Simple Plots in Python

Leigh J. Little SUNY Brockport December 4, 2018

Contents

1 Introduction

Python has no native plotting capabilities. This gives rise to a large number of third party modules that provide these features. These modules are varying in quality. Historically, the `pyplot` module has been the standard module, but this has been superceeded by an updated module. The module that is recommend for plotting is the `matplotlib` module. This module incorporates features of the original `pyplot` module, but is more robust and has better capabilities. This module more or less provides the same plotting ability as MATLAB. The main change is in the syntax of the commands.

2 Examples

2.1 Example 1 - Simple x-y plot

The code below will generate a simple plot of x versus y.

```
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [5,4,3,2,1]

plt.figure(1)
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Simple plot')
plt.show()
```

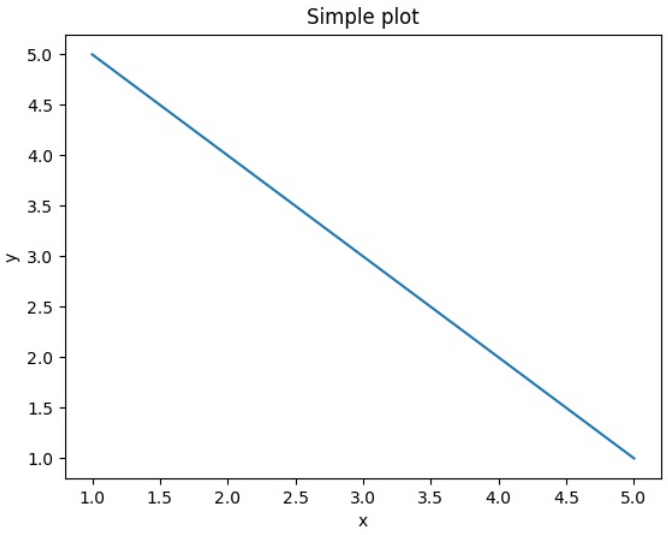

Figure 1: Simple x-y plot example.

Some comments on the above example:

1) The entire `matplotlib` module is not being pulled into the workspace (just the `pyplot` submodule is being imported).

2) Every plot command gets preceded by `plt.`

3) The `plt.show()` command is used to make the figure visible.

2.2 Example 2 - Simple plot with a legend

The code below will generate several plots in the same frame with a legend

```
import matplotlib.pyplot as plt

x = [1,2,3,4,5]
y = [5,4,3,2,1]
z = [3,3,3,3,3]

plt.figure(2)
plt.plot(x,y,label='y')    # Legend labels specified when plotting
plt.plot(x,z,label='z')
plt.xlabel('x')
legend()                   # Displays all legend items
plt.title('Simple plot with legend')
plt.show()
```

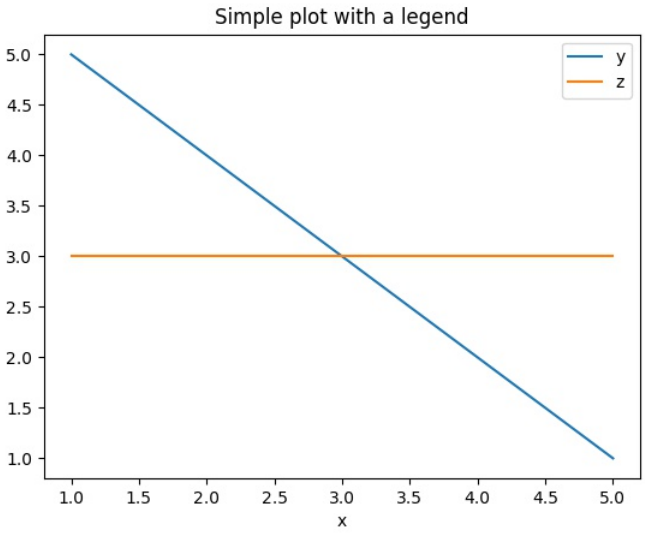

Figure 2: Simple plot with a legend.

2.3 The `numpy` Module

The `numpy` module is used to make solving mathematical problems in Python easier. It does this by introducing a new data structure called an `array` that behaves more like an array/matrix in MATLAB. In fact, augmenting Python with the `numpy` and `matplotlib` modules results in an environment very similar to MATLAB.

This will be illustrated using some 3D graphics capabilities.

2.4 Three dimensional graphics

To date we have not had a need for 3D graphics, but there are many types of 3D graphs that can be created. For example, if we wanted to plot the function

$$z = e^{-(x^2+y^2)}$$

over some rectangular region of the x-y plane we could use a surface plot. However, this requires a more user friendly way of working with vectors than the standard Python list structure. The solution to this is the `numpy` array structure.

To plot this surface, we first need to create what is called a mesh grid. This is a set of two matrices. One contains the x-coordinates and one contains the y-coordinates. When these two matrices are overlaid, they generate a discrete representation of the x-y plane. The function z can them be evaluated on the mesh grid and the surface plot can be generated.

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits import mplot3d

plt.figure(3)
x = np.linspace(-1,1,51)    # Generate 51 points from -1 to 1
y = np.linspace(-2,2,101)   # Generate 101 points from -2 to 2
X,Y = np.meshgrid(x,y)      # Generate mesh grid (note capital X,Y)
Z = np.exp(-(X**2 + Y**2))  # Compute Z on the mesh grid (X,Y)
                            # Note that there are no . operators

ax = plt.axes(projection='3d')  # Generate 3D axes
```

```
ax.plot_surface(X,Y,Z)      # Generate surface plot of Z on ax
plt.show()
```

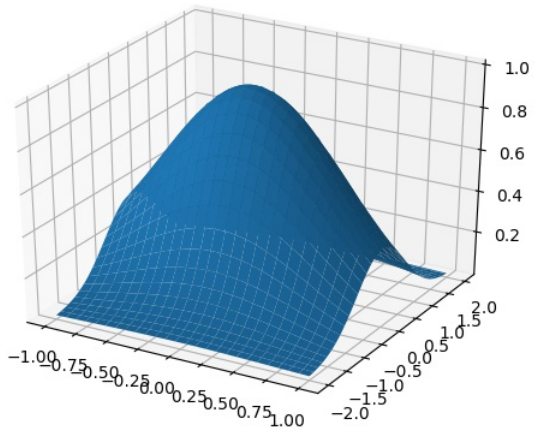

Figure 3: Surface plot of z.

2.5 Example 4 - Contour Map

Contour maps can also be generated. You can do a 2D contour map

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits import mplot3d

plt.figure(4)
x = np.linspace(-1,1,51)
y = np.linspace(-2,2,101)
X,Y = np.meshgrid(x,y)
Z = np.exp(-(X**2 + Y**2))
plt.contour(X,Y,Z)          # No ax here because the plot is 2d
plt.show()
```

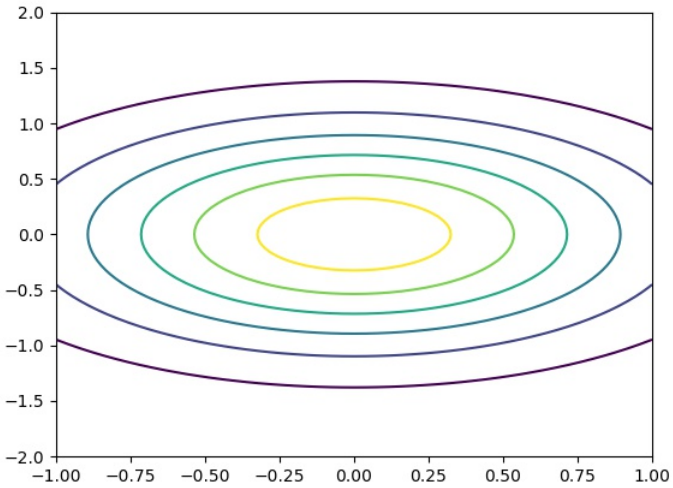

Figure 4: 2D contour map of z.

You can also do 3D contour maps.

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits import mplot3d

plt.figure(5)
x = np.linspace(-1,1,51)
y = np.linspace(-2,2,101)
X,Y = np.meshgrid(x,y)
Z = np.exp(-(X**2 + Y**2))
ax = plt.axes(projection='3d')
ax.contour(X,Y,Z)               # Plot 3D version on ax
plt.show()
```

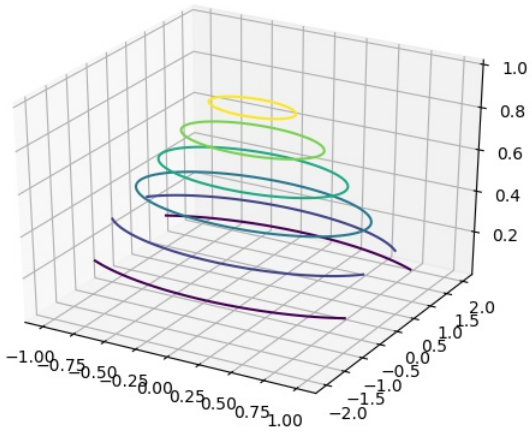

Figure 5: 3D contour map of z.