TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Maschinenwesen Institut für Verarbeitungsmaschinen und mobile Arbeitsmaschinen

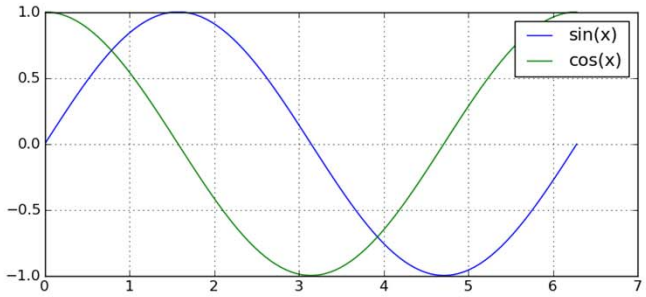
2D - Plotten / Visualisierung

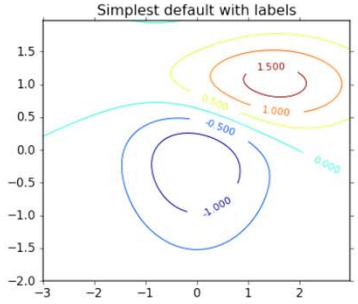
mit numpy und matplotlib

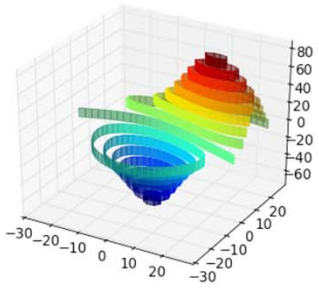
06.06.2011 Sebastian Voigt

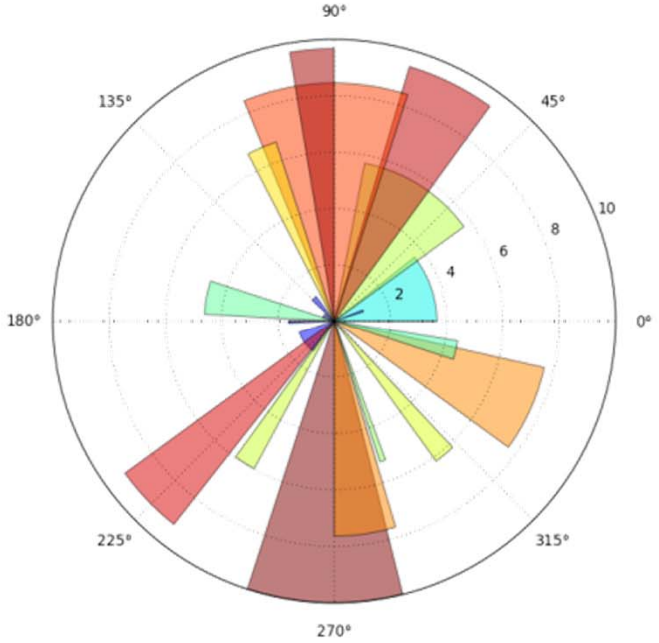
Wie was wo warum?

- Visualisierung von Messdaten, Simulationsergebnissen etc.
- Publikationsfertige Grafiken für Studien- und Diplomarbeiten, Dissertationen wissenschaftliche Artikel und Beiträge







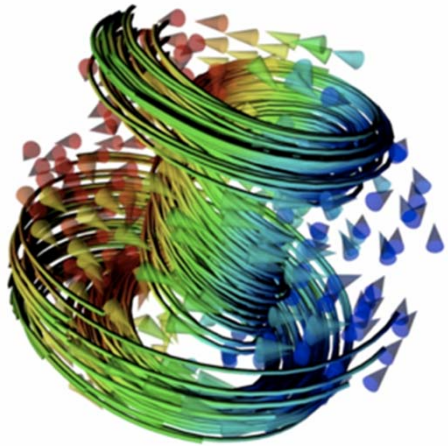


Python Pakete für Visualisierung allg.

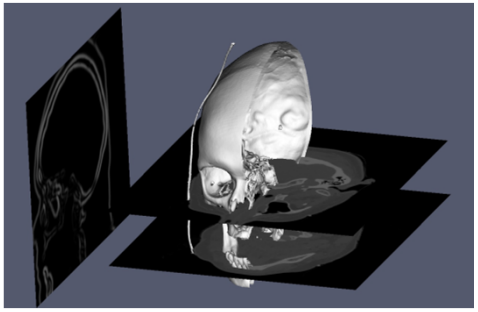
zahlreiche Pakete, die (nicht nur 2D) Plotfunktionalitäten bereitstellen:

- gnuplot-py – GnuPlot interface
- Chaco – schnelle 2D-lib von Enthought
- PyQwt – Plotwidgets für Qt

- Mayavi



- Vtk
- ...



Übersicht: http://wiki.python.org/moin/NumericAndScientific/Plotting

Quasi-Standard für 2D

numpy – numeric python

- Grundlage für nahezu alle Pakete, die irgendwie numerisch rechnen
- Kern: n-dimensionale Array-Klasse
- Lineare Algebra …

→ vertiefende Veranstaltung im Pythonkurs

http://numpy.scipy.org/

matplotlib

- Quasi-Standard für 2D-Plotten mit Python
- Syntax an Matlab angelehnt → einfach für Umsteiger
- Sehr umfangreiche Plotfunktionen
- Qualitativ hochwertige Ergebnisse
- Gute Doku

http://matplotlib.sourceforge.net/

Einführung numpy

Tutorial zum Nachlesen:
http://www.scipy.org/Tentative_NumPy_Tutorial

Und los geht‘s:

```
>>> from numpy import *

>>> a = array( (1,2,3) )
>>> a
array([1,2,3])

>>> b = array( ((1,2,3),(2,3,4)) )
>>> b
array([[1,2,3],
       [3,4,5]])
```

Weitere Möglichkeiten, Arrays zu generieren:

```
>>> linspace(0,1,6)
array([ 0. ,  0.2,  0.4,  0.6,  0.8,  1. ])

>>> arange(0,10,1)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Spezielle Arrays / Matrizen:

```
>>> zeros((2,3))
>>> ones((2,3))
>>> diag((1,2,3))
>>> eye(3)
```

Rechnen mit arrays

Alle Operationen werden elementweise ausgeführt!

```
>>> 2*a
array([2, 4, 6])
>>> sin(a)
array([ 0.84147098,  0.90929743,  0.14112001])
>>> abs(-a)
array([1, 2, 3])
```

Länge eines Vektors, „echte“ Matrix-Multiplikation:

```
>>> from numpy.linalg import norm
>>> norm(a)
3.7416573867739413
>>> dot(a,a)
14
```

Stapeln

Zusammenfügen von Arrays:

```
>>> hstack((a,a))
array([1, 2, 3, 1, 2, 3])

>>> vstack((a,b))
array([[1, 2, 3],
       [1, 2, 3],
       [2, 3, 4]])

>>> hstack((a, b.flatten()))
array([1, 2, 3, 1, 2, 3, 2, 3, 4])
```

Slicing

Wie komme ich an die Werte im Array ran?

grundsätzlich: `a[start:stop:step]`

```
>>> a[0]                → einzelnes Element
1
>>> a[0:1]              → Bereich aus dem Array
array([1])
>>> a[0:2]
array([1,2])
>>> a[::-1]             → kehrt Reihenfolge um
array([3,2,1])
>>> b[:,0]              → alle Werte der ersten Spalte
array([1,2])
>>> b[0,1]              → äquivalent zu b[0][1]
2
```

Zwei Möglichkeiten der Verwendung:

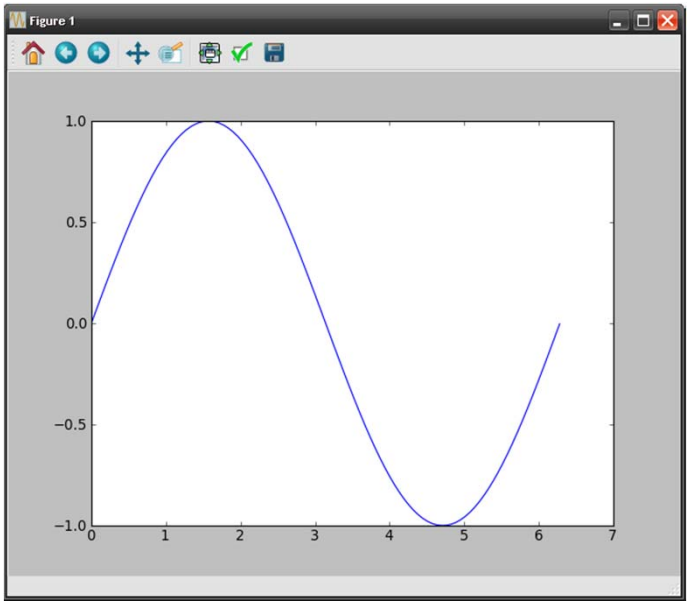
(1) interaktiv in der Konsole (pylab)
(2) objektorientiert in Skripten und Anwendungen (pyplot)

Einführungsbeispiel

```
from matplotlib.pylab import *

x = linspace(0,6.28,100)
y = sin(x)

plot(x,y)
show()
```



Anständiger Plot

Komplettierung mit Beschriftungen, Legende, Hilfslinien, Titel:

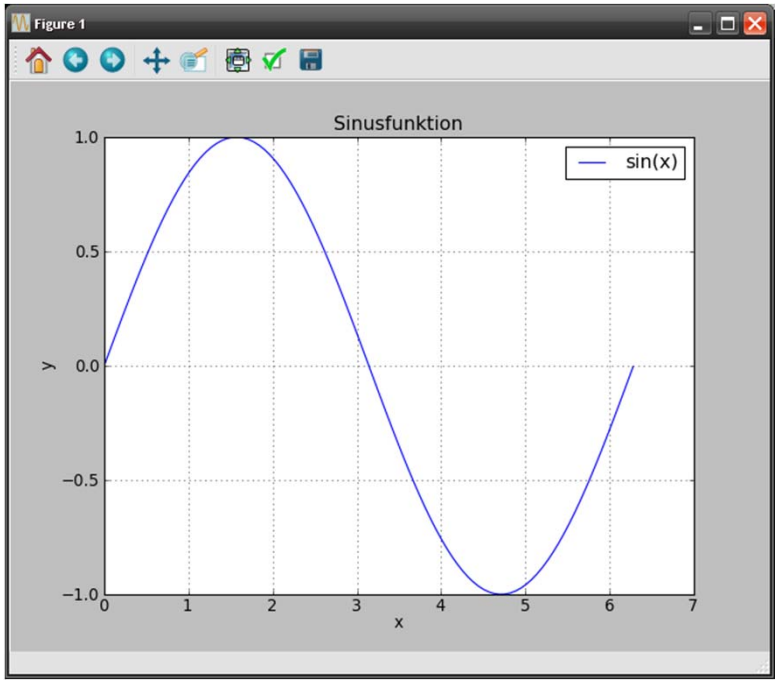
```
from matplotlib.pylab import *

x = linspace(0,6.28,100)
y = sin(x)

title('Sinusfunktion')
plot(x,y, label='sin(x)')

xlabel('x')
ylabel('y')

legend()
grid()
show()
```



Objektorientierte Variante

Für komplexere Aufgaben wird i.d.R. Skript geschrieben:

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

x = np.linspace(0,6.28,100)
y = np.sin(x)

fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(1,1,1)

ax.set_title('Sinusfunktion')
ax.plot(x,y, label='sin(x)')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.legend()
ax.grid()

plt.show()
```

Begriffe

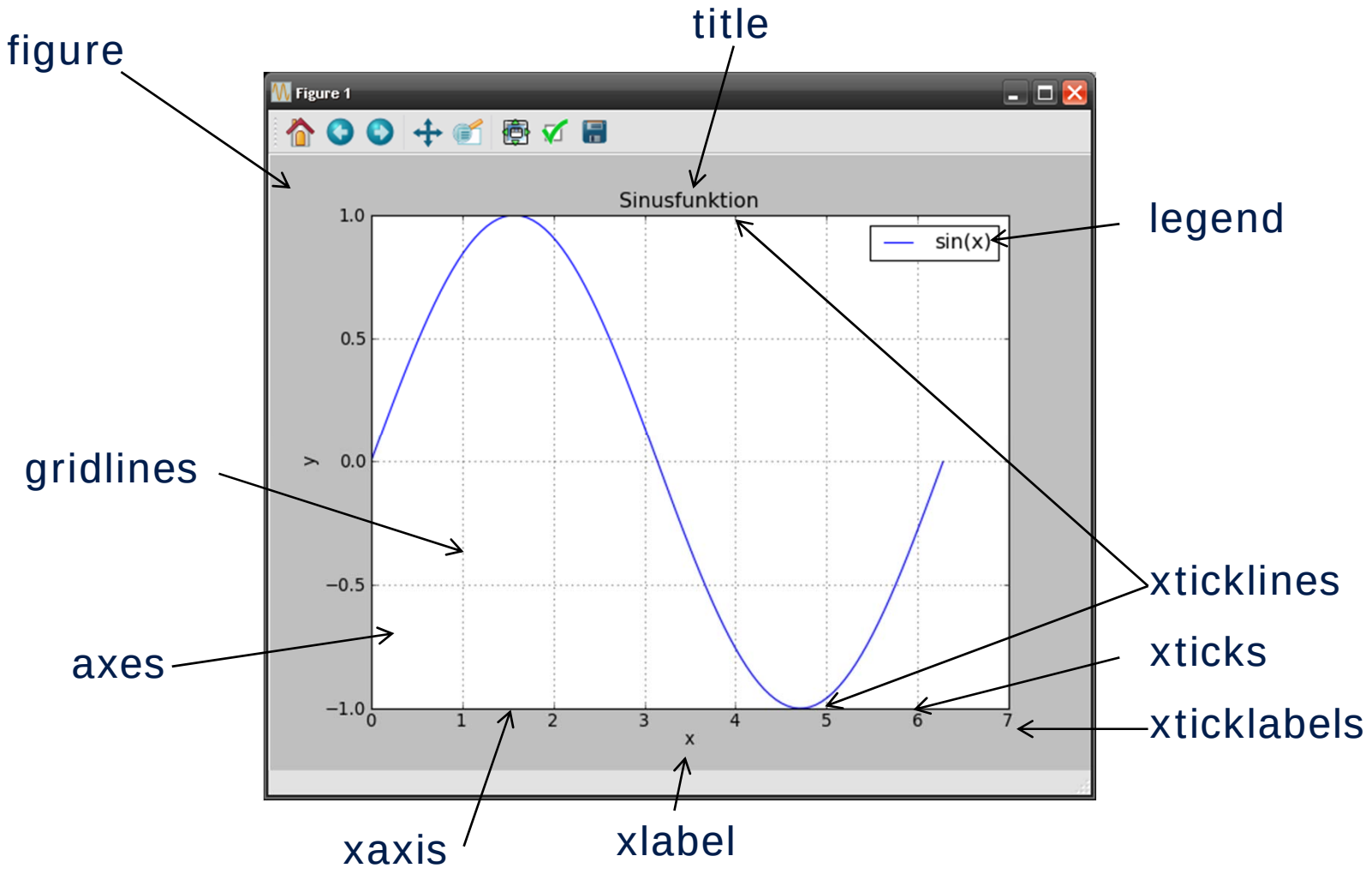


figure
title
Figure 1
Sinusfunktion
sin(x)
legend
gridlines
axes
xticklines
xticks
xticklabels
xaxis
xlabel
1.0
0.5
0.0
−0.5
−1.0
y
0
1
2
3
4
5
6
7
x

Wo bekomme ich Hilfe?

Matplotlib ist sehr umfangreich und komplex → Doku unter

http://matplotlib.sourceforge.net/contents.html

Tipp 1: gallery und von da abgucken

Tipp 2: Dokumentation der Axes-Klasse

http://matplotlib.sourceforge.net/api/axes_api.html?highlight=axes subplot#matplotlib.axes.Axes

Alle Plot- und Zeichenfunktionen sind über die Axes-Klasse zu erreichen!

- plot(), bar(), scatter(), arrow(), …
- Besonders wichtig: Keyword-Argumente

Siehe auch Cheatsheet!

TECHNISCHE UNIVERSITÄT DRESDEN

Häufig benötigte Einstellungen

```
ax.set_aspect('equal')          → Seitenverhältnis 1:1
ax.set_xlim(0,10)               → Wertebereich der x-Achse
ax.set_xticklabels(['a','b']) → eigene Beschriftungen

ax.legend(loc=1)                → Position der Legende
ax.tick_params(kwargs**)        → Optik Achsenbeschriftung

leg = ax.legend(loc=0)          → Schriftgröße Legende
    for t in leg.get_texts():
        t.set_fontsize(13)
```

Grundsätzlich auch:
setp() ermöglicht das setzten von Eigenschaften von Objekten
getp() liefert die Objekte und deren kwargs (Introspection)

LaTeX und matplotlibrc

- Matplotlib bringt eigenen (reduzierten) LaTeX Compiler mit
- Ausprobieren:
  - `ax.plot(x,y, label='$sin(x)$')`
  - `ax.set_ylabel('$\mu_2$')`

- Optionen können global in rc gesetzt werden:

```
import matplotlib as mpl

mpl.rc('font',**{'family':'serif',
                 'size':11,
                 'serif':['Computer Modern Roman']})

mpl.rc('text', usetex=True)
```

→ Damit werden alle strings mit LaTeX compiliert