Lecture 3: Data Plotting and Fitting

Hendrik Weimer

Institute for Theoretical Physics, Leibniz University Hannover

Quantum Physics with Python, 25 April 2016

Outline of the course

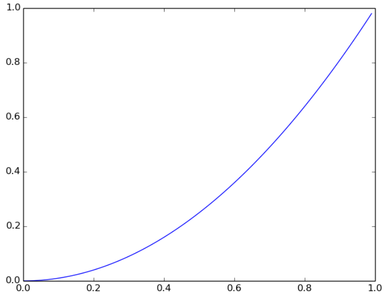
1. Introduction to Python
2. SciPy/NumPy packages
3. **Plotting and fitting**
4. QuTiP: states and operators
5. Ground state problems
6. Non-equilibrium dynamics: quantum quenches
7. Quantum master equations
8. Generation of squeezed states
9. Quantum computing
10. Grover's algorithm and quantum machine learning
11. Student presentations

The matplotlib package

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

x = np.arange(0,1,0.01)
y = x*x
plt.plot(x, y)
plt.show()
```

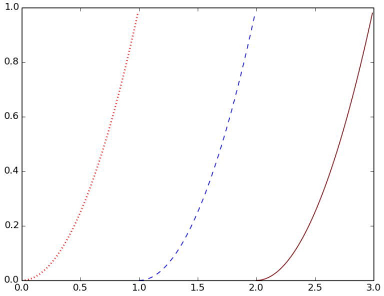
Output:



Colors and styles

```
plt.plot(x, y, 'r:', linewidth=2)
plt.plot(x+1, y, 'b--')
plt.plot(x+2, y, '#800000')
plt.show()
```

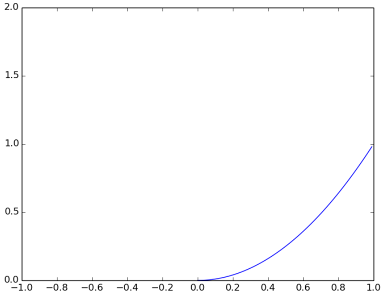
Output:



Plot ranges and axes

```
plt.plot(x, y)
plt.axis([-1, 1, 0, 2])
plt.xticks(np.linspace(-1, 1, 11))
plt.show()
```

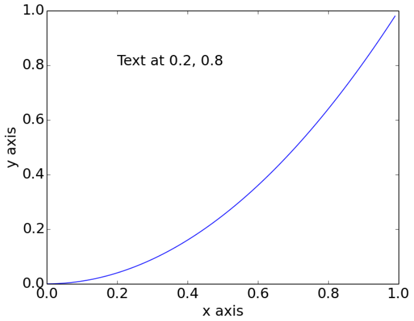
Output:



Labels

```
import matplotlib
matplotlib.rcParams['font.size'] = 18
plt.plot(x, y)
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.text(0.2, 0.8, 'Text at 0.2, 0.8')
plt.show()
```

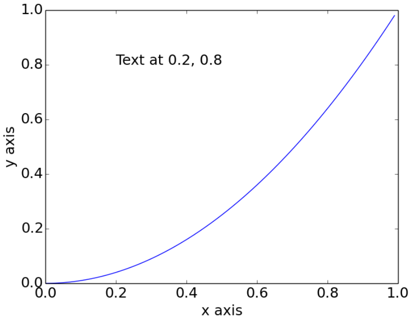
Output:

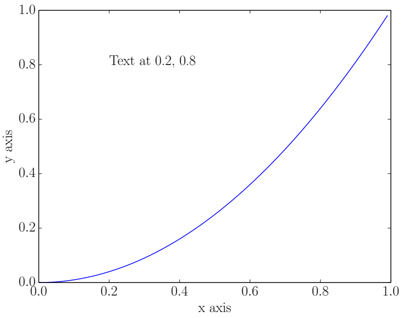


LaTeX

```
matplotlib.rc('text', usetex=True)
matplotlib.rc('font', family='serif')
matplotlib.rc('font', serif='cm')
```

Before and after:

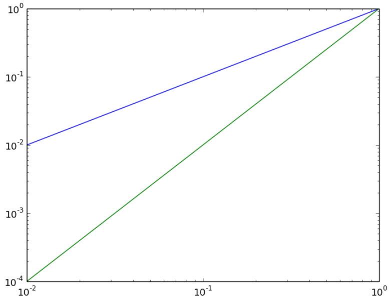




Logarithmic plots

```
plt.loglog(x, x)
plt.loglog(x, y)
plt.show()
```

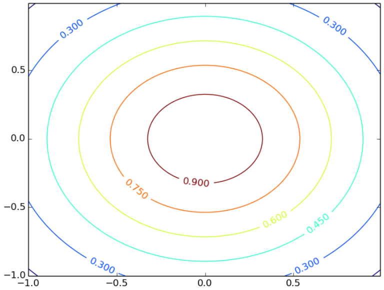
Output:



Contour plots

```
x = y = np.arange(-1,1,0.01)
xx, yy = np.meshgrid(x, y)
z = np.exp(-2*(xx**2+yy**2))
cs = plt.contour(x, y, z)
plt.clabel(cs)
plt.show()
```

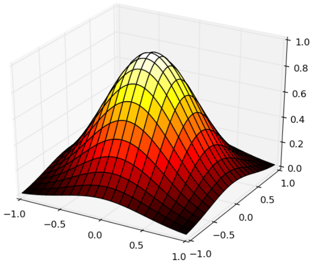
Output:



3D plots

```
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(xx, yy, z, cmap=cm.hot)
plt.show()
```

Output:



Save as file

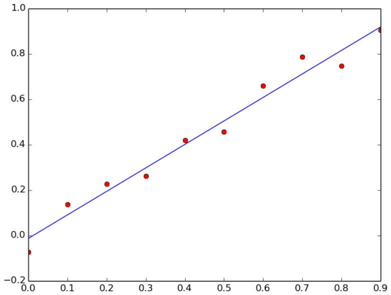
```
plt.savefig('foo.png')
plt.savefig('foo.pdf')
```

Other supported file formats: EPS, SVG, JPG, ...

Data fitting

```
from scipy import optimize
x = np.arange(0,1,0.1)
y = np.random.normal(x, 0.05)
fitfun = lambda x, a, b: a*x + b
fit = optimize.curve_fit(fitfun, x, y)
plt.plot(x, y, 'ro')
plt.plot(x, fitfun(x, fit[0][0], fit[0][1]))
plt.show()
```

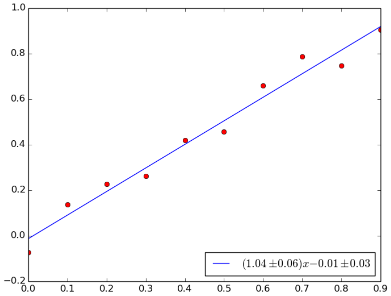
Output:



Parameter uncertainties

```
dfit = np.sqrt(np.diag(fit[1]))
flabel = r'$({:.2f}\pm {:.2f}) x {:.2f} \pm {:.2f}$'\
    .format(fit[0][0], dfit[0], fit[0][1], dfit[1])
plt.plot(x, y, 'ro')
plt.plot(x, fitfun(x, fit[0][0], fit[0][1]), label=flabel)
plt.legend(loc='lower right')
plt.show()
```

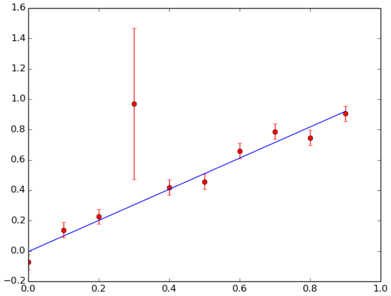
Output:



Dealing with noisy data

```
err = [0.05]*len(x)
y[3] = 0.97
err[3] = 1
fit = optimize.curve_fit(fitfun, x, y, sigma=err)
plt.errorbar(x, y, fmt='ro', yerr=err)
plt.plot(x, fitfun(x, fit[0][0], fit[0][1]))
plt.show()
```

Output:



Example: Laser excitation of atoms to Rydberg states

- Data from [R. Löw et al., Phys Rev. A **80**, 033422 (2009)]
- First column: dimensionless ratio of laser power and atom density (α)
- Second column: fraction of atoms in the Rydberg state (f_R)
- Third column: Standard deviation of f_R
- Theory prediction: $f_R = c\alpha^\nu$