Plotting: gnuplot & pyplot

Shivam Verma
Email address: shivam.59910103@gmail.com
Research Scholar, Department of Physics

Ramakrishna Mission Vivekananda Educational and Research Institute
(RKMVERI)

gnuplot

Installation

If you are working on Linux simply download gnu from apt repository with the command,

```
sudo apt install gnuplot
```

If you are working on windows download and run appropriate version from,

Click here to download GnuPlot

Mathematical expression Visualization

To visualize a function $(f(x),\ g(x))$ in *GnuPlot*, following line of code does the job,

```
gnuplot> f(x) =  <1st mathematical expression>
gnuplot> g(x) =   <2nd mathematical expression>
gnuplot> set xlabel "<put x label>"
gnuplot> set ylabel "<put y label>"
gnuplot> plot[t=<range>] f(t) title "<1st title>", g(t) title "<2nd title>"
```

Saving a plot

```
gnuplot> set term png
gnuplot> set output "<name of the file>.png"
gnuplot> plot[t=<range>] f(t) title "<1st title>", g(t) title "<2nd title>"
gnuplot> set term qt
```

Plotting a data file

Instead of writing the function just substitute it with the data file and since it's a 2d plot give the columns corresponding to x and y values

```
gnuplot> plot "<filename>" using 1:2
```

if you have error in a 3rd column include it and add "*using*/*u*"

```
gnuplot> plot "<filename>" using 1:2:3 with yerr
```

To plot multiple columns on the same plot

```
gnuplot> plot "<filename>" using 1:2 , <filename> using 1:3
```

Line style

- **linetype/lt** : In gnuplot there are various predefined linetypes that can be seen by typing test in gnuplot interface.
- **linewidth/lw** : defines thickness of the line.
- **linecolor/lc** : defines the color of the plot line, can take rgb values or just define the colorspec code.
- **pointtype/pt** : In gnuplot there are various predefined pointypes that can be seen by typing test in gnuplot interface.
- **pointsize/ps** : defines thickness of the point.
- **pointcolor/pc** : defines the color of the plot points, can take rgb values or just define the colorspec code.

```
gnuplot> gnuplot> set style line <index> lc rgb 'red' lt 1 lw 2 pt 7 pi -1 ps 1
gnuplot> plot "<filename>" using 1:2 title "<Some title>" w lp ls 1
```

Scaling

One can scale the axes as per requirement

```
gnuplot> set logscale <axis>
```

To reset the axis scale,

```
gnuplot> unset logscale
```

Multi plot

If you want to plot multiple plots in the same canvas then invoke the *multiplot/multi* environment.

```
gnuplot> set multi layout <nrows,nrows> title '<label>'
```

when done with plotting unset multi environment,

```
gnuplot> unset logscale
```

Multi plot

Example

If you want to plot multiple plots in the same canvas then invoke the *multiplot/multi* environment.

```
gnuplot> set multi layout 1,2 title 'All plots'
gnuplot> set style line 1 lc rgb 'red' lt 1 lw 2 pt 7 pi -1 ps 1
gnuplot> plot x**2/16 title 'x^{2}/16' w lp ls 1
gnuplot> set style line 2 lc rgb 'blue' lt 1 lw 2 pt 7 pi -1 ps 1
gnuplot> plot sin(x) title 'sin(x)' w lp ls 2
gnuplot> unset multiplot
```

2D Color map

A surface plot or a 2d color map can be plotted using [1]pm3d style for a 3d/4d data.

```
gnuplot>  set pm3d map
gnuplot> splot "<filename>"
```

[1]For more examples:http://gnuplot.sourceforge.net/demo/pm3d.html

Pyplot

Line plot of f(x)

The simplest kind of plotting is to visualize a single function $y = f(x)$. we will create a simple plot of this type.

```
import matplotlib.pyplot as plt
plt.style.use('classic')
import numpy as np
```

For any type of plot first create a figure and an axes like,

```
fig = plt.figure()
ax = plt.axes()
```

```
x = np.linspace(0, 10, 100)
y = np.sin(x)
fig = plt.figure()
ax = plt.axes()
ax.plot(x,y);
```

Scatter plot

One can use *plt.scatter* or *plt.plot* for drawing a scatterplot.

```
plt.scatter(x, y, marker='o');
```

The difference is evident in a larger dataset, since *plt.scatter* has the capability to render a different size or color for each point, so the renderer must do the extra work of constructing each point individually.
plt.plot, on the other hand, the points are always essentially clones of each other, so the work of determining the appearance of the points is done only once for the entire set of data.

Multiple plots

The role of defining an axes shines here.

```
x = np.linspace(0, np.pi*2, 100)
figure, axis = plt.subplots(2, 2)

axis[0, 0].plot(x, np.sin(x))
axis[0, 0].set_title("Sine Function")

axis[0, 1].plot(x, np.cos(x))
axis[0, 1].set_title("Cosine Function")

axis[1, 0].plot(x, np.tan(x))
axis[1, 0].set_title("Tangent Function")

axis[1, 1].plot(x, np.sinh(x))
axis[1, 1].set_title("sinh Function");
```

Data loading and plotting

Load the data using pandas API.

```
df = pd.read_csv('<filename>', sep="<sep used in file>", names=[<colnames>],header
```

Now you can plot different columns of the loaded dataframe.

```
fig,ax = plt.subplots(1,1,figsize = (7,5))
ax.set(xlabel='<label>', ylabel='<label>',title='<title>')
ax.plot(df['<colname>'], df['<colname>']);
```

2D color plot

The role of defining an axes shines here.

```
x = np.linspace(0, 10, 1000)
I = np.sin(xc) * np.cos(x[:, np.newaxis])

fig,ax = plt.subplots(1,1,figsize=(7,5))
im = ax.imshow(I,cmap = "rainbow")
fig.colorbar(im);
```


thank you!