MATPLOTLIB

Python for computational science

28 - 30 May 2012

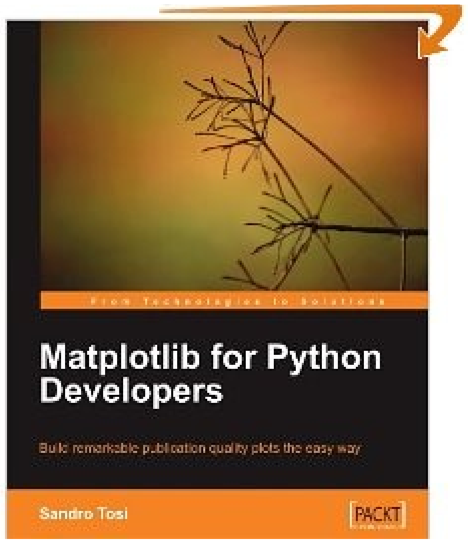
CINECA

m.cestari@cineca.it

Bibliography

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

Matplotlib for Python developers (Sandro Tosi, Packt Publishing Ltd., 2009)


From Technologies to Solutions
Matplotlib for Python Developers
Build remarkable publication quality plots the easy way
Sandro Tosi
PACKT

Introduction (1)

- plotting the data gives us **visual feedback** in the working process

- Typical workflow:
  - ➔ write a python program to parse data
  - ➔ pass the parsed data to **gnuplot** to plot the results

- with **Matplotlib** we can achieve the same result in a single script and with more flexibility

Introduction (2)

Matplotlib:

- makes use of Numpy to provide good performance with large data arrays
- allows **publication quality** plots
- allows to make plots easily
- since it's a Python module can be easily integrated in a Python program

Module import

Let us be consistent with the official documentation

```
$ (i)python
>>> import matplotlib.pyplot as plt
```

Matplotlib 1st example

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> plt.interactive('on') # set interactive
                          # no need with Ipython
>>> x = np.arange(0,7,0.00001)
>>> plt.plot(x,x**3) # x,y values of the plot
[<matplotlib.lines.Line2D object at 0xa1750cc>]
>>> plt.show()
```

Matplotlib: multiple line plot

```
>>> x = np.arange(0,7,0.00001)
>>> plt.plot(x,x**3)
>>> plt.plot(x,x**2)
>>> plt.show()
```

Or by passing multiple (x,y) arguments to the plot function

```
>>> x = np.arange(0,7,0.00001)
>>> plt.plot(x,x**3, x,x**2)
>>> plt.show()
```

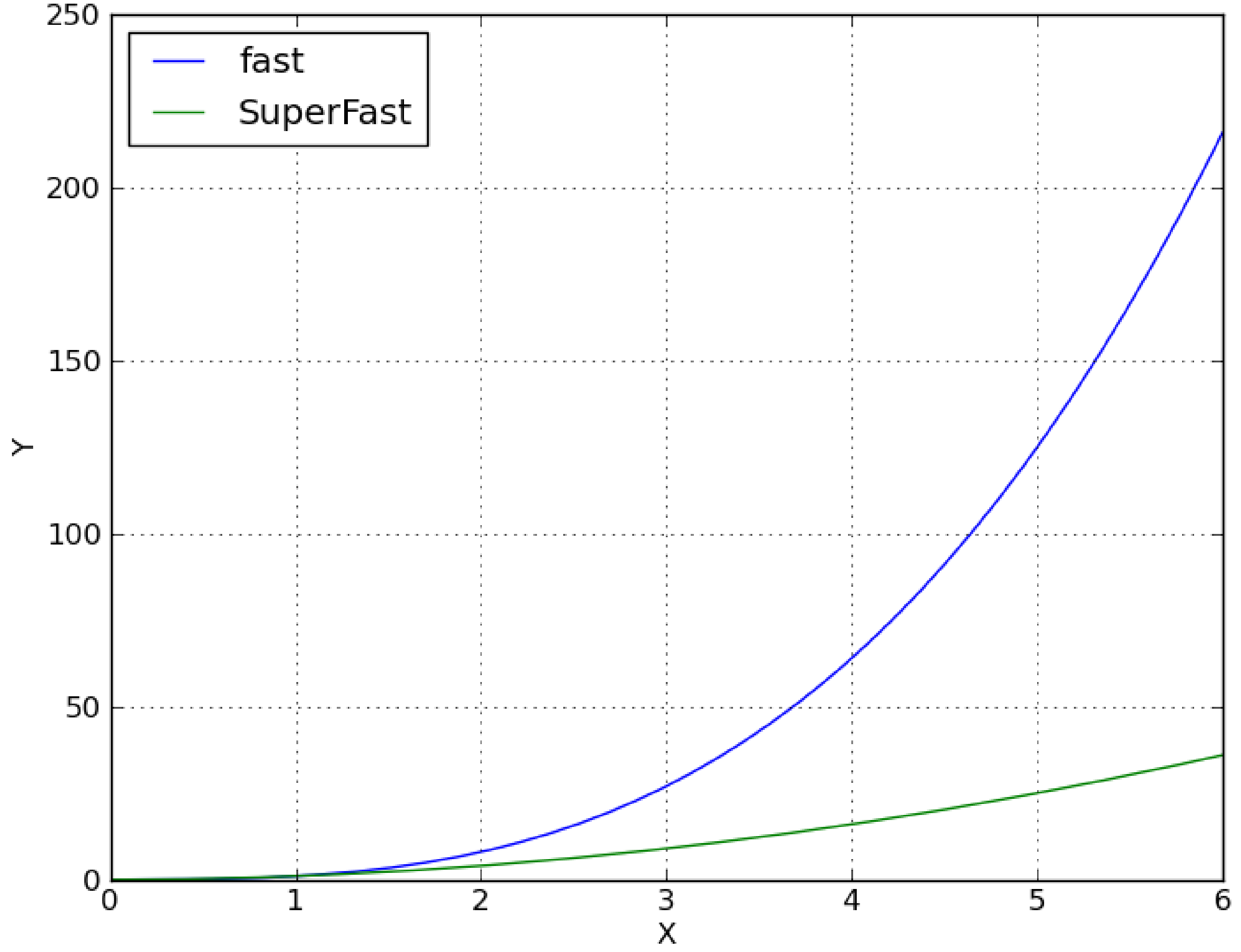
Plot control commands

Classic plot interaction is available

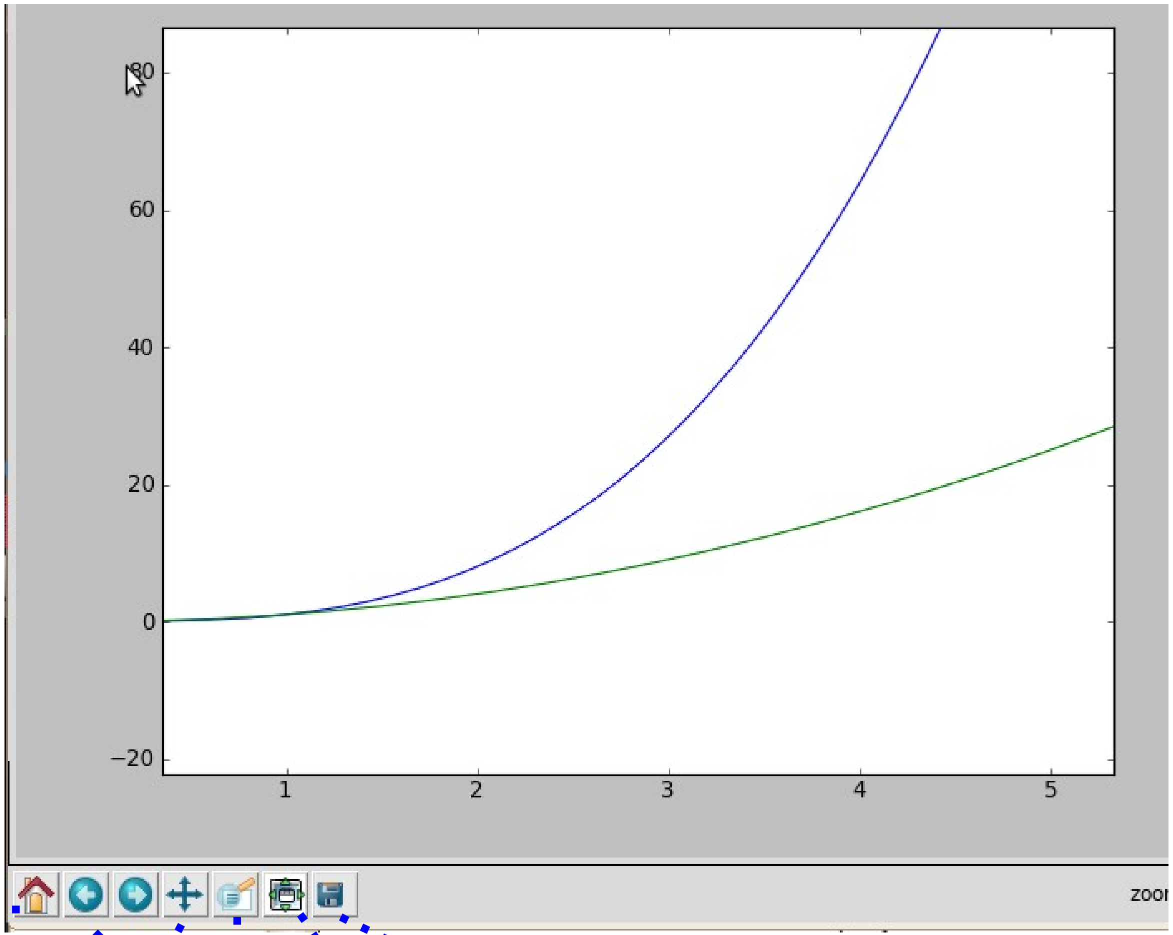
```
>>> plt.grid()
>>> plt.axis() # shows the current axis limits values
>>> plt.axis([0,5,0,10]) #[xmin,xmax,ymin,ymax]
>>> plt.xlabel('This is the X axis')
>>> plt.title('Outstanding title here')
>>> plt.legend(['Fast', 'SuperFast'],loc=2)
>>> plt.savefig('plot123.png', dpi=250)
```

extension determines the file format

Plot example



Plot window



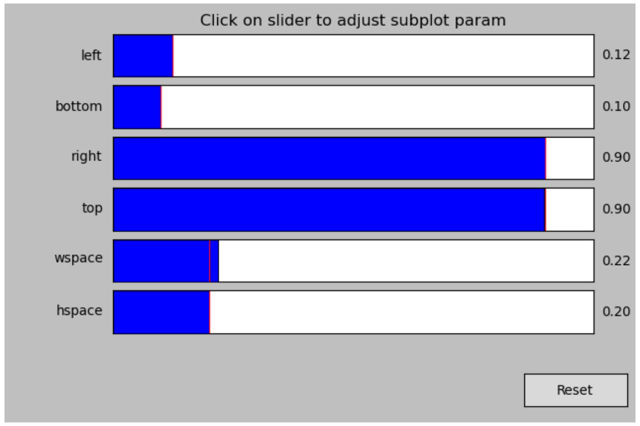
First view of the plot

Back and forward among views

Move (left clik) and zoom (right click)

Draw a view of the plot

Save file



Interactive usage

Ipython is recommended:

- It has already a matplotlib support mode

  ```
  $ Ipython -pylab
  ```

- no need to `import` any modules; merges matplotlib.pyplot (for plotting) and numpy (for mathematical functions)
- spawn a thread to handle the GUI and another one to handle the user inputs
  - every plot command triggers a plot update

Object-oriented interface

A figure is composed by a hierarchical series of Matplotlib objects

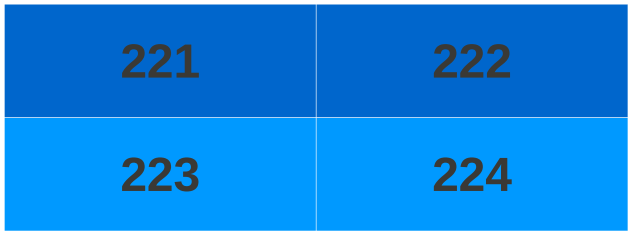
- **FigureCanvas**: Container class for the Figure instance
- **Figure**: Container for one or more Axes instances
- **Axes**: The rectangular areas that holds the basic elements, such as lines, text, and so on

```
>>> ax = fig.add_subplot(221)
```

numb of rows

numb of cols

fig number



Object-oriented interface

- OO use of matplotlib makes the code **more explicit** and allows a lot more **customizations**

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> x = np.arange(0, 10, 0.1)
>>> y = np.random.randn(len(x))
>>> fig = plt.figure()      # instance of the fig obj
>>> ax = fig.add_subplot(111) # instance of the axes
                              # obj
>>> l, m = ax.plot(x, y, x, y**2) # returns a tuple of obj
>>> l.set_color('blue')
>>> m.set_color('red')
>>> t = ax.set_title('random numbers')
>>> plt.show()
```

Object-oriented interface

- Multiple figures are allowed

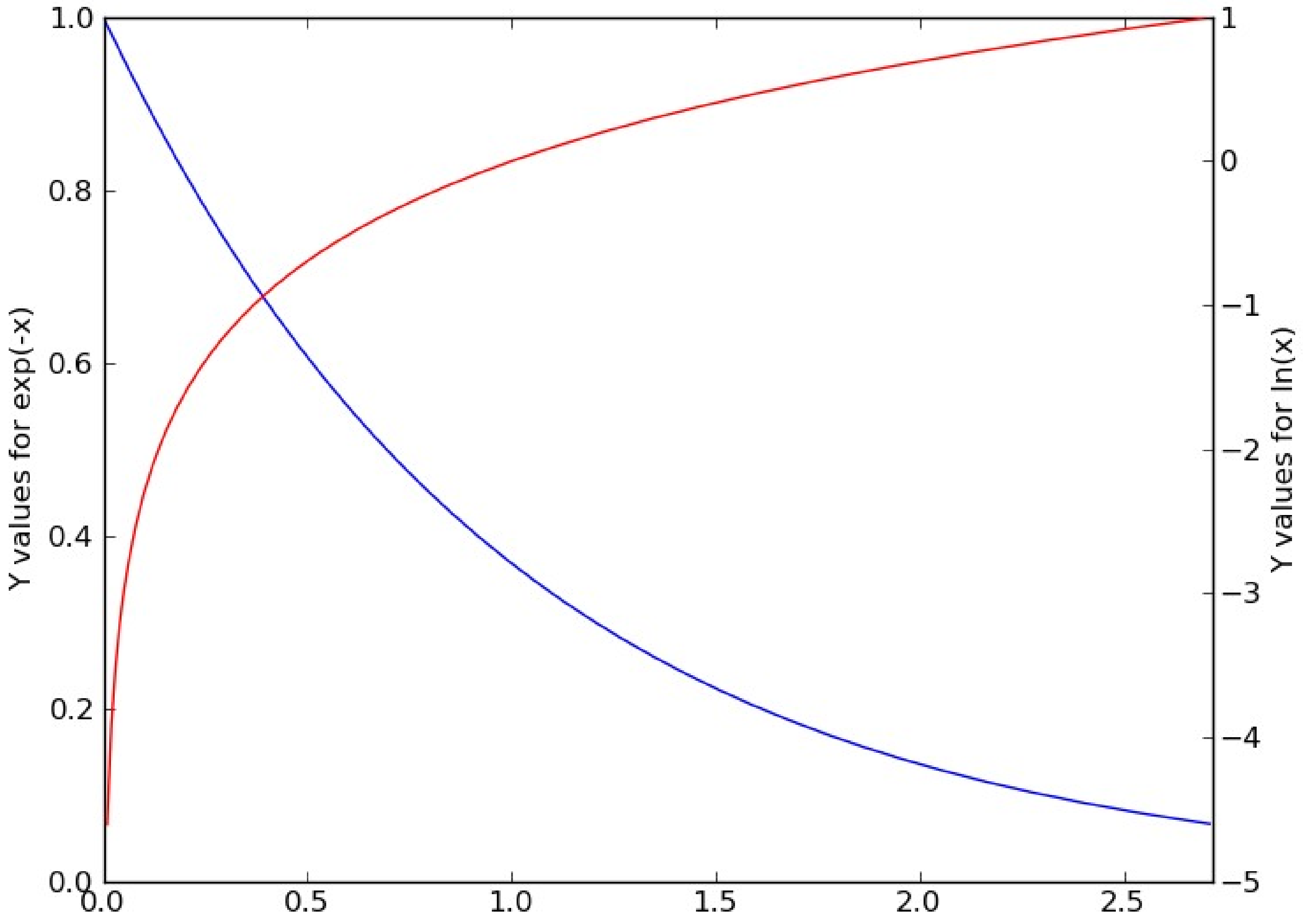
```
>>> import matplotlib.pyplot as plt
>>> fig1 = plt.figure()
>>> ax1 = fig1.add_subplot(111)
>>> ax1.plot([1, 2, 3], [1, 2, 3]);
>>> fig2 = plt.figure()
>>> ax2 = fig2.add_subplot(111)
>>> ax2.plot([1, 2, 3], [3, 2, 1]);
>>> plt.show()
```

Object-oriented interface

- Additional axes are allowed as well

```
>>> x = np.arange(0., np.e, 0.01)
>>> y1 = np.exp(-x)
>>> y2 = np.log(x)
>>> fig = plt.figure()
>>> ax1 = fig.add_subplot(111)
>>> ax1.plot(x, y1);
>>> ax1.set_ylabel('Y values for exp(-x)');
>>> ax2 = ax1.twinx() # this is the important function
>>> ax2.plot(x, y2, 'r');
>>> ax2.set_xlim([0, np.e]);
>>> ax2.set_ylabel('Y values for ln(x)');
>>> ax2.set_xlabel('Same X for both exp(-x) and ln(x)');
>>> plt.show()
```

Object-oriented interface



Other examples

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

To summarize

- This was a very brief (and incomplete) introduction to Matplotlib.
- We don't need to cover everything; just go with your needs
- It can be used interactively, *ala* Matlab, or Object-oriented
- It can be fully integrated in a Python program;
    - ➔ your analysis code can be integrated with a plot tool, tailored to the application needs