

---

# Skriptsprache Python

Arbeiten mit  
NumPy und Matplotlib

---

# NumPy

- NumPy → Python-Paket für mathematische Berechnungen
- Programmlogik stark an Mathematik-Software Matlab angelehnt
- NumPy kein Standardbestandteil von Python → muss nachträglich installiert werden → für Kurs schon gemacht
- NumPy implementiert Funktionen für wissenschaftliche Berechnungen und Analysen in Python:
  - Container für Zahlenreihen, Matrizen
  - Berechnungsverfahren für lineare Algebra
  - Methoden für Zufallszahlenberechnung
  - Fourier-Analyse
  - usw.

# NumPy-Array

- Ein Array ist ein Vektor oder eine Matrix von Werten gleichen Typs
- NumPy-Funktionen führen automatisch Berechnungen auf allen Elementen eines Arrays aus

```
>>> import numpy as np, math as m
>>> v=np.array([1,2,3.4])
>>> v
array([ 1. ,  2. ,  3.4])
```

math.sin() geht  
nur für Skalare

```
>>> m.sin(v)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: only length-1 arrays can be converted to Python
scalars
>>> np.sin(v)
array([ 0.84147098,  0.90929743, -0.2555411 ])
```

numpy.sin() arbeitet  
auch auf Arrays

# Dimensionen von Arrays

---

- Attribut `shape` → Tupel mit Anzahl von Werten im Array pro Dimension
- Methode `reshape(tupel)`
  - Erzeuge neues Array mit angegebenen Dimensionen.
  - Nach Möglichkeit Speicher für ursprüngliche Daten referenzieren

```
>>> import numpy as np
>>> v1=np.array([1,2,3,4])
>>> v2=np.array([[1,2,3],[4,5,6]])
>>> v1.shape
(4,)
>>> v2.shape
(2, 3)
>>> v2.reshape((6,))
array([1, 2, 3, 4, 5, 6])
```

# Funktionen für Array-Erzeugung

---

Methode (np=numpy)	Aufgabe / Operation
<code>np.zeros(dim)</code>	Erzeuge ein Arrays aus lauter Nullen mit den in <code>dim</code> angegebenen Dimensionen
<code>np.ones(dim)</code>	Erzeuge ein Arrays aus lauter Einsen mit den in <code>dim</code> angegebenen Dimensionen
<code>np.arange(s, e, i)</code>	Erzeuge einen Vektor mit Werten von <code>s</code> bis <code>e</code> , die in der Schrittweite <code>i</code> inkrementiert werden
<code>np.linspace(s, e, n=50)</code>	Erzeuge einen Vektor mit <code>n</code> Werten in gleichen Abständen von <code>s</code> nach <code>e</code>
<code>np.logspace(s, e, n=50)</code>	Erzeuge einen Vektor mit <code>n</code> Werten in gleichen Abständen von $\text{base}^s$ nach $\text{base}^e$ . <code>base</code> kann als benannter Parameter angegeben werden (Default 10)

Weitere ...

---

# Rechnen mit Arrays

---

- NumPy-Funktionen führen automatisch Berechnungen auf allen Elementen eines Arrays aus
- Auch die mathematischen Operatoren arbeiten elementweise
- Für das Matrixprodukt existiert die Funktion `numpy.dot(a, b)`

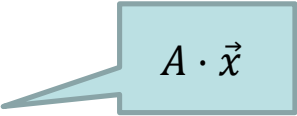
# Lineare Algebra über Modul linalg

- Numpy enthält Funktionen zur Lösung linearer Gleichungssysteme
- Beispiel: Für das folgende Gleichungssystem sei der Lösungsvektor gesucht:

$$\begin{pmatrix} 2 & 1 \\ 1 & -2 \end{pmatrix} \cdot \vec{x} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$

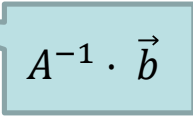
Lösung mit `np.linalg.solve()`

```
>>> import numpy as np
>>> A=np.array([[2,1],[1,-2]])
>>> b=np.array([[3],[4]])
>>> x=np.linalg.solve(A,b)
>>> x
array([[ 2.],
       [-1.]])
>>> np.dot(A,x)
array([[ 3.],
       [ 4.]])
```



Lösung über  $\vec{x} = A^{-1}\vec{b}$

```
>>> import numpy as np
>>> A=np.array([[2,1],[1,-2]])
>>> b=np.array([[3],[4]])
>>> Ai=np.linalg.inv(A)
>>> x=np.dot(Ai,b)
>>> x
array([[ 2.],
       [-1.]])
```





---

# Matplotlib

- Matplotlib → Python-Paket für wissenschaftliches Plotten von Daten
- Programmlogik stark an Mathematik-Software Matlab angelehnt
- Matplotlib kein Standardbestandteil von Python → muss nachträglich installiert werden → für Kurs schon gemacht
- Matplotlib stellt Diagrammtypen für wissenschaftliche Daten dar
  - XY-Plots
  - Histogramme, Balkendiagramme
  - Kuchendiagramme
  - usw.

# Beispiel: Matplotlib für XY-Plots

---

```
# tb_np_plot.py
```

```
import numpy as np
import matplotlib.pyplot as plt
```

```
a=3.4
b=0.4
x=np.linspace(0, 30)
y=a*np.exp(x*b)
```

```
plt.subplot(311)
plt.plot(x, y)
plt.grid()
plt.xlabel("x")
plt.ylabel("y")
```

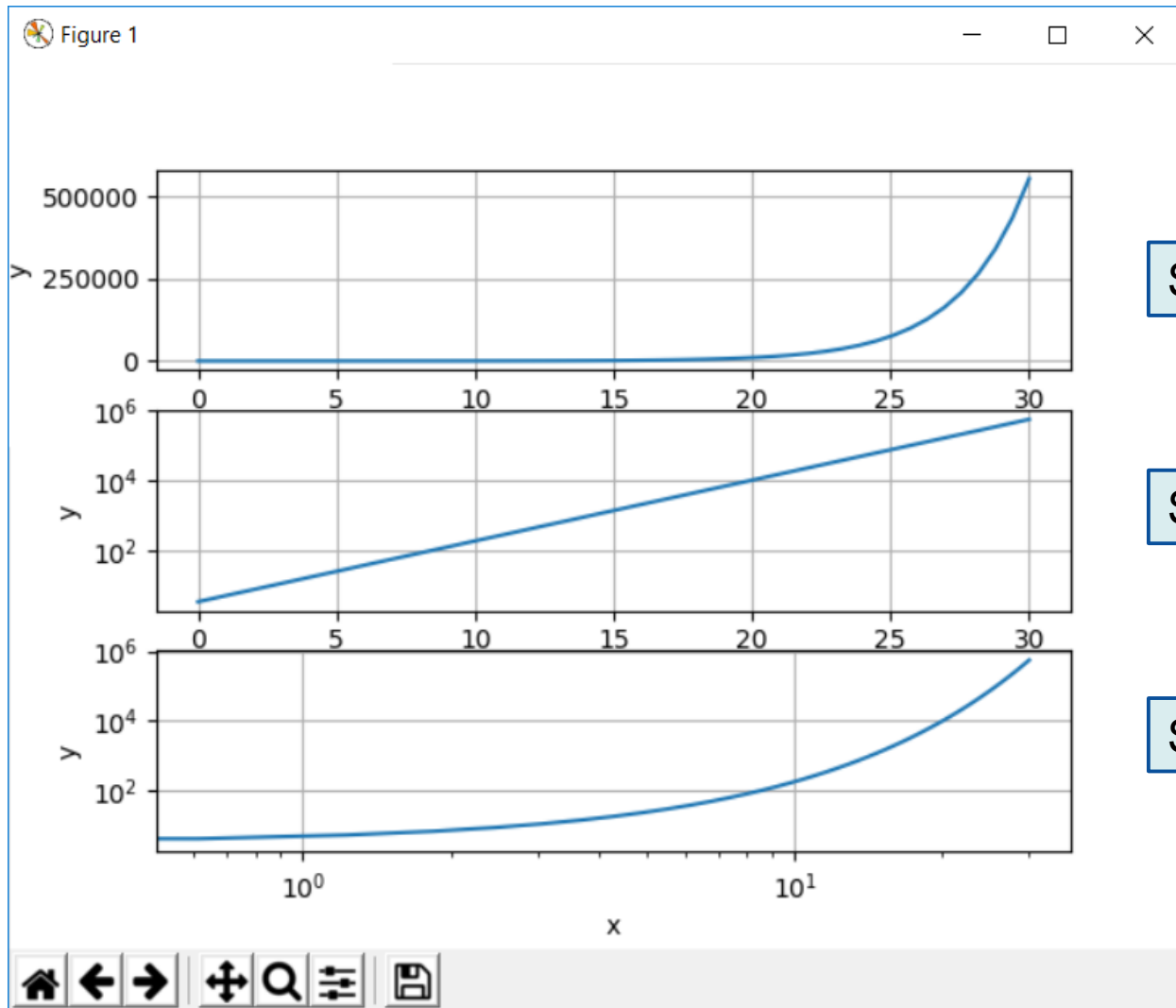
```
# Fortsetzung ...
```

```
plt.subplot(312)
plt.semilogy(x, y)
plt.grid()
plt.xlabel("x")
plt.ylabel("y")
```

```
plt.subplot(313)
plt.loglog(x, y)
plt.grid()
plt.xlabel("x")
plt.ylabel("y")
```

```
plt.show()
```

# Ausgabe XY-Plots



Subplot 311

Subplot 312

Subplot 313

- **Autor**

Prof. Dr.-Ing. Jürgen Krumm

- **Impressum**

Prof. Dr.-Ing. J. Krumm, TH Nürnberg Georg Simon Ohm,  
Fakultät Elektrotechnik Feinwerktechnik Informationstechnik,  
Postfach 210320, 90121 Nürnberg, Germany,  
Tel:+49-911-5880-1111,  
E-mail: [juergen.krumm@th-nuernberg.de](mailto:juergen.krumm@th-nuernberg.de)

Dieses Skriptum ist nur für den eigenen Gebrauch im Studium gedacht. Eine Weitergabe ist nur mit Zustimmung des Autors gestattet.