

NUMERISCHE DATENAUSWERTUNG

Carsten Knoll, Institut für Regelungs- und Steuerungstheorie (RST)

Dresden, Wintersemester 2013/14

Was macht man wenn man (Mess-) Daten auswertet?

- Daten laden
- Relevante Daten (bzw. Abschnitte) heraussuchen
- Aus vorhanden Größen neue bestimmen
- Informationsdichte erhöhen
- Grafisch darstellen



- Lernziele:
 - Daten Laden / Speichern
 - Produktiver Umgang mit Numpy Arrays
 - Überblick, welche Algorithmen schon implementiert sind

```
import numpy as np

array1 = np.load("file1.npy") # Binaerformat
array2 = np.loadtxt("file2.dat") # TXT-Format

# Rechnungen

np.save("ergebnis.bin", result)
```

Weitere Möglichkeiten (z.B. Matlab-Format: `scipy.io`)

Indizierungsmöglichkeiten von Numpy Arrays:

- `int`-Zahlen
- "Slices" (`x[1:10]`)
- `int`-arrays
- `boolean`-arrays

```
x = np.arange(20)
```

```
# Alle Werte die kleiner 10 sind negieren:
```

```
x[x<10]*=-1
```

- Nachfolger - Vorgänger
- Für Ableitungsnaherung: noch durch Δx teilen
„Differenzenquotient“
- Ergebnis ist um eins kürzer als Eingangsdaten

```
x = np.arange(20) # [0, 1, 2, ...]  
xd = np.diff(x) # [1, 1, 1, ...]
```

- Filterung (Tiefpass, Gleitender Mittelwert, ...)
→ Paket: `scipy.signal`

- Darstellung des Frequenzspektrums
→ Paket: `numpy.fft` (Fast Fourier Transform)

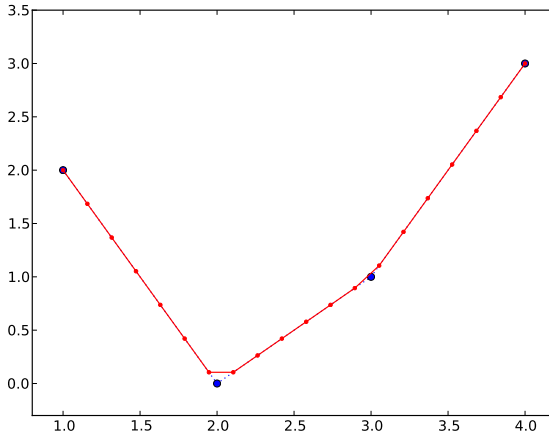
Beides erfordert Hintergrundwissen (darum hier nicht Thema)

- Interpolation (Zwischenwerte ergänzen, Abtaste ändern, ...)
- Paket: `scipy.interpolate`

```
import scipy.interpolate as ip
import numpy as np
import pylab as pl

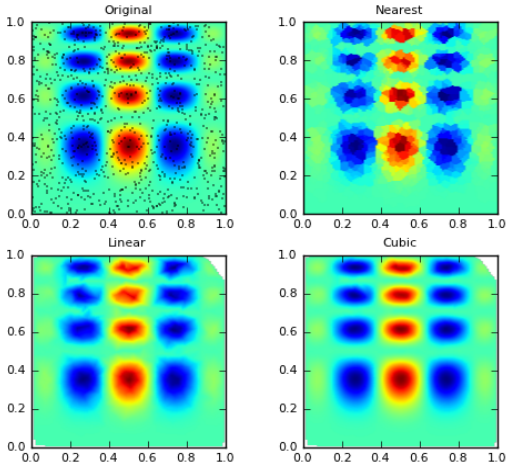
x = [1,2,3,4]
y = [2,0,1,3]
pl.plot(x,y, "bo:")

fnc1 = ip.interp1d(x,y)
xx = np.linspace(1, 4, 20)
pl.plot(xx,fnc1(xx), "r.-")
pl.show()
```

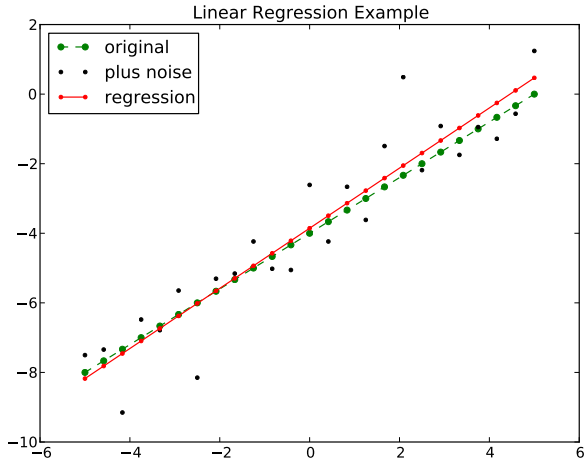


Interpolation (III)

- Geht auch N-dimensional mit irregulär verteilten Eingangsdaten



- Regression → `scipy.polyfit`



```
import scipy as sc
import numpy as np
import pylab as pl

n = 25
t = np.linspace(-5,5,n)
a, b = 0.8 , -4

x = sc.polyval([a,b],t) #  $x = a*t + b$ 
xn = x+sc.randn(n) # mit rauschen

ar, br = sc.polyfit(t,xn,1) # Regressionsgerade
xr = sc.polyval([ar,br],t)

pl.plot(t,x,'go--')
pl.plot(t,xn,'k.')
pl.plot(t,xr,'r.-')

pl.show()
```