CSC 7700: Scientific Computing

Module A: Basic Skills

Lecture 4: Advanced 1D and 2D Visualization

Dr Frank Löffler

Center for Computation and Technology
Louisiana State University, Baton Rouge, LA

Fri, Sep 13 2013

Overview

1D and/or 2D visualization

- Publications
- Debugging
- Quickly put together
- Does not require fast network connections if done remotely

3D visualization

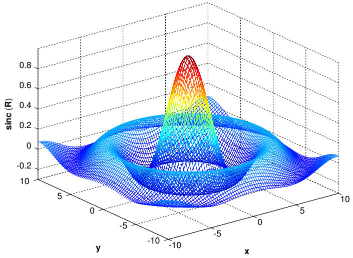
- Web / PR (movies)
- Detailed Debugging
- Comparably slower to achieve than 1D/2D plots

1D/2D tools

Common (biased):

- Gnuplot
- Matplotlib
- Supermongo (*)
- Tecplot (*)
- GNU Octave
- Grace
- Maple/Mathematica (*)
- Matlab (*)
- Maxima/Sage

(*): license necessary



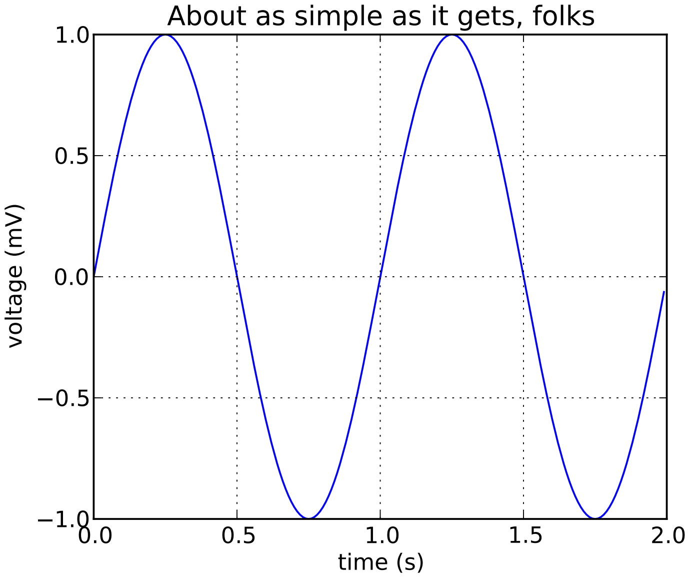
Matplotlib Overview

Matplotlib

- Python
  - programming language
  - high-level
  - large and comprehensive standard library
  - often used as scripting language
  - available for a wide range of OSs
- NumPy extension
  - support for large, multi-dimensional arrays and matrices
  - high-level math functions
- SciPy extension
  - linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers, ...
  - data structures provided by Numpy
- Interactive: pylab (closely resembles Matlab interface)

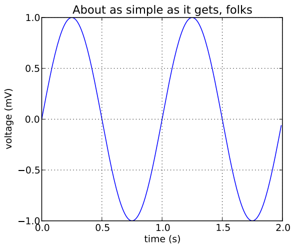
Matplotlib Examples

Hello World: plot sin(x)

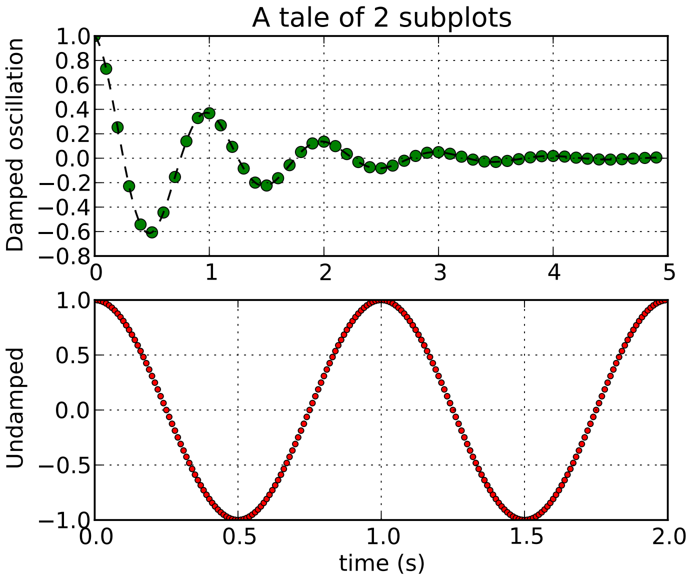


Hello World: plot sin(x)

```
from pylab import *

t = arange(0.0, 2.0, 0.01)
s = sin(2*pi*t)
plot(t, s, linewidth=1.0)

xlabel('time (s)')
ylabel('voltage (mV)')
title('About as simple as it gets, folks')
grid(True)
show()
```



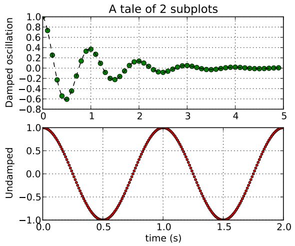
Subplots



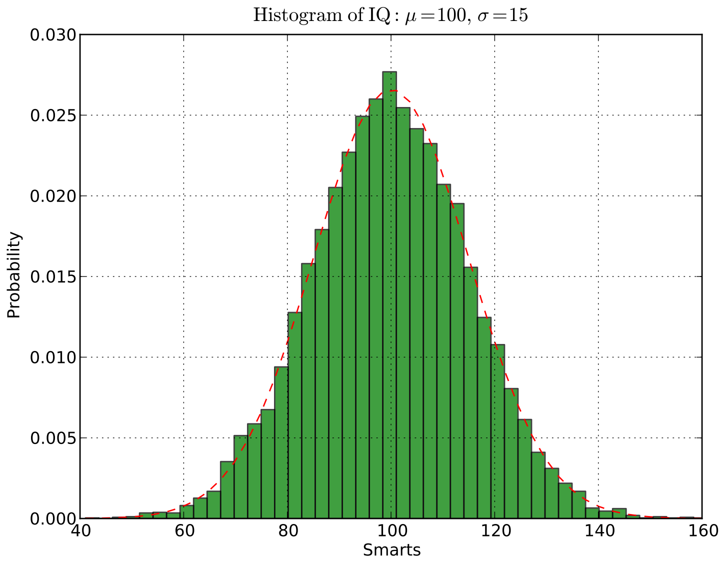
A tale of 2 subplots
Damped oscillation
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
1
2
3
4
5
Undamped
1.0
0.5
0.0
−0.5
−1.0
0.0
0.5
1.0
1.5
2.0
time (s)

Subplots

```
#!/usr/bin/env python
from pylab import *

def f(t):
    s1 = cos(2*pi*t)
    e1 = exp(-t)
    return multiply(s1,e1)

t1 = arange(0.0, 5.0, 0.1)
t2 = arange(0.0, 5.0, 0.02)
t3 = arange(0.0, 2.0, 0.01)

subplot(211)
l = plot(t1, f(t1), 'bo', t2, f(t2), 'k--', markerfacecolor='green')
grid(True)
title('A tale of 2 subplots')
ylabel('Damped oscillation')

subplot(212)
plot(t3, cos(2*pi*t3), 'r.')
grid(True)
xlabel('time (s)')
ylabel('Undamped')
show()
```

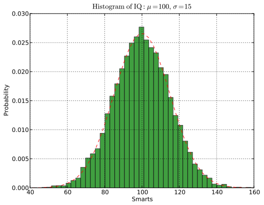


Histograms

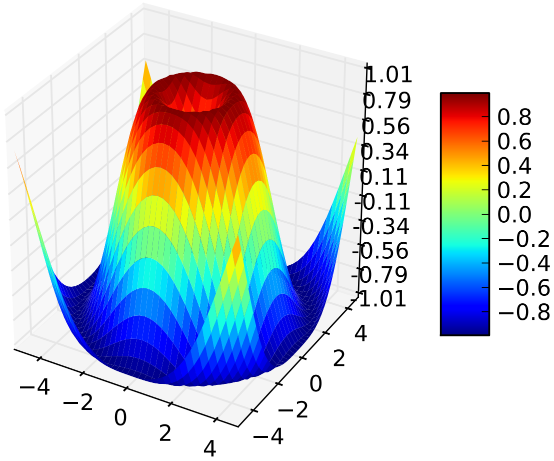

Histogram of IQ : $\mu = 100$, $\sigma = 15$
Probability
Smarts
0.030
0.025
0.020
0.015
0.010
0.005
0.000
40
60
80
100
120
140
160

Histograms

```
#!/usr/bin/env python
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma*np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, normed=1,
                            facecolor='green', alpha=0.75)

# add a 'best fit' line
y = mlab.normpdf( bins, mu, sigma)
l = plt.plot(bins, y, 'r--', linewidth=1)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title(r'$\mathrm{Histogram\ of\ IQ:}\ \mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)

plt.show()
```

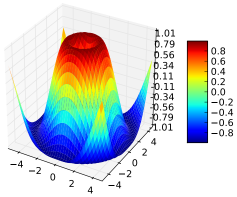


Surfaces

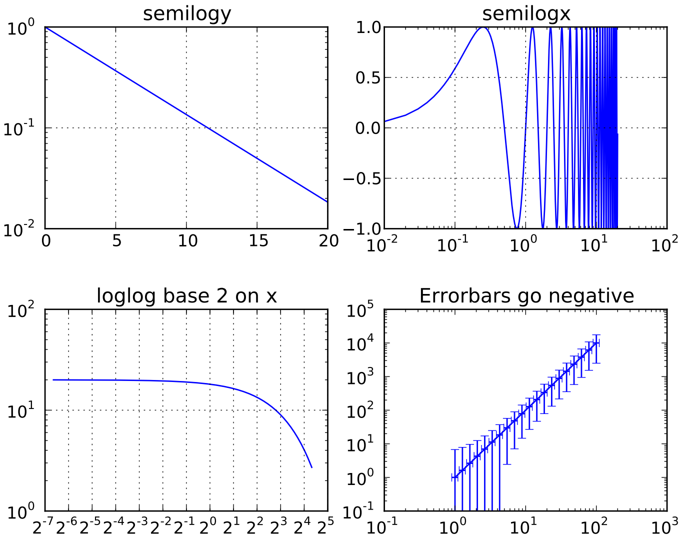

1.01
0.79
0.56
0.34
0.11
-0.11
-0.34
-0.56
-0.79
-1.01
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−4
−2
0
2
4
4
2
0
−2
−4

Surfaces

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure()
ax = fig.gca(projection='3d')
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.jet,
        linewidth=0, antialiased=False)
ax.set_zlim(-1.01, 1.01)

ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```

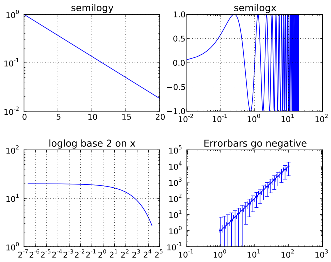


Log plots



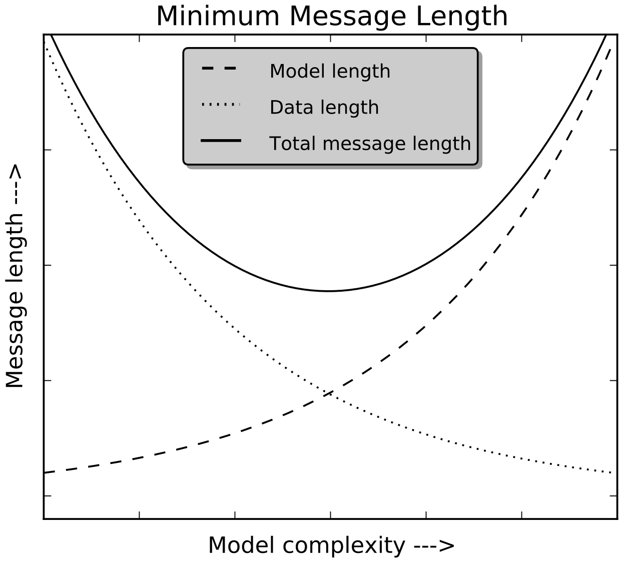
Log plots

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

plt.subplots_adjust(hspace=0.4)
t = np.arange(0.01, 20.0, 0.01)

# log y axis
plt.subplot(221)
plt.semilogy(t, np.exp(-t/5.0))
plt.title('semilogy')
plt.grid(True)

# log x axis
plt.subplot(222)
plt.semilogx(t, np.sin(2*np.pi*t))
plt.title('semilogx')
plt.grid(True)

# log x and y axis
plt.subplot(223)
plt.loglog(t, 20*np.exp(-t/10.0), basex=2)
plt.grid(True)
plt.title('loglog base 2 on x')
```



```
# with errorbars: clip non-positive values
ax = plt.subplot(224)
ax.set_xscale("log", nonposx='clip')
ax.set_yscale("log", nonposy='clip')

x = 10.0**np.linspace(0.0, 2.0, 20)
y = x**2.0
plt.errorbar(x, y, xerr=0.1*x, yerr=5.0+0.75*y)
ax.set_ylim(ymin=0.1)
ax.set_title('Errorbars go negative')

plt.show()
```

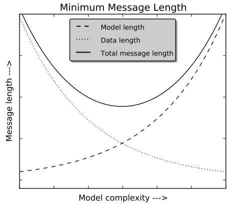
Legends



Minimum Message Length
Model length
Data length
Total message length
Message length --->
Model complexity --->

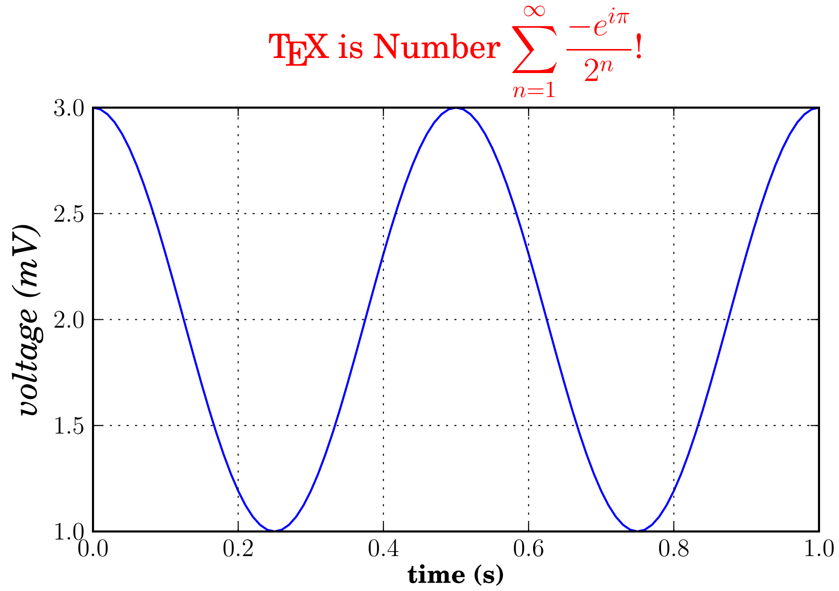
Legends

```python
#!/usr/bin/env python

import numpy as np
import matplotlib.pyplot as plt

a = np.arange(0,3,.02)
b = np.arange(0,3,.02)
c = np.exp(a)
d = c[::-1]

ax = plt.subplot(111)
plt.plot(a,c,'k--',a,d,'k:',a,c+d,'k')
plt.legend(('Model length', 'Data length',
            'Total message length'),
           'upper center', shadow=True,
            fancybox=True)
plt.ylim([-1,20])
plt.grid(False)
plt.xlabel('Model complexity --->')
plt.ylabel('Message length --->')
plt.title('Minimum Message Length')

plt.setp(plt.gca(), 'yticklabels', [])
plt.setp(plt.gca(), 'xticklabels', [])
```



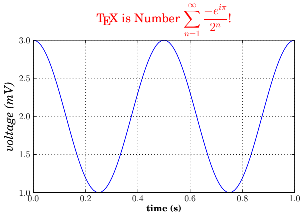
```python
# set some legend properties.  All the code below is optional.
# The defaults are usually sensible but if you need more control,
# this shows you how
leg = plt.gca().get_legend()
ltext  = leg.get_texts()  # all the text.Text instance
llines = leg.get_lines()  # all the lines.Line2D instance
frame  = leg.get_frame()  # the surrounding patch.Rectangle inst.

# see text.Text, lines.Line2D, and patches.Rectangle for more
# info on the settable properties of lines, text, and rectangles
frame.set_facecolor('0.80')      # set the frame face color
plt.setp(ltext, fontsize='small') # the legend text fontsize
plt.setp(llines, linewidth=1.5)   # the legend linewidth
#leg.draw_frame(False)           # don't draw the legend frame
plt.show()
```

Native TeX rendering



Native TeX rendering

```
#!/usr/bin/env python
from matplotlib import rc
from numpy import arange, cos, pi
from matplotlib.pyplot import figure, axes, plot, \
     xlabel, ylabel, title, grid, savefig, show

# comment this if you have trouble using LaTeX
rc('text', usetex=True)
rc('font', family='serif')
figure(1, figsize=(6,4))
ax = axes([0.1, 0.1, 0.8, 0.7])
t = arange(0.0, 1.0+0.01, 0.01)
s = cos(2*2*pi*t)+2
plot(t, s)

xlabel(r'\textbf{time (s)}')
ylabel(r'\textit{voltage (mV)}',fontsize=16)
title(r"\TeX\ is Number $\displaystyle\sum_{n=1}^\infty\frac{-e^{i\pi}}{2^n}$!",
      fontsize=16, color='r')
grid(True)
savefig('tex_demo')

show()
```



File Input

Text files

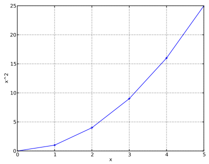
data.txt:

```
# Some data points
0 0
1 1
2 4
3 9
4 16
5 25
```

Text files - loadtxt()

Note: Each row in the text file must have the same number of values

```
from pylab import *

Fx, Fy = loadtxt("data.txt", comments="#",
                  usecols=(0,1), unpack=True)

plot(Fx, Fy, marker='+', linestyle='-', linewidth=1.0)

xlabel('x')
ylabel('x^2')
grid(True)
show()
```



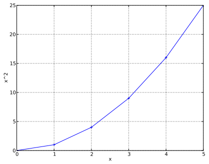
Text files with missing columns

data2.txt

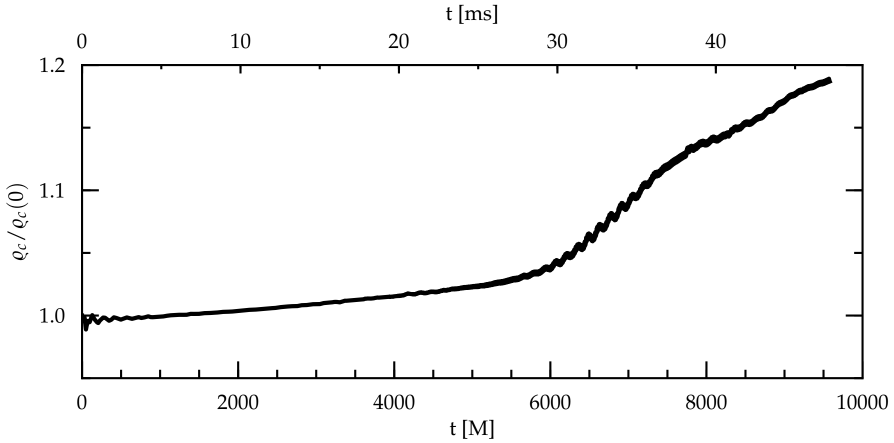
```
# Some data points
0 0
1 1
2 4
3 9
4 16
4.5
5 25
6
```

Text files - genfromtxt()

```
from pylab import *

Fx, Fy = genfromtxt("data2.txt", comments="#",
                    usecols=(0,1), unpack=True,
                    invalid_raise=False)

plot(Fx, Fy, marker='+', linestyle='-', linewidth=1.0)

xlabel('x')
ylabel('x^2')
grid(True)
show()
```

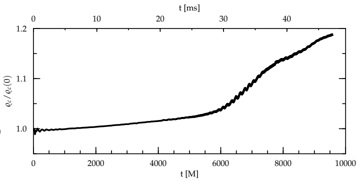


Production example



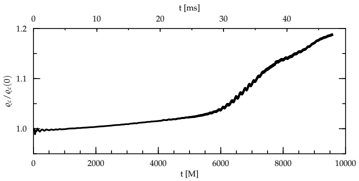
Production example

```
#!/usr/bin/python

# Plot evolution of central density
from plot_defaults import *

# load data
Fx, Fy = numpy.loadtxt("hydrobase::rho.maximum.asc.bz2",
                       comments="#", usecols=(1,2), unpack=True)

# plot basics
fig = plt.figure(figsize=(6, 3))
fig.subplots_adjust(top=0.85, bottom=0.16, left=0.11,right=0.97)
ax = fig.add_subplot(1,1,1)

# plot
ax.plot(Fx, Fy/Fy[0], linestyle='-', color='black')

ax.set_xlabel(r't [M]')
ax.xaxis.set_minor_locator(mticker.AutoMinorLocator())
ax.xaxis.grid(False)
ax2 = ax.twiny()
ax2.set_xlabel(r't [ms]')
ax2.set_xlim((ax.get_xlim()[0]/M_to_ms, ax.get_xlim()[1]/M_to_ms))
ax2.xaxis.set_minor_locator(mticker.AutoMinorLocator())
ax.set_ylabel(r'$\varrho_c/\varrho_c(0)$')
ax.yaxis.set_major_locator(mticker.MaxNLocator(4))
ax.yaxis.set_minor_locator(mticker.AutoMinorLocator())
ax.yaxis.grid(False)
set_tick_sizes(ax, 8, 4)

#plt.show()
plt.savefig('rho_max.pdf')
```



Production example

```
#!/usr/bin/python
import matplotlib.pyplot as plt
import matplotlib.ticker as mticker
import numpy
from matplotlib import rc
fontsize  = 10
linewidth = 2
# comment the following line in case you have trouble with LaTeX
rc('text', usetex=True)
rc('font', family='serif'); rc('font', serif='palatino');
rc('font', weight='bolder'); rc('mathtext', default='sf')
rc("lines", markeredgewidth=1)
rc("lines", linewidth=linewidth)
rc('axes', labelsize=fontsize); rc("axes", linewidth=(linewidth+1)//2)
rc('xtick', labelsize=fontsize); rc('ytick', labelsize=fontsize)
rc('legend', fontsize=fontsize)
rc('xtick.major', pad=8); rc('ytick.major', pad=8)

def set_tick_sizes(ax, major, minor):
    for l in ax.get_xticklines() + ax.get_yticklines():
        l.set_markersize(major)
    for tick in ax.xaxis.get_minor_ticks() + ax.yaxis.get_minor_ticks():
        tick.tick1line.set_markersize(minor); tick.tick2line.set_markersize(minor)
    ax.xaxis.LABELPAD       = 10.
    ax.xaxis.OFFSETTEXTPAD = 10.

# constants, in SI
G = 6.673e-11         # m^3/(kg s^2)
c = 299792458         # m/s
M_sol = 1.98892e30   # kg
# convertion factors
M_to_ms = 1./(1000*M_sol*G/(c*c*c))
```

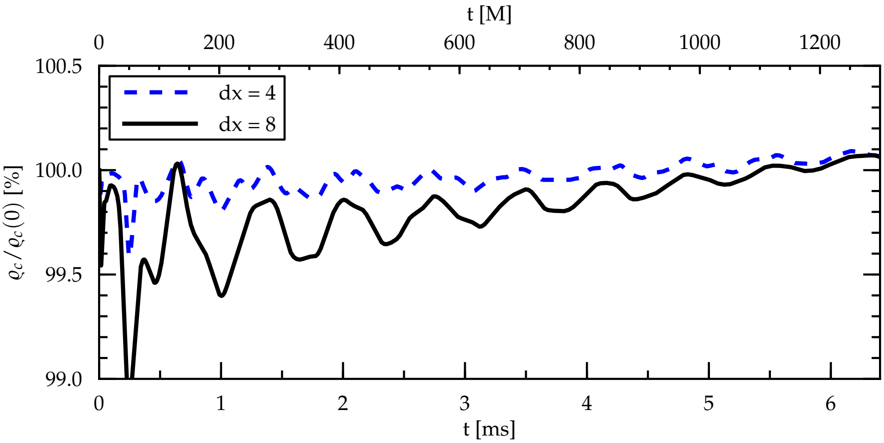


Interactive Usage

```
$ ipython --pylab
Python 2.7.3 (default, Jan  2 2013, 13:56:14)
Type "copyright", "credits" or "license" for more information.

IPython 0.13.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

Welcome to pylab, a matplotlib-based Python environment [backend: TkAgg].
For more information, type 'help(pylab)'.

In [1]: x = randn(10000)

In [2]: hist(x,100)
```

Course Work

Use template and data [1] to create this plot, send Python source and pdf-file.



[1] https://svn.cct.lsu.edu/repos/courses/sci-comp-2013-public/coursework/A4/

Course Work

Things to change from given example:

- Plot data from two files in one plot, using line styles as in example (blue and dotted, black and solid)
- Switch the two *x*-axes and make sure they line up properly
- Add legend (key) as shown (length of lines not important)
- Show percentage on *y* axis instead of raw quotient
- Add [%] to *y*-label
- Show ticks on y axes as in example
- Commit: python script, resulting pdf file, no full report (coursework/A4)

Due: Fri, Sep 20th 2013