Matplotlib Introduction

Molecular Statistics

2014

1 Introduction

Matplotlib (MPL) is a module for Python used for plotting. It is very useful for visualizing all kinds of data types and extremely customizable. For plot examples see matplotlib.org/gallery.html and webloria.loria.fr/~rougier/teaching/matplotlib.

This short introduction will go through small plots with code examples. For advanced examples (like 3D plots), see the examples above or use "Google".

To utilize the power of matplotlib in a simple way we use a wrapper function called **pyplot** (PLT), which basically means we can use matplotlib without any crazy coding (involving class and instances etc). This is simply done by importing the pyplot module in the head of your `.py` file.

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

There are several ways to show the results of MPL, open a windows show the results or saving the plot as an image or other formats.

```python
plt.show()                  # Show the result in a new window
plt.savefig('my_plot.png')  # Save the result in a file
```

The `savefig()` function can save to `.png`, `.eps`, `.svg`, as well as `.pdf` formats. The last formats are very useful for LaTeX work.

When working with multiple plots in the same script, remember to clear the PLT data, or you will see a mess of data in the plot. This is done using the `clf()` function from PLT.
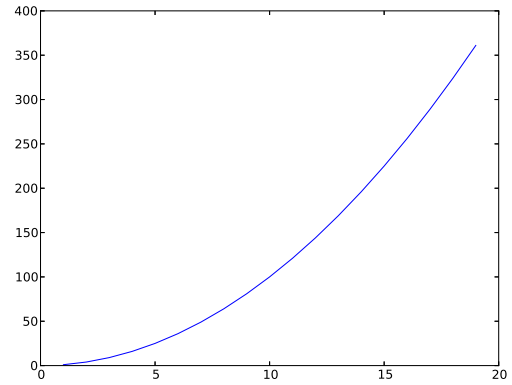
```python
plt.clf()
```

Happy Pyplot'ing!

P.S. Something about mpl_header on Github.

2 Examples

2.1 XY

A simple example is to plot x and y coordinates based on two simple python lists. For $y(x) = x^2$.

```python
import matplotlib.pyplot as plt

x = range(1, 20)
y = [i**2 for i in x]

plt.plot(x, y)
plt.savefig('figure_xy.eps')
```

2.2 Titles and labels

However a graph is worth nothing without title and axis-labels, which can be included by the following syntax. It is also possible to include latex code in the strings.

```python
import matplotlib.pyplot as plt

# Data
x = range(1, 20)
y = [i**2 for i in x]


# First graph
plt.title('Nice Graph Title')
plt.xlabel('Energy [kcal/mol]')
plt.ylabel('Something else $\hat \
    lambda^2$')

plt.plot(x, y)
plt.savefig('figure_title.eps')


# Clear data
plt.clf()


# Second graph
plt.title('Another nice title')
plt.xlabel('X [$R^2$]')
plt.ylabel('Y [$\hat \epsilon$]')

# Place some text
plt.text(5, 200, r'$\mu=100,\ \sigma=15
    $')

plt.plot(x, y)
plt.savefig('figure_title_text.eps')
```

figure_title.eps

figure_title_text.eps
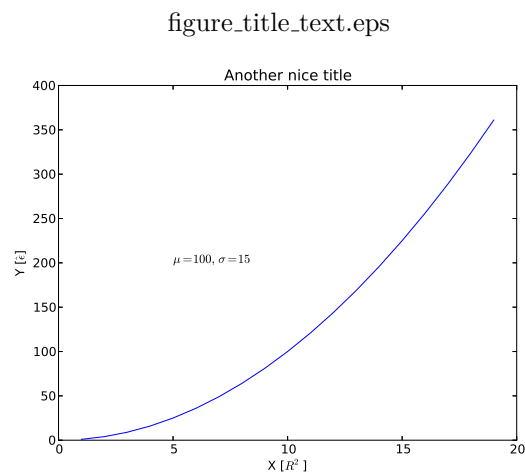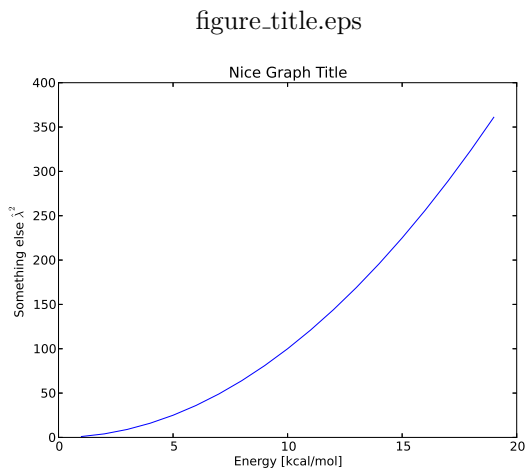
2.3 Multiple lines

Usually you will need to plot multiple datasets in the same figure. This is practically just adding more `plot()` functions. For customization you can add labels and choose color and line-type for the specific data sets. See appendix A for a full list of colors and line styles. If you do not choose a color, a color will be chosen by default.

Note that the location of the legend box can be set by changing the string location.

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

# Data
x = np.arange(0, 14, 0.1)
y_cos = [np.cos(i) for i in x]
y_sin = [np.sin(i) for i in x]

# First plot
plt.plot(x, y_cos, 'r-')
plt.plot(x, y_sin, 'b-')

plt.savefig('figure_sincos1.eps')

# Clear MPL data
plt.clf()

# Second figure
plt.plot(x, y_cos, '-', label="Cos
    function")
plt.plot(x, y_sin, '.-', label="Sin
    function")

# Location of legend
plt.legend(loc='upper left')

plt.savefig('figure_sincos2.eps')
```

figure_sincos2.eps

figure_sincos2.eps

4
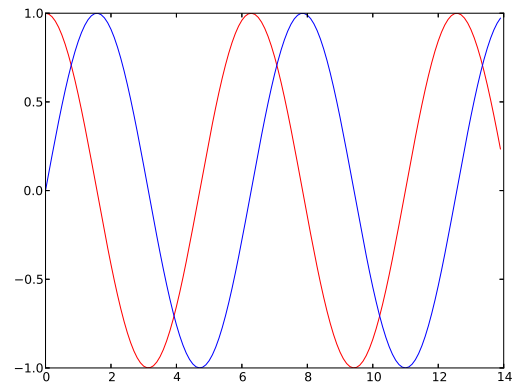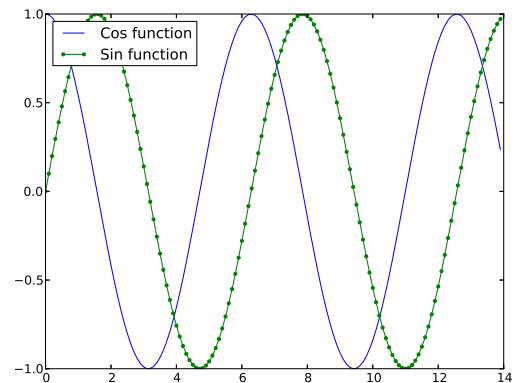
A Plot styling

A.1 Colors

b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black
w	white

A.2 Line styles

0	tickleft
1	tickright
2	tickup
3	tickdown
4	caretleft
D	diamond
6	caretup
7	caretdown
s	square
—	vline
x	x
5	caretright
_	hline
^	triangle up
d	thin diamond
h	hexagon1
+	plus
*	star
,	pixel
o	circle
.	point
'1'	tri down
p	pentagon
'3'	tri left
'2'	tri up
'4'	tri right
H	hexagon2
v	triangle down
'8'	octagon
<	triangle left
>	triangle right