HOCHSCHULE
FÜR ANGEWANDTE
WISSENSCHAFTEN
MÜNCHEN

Prof. Dr.
Christian Münker

(I)Python in Labor & Lehre

Hochschule München, Oktober 2014

… und wie sieht es aus?

```
def fib(n):
    """
    Berechne Fibonacci-Zahlen
    """
    print('n =', n)  # für Python 3.x
    if n > 1:
        return fib(n - 1) + fib(n - 2)
    else:
        print 'Fertig'
        return 1
x = 9
print('F_n(%d) = %d') %(x, fib(x))
```

Namespaces

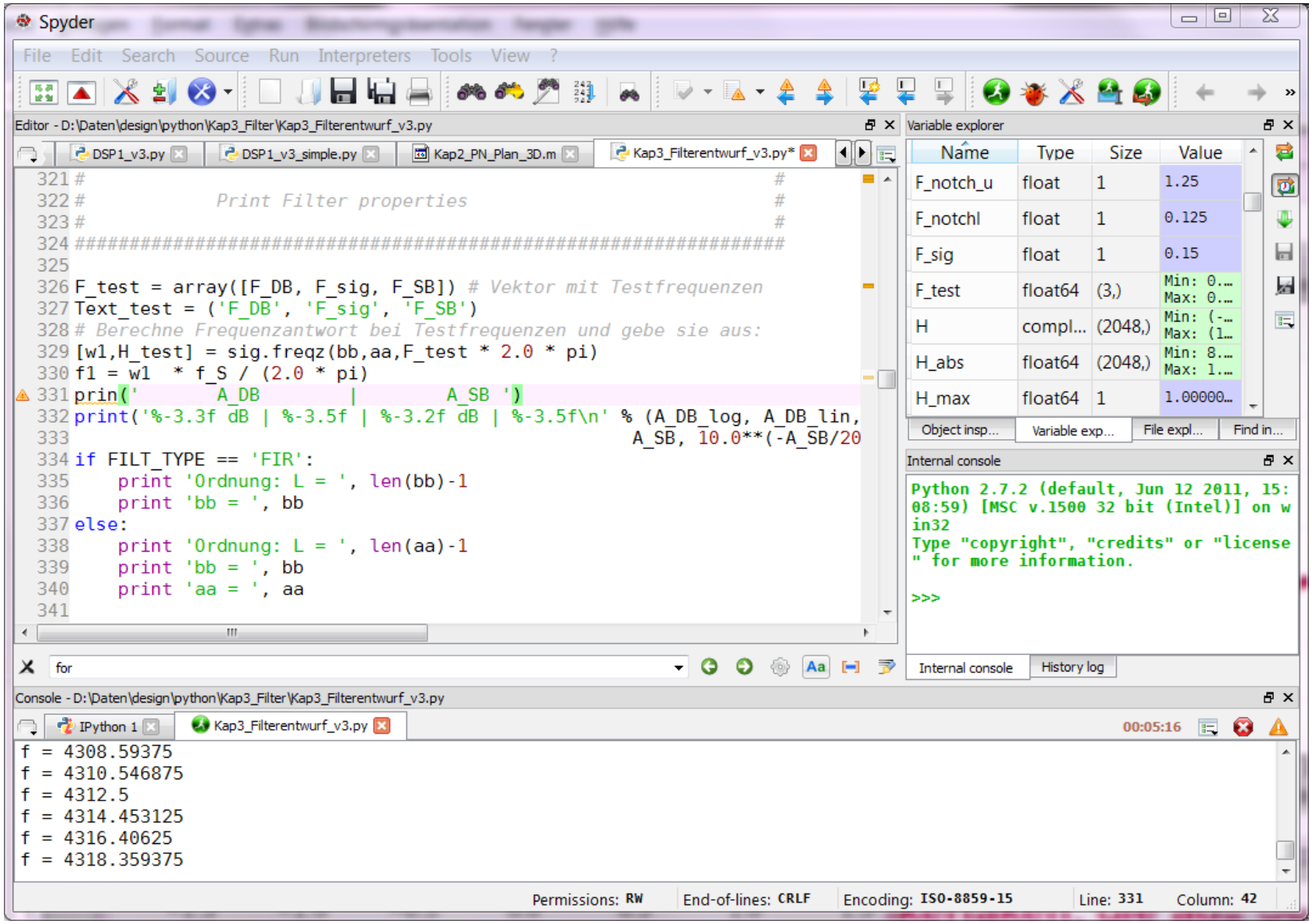
```
import numpy #  importiere numpy Modul mit eigenem Namespace
x = numpy.pi

import numpy as np              # importiere Modul und Untermodul
import numpy.random as rnd      # mit abgekürztem Namespace
x = rnd.normal() * np.pi        # EMPFOHLENE VARIANTE !!

from numpy import *             # Import aller Funktionen eines Moduls in
# alternativ *:                 # gemeinsamen Namespace (bequem, aber
import pylab                    # nur für interaktives Arbeiten empfohlen:
x = sin(pi * 0.1)               # "besudelt" den Namespace)

from numpy import pi,log10      # Kompromiss: Import oft benutzter
x = log10(1000)                 # Funktionen in gemeinsamen Namespace
```

* pylab: alle Funktionen aus numpy, matplotlib, pyplot

Entwicklungsumgebungen (IDEs)

- IDLE - das vi unter den IDEs (geht immer)
- **Spyder** - der Matlab-Clone (siehe nächste Folie)
- IPython – für Konsole (auch unter Spyder) und Browser
- Eclipse mit pydev-Plugin – nicht primär für wiss. Anwendungen
- … Raspberry Pi ;-)

Spyder als Python IDE für wiss. Anwendungen



Spyder als IDE (2)

- Sehr ähnlich zur (alten) Matlab – IDE
- Komfortabel durch Syntax-Highlighting (auch für Matlab!) und Code-Completion
- Rich Text Dokumentation zu Funktionen und Klassen
- Debugger, Delinting, Refactoring, Variable-Explorer, Profiler, Path Manager, …
- Verschiedene Run-Optionen: Nützlich, aber gewöhnungsbedürftig
- Einbindung von IPython-Terminals
- Projekt Manager und Git-Support (ein bisschen ...)
- Unterstützung von Event Loops für GUI-Applikationen

„One-Click Install“ Scientific Python Distributions

WinPython (Open Source)

- „portable Distribution“ für Windows (startet von USB-Stick)

Anaconda (Continuum Analytics)

- Für alle „großen“ Betriebssysteme
- Numba (Pro) für CPU (GPU) beschleunigte Simulationen, Wakari

Canopy (Enthought)

- Eigener Desktop, aber keine Spyder IDE

Alle drei mit eigenem Paketmanager, kostenlos für Hochschulen

- Was ist das und woher bekomme ich es?
- **Wissenschaftliche Anwendungen**
- Python als Ersatz für Matlab?
- IPython Notebooks und Wakari
- Python im Labor
- Was gibt's noch?

Die Module des **SciPy Software Stacks** machen aus dem „Schweizer Taschenmesser“ Python eine Sprache für wissenschaftliche Anwendungen:

- Numpy [Arrays und schnelle lineare Algebra]
- Scipy [Funktionen]
- Matplotlib [2D- und ein bisschen 3D-Grafik]
- IPython [interaktive Konsole / Webbased Computing]
- Sympy [Symbolische Mathematik]

SciPy bezeichnet den Stack, die Bibliothek und eine Konferenz

NumPy (numpy.org)

Grundlegendes Paket für Scientific Computing unter Python:

- ~ Matlab-Funktionalität ohne Toolboxen und Grafik
- *N*-dimensionale Array-Objekte
- Schnelle Routinen u.a. für lineare Algebra, Arraymanipulationen Fouriertransformationen und Erzeugung von Zufallszahlen
- Integration von C/C++ (z.B. DLLs) und Fortran Code (f2py)

Wichtig: http://wiki.scipy.org/NumPy_for_Matlab_Users

Beispiel:

```
import numpy as np
from numpy.linalg import det
A = np.array([[1., 2.],[3., 4.]])
print det(A)
```

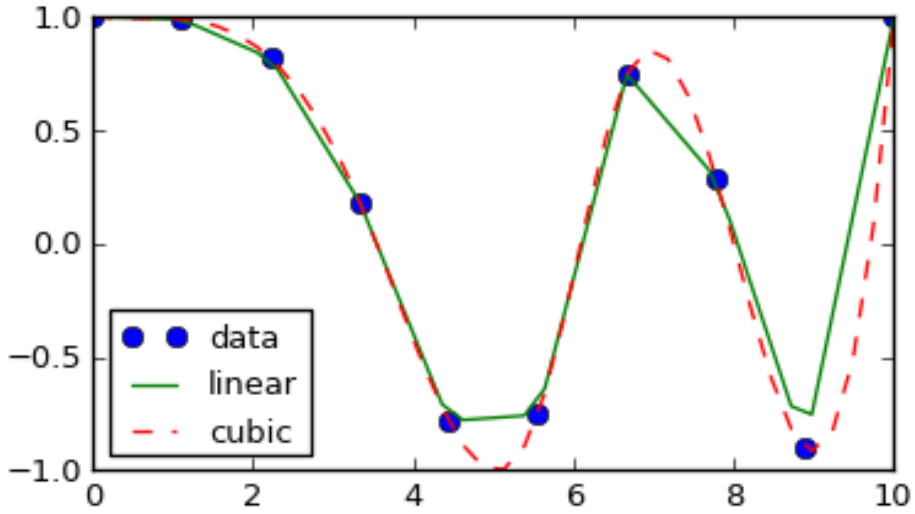
SciPy Bibliothek (scipy.org)

Ein Kernpaket des SciPy Stacks mit zahlreichen Funktionen, es ersetzt einige Matlab-Toolboxen:

- Spezielle Funktionen (scipy.special)
- Numerische Integration (scipy.integrate), Optimierung (scipy.optimize) und Interpolation (scipy.interpolate)
- Signalverarbeitung (scipy.signal)

Beispiel:

```
from scipy.interpolate import interp1d
x = np.linspace(0, 10, 10)
xnew = np.linspace(0, 10, 40)
y = np.cos(-x**2/8.0)
f = interp1d(x, y, kind='cubic')
y_ip =  f(xnew)
```



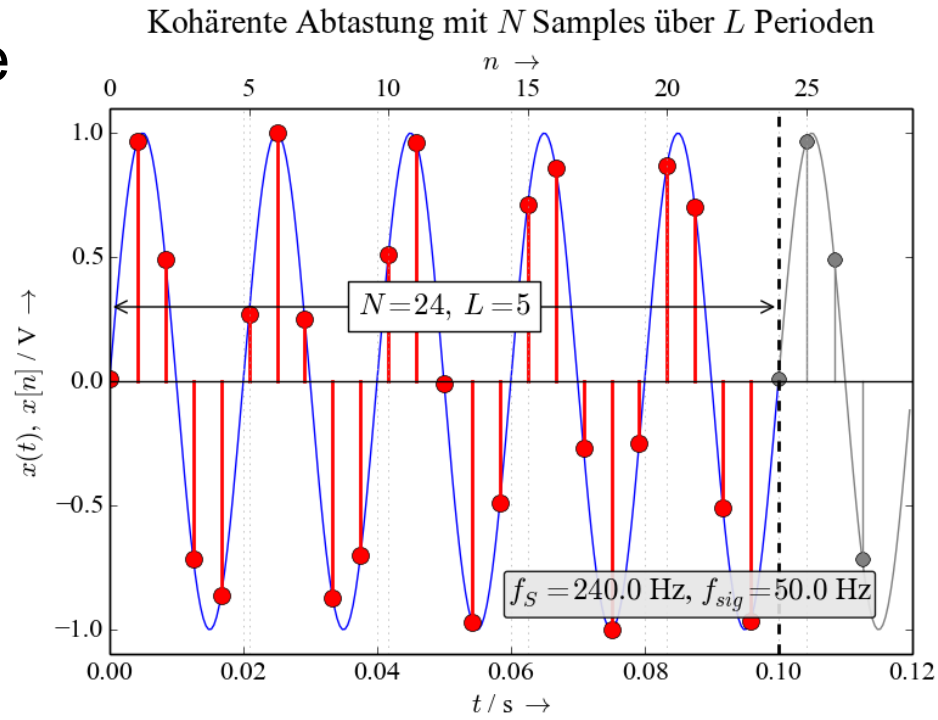
docs.scipy.org/doc/scipy/reference/tutorial/interpolate.html

matplotlib (matplotlib.org)

Bibliothek für interaktive und statische 2D- und (einfache) 3D-Plots

Anleitung und Beispiele u.a. unter

- www.loria.fr/~rougier/teaching/matplotlib/
- matplotlib.org/gallery.html



OO-API

```
import matplotlib.pyplot as plt
from numpy import arange
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(arange(9),arange(9)**2)
plt.show()
```

Matlab-Style-API

```
import matplotlib.pyplot as plt
from numpy import arange
plt.figure()
plt.plot(arange(9),arange(9)**2)
plt.show()
```

SymPy (sympy.org)

Bibliothek für symbolische Mathematik
Anleitung und Beispiele u.a. unter

- live.sympy.org/ (interaktiv)
- http://gamma.sympy.org/ (~ Wolfram Alpha)
- IPython.org Notebook zu SymPy

```
In [7]: eq = ((x+y)**2 * (x+1))
        eq
```

Out[7]: $(x+1)(x+y)^2$

```
In [8]: expand(eq)
```

Out[8]: $x^3 + 2x^2y + x^2 + xy^2 + 2xy + y^2$

```
In [9]: a = 1/x + (x*sin(x) - 1)/x
        a
```

Out[9]: $\frac{1}{x}\left(x\sin\left(x\right) - 1\right) + \frac{1}{x}$

```
In [10]: simplify(a)
```

Out[10]: $\sin\left(x\right)$

Sympy Session in IPython

Weitere Module

IPython

Interaktive Shell und browser-basiertes Notebook für Code, Text, Plots und andere Medien → eigene Folien

Pandas

Datenstrukturen und -analyse für Python

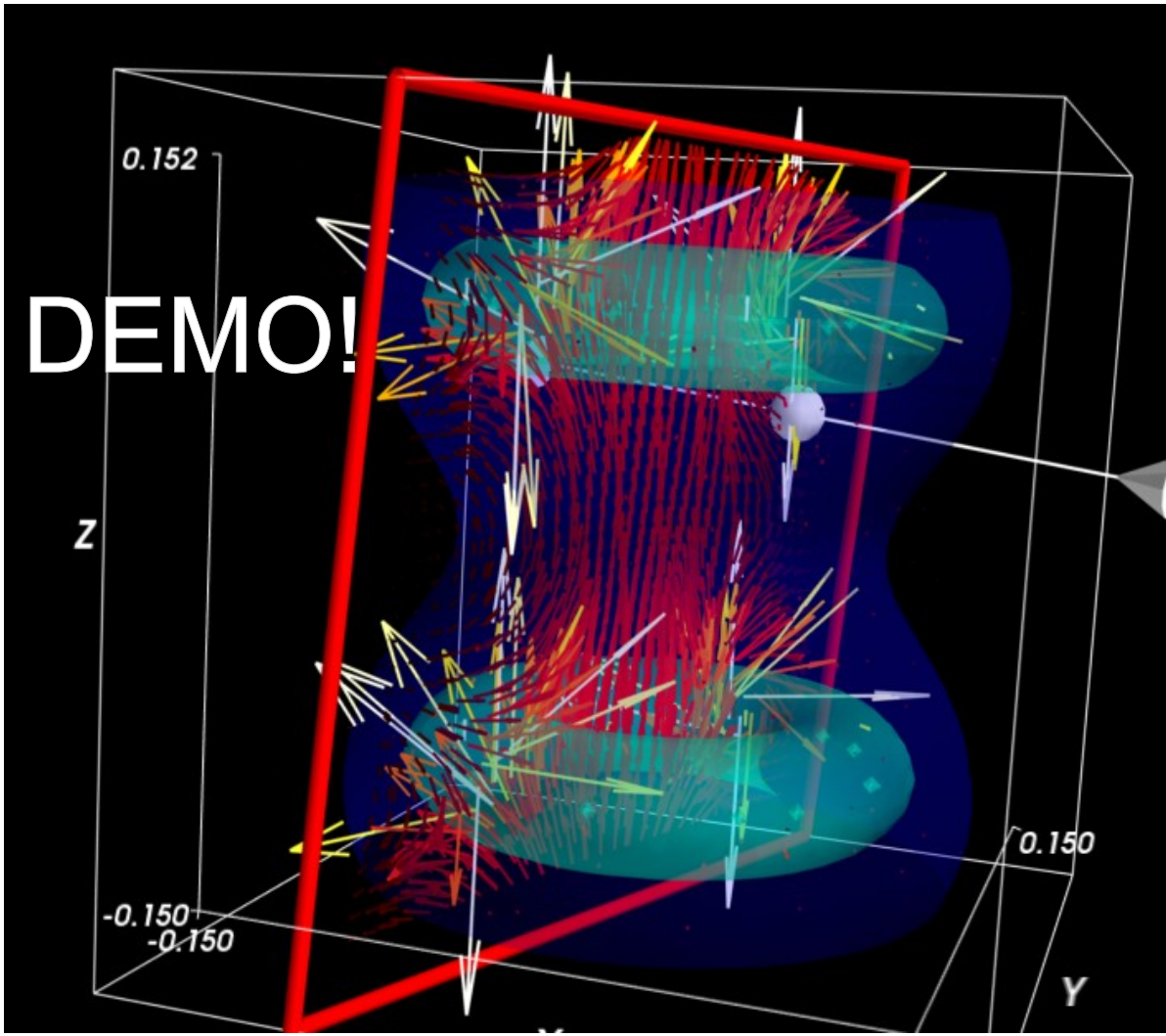
Mayavi2

3D-Datenvisualisierung und Animation (VTK-basiert) → nächste Folie

Mayavi – Interaktive High-End 3D-Visualisierung

mlab_interactive_example.py

magnetic_field_lines.py, B_field_visualize.py



Python 2.x oder 3.x?

- 2000 wurde Python 2.0, 2009 wurde Python 3.0 eingeführt (Feb. 2017: Python 2.7.12 und 3.6.0)
- Offizielles End-of-Life für Python 2.7 in 2020
- Verschiedene „Unsauberkeiten“ wurden behoben (print und input Funktion, automatische Float-Division, Unicode-Behandlung …) siehe docs.python.org/py3k/whatsnew/3.0.html
- 2017 sind fast alle Pakete für wissenschaftliche Anwendungen auch für Python 3.x verfügbar
- Nutze: `from __future__ import print_function, …` um Python 3 Syntax in Python 2 zu verwenden
- Neue Projekte nicht mehr mit Python 2 anfangen

Module für Simulation physikalischer Systeme

PyGame

Library zur Erstellung von Spielen mit Physics Engine und 2D- sowie 3D-Grafik. Auch geeignet für die Simulation / Demonstration von physikalischen Systemen

Siehe z.B. http://www.petercollingridge.co.uk/pygame-physics-simulation

VPython

Leicht bedienbare Bibliothek für die schnelle 3D-Visualisierung von Objekten (Animationen, physikalische Simulationen) – siehe z.B. www.youtube.com/user/physicsfiend oder www.visualrelativity.com/vpython/

Neu (Ende 2016): Integration mit IPython / Jupyter!

Python

- Was ist das und woher bekomme ich es?
- Wissenschaftliche Anwendungen
- **Ersatz für Matlab?**
- IPython Notebooks und Wakari
- Einsatz im Labor
- Python / Git für verteilte Entwicklungen und Open Source Projekte
- Was gibt's noch?

Umstieg von Python auf Matlab?

- ☹ Kein Äquivalent zu Simulink
- ☹ Unübersichtliches Ökosystem für Einsteiger
- ☹ Keine automatische Code-Konvertierung!
- 😐 … aber einfache Anpassung des Codes in 95% der Fälle
- 😐 „Schöne“ Plots erfordern Arbeit (wie bei Matlab)
- 😐 Vergleichbare Geschwindigkeit (NumPy basiert ebenfalls auf BLAS / LAPACK / Intel MKL)
- 😐 Kann durch Cython und andere C-Interfaces, Numba, pyCUDA, … beschleunigt werden (wie Matlab, aber besser integriert)
- 😐 Python kann nichts wirklich gut, ist aber ...

Migration Matlab → Python?

```
7  b = [0.1, 0]; a = [1, -0.9];
8  [h, k] = impz(b, a, 30);
9  figure(1);
10 subplot(211);
11 stem(k, h, 'r-');
12 ylabel('h[n] \rightarrow'); grid on;
13 title('Impulse Response h[n]');
14 subplot(212);
15 stem(k, 20*log10(abs(h)), 'r-');
16 xlabel('n \rightarrow'); grid on;
17 ylabel('20 log h[n] \rightarrow');
18 % ------- Filtered signal ---
19 figure(2);
20 yn = filter(b,a,xn);
21 yt = interp1(n, yn, t, 'cubic');
22 hold on; % don't overwrite plots
23 plot(t, yt,'color',[0.8,0,0],'LineWidth',3);
24 stem([0:length(yn)-1],yn,'b');
25 xlabel('n \rightarrow'); grid on;
26 ylabel('y[n] \rightarrow');
27 title('Filtered Signal');
28 %
```

```
b = [0.1, 0]; a = [1, -0.9] #
[h, k] = dsp.impz(b, a, N = 30) # calc. h[n]
figure(1)
subplot(211); grid(True)
stem(k, h, linefmt='r-') # x[n], red stems
ylabel(r'$h[k] \rightarrow$')
title(r'Impulse Response $h[n]$')
subplot(212); grid(True)
stem(k, 20*log10(abs(h)), linefmt='r-')
xlabel(r'$n \rightarrow$')
ylabel(r'$20 \log h[n] \rightarrow$')
# ------- Filtered signal ------------
figure(2); grid(True)
yn = sig.lfilter(b,a,xn) #filter xn with h
f = intp.interp1d(n, yn, kind = 'cubic')
yt = f(t) # y(t), interpolated
plot(t, yt, color='#cc0000', linewidth=3)
stem(n, yn, 'b') # y[n]
xlabel(r'$n \rightarrow$')
ylabel(r'$y[n] \rightarrow$')
title('Filtered Signal')
plt.show()       # draw and show the plots
```

- Was ist das und woher bekomme ich es?
- Wissenschaftliche Anwendunge
- Python als Ersatz für Matlab?
- **IPython / Jupyter Notebooks und Wakari**
- Python im Labor
- Was gibt's noch?

Im Terminal: IPython als IDE

IPython ist ein komfortables Terminal mit Rich Text

- Einfaches interaktives Arbeiten mit `from pylab import *`
- Zum „Herumspielen“ mit Code und Dokumentieren
- Für interaktive Lehrmaterialien
- „The IPython Notebook - Software Infrastructure for Reproducibility“
  http://nbviewer.ipython.org/urls/raw.github.com/ellisonbg/talk-softwarerepro2013/master/SoftwareInfrastructure.ipynb

Im Browser: IPython-Notebooks

One-Click Installation ist immer noch zu aufwändig?!

- Keine Installation, nach Anmeldung auf IPython Server steht im Browser die volle Python-Funktionalität zur Verfügung (inkl. Zugriff auf Hardware)
- Lehrmaterialien („Notebooks“) können gemeinsam im Unterricht entwickelt und sofort von Studierenden übernommen werden
- Durch Einbinden von Medien (Bilder, Audio, Video) können schnell interaktive Lernmaterialien erstellt werden
- Seit IPython Version 1.0 bequeme Exportmöglichkeiten (PDF, statisches HTML, Slideshow)
- Aber: Einrichtung eines Servers notwendig, innerhalb des Hochschulnetzwerks ein Sicherheitsrisiko

Im Blog: IPython Notebooks und Wakari

- „The IPython Notebook - Software Infrastructure for Reproducibility“
  http://nbviewer.ipython.org/urls/raw.github.com/ellisonbg/talk-softwarerepro2013/master/SoftwareInfrastructure.ipynb
- „IPython: Python at your fingertips“
  https://pycon-2012-notes.readthedocs.org/en/latest/ipython.html
- „Easily teach scientific computing with Wakari“
  http://continuum.io/blog/teaching-with-wakari

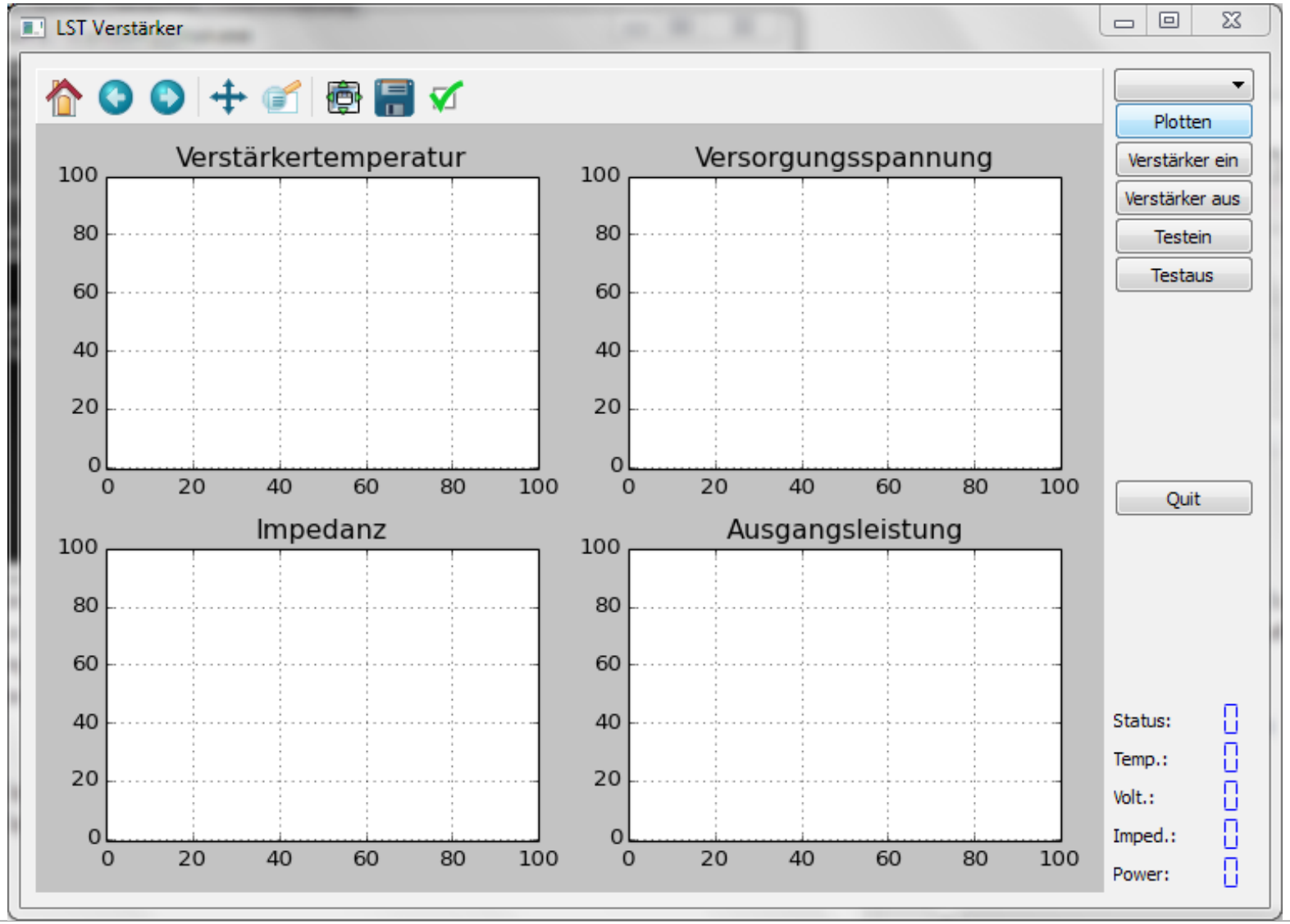
- Was ist das und woher bekomme ich es?
- Wissenschaftliche Anwendungen
- Python als Ersatz für Matlab?
- IPython Notebooks und Wakari
- **Python im Labor**
- Was gibt's noch?

(Studentische) Projekte im Labor Schaltungstechnik

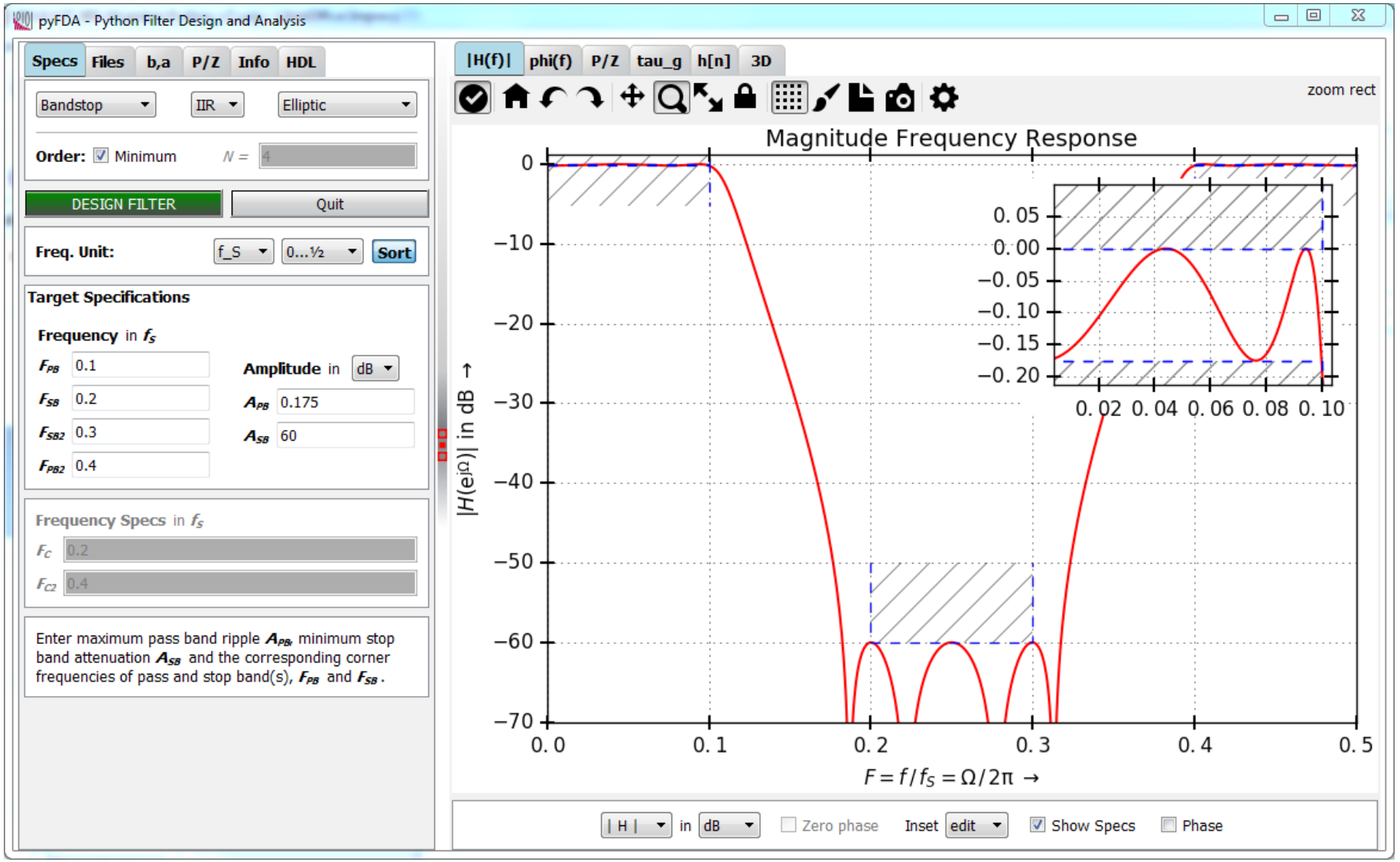
Python für DSP Simulation, Application Building, Messautomatisierung

- **Einfache FPGA-basierte ΣΔ-ADCs**: Entwicklung und Charakterisierung, Python für Messtechnik und Datenverarbeitung, ggf. Simulation
- **PyFDAS:** Python Filter Design, Analysis and Synthesis Tool als Weiterentwicklung von Mathworks fdatool [→ GitHub]
- **Datenlogger** mit „Analog Discovery“ (USB-Messmodul) [→ GitHub]
- **ASRC**: Simulation (Python) und Implementierung (Xilinx) effizienter Algorithmen auf FPGAs zur asychronen Sampleraten Konvertierung

Qt-GUI zur Steuerung des Digitalverstärkers



pyFDA für Analyse / Entwurf digitaler Filter



www.pyfda.org

- Was ist das und woher bekomme ich es?
- Wissenschaftliche Anwendungen
- Python als Ersatz für Matlab?
- IPython Notebooks und Wakari
- Python im Labor
- **Was gibt's noch?**

Literatur und Kurse

Videos

- MIT, OCW 6.189 „A Gentle Introduction to Programming Using Python“
- MIT, OCW 6.00 „Introduction to Computer Science and Programming“ [Python]
- Khan Academy, ”Introduction to Computer Science / Python Programming“
- pyvideo.org, Youtube, ...

Online

- Computer Science Circles, „Grundlagen des Programmierens mit Python“ - interaktiv!
- scipy-lectures.github.io/ - Kurs speziell zu NumPy / SciPy
- github.com/numpy-tutorial/PyCon.DE13 - Numpy Tutorial (IPython Notebooks)
- www.python-kurs.eu/ - Kurse (auf deutsch) zu Python 2, 3 und NumPy
- learnpythonthehardway.org - „Hard way“ = selbst programmieren“
- docs.python.org - gut, aber nicht unbedingt für Einsteiger

Bücher

- Mark Summerfield, „Rapid GUI Programming with Python and Qt„
  sehr gut erklärt, viele Beispiele, exzellente Schnell-Einführung in Python für Quereinsteiger, Demo-Kapitel „Introduction to GUI-Programming“
- Hans-Petter Langtangen, „A Primer on Scientific Programming with Python“,
  sowohl Matlab-mäßige als auch OO Programmiertechniken für (angehende) Ingenieure und Wissenschaftler. Dozentenexemplare erhältlich (Springer)
- Cyrille Rossant,
  „Learning Ipython for Interactive Computing and Data Visualization“, Packt Publishing, 28 € (inkl. eBook)
- RRZN, „Python“, für 5,- konkurrenzlos günstig, aber eher als Nachschlagewerk geeignet.