Dokumentation

Inhaltsverzeichnis

Inhalt

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

```
def mult(A, B):
```

Übergabeparameter:

- sympy Matrizen A und B

Rückgabewert:

- sympy Matrix

```
return simplify(C)
```

Funktion:

- Es wird die Matrizenmultiplikation von A und B berechnet

Gaußsches Eliminationsverfahren

Skript: gauss.py

```
def gauss(A):
```

Übergabeparameter:

- sympy Matrix A

Rückgabewert:

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

```
return (A, tuple(pivots))
```

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

```
def inverse(A):
```

Übergabeparameter:

- sympy Matrix A

Rückgabewert:

- sympy Matrix oder None

```
return (gauss(A)[0])[:, list(range(n, 2*n))]
```

Funktion:

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

Determinante

Skript: laplace.py

```
def laplace(A):
```

Übergabeparameter:

- sympy Matrix A

Rückgabewert:

- sympy Zahl oder Ausdruck

```
return detA
```

Funktion:

- Berechnet die Determinante der Matrix A

Eigenwerte

Skript: eigenwerte.py

```
def eigenVal(A):
```

Übergabeparameter:

- sympy Matrix A

Rückgabewert:

- Liste von symbolischen Eigenwerten

```
return solve(charPoly, x)
```

Funktion:

- Berechnet symbolisch die Eigenwerte der Matrix A

```
def eigenValNumeric(A):
```

Übergabeparameter:

- sympy Matrix A

Rückgabewert:

- Liste von nummerischen Eigenwerten

```
return list(map(lambda x: x.evalf(), eigenVal(A)))
```

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:

```
def generateHamming(bitList):
```

Übergabeparameter:

- bitList (4 Bit Liste)
  - Form (Beispiel):
    ```
    [1, 0, 1, 1]
    ```

Rückgabewert:

- 7 Bits Liste. Dies sind die Bits, die bei der Kommunikation an den Empfänger übertragen werden.
  ```
  return list(y)
  ```

Funktion:

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

```
def decodeHamming(bitList):
```

Übergabeparameter:

- bitList (7 Bit Liste)
  - Form (Beispiel):
    ```
    [0, 1, 1, 0, 0, 1, 0]
    ```

Rückgabewert:

- Tupel bestehend aus einer 4 Bit Liste und einem String.
  ```
  return ([rNew[2], rNew[4], rNew[5], rNew[6]], "Bit"+str(i+1)+
  " korrigiert")
  ```

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:

`def start():`

Übergabeparameter:

- keine

Rückgabewert:

- keiner

Funktion:

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

`def rotateMatrix(points, alphaDeg):`

Übergabeparameter:

- points (Liste von Punkten)
  - Form:
    ```
    [[x1, y1], [x2, y2], ...]
    ```
- alphaDeg (Rotationswinkel in Grad)

Rückgabewert:

- Liste von Punkten welche rotiert wurde
  ```
  return newMatrix
  ```

Funktion:

- Übergebene Punkte Liste wird am übergebenen Winkel rotiert und zurückgegeben

`def scaleMatrix(points, lambdaValue):`

Übergabeparameter:

- points (Liste von Punkten)
  - Form:
    ```
    [[x1, y1], [x2, y2], ...]
    ```
- lambdaValue (Skalierungswert)

Rückgabewert

- Liste von Punkten welche skaliert wurde
  ```
  return newMatrix
  ```

Funktion:

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

```
def mirrorMatrix(points, axis):
```

Übergabeparameter:

- points (Liste von Punkten)
  - Form:
    ```
    [[x1, y1], [x2, y2], ...]
    ```
- axis (Spiegelungsachse oder Wert des Winkels in Grad einer Ursprungsgeraden zur x-Achse)

Rückgabewert

- Liste von Punkten welche gespiegelt wurde
  ```
  return newMatrix
  ```

Funktion:

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

```
def plotPoints(pointsList, newPointsList):
```

Übergabeparameter:

- pointsList (originale Liste von Punkten)
  - Form:
    ```
    [[x1, y1], [x2, y2], ...]
    ```
- newPointsList (verarbeitete Liste von Punkten)
  - Form:
    ```
    [[x1, y1], [x2, y2], ...]
    ```

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

```
def checkAndUseMatrix(pointsList):
```

Übergabeparameter:

- pointsList (originale Liste von Punkten)
  - Form:
    ```
    [[x1, y1], [x2, y2], ...]
    ```

Rückgabewert

- veränderte Liste von Punkten
  ```
  return newPointsList
  ```

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:

```
def start():
```

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

```
def code(text, matrix):
```

Übergabeparameter:

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

Rückgabewert

- ver-/entschlüsselter Text

```
return codedText
```

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

```
def keyGenerator(dim):
```

Übergabeparameter:

- dim (Dimension der zu generierenden quadratischen Matrix)

Rückgabewert

- Quadratische Matrix und deren Inverse

```
return K, K.inv_mod(94)
```

Funktion:

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

```
def is_square_matrix(matrix):
```

Übergabeparameter:

- matrix (zu überprüfende Matrix)

Rückgabewert

- True oder False (Matrix gerade oder ungerade)

```
return rows == cols
```

Funktion:

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