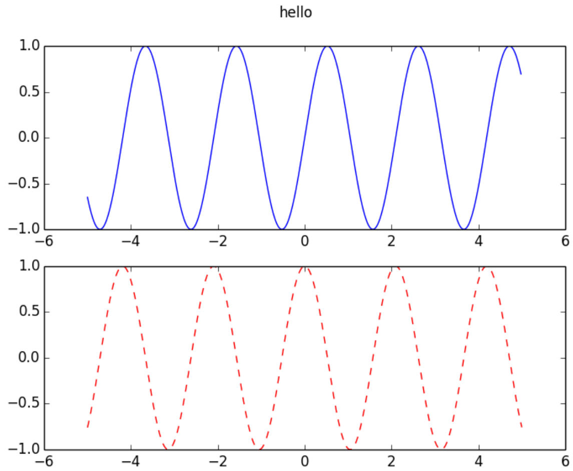
Sept 13, 2016

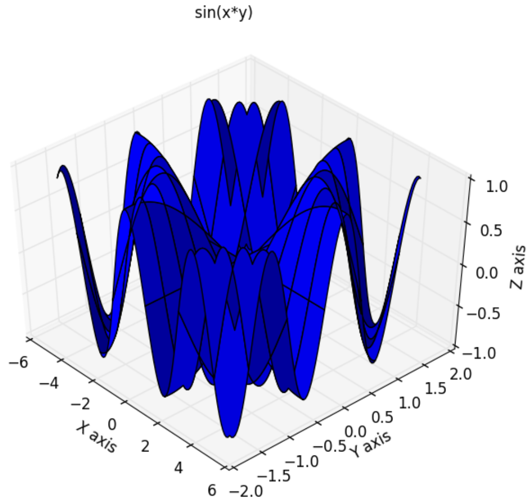
Roy Rinberg
Professor Haas
Computational Physics Project 1 – Plotting Graphs

This project deals with plotting, and learning basic python techniques.

In this project I largely do not know what to write, so I will post a picture and describe the process of creating the image.

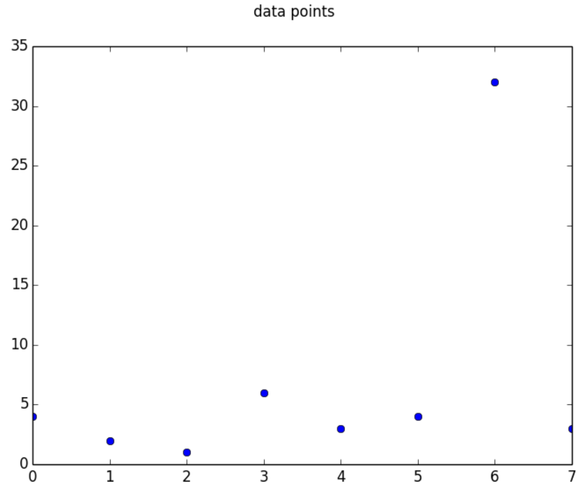


Create a figure, create 2 subplots – create two 2-D arrays for 2 different functions, then plot them in their respective subplots.

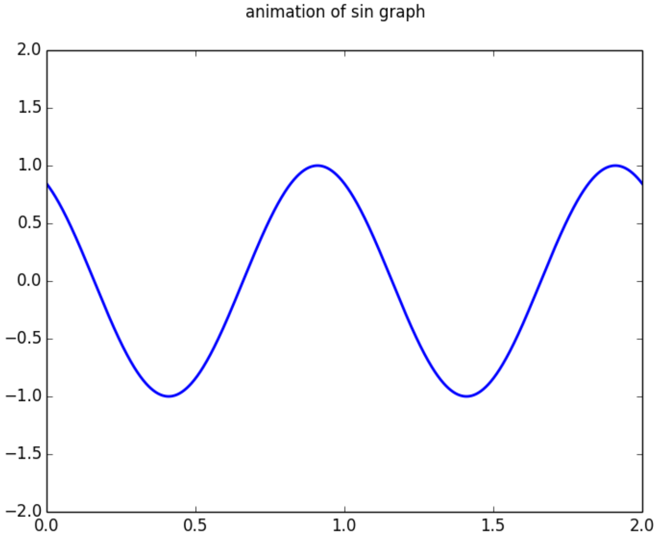


Make a 2D array for the x and y directions. Use meshgrid, in order to make 2 different 2D arrays from 2 different vectors, of the correct size. Take z to be a function of x and y. Create a 3D figure, by the function call ax = fig3.gca(projection='3d') (get current axis – project onto 3 axes). Then do : ax.plot_surface(x,y,z) to plot the surface onto the 3d plot.

Sept 13, 2016



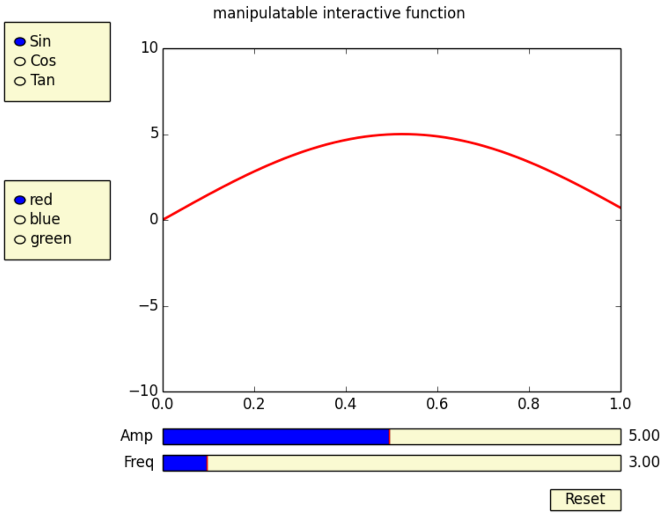
This is just a plot of 2 arrays, such that the x array is evenly spaced, and the y array is data points.



This is a plot of the animation. To make an animation you need to define an "init" and an "animate" function. The animation function takes an argument, and then for every time the animate function is called, the plot is redrawn. Then to actually animate it the animation.FuncAnimation call:

```
anim = animation.FuncAnimation(fig, animate, init_func=None, frames=200,
interval=20, blit=True)
```

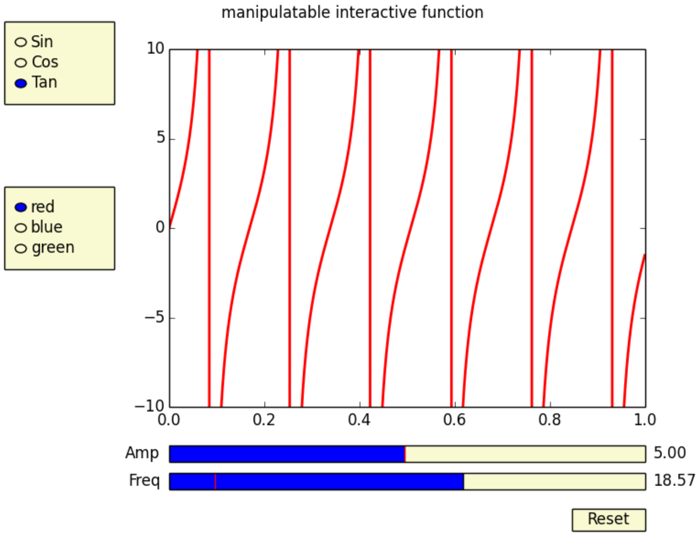
Sept 13, 2016



● Sin
○ Cos
○ Tan

● red
○ blue
○ green

Amp 5.00
Freq 3.00



○ Sin
○ Cos
● Tan

● red
○ blue
○ green

Amp 5.00
Freq 18.57

This is a plot of the manipulable function. To make a manipulable function, it is very similar to the way to create an animation, except you need to also create something interactive, like a slider or a radio button, or something like that. This is in theory a very easy topic, and I was able to do it quite easily, however to add a certain level of functionality, I spent a long time working on the code.

To simply have the same function take 2 arguments, and change those arguments under different slider values was a basic problem, you just take an update and a redraw function. However, I attempted to add the functionality of being able to change the actual function as well as the arguments using GUI outputs – in this case, setting the amplitude and frequency using sliders, as well as the trig function using radio buttons.

The difficulty of this is not adding in the different functions, but making it so that when the function is updated- and then the slider updates the arguments, the new function is preserved, and it doesn't default to the original function that was used.