Grundkenntnisse in Python's Numerik-Library numpy

1 Lernziele

Ein zentraler Eckpunkt des Numerik-Unterrichts ist die systematische praktische Umsetzung theoretischer mathematischer Konzepte sowie das Experimentieren am Computer. Dazu wird im Kurs systematisch und ausschliesslich Python und numpy eingesetzt, welches auch Bestandteil der SEP sein wird. In dieser Lektion werden die Grundlagen und ein gemeinsamer Wissensstand aller Studierenden erarbeitet.

2 Rechnen mit Daten

Generell werden wir stets die folgenden Libraries einbinden:

```
import math
import numpy as np
from matplotlib import pyplot as plt
```

2.1 Spyder als Taschenrechner

- Summe, Differenz, Produkt, Quotient.
- Potenzieren

  ```
  In[1]: 2**5
  Out[1]: 32
  ```

- Wurzeln `math.sqrt` und `math.pow`

  ```
  In[2]: math.sqrt(45)
  Out[2]: 6.708203932499369

  Out[3]: math.pow(23,(1/5))
  Out[3]: 1.8721712305548575
  ```

2.2 Vektoren und Matrizen

- Datenreihen als Vektoren (einfache Indizierung)

```
In [4]: x = np.array([1, 2, 3, 4])
In [5]: x
Out[5]: array([1, 2, 3, 4])
```

- Datenreihen als 2D Array

```
In [6]: y = np.array([[-1], [2], [3], [4]])
In [7]: y
Out[7]:
array([[-1],
       [2],
       [3],
       [4]])
```

- Skalare Operationen

```
In 8: 2 * x
Out[8]: array([2, 4, 6, 8])

In [9]: 2 * x + 4
Out[9]: array([ 6,  8, 10, 12])
```

- Transposition

```
In [10]: y.T
Out[10]: array([[1, 2, 3, 4]])
```

- Vektoraddition

```
In [11]: x + y.T
Out[11]: array([[0, 4, 6, 8]])
```

- Skalarprodukt

```
In [12]: np.dot(x,x)
Out[12]: 30
```

- Punktweise Operationen:

```
In [13]: x*x
Out[13]: array([ 1,  4,  9, 16])
```

- Matrizen (zweifache Indizierung)

```
In [14]: A = np.array([[16, 3, 2, 13], [5, 10, 11, 8],
                       [9, 6, 7, 12], [4, 15, 14, 1]])
In [15]: A
Out[15]:
array([[16,  3,  2, 13],
       [ 5, 10, 11,  8],
       [ 9,  6,  7, 12],
       [ 4, 15, 14,  1]])
```

- Matrixbildende Funktionen

`eye`	Einheitsmatrix
`zeros`	Nullmatrix
`ones`	Einser-Matrix
`diag`	Diagonalmatrix
`random`	Zufallszahlenmatrix

```
In [16]: np.random.random((2,3))
Out[16]:
array([[0.12577127, 0.89748488, 0.12893368],
       [0.66688226, 0.92110059, 0.00582778]])
```

- Verschiedene Funktionalitäten können sich hinter dem gleichen Funktionsnamen verbergen: `help(np.eye)`
- Die Befehle `shape` und `len`

```
In [17]: A = np.array([[16, 3, 2, 13], [5, 10, 11, 8], [9, 6, 7, 12]])

In [18]: np.shape(A)
Out[18]: (3, 4)

In [19]: len(A)
Out[19]: 3
```

- Matrix-Vektor- und Matrix-Matrix-Multiplikation

```
In [20]: z = A.dot(x) # oder z = A @ x
In [21]: z
Out[21]:
array([[80],
[90],
```

```
[90]])

In [22]: B = np.array([[1, 2], [3, 4], [5, 6], [7,8]])
In [23]: C = A @ B # oder C = A.dot(B)
In [24]: C
Out[24]:
array([[126, 160],
[146, 180],
[146, 180]])
```

- Achtung: Wird der Typ nicht explizit via `np.array(...,dtype=...)` angegeben, kann es zu Überraschungen kommen:

```
In [25]: x = np.array([1])
In [26]: x[0] = x[0] / 2
In [27]: x
Out[27]: array([0])

In [28]: y = np.array([1.])
In [29]: y[0] = y[0] / 2
In [30]: y
Out[30]: array([0.5])

In [31]: z = np.array([1],dtype=np.float64) # safe
In [32]: z[0] = z[0] / 2
In [33]: z
Out[33]: array([0.5])
```

2.3 Aufgaben

1. Finden Sie heraus, wie sich der Sinus von 30 Grad berechnen lässt.
2. Überlegen Sie sich Varianten zur Definition der Matrix

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

3 Graphische Darstellungen

3.1 Grundlegendes

- Der Operator `numpy.linspace(start, stop, num)` dient u.a. zur komfortablen Vektor-Definition:

```
In [20]: np.linspace(1,4,4)
Out[20]: array([1., 2., 3., 4.])

In [21]: np.linspace(1,5,9)
Out[21]: array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ])

In [22]: np.linspace(1,10,4)
Out[22]: array([ 1.,  4.,  7., 10.])
```

- Visualisierung einer Funktion

```
x = np.linspace(0,2*math.pi,100)
y = np.sin(x) # Vorsicht: math.sin() geht nicht für Vektoren!
plt.plot(x, y, color='red')
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.title('Plot von y = sin(x)')
plt.show()
```

- Visualisieren Sie analog das Polynom $p(x) = x^2 + x + 17$.

3.2 Aufgaben

1. Probieren Sie selbstständig folgende Befehlskette aus.

```
x = np.linspace(0,20,100)
y1 = np.sin(x) / np.sqrt(x+1)
y2 = np.sin(x/2)/np.sqrt(x+1)
y3 = np.sin(x/3)/np.sqrt(x+1)
plt.xlabel( 'x' )
plt.plot(x, y1, 'blue', x, y2, 'red', x, y3, 'orange')
plt.legend(['y1','y2','y3'])
plt.show()
```

2. Finden Sie bei jedem der folgenden Befehle heraus, was passiert.

```
fig, axs = plt.subplots(2)
fig.suptitle('Vertically stacked subplots')
```

```
x = np.linspace(0,2*math.pi,100)
y = np.sin(x)
axs[0].plot( x, y, color='red' )
axs[1].plot( x, -y, color='blue' )
```

3. Was müssen Sie ändern, damit die Plots nebeneinander stehen?

4 Erstellung eigener Funktionen

- Im folgenden Beispiel einer Funktion ist die Eingabevariable x. Programmieren Sie folgendes Beispiel:

```
def log17(x):
    y = np.log(x)/np.log(17)
    return y
```

- Welchen Wert hat log17(17) und log17(17**2)?
- Implementieren Sie analog das folgende Beispiel:

```
def pythag(a,b):
    c = np.sqrt(a**2 + b**2)
    return c
```

- Die natürlichen Zahlen a=3, b=4 und c=5 bilden ein sogenanntes Pythagoreisches Tripel. Verifizieren Sie auf Papier sowie mit einem Python-Experiment, dass mit (a,b,c) und einer natürlichen Zahl k auch (ka,kb,kc) ein Pythagoreisches Tripel ist.

4.1 Aufgaben

1. Schreiben Sie ein Skript, das die Funktionen $x = \cos(t)$ und $y = \sin(t)$ mit t im Intervall $[0, 2\pi]$ auswertet und mittels `plot(x,y)` darstellt. (Hilfestellung: Sie brauchen geeignete Stützstellen t im Intervall.)

2. Modifizieren Sie das vorangegangene Programm zu einer Python-Funktion, die als variable Eingabeparameter die linke und rechte Intervallgrenzen a, b des t-Intervalls verwendet.

3. Schreiben Sie eine Python-Funktion, die die Funktionen $x = \sin(At+a)$ und $y = \sin(Bt + b)$ für t im Intervall $[0, 2\pi]$ auswertet und mittels `plot(x,y)` darstellt. Dabei sollen A, a und B, b variable Eingabeargumente der Funktion sein.

4. Experimentieren Sie mit folgenden Werten:
   $A = 2$, $a = \pi/4$, $B = 1$, $b = 0$
   $A = 3$, $a = \pi/2$, $B = 1$, $b = 0$
   $A = 1$, $a = 0$, $B = 1$, $b = 0$
   Variieren Sie dann weiter A, a, B und b. Was stellen Sie fest? Beschreiben Sie ihre Beobachtungen in Worten. Welche Bedeutung haben a, b sowie A, B? Recherchieren Sie asserdem im Internet das Thema "Lissajous Figuren".

A Lösungen

4.1

2.
```
def lsg(a,b):
    t = np.linspace(a,b,1000)
    x = np.cos(t);
    y = np.sin(t);
    plt.plot( x, y, color='red' )
    plt.show()
```

3.
```
def lissajous(A,a,B,b):
    t = np.linspace(0,2*math.pi,1000)
    x = np.sin(A*t + a);
    y = np.sin(B*t + b);
    plt.plot( x, y, color='red' )
    plt.show()
```