Data visualization with matplotlib

Ashley DaSilva

Download presentation and sample data:
http://goo.gl/LCMIRD

Have you installed matplotlib and numpy?

Overview

- Straight from calculation or analysis to figures
- High-level control
- High quality
- Examples in the matplotlib cookbook

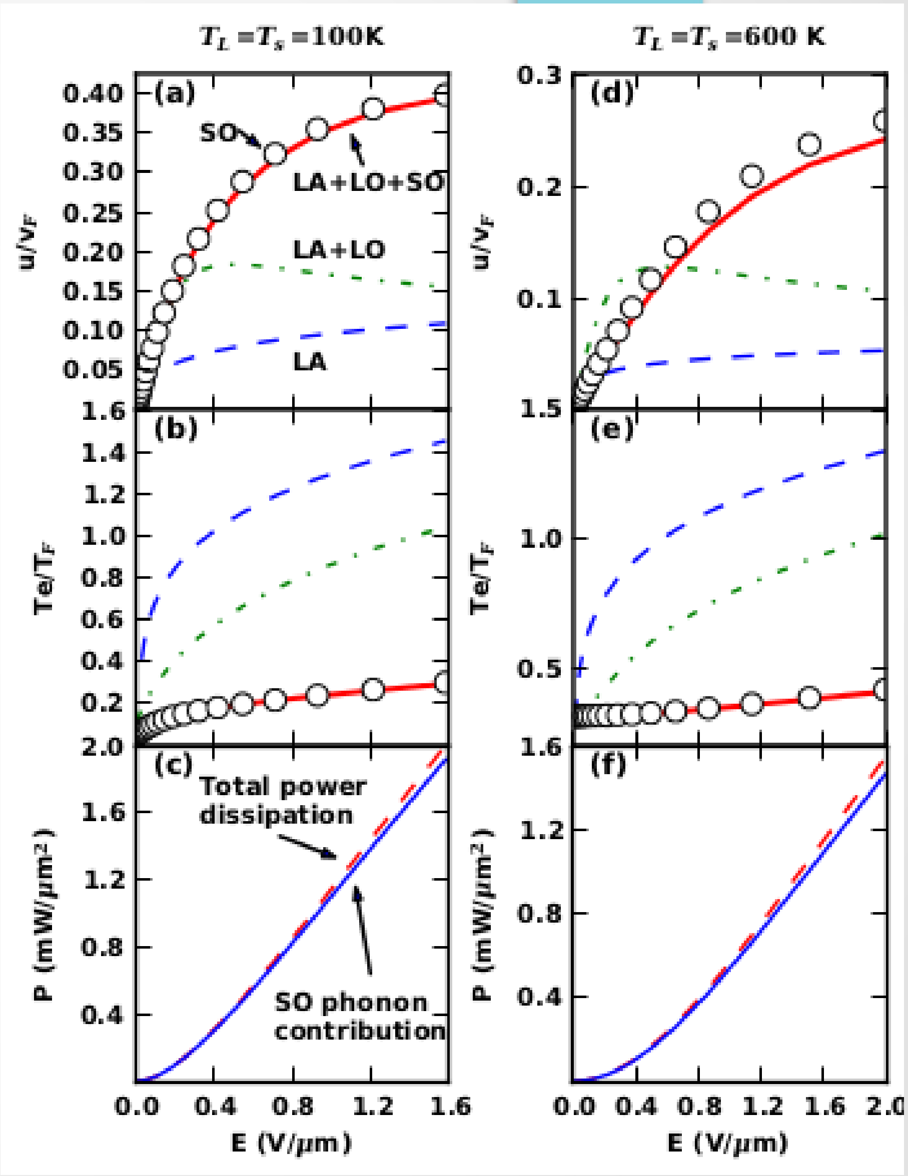


Figure from: DaSilva et al.
Phys. Rev. Lett. **104**, 236601 (2010)

What packages do I need?

- Matplotlib – includes plotting functionality through the pyplot module
- Numpy – numerical and linear algebra package (including arrays)
- Pylab – a separate module installed at the same time as matplotlib; provides both pyplot and numpy (as np)
- Ipython – interactive python environment
- Scipy – crucial for science!
  - Fourier transforms, integration, optimization, etc.

http://matplotlib.org/faq/usage_faq.html#matplotlib-pylab-and-pyplot-how-are-they-related

Interactive plotting with IPython

- Best thing about Ipython is interactive plotting!
- Use pylab option: $ ipython --pylab

```
ashley@ashley-laptop:~$ ipython --pylab
Python 2.7.7rc1 (default, May 21 2014, 11:15:30)
Type "copyright", "credits" or "license" for more information.

IPython 2.1.0 -- 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.
Using matplotlib backend: TkAgg

In [1]:
```

Loading data

- Many available modules:
  - Plain text: loadtxt
  - Numpy arrays: numpy.load
  - HDF datasets: h5py.File
  - Collaborate with matlab users: scipy.io.loadmat

Data and arrays in numpy

- Load data from ohm1.npy
- Experiment with numpy arrays and array slices

```
with open('ohm1.npy','r') as f:
    ohm1=numpy.load(f)

print ohm1
print ohm1[0,:]
print ohm1[1,:]
```

Simple plots with pylab

- Numpy included with pylab under the name np
- Matplotlib included with pylab under the name mpl

```
from pylab import *
```

```
plot(ohm1[0,:],ohm1[1,:],'.y-',lw=.75,label='First example')
xlabel('Voltage')
ylabel('Current')
title("Ohm's law")
legend(loc='lower right')
```

Line properties

- Simple combinations are recognized in a single string
  - Markers: dot (.), circle (o), triangle (v), square (s), etc.
  - Colors: black (k), blue (b), yellow (y), green (g), etc.
  - Line styles: solid (-), dashed (--), dash-dotted (-.), etc.
- Defaults are used to “fill in” missing info (e.g. 'or' is valid!)
- Note: the combination .- is a solid line with dot marker

```
plot(ohm1[0,:],ohm1[1,:],'.y-',lw=.75,label='First example')
xlabel('Voltage')
ylabel('Current')
title("Ohm's law")
legend(loc='lower right')
```

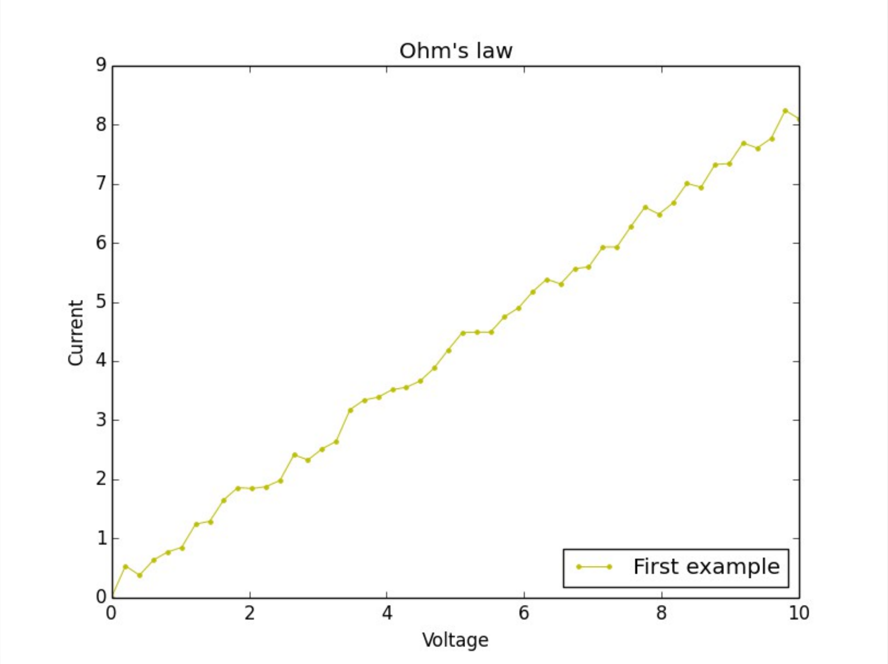
Line properties: lots of options!

- Linewidth: lw=1.5
- Color:
  - Shortcuts: c='m'
  - HTML codes: c='#736AFF'
  - RGB triplets: c=(.8,.5,.5)

```
plot(ohm1[0,:],ohm1[1,:],'.y-',lw=.75,label='First example')
xlabel('Voltage')
ylabel('Current')
title("Ohm's law")
legend(loc='lower right')
```

Line properties: lots of options!

- Linewidth: lw=1.5
- Color:
  - Shortcuts: c='m'
  - HTML codes: c='#736AFF'
  - RGB triplets: c=(.8,.5,.5)

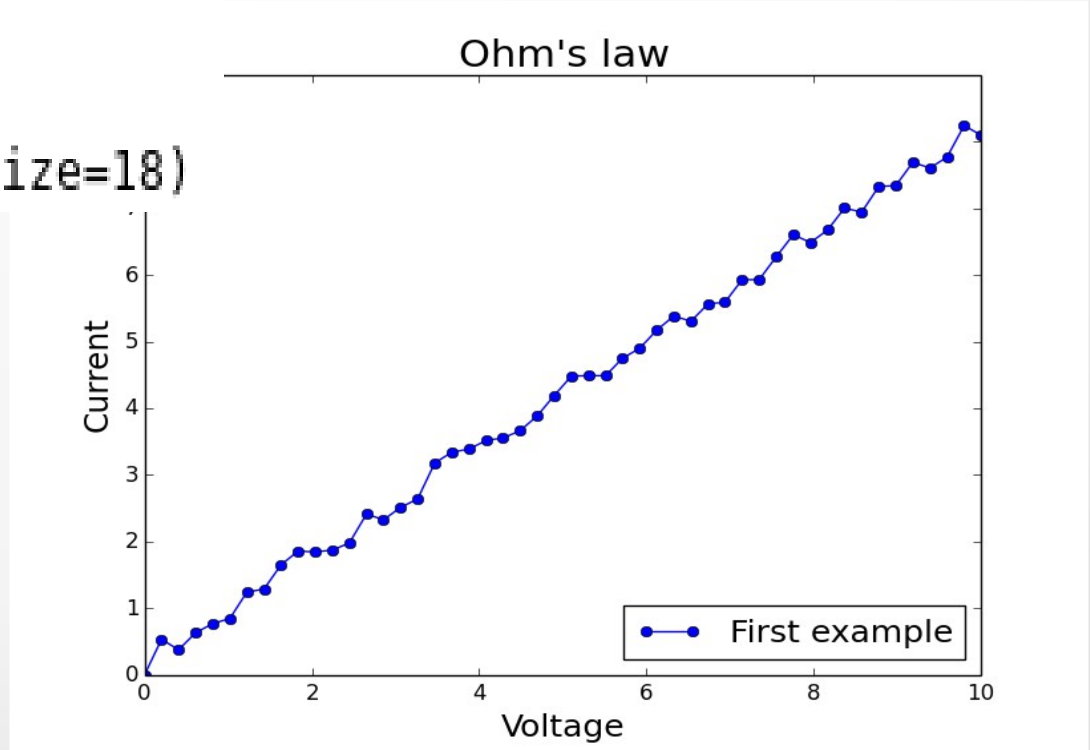



```
plot(ohm1[0,:],ohm1[1,:],'.y-',lw=.75,label='First example')
xlabel('Voltage')
ylabel('Current')
title("Ohm's law")
legend(loc='lower right')
```

Modifying the font sizes

- Optional argument for fontsize

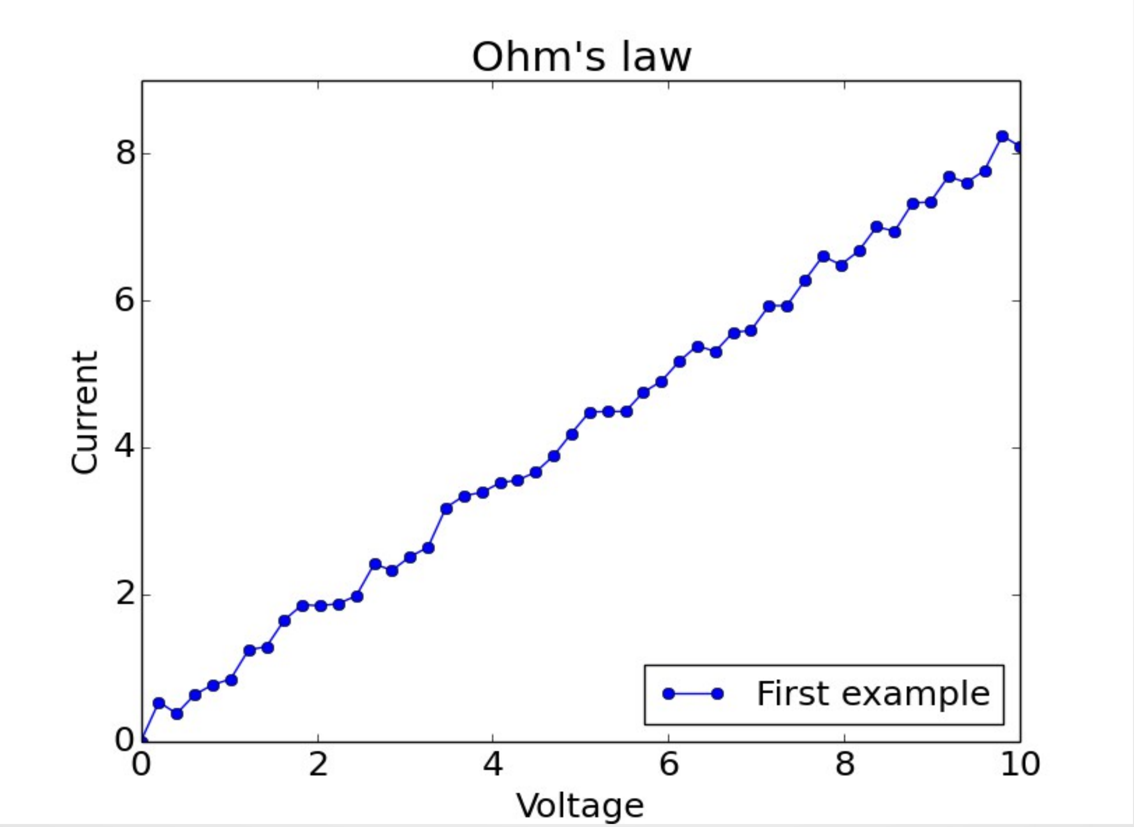
```
plot(ohm1[0,:],ohm1[1,:],'ob-',lw=1,
     label='First example')
xlabel('Voltage',fontsize=18)
ylabel('Current',fontsize=18)
title("Ohm's law",fontsize=22)
legend(loc='lower right',fontsize=18)
```



Control of the axes

- Control over font size and ticks used

```
xticks(fontsize=18)
yticks(np.arange(0,10,2),fontsize=18)
```



Trendlines

- Adding lines – not a problem
- Numpy polynomial fits: $y = c_0 + c_1 x + c_2 x^2 + \ldots + c_n x^n$

x-data y-data n

```
In [49]: polynomial = np.polyfit(ohm1[0,:],ohm1[1,:],1)

In [50]: print polynomial
[ 0.80378687  0.17198942]
```

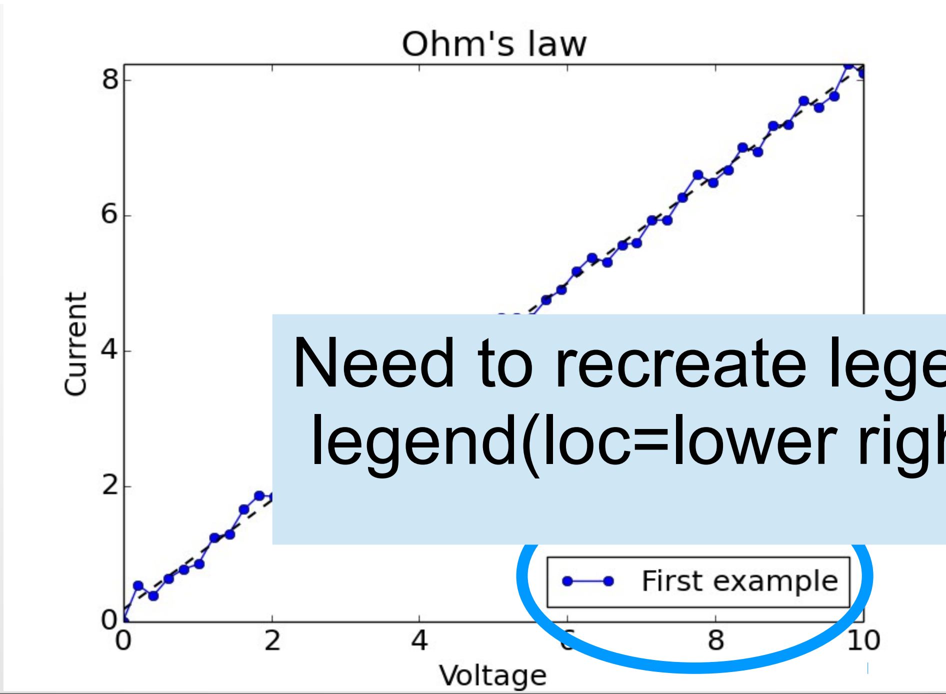
Trendlines

- Numpy polynomial evaluation: polyval

List of coefficients

x-points

```
pylab.plot(ohm1[0,:],numpy.polyval(polynomial,ohm1[0,:]),
        '--k',lw=1.5,label='linear fit')
```



No new legend item?

Need to recreate legend!
legend(loc=lower right, fontsize=18)

Easy subplots

- Set size, figure number

```
figure(1,(5,9))
```

- Pylab provides subplot grids
  - Easy to use
  - But not so versatile

Index of subplot axes

```
subplot(311)
```

Number of rows

Number of columns

Projectile motion example

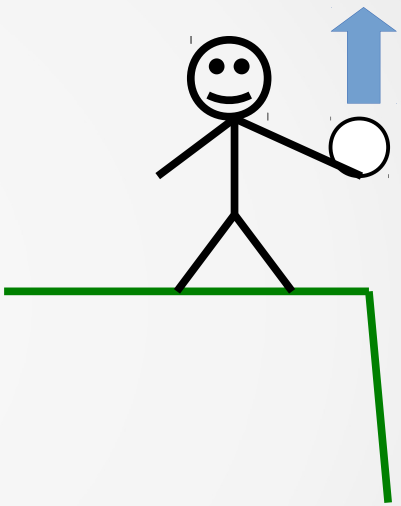
- Example of projectile motion
  - Gravitational acceleration is constant: a = -9.8 m/s^2
  - Subplots of acceleration, velocity, and position vs time

$$v = v_0 + at$$
$$x = x_0 + v_0 t + \frac{1}{2}at^2$$

Projectile motion example

- Download data: projectile.npy

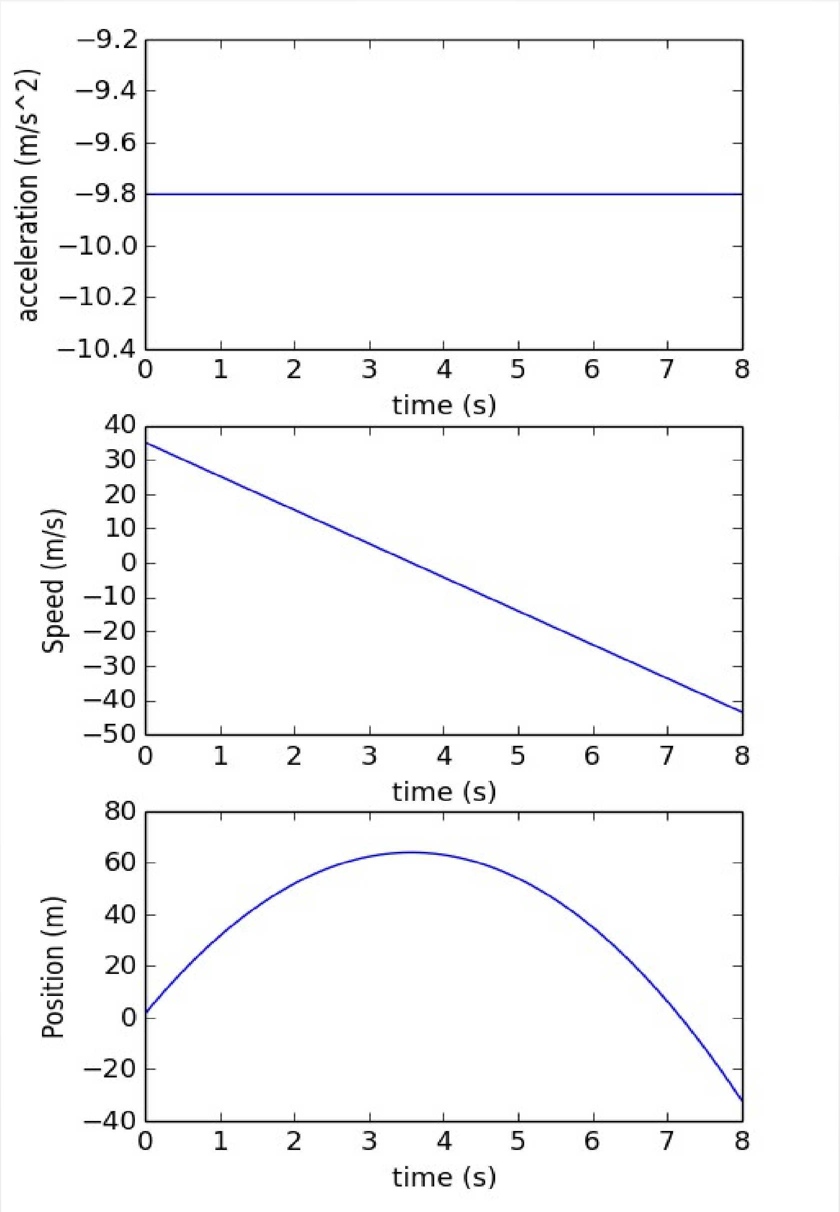
```
with open('projectile.npy','r') as f:
    data=np.load(f)

t=data[0,:]
v=data[1,:]
x=data[2,:]
a=-9.8*np.ones(len(t))
```



- Set up figure `figure(1,(5,9))`
- plot is also a method of the subplot object

```
aplt=subplot(311)
aplt.plot(data[0,:],accel*np.ones(len(t)))
show()
```

Sample solution

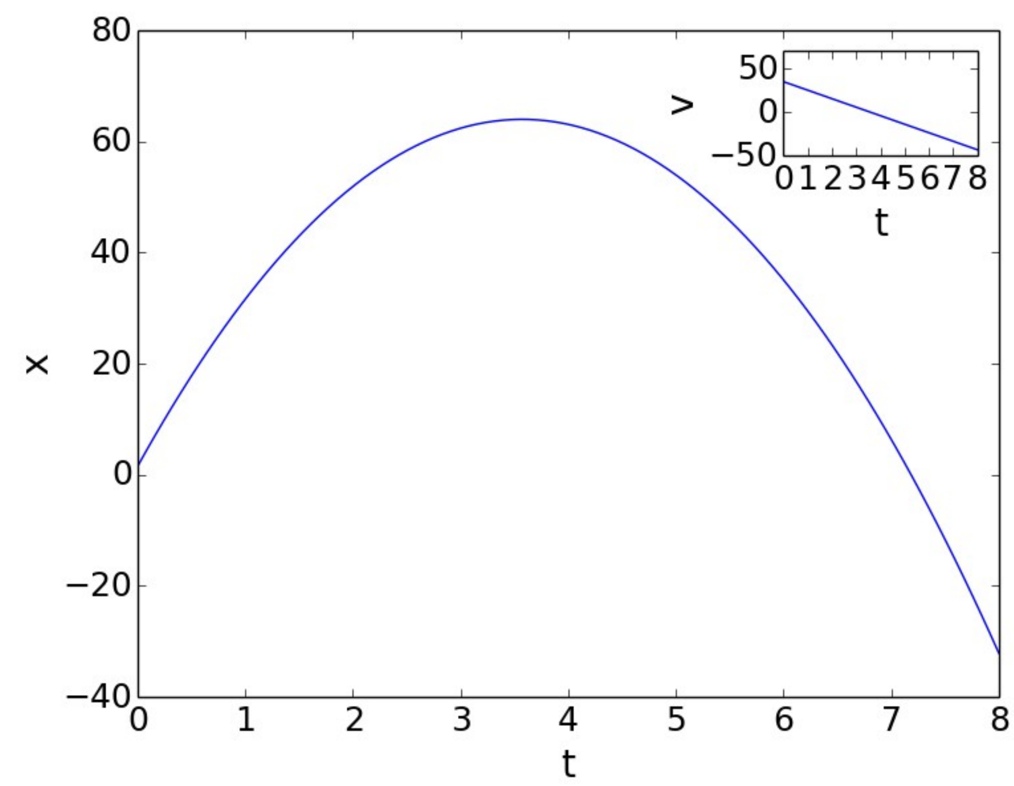


Subplots with matplotlib axes

- Why?
  - Different sized subplots
  - Shared axis
  - Insets
- How?
  - Specify a rectangle: [x, y, dx, dy]
  - Normalized units (0=0% of figure width, 1=100% figure width)
  - Get current axes: gca()

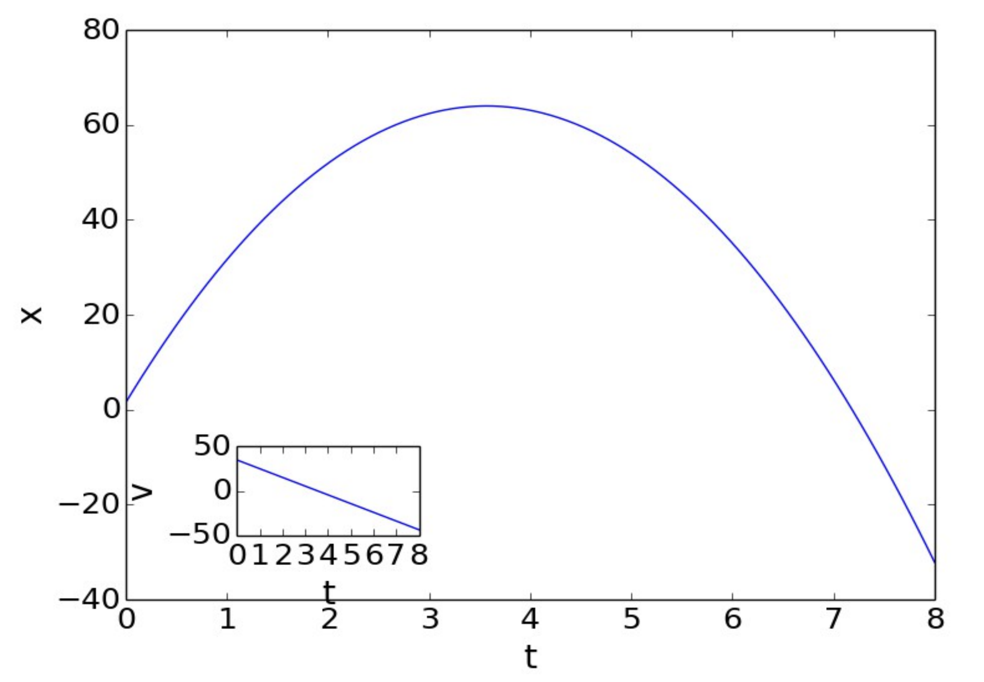
Subplots with matplotlib axes

```
figure(2)

main=axes([.15,.15,.8,.8])
plot(t,x)
xlabel('t',fontsize=20)
ylabel('x',fontsize=20)
xticks(fontsize=18)
yticks(fontsize=18)

inset=axes([.75,.8,.18,.125])
plot(t,v)
xlabel('t',fontsize=20)
ylabel('v',fontsize=20)
xticks(fontsize=18)
yticks(arange(-50,100,50),fontsize=18)
```



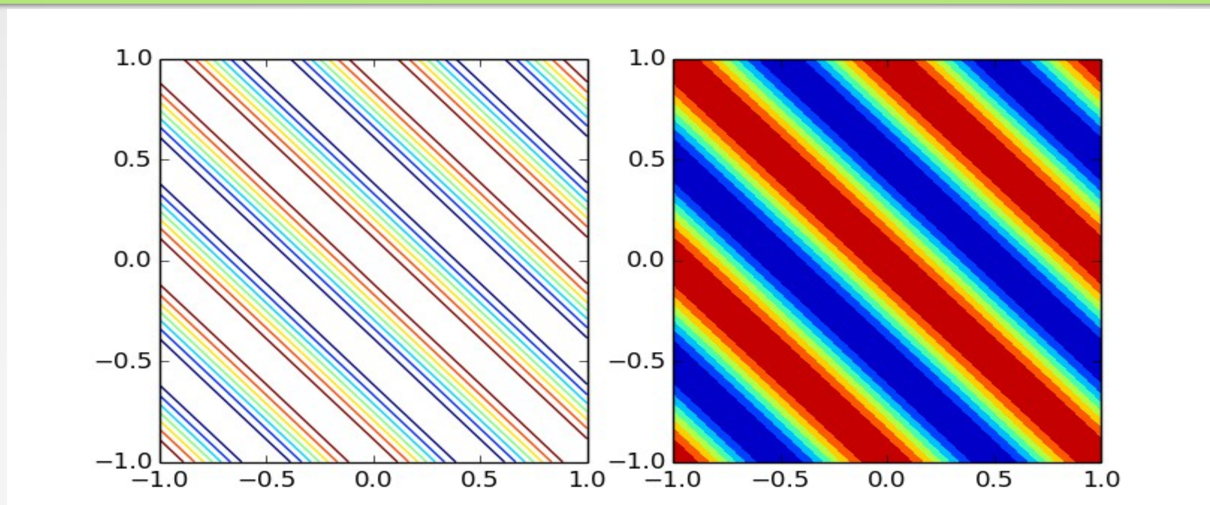
Subplots with matplotlib axes

- Clear figure with clf()
- Clear axes with cla()

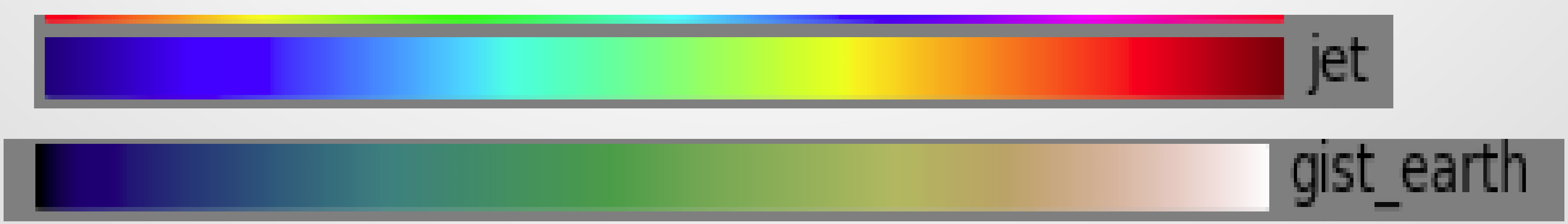



```
# try again
clf()
main=axes([.15,.15,.8,.8])
plot(t,x)
xlabel('t',fontsize=20)
ylabel('x',fontsize=20)
xticks(fontsize=18)
yticks(fontsize=18)

inset=axes([.26,.24,.18,.125])
plot(t,v)
xlabel('t',fontsize=20)
ylabel('v',fontsize=20)
xticks(fontsize=18)
yticks(arange(-50,100,50),fontsize=18)
```

Contour plots



- Filled or empty
- Built in color maps in matplotlib.cm (see cookbook)
- Or make your own
- Default is “jet”

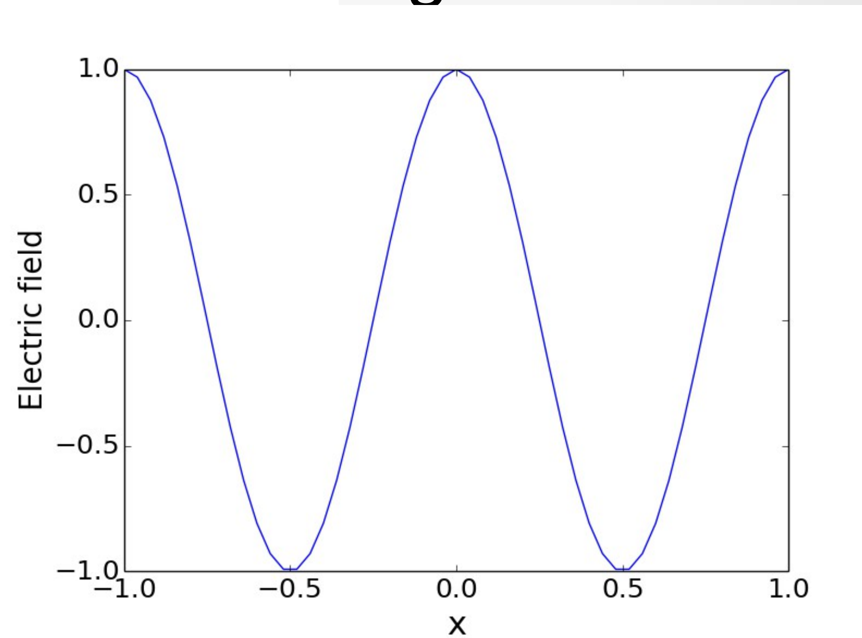


Contour plots: electric field

- Electric field is a wave
- Emanates from a source (e.g. antennas, flashlight, ...)

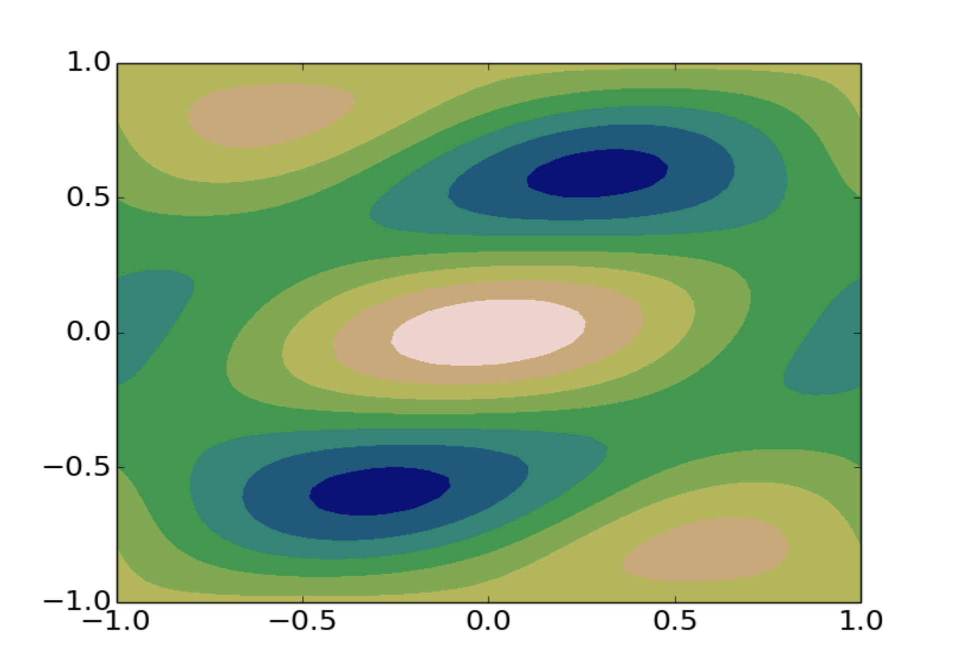
```
x=np.linspace(-1,1,51)
y=np.linspace(-1,1,53)
E=np.zeros((len(x),len(y)))

k=[1,1]
for i in range(len(x)):
    for j in range(len(y)):
        E[i,j]=cos(2*pi*(k[0]*x[i]+

figure(1,(8,4))
subplot(121)
contour(x,y,E.T)

subplot(122)
contourf(x,y,E.T)
```

Wave vector = inverse of wavelength



Contour plots: wave interference

- Two or more waves of different wavelength will interfere
- Try it:
  - Reset E to zeros
  - Use loop over k's




```
E=np.zeros((len(x),len(y)))
for k in [[0,1],[.5,-1],[.5,.5]]:
    for i in range(len(x)):
        for j in range(len(y)):
            E[i,j]=E[i,j]+cos(2*pi*(k[0]*x[i]+k[1]*y[j]))

contourf(x,y,E.T,cmap=cm.gist_earth)
xticks(fontsize=18)
yticks(fontsize=18)
```

Going further...

- Surface plots
- Text
- LaTeX (pretty math)
- Arrows
- Photos
- “Zooming in”
- Animation