Dokumentation

# Inhaltsverzeichnis

Inhalt

[Inhaltsverzeichnis 1](#_Toc137551221)

[Mathematische Skripte 2](#_Toc137551222)

[Vorbemerkungen: 2](#_Toc137551223)

[Multiplikation 2](#_Toc137551224)

[ 3](#_Toc137551225)

[Gaußsches Eliminationsverfahren 3](#_Toc137551226)

[ 3](#_Toc137551227)

[Inverse Matrix 3](#_Toc137551228)

[ 4](#_Toc137551229)

[Determinante 4](#_Toc137551230)

[ 4](#_Toc137551231)

[Eigenwerte 4](#_Toc137551232)

[ 4](#_Toc137551233)

[ 5](#_Toc137551234)

[Hamming Code (7,4) 5](#_Toc137551235)

[Verwendungsbeispiel: 5](#_Toc137551236)

[Funktionen: 6](#_Toc137551237)

[ 6](#_Toc137551238)

[ 6](#_Toc137551239)

[Bildverarbeitung: 6](#_Toc137551240)

[Verwendung: 6](#_Toc137551241)

[Funktionen: 7](#_Toc137551242)

[ 7](#_Toc137551243)

[ 7](#_Toc137551244)

[ 7](#_Toc137551245)

[ 8](#_Toc137551246)

[ 8](#_Toc137551247)

[ 8](#_Toc137551248)

[Hill Chiffre: 9](#_Toc137551249)

[Verwendung: 9](#_Toc137551250)

[Funktionen: 9](#_Toc137551251)

[ 9](#_Toc137551252)

[ 9](#_Toc137551253)

[ 9](#_Toc137551254)

[ 10](#_Toc137551255)

# Mathematische Skripte

## Vorbemerkungen:

Wir verwenden die sympy Bibliothek. Diese ist als freie Software (BSD-Lizenz) verfügbar und kann mit einem from sympy import \* einfach in Python Skripte eingebunden werden.

Eine 2x3 sympy Matrix kann mit dem Befehl Matrix([[1,2,3],[4,5,6]]) erzeugt werden.

Für weiter Informationen über sympy verweißen wir auf https://www.sympy.org/en/index.html .

## Multiplikation

Skript: multiplikation.py

### 

#### Übergabeparameter:

* + sympy Matrizen A und B

#### Rückgabewert:

* + sympy Matrix



#### Funktion:

* + Es wird die Matrizenmultiplikation von A und B berechnet

## Gaußsches Eliminationsverfahren

Skript: gauss.py

### 

#### Übergabeparameter:

* + sympy Matrix A

#### Rückgabewert:

* + sympy Matrix in Zeilenstufenform und Liste mit den Pivotelementen (Spaltezahlen)



#### Funktion:

* + Es wird eine (normierte) Zeilenstufenform von A berechnet. Die "Stufen" werden durch die Pivotelemente gekennzeichnet.

Hilfsfunktionen für gauss:

* + common, multRow, multRowAdd, swapRows (Für eine Beschreibung siehe Code Kommentare)

## Inverse Matrix

Skript: invers.py

### 

#### Übergabeparameter:

* + sympy Matrix A

#### Rückgabewert:

* + sympy Matrix oder None

#### Funktion:

* + Es wird die zu A inverse Matrix berechnet, falls sie existiert

## Determinante

Skript: laplace.py

### 

#### Übergabeparameter:

* + sympy Matrix A

#### Rückgabewert:

* + sympy Zahl oder Ausdruck

#### Funktion:

* + Berechnet die Determinante der Matrix A

## Eigenwerte

Skript: eigenwerte.py

### 

#### Übergabeparameter:

* + sympy Matrix A

#### Rückgabewert:

* + Liste von symbolischen Eigenwerten

#### Funktion:

* + Berechnet symbolisch die Eigenwerte der Matrix A

### 

#### Übergabeparameter:

* + sympy Matrix A

#### Rückgabewert:

* + Liste von nummerischen Eigenwerten

#### Funktion:

* + Berechnet die nummerischen Eigenwerte der Matrix A

# Hamming Code (7,4)

Skript: hamming.py

## Verwendungsbeispiel:

Will ein Sender die vier Bits "1011" and einen Empfänger übertragen so erzeugt er mit

In: generateHamming([1,0,1,1])

Out: [0, 1, 1, 0, 0, 1, 1]

7 zu übertragenden Bits. Das Resultat, nämlich "0110011", sendet er dann an den Empfänger.

Angenommen bei der Übertragung tritt ein Fehler im letzten Bit auf, sodass beim Empfänger "0110010" anstatt "0110011" ankommt.

Diesen Fehler kann der Empfänger nun wie folgt korrigiert werden:

In: decodeHamming([0, 1, 1, 0, 0, 1, 0])

Out: ([1, 0, 1, 1], 'Bit 7 korrigiert')

Also kann der Empfänger die ursprüngliche Nachricht "1011" trotz eines Übertragungsfehlers rekonstruieren.

## Funktionen:

### 

#### Übergabeparameter:

* + bitList (4 Bit Liste)
    - Form (Beispiel): 

#### Rückgabewert:

* + 7 Bits Liste. Dies sind die Bits, die bei der Kommunikation an den Empfänger übertragen werden. 

#### Funktion:

* + Die eingegebenen 4 Bits werden redundant zu einer Liste von 7 Bits codiert.

### 

#### Übergabeparameter:

* + bitList (7 Bit Liste)
    - Form (Beispiel): 

#### Rückgabewert:

* + Tupel bestehend aus einer 4 Bit Liste und einem String.

#### Funktion

* + Aus der gesendeten Nachricht (7 Bits) wird die ursprüngliche Nachricht rekonstruiert, wobei bis zu ein Übertragungsfehler korrigiert werden kann.

# Bildverarbeitung:

## Verwendung:

* Starten des Python Scripts (Bildverarbeitung.py)
  + start() Methode wird aufgerufen
  + User Input Felder fordern die benötigten Eingaben
  + Ergebnis wird in einem extra Fenster per Plot dargestellt
    - Fenster schließen -> Script wird beendet
  + Für neue Eingabe: Script neustarten

## Funktionen:

### 

#### Übergabeparameter:

* + keine

#### Rückgabewert:

* + keiner

#### Funktion:

* + startet das Programm, fordert die ersten benötigten User Inputs und ruft alle weiteren Methoden auf

### 

#### Übergabeparameter:

* + points (Liste von Punkten)
    - Form:



* + alphaDeg (Rotationswinkel in Grad)

#### Rückgabewert:

* + Liste von Punkten welche rotiert wurde

#### Funktion:

* + Übergebene Punkte Liste wird am übergebenen Winkel rotiert und zurückgegeben

### 

#### Übergabeparameter:

* + points (Liste von Punkten)
    - Form:



* + lambdaValue (Skalierungswert)

#### Rückgabewert

* + Liste von Punkten welche skaliert wurde

#### Funktion:

* + Übergebene Liste von Punkten wird mit dem übergebenen Wert skaliert und zurückgegeben

### 

#### Übergabeparameter:

* + points (Liste von Punkten)
    - Form:



* + axis (Spiegelungsachse oder Wert des Winkels in Grad einer Ursprungsgeraden zur x-Achse)

#### Rückgabewert

* + Liste von Punkten welche gespiegelt wurde

#### Funktion:

* + Übergebene Liste von Punkten wird an der übergebenen Achse oder der Ursprungsgerade mit übergebenem Winkel zur x-Achse gespiegelt und zurückgegeben

### 

#### Übergabeparameter:

* + pointsList (originale Liste von Punkten)
    - Form:



* + newPointsList (verarbeitete Liste von Punkten)
    - Form:

#### Rückgabewert

* + keiner, öffnet ein Fenster mit dem Plot

#### Funktion:

* + Zwei übergebene Listen mit Punkten werden in einem Plot geplottet, welcher in einem neuen Fenster geöffnet wird

### 

#### Übergabeparameter:

* + pointsList (originale Liste von Punkten)
    - Form:



#### Rückgabewert

* + veränderte Liste von Punkten

#### Funktion:

* + Fordert und validiert den User Input, um die übergebene Liste von Punkten zu bearbeiten und zurückzugeben

# Hill Chiffre:

## Verwendung:

* Starten des Python Scripts (HillChiffre.py)
  + start() Methode wird aufgerufen
  + User Input Felder fordern die benötigten Eingaben
  + Ergebnis wird in der Konsole geprintet (Matrix, Inverse, Verschlüsselter und Entschlüsselter Text)
  + Für neue Eingabe: Script neustarten

## Funktionen:

### 

#### Übergabeparameter:

* + keine

#### Rückgabewert

* + keiner

#### Funktion:

* + Startet das Script und fordert über Userinput die benötigten Eingaben, ruft die weiteren Methoden auf und gibt zum Schluss das Ergebnis in der Konsole aus

### 

#### Übergabeparameter:

* + text (zu verschlüsselnder Text)
  + matrix (Matrix zum ver-/entschlüsseln)

#### Rückgabewert

* + ver-/entschlüsselter Text

#### Funktion:

* + Ver-/entschlüsselt den übergebenen Text mit der übergebenen Matrix nach dem Prinzip der Hill Chiffre mit dem druckbaren ASCII Zeichensatz und gibt den ver-/entschlüsselten Text zurück

### 

#### Übergabeparameter:

* + dim (Dimension der zu generierenden quadratischen Matrix)

#### Rückgabewert

* + Quadratische Matrix und deren Inverse

#### Funktion:

* + Generiert eine quadratische Matrix und die zugehörige Inverse mit der übergebenen Dimension, welche zur Verwendung mit der Hill Chiffre geeignet ist

### 

#### Übergabeparameter:

* + matrix (zu überprüfende Matrix)

#### Rückgabewert

* + True oder False (Matrix gerade oder ungerade)

#### Funktion:

* + Überprüft, ob die gegebene Matrix quadratisch ist oder nicht und liefert das Ergebnis zurück