $A_1 := A$. Für $k = 1, 2, \dots$ berechne QR-Zerlegung $A_k = QR$ und definiere

$$A_{k+1} := Q^t A_k Q = RQ.$$

Satz

Haben die Eigenwerte von A paarweise verschiedene Beträge, so konvergiert A_k gegen eine obere Dreiecksmatrix.

Eigenwerte mittels QR-Zerlegung

Francis-Algorithmus

Setze $A_1 := A$. Für $k = 1, 2, \dots$ berechne QR-Zerlegung $A_k = QR$ und definiere

$$A_{k+1} := Q^t A_k Q = RQ.$$

Satz

Haben die Eigenwerte von A paarweise verschiedene Beträge, so konvergiert A_k gegen eine obere Dreiecksmatrix.

- Aus der Voraussetzung folgt, dass A nur reelle Eigenwerte hat, denn $|\lambda| = |\bar{\lambda}|$ für $\lambda \in \mathbb{C}$.

Eigenwerte mittels QR-Zerlegung

Francis-Algorithmus

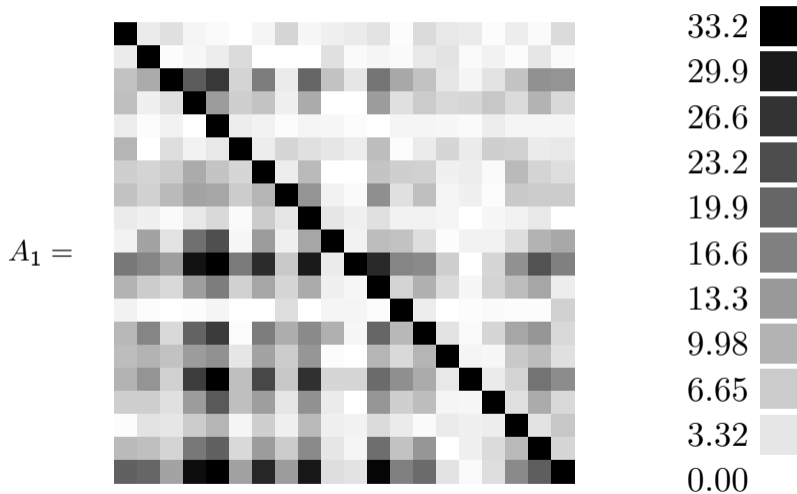
Setze $A_1 := A$. Für $k = 1, 2, \dots$ berechne QR-Zerlegung $A_k = QR$ und definiere

$$A_{k+1} := Q^t A_k Q = RQ.$$

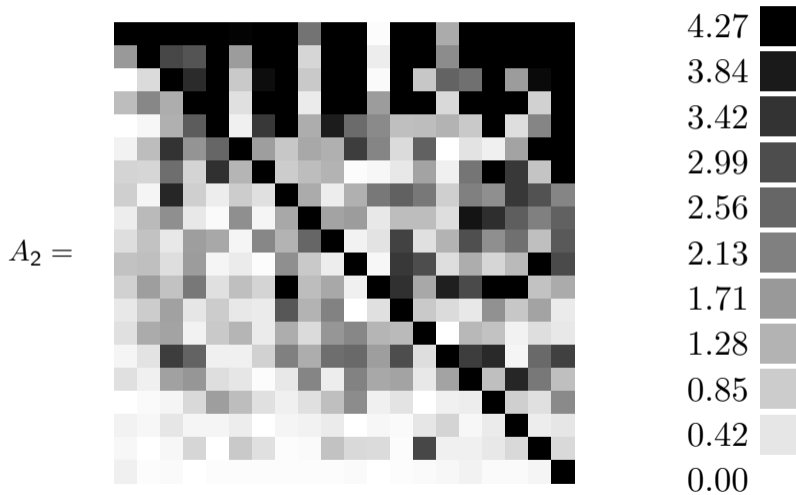
Satz

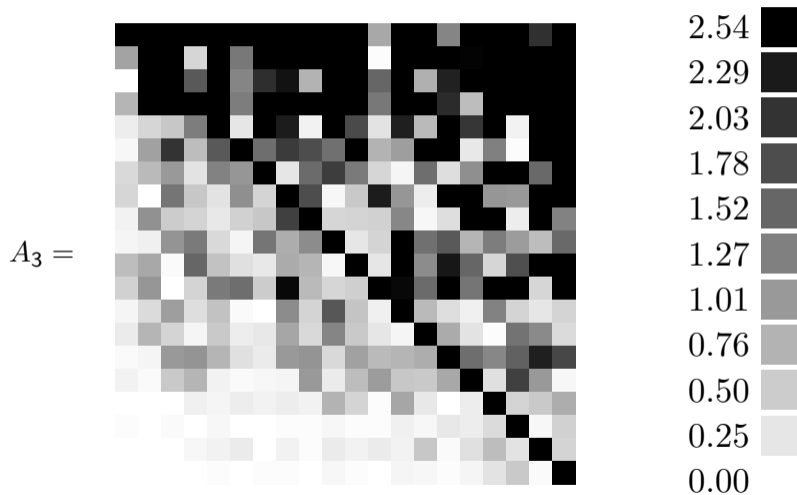
Haben die Eigenwerte von A paarweise verschiedene Beträge, so konvergiert A_k gegen eine obere Dreiecksmatrix.

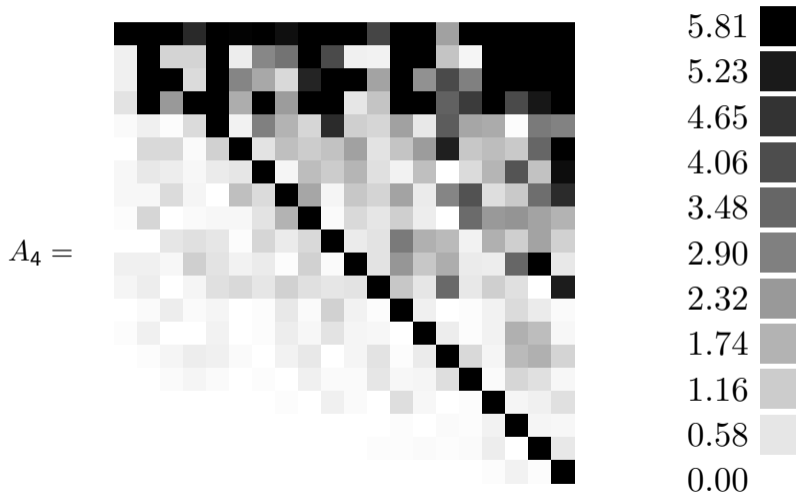
- Aus der Voraussetzung folgt, dass A nur reelle Eigenwerte hat, denn $|\lambda| = |\bar{\lambda}|$ für $\lambda \in \mathbb{C}$.
- Besitzt A nicht-reelle Eigenwerte, so kann man erreichen, dass A_k gegen eine Dreiecksmatrix aus 2×2 -Blöcken konvergiert (je ein Block für $(\lambda, \bar{\lambda})$).

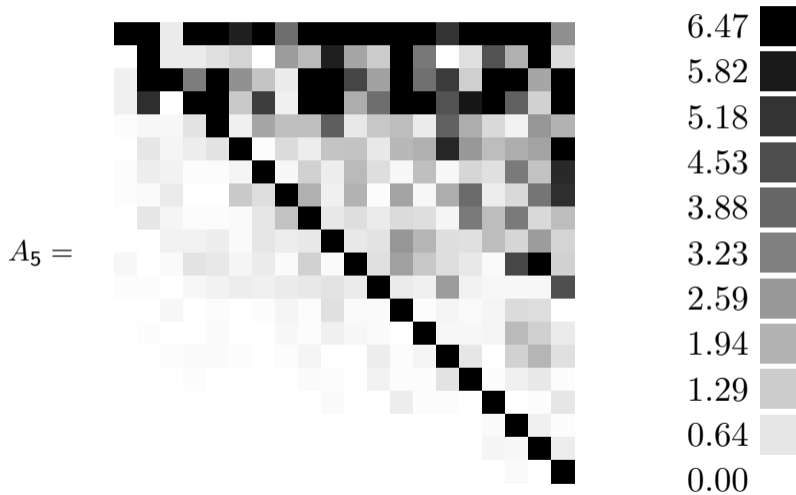
Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20

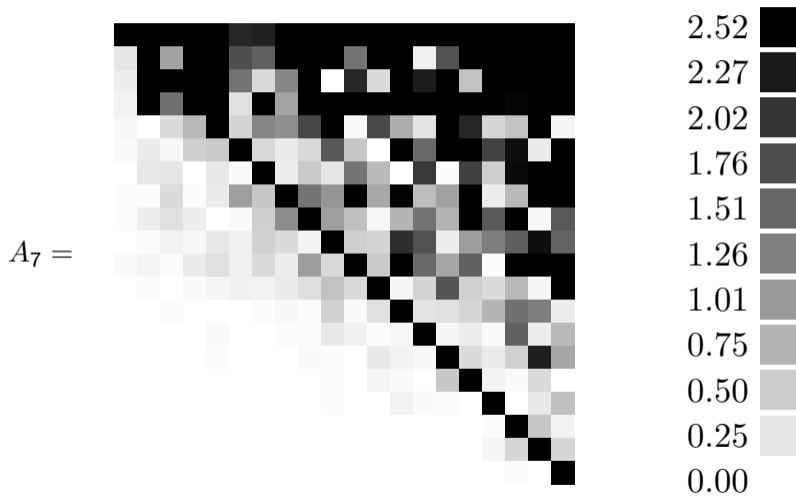


Zufallsmatrix der Größe 20×20 

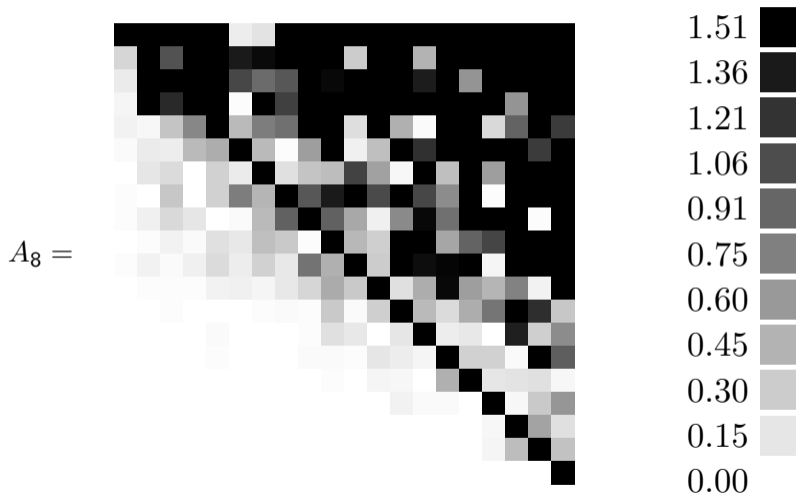
Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20 

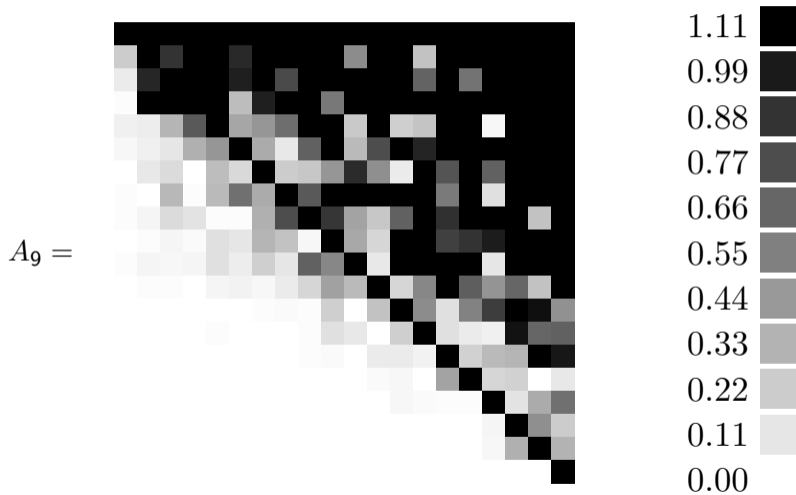
Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20 

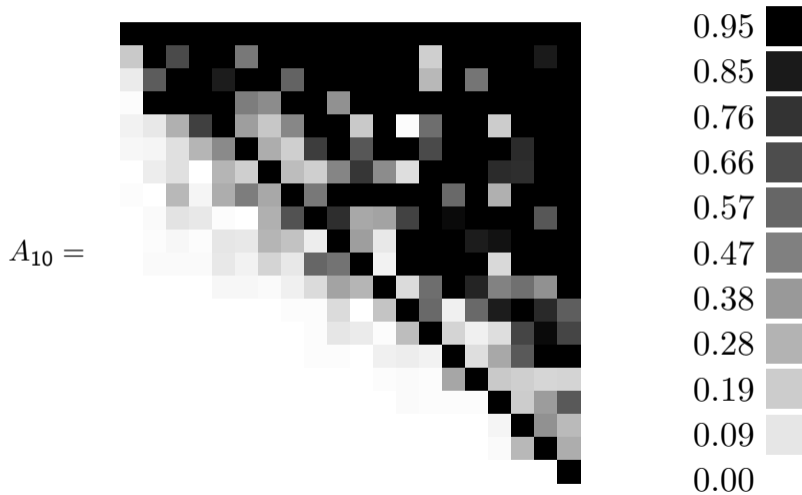
Zufallsmatrix der Größe 20×20

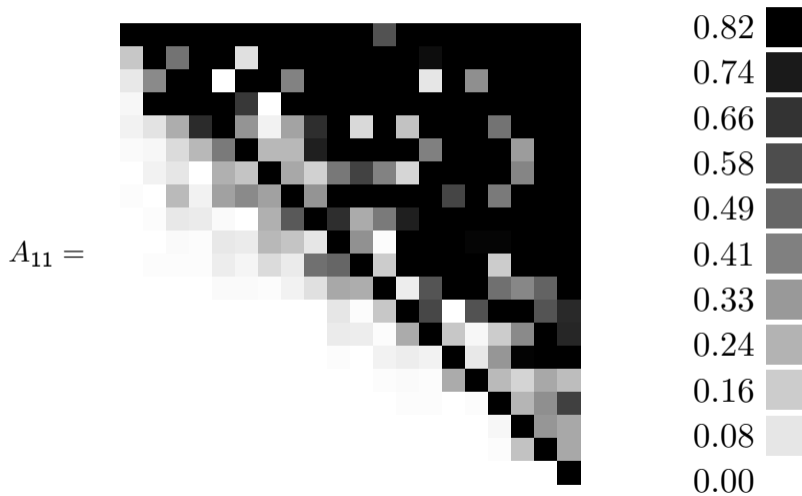


Zufallsmatrix der Größe 20×20

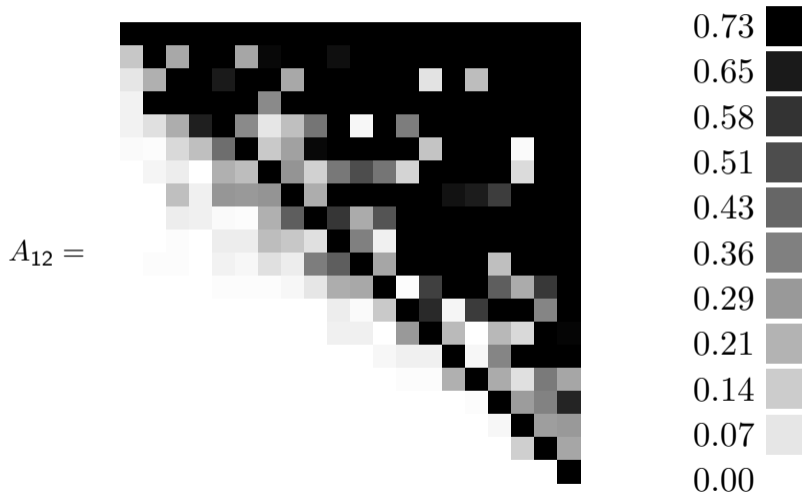


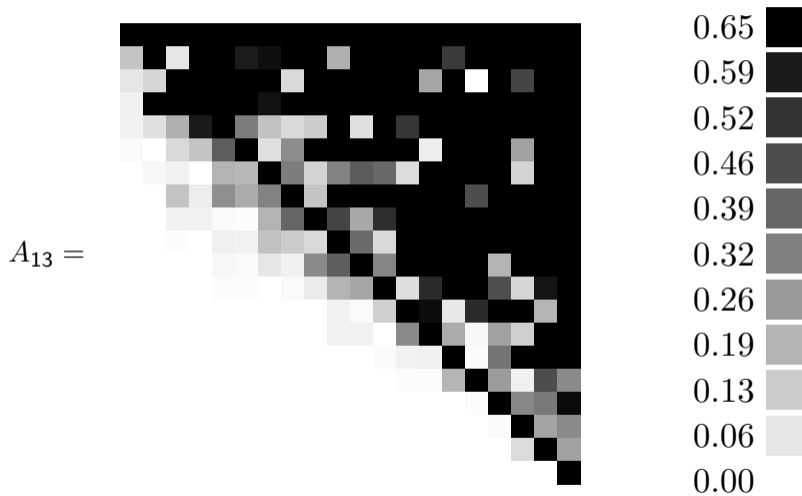
Zufallsmatrix der Größe 20×20

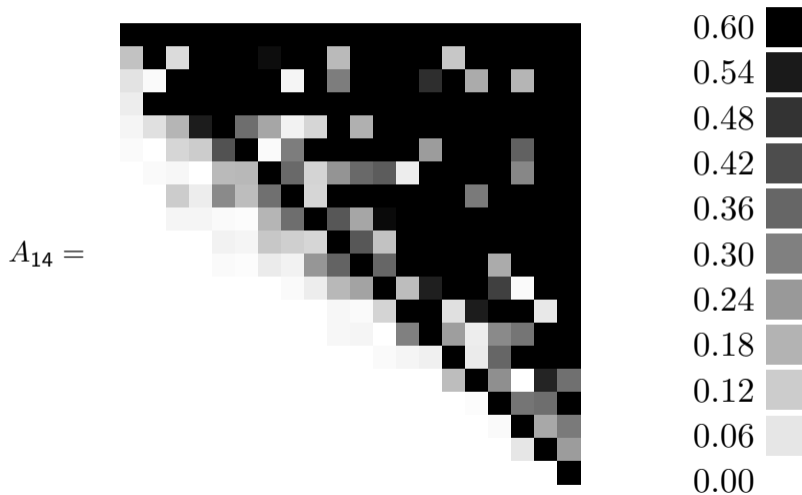


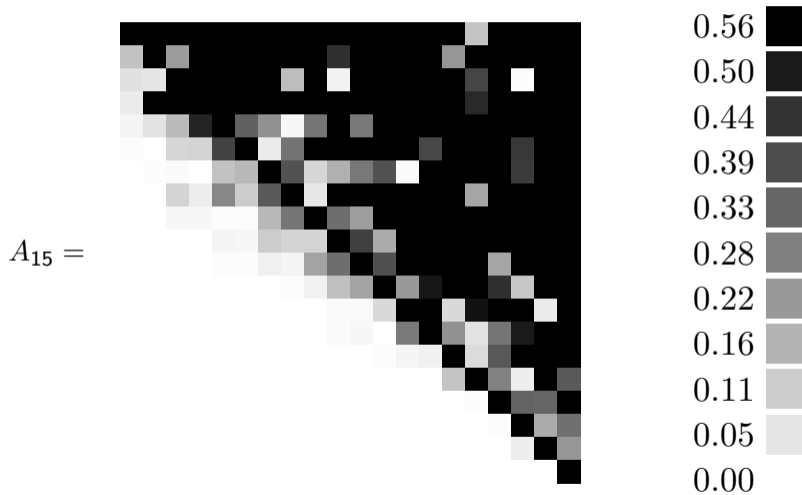
Zufallsmatrix der Größe 20×20 

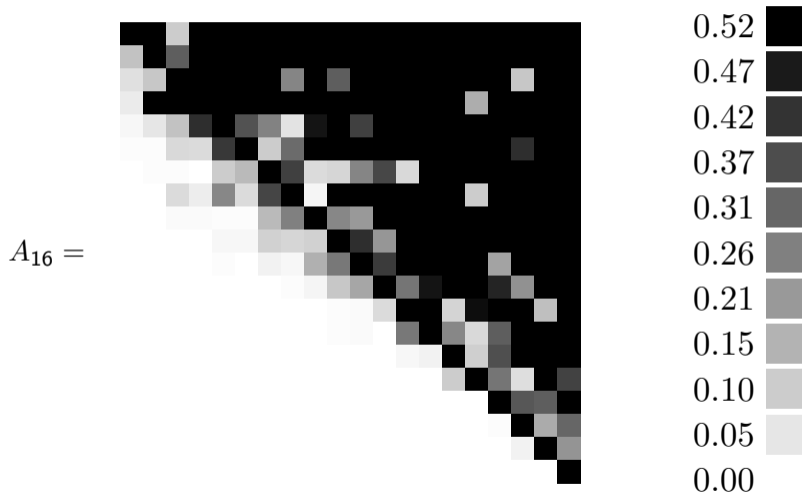
Zufallsmatrix der Größe 20×20

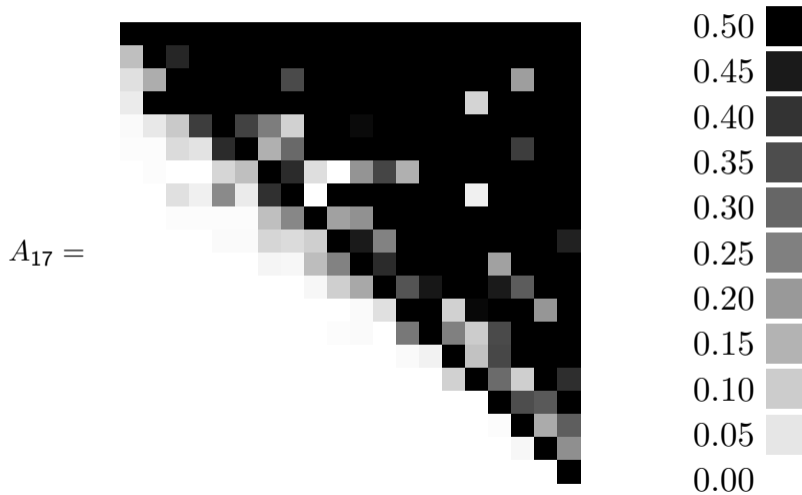


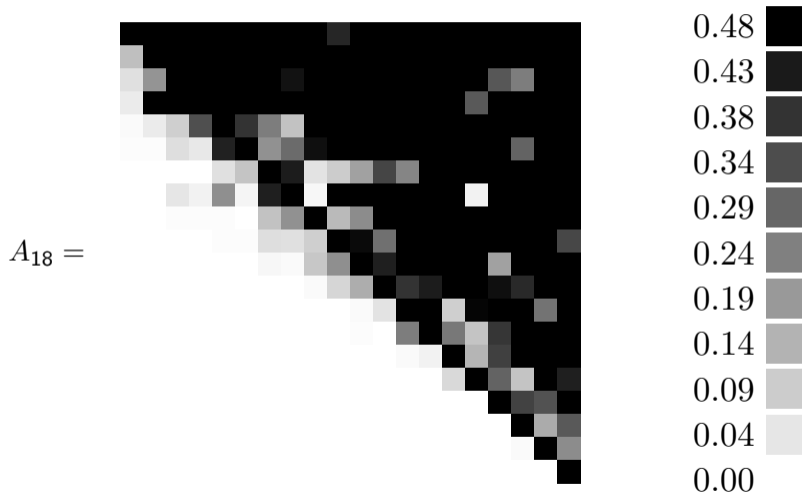
Zufallsmatrix der Größe 20×20 

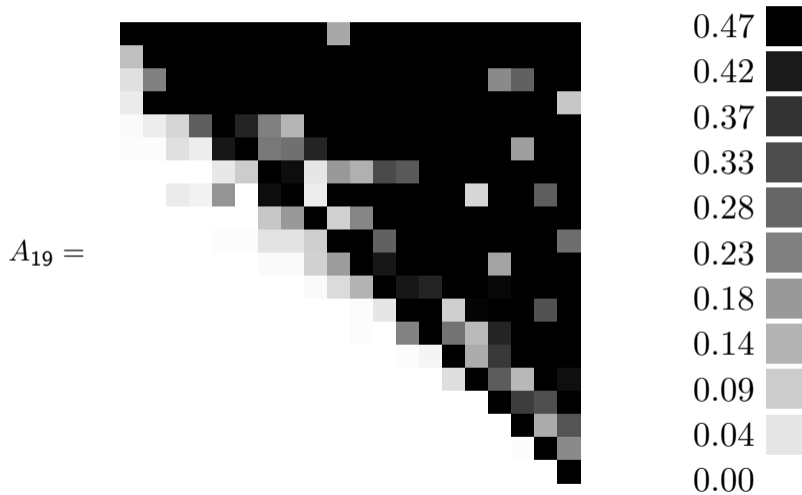
Zufallsmatrix der Größe 20×20 

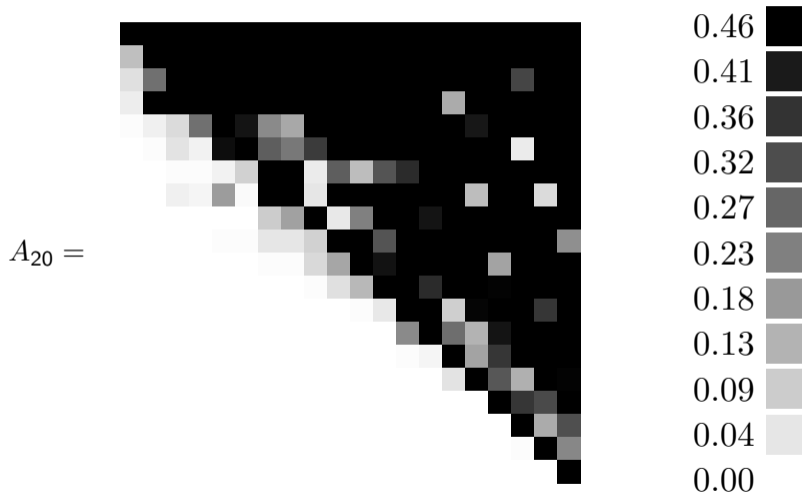
Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20 

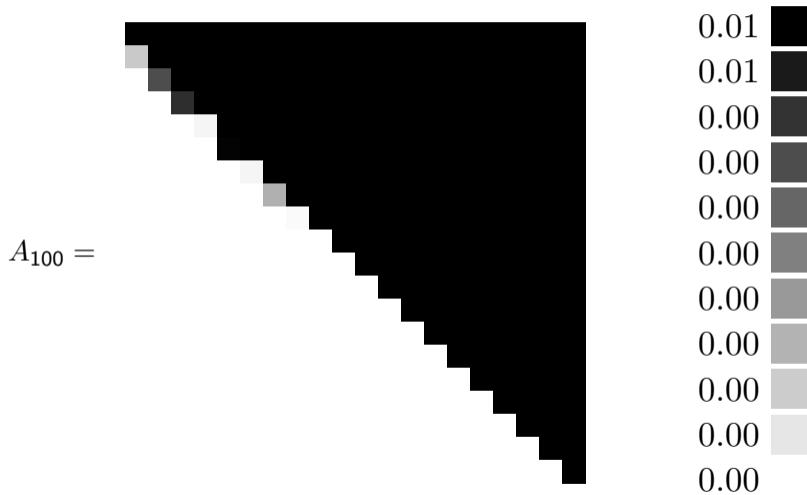
Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20 

Zufallsmatrix der Größe 20×20



Gram-Schmidt modifiziert

In der Gram-Schmidt-Formel

$$b'_k := v_k - \sum_{i=1}^{k-1} [v_k, b_i] b_i, \quad b_k := \frac{b'_k}{|b'_k|}$$

akkumulieren sich Rundungsfehler.

Gram-Schmidt modifiziert

In der Gram-Schmidt-Formel

$$b'_k := v_k - \sum_{i=1}^{k-1} [v_k, b_i] b_i, \quad b_k := \frac{b'_k}{|b'_k|}$$

akkumulieren sich Rundungsfehler.

Numerisch stabiler ist die iterierte Berechnung:

$$\begin{aligned} b_{k,1} &:= v_k - [v_k, b_1] b_1, \\ b_{k,2} &:= b_{k,1} - [b_{k,1}, b_2] b_2, \\ &\vdots \\ b_k &:= \frac{b_{k,k-1}}{|b_{k,k-1}|}. \end{aligned}$$

Alternative zu Gram-Schmidt

- Die QR-Zerlegung $A = QR$ lässt sich noch stabiler als Komposition von Spiegelungen realisieren.

Alternative zu Gram-Schmidt

- Die QR-Zerlegung $A = QR$ lässt sich noch stabiler als Komposition von Spiegelungen realisieren.
- Sei a_1 die erste Spalte von A , $e_1 = (1, 0, \dots, 0)^t$ und $b := a_1 - |a_1|e_1$.

Alternative zu Gram-Schmidt

- Die QR-Zerlegung $A = QR$ lässt sich noch stabiler als Komposition von Spiegelungen realisieren.
- Sei a_1 die erste Spalte von A , $e_1 = (1, 0, \dots, 0)^t$ und $b := a_1 - |a_1|e_1$.
- Die Spiegelung $Q_1 \in O(n, \mathbb{R})$ an der Hyperebene $\langle b \rangle^\perp$ lässt sich bequem als **Householder-Transformation** berechnen:

$$Q_1 = 1_n - \frac{2}{|b|^2}bb^t.$$

Alternative zu Gram-Schmidt

- Die QR-Zerlegung $A = QR$ lässt sich noch stabiler als Komposition von Spiegelungen realisieren.
- Sei a_1 die erste Spalte von A , $e_1 = (1, 0, \dots, 0)^t$ und $b := a_1 - |a_1|e_1$.
- Die Spiegelung $Q_1 \in O(n, \mathbb{R})$ an der Hyperebene $\langle b \rangle^\perp$ lässt sich bequem als **Householder-Transformation** berechnen:

$$Q_1 = 1_n - \frac{2}{|b|^2}bb^t.$$

- Wegen $a_1 + |a_1|e_1 \in \langle b \rangle^\perp$ gilt

$$Q_1 a_1 = \frac{1}{2}(Q_1 b + Q_1(a_1 + |a_1|e_1)) = \frac{1}{2}(-b + a_1 + |a_1|e_1) = |a_1|e_1.$$

Alternative zu Gram-Schmidt

- Es folgt

$$Q_1 A = \begin{pmatrix} |a_1| & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{pmatrix}.$$

Alternative zu Gram-Schmidt

- Es folgt

$$Q_1 A = \begin{pmatrix} |a_1| & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{pmatrix}.$$

- Man kann den Prozess mit den Spalten $2, \dots, n$ wiederholen und erhält eine obere Dreiecksmatrix $R := Q_n \dots Q_1 A$.

Alternative zu Gram-Schmidt

- Es folgt

$$Q_1 A = \begin{pmatrix} |a_1| & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{pmatrix}.$$

- Man kann den Prozess mit den Spalten $2, \dots, n$ wiederholen und erhält eine obere Dreiecksmatrix $R := Q_n \dots Q_1 A$.
- Mit $Q := Q_1 \dots Q_n \in O(n, \mathbb{R})$ gilt $A = QR$ (beachte: $Q_i^{-1} = Q_i$).

Alternative zu Gram-Schmidt

- Es folgt

$$Q_1 A = \begin{pmatrix} |a_1| & * & \cdots & * \\ 0 & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{pmatrix}.$$

- Man kann den Prozess mit den Spalten $2, \dots, n$ wiederholen und erhält eine obere Dreiecksmatrix $R := Q_n \dots Q_1 A$.
- Mit $Q := Q_1 \dots Q_n \in O(n, \mathbb{R})$ gilt $A = QR$ (beachte: $Q_i^{-1} = Q_i$).
- Anstelle von Spiegelungen kann man **Givens-Rotationen** benutzen:

$$\begin{pmatrix} 1_r & & & & \\ & \cos \varphi & & -\sin \varphi & \\ & \sin \varphi & & \cos \varphi & \\ & & 1_s & & \\ & & & & 1_t \end{pmatrix} \in O(n, \mathbb{R}).$$

Ein Vergleich

Wir berechnen die QR-Zerlegung einer zufälligen 20×20 -Matrix. **Gram-Schmidt:**



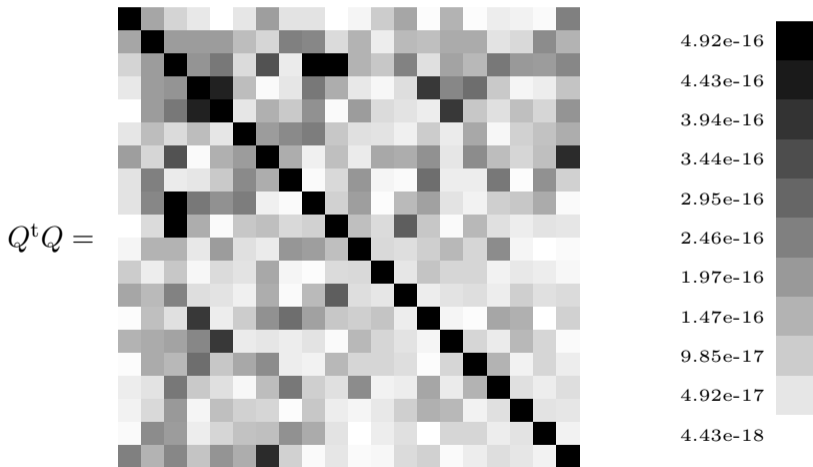
Ein Vergleich

Wir berechnen die QR-Zerlegung einer zufälligen 20×20 -Matrix. **Gram-Schmidt modifiziert:**



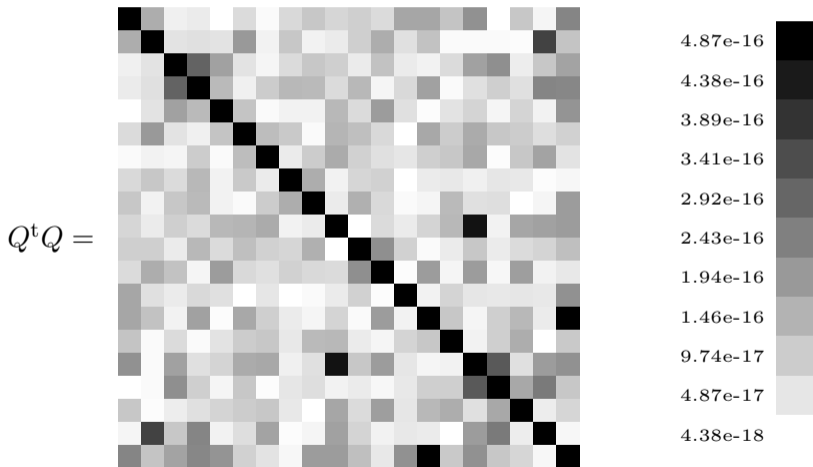
Ein Vergleich

Wir berechnen die QR-Zerlegung einer zufälligen 20×20 -Matrix. **Householder-Transformation:**



Ein Vergleich

Wir berechnen die QR-Zerlegung einer zufälligen 20×20 -Matrix. **Givens-Rotation:**





Software

Bibliotheken: LAPACK, BLAS

Programmiersprache:  (open source, beste Performance)

Software:  Matlab (LUH-Lizenz)

 (open source)

 Octave (open source)

Ende

Viel Erfolg in Ihrem weiteren Studium!