PHZ4151C: Exercise 3

Computational Physics Lab
Due February 08

For each problem you will write one or more python programs. These programs should follow the Python 2.7.x coding and formatting conventions outlined for our course. You must hand in copies of the programs and outputs as prescribed in each problem.

In addition, you must submit all Python programs as an archive tgz file via email to phz4151c@hadron.physics.fsu.edu. Place copies of only your Python programs in a directory called <last_name>-exercise3/ where <last_name> is your last name and use similar command as provided in earlier exercises.

1. Sunspots: A file called sunspots.txt can be obtained using the wget command:

```
wget http://hadron.physics.fsu.edu/~eugenio/comphy/mat/sunspots.txt
```

This text data file contains the observed number of sunspots on the Sun for each month since January 1749. The file contains two columns of numbers, the first being the month and the second being the sunspot number.

(a) Write a program that reads in the data and makes a graph of sunspots as a function of time.

(b) Modify your program to display only the first 1000 data points on the graph.

(c) Modify the program further to calculate and plot the running average of the data , defined by

$$Y_k = \frac{1}{(2r+1)} \sum_{m=-r}^{r} y_{k+m},$$

where r = 5 in this case (and y_k are the sunspot numbers). Have the program plot both the original data and the running average on the same graph, again over the range covered by the first 1000 data points.

For full credit turn in a printout of your final program and printout of your final plot.

2. Curve Plotting: Although the `plot` function is designed primarily for plotting standard `xy` graphs, it can be adapted to plot other kinds of plotting as well.

(a) Make a program to plot the so-called *deltoid* curve, which is defined parametrically by the equations

$$x = 2\cos(\theta)+\cos(2\theta), \qquad y = 2\sin(\theta)-\sin(2\theta)$$

where 0 <= θ < 2π. Take a set of values of θ between zero and 2π and calculate `x` and `y` for each equation above, then plot `y` as a function of `x`.

(b) Taking this approach a step further, one can make a polar plot `r=f(θ)` for some function `f` by calculating `r` for a range of values of θ and then converting `r` and θ to Cartesian coordinates using

the standard equations `x = r cos(θ), y = r sin(θ)`. Use this method to make a program to plot the Galilean spiral $r=\theta^2$ for `0 <= θ <= 10π`.

(c) Using the same method, make a polar plot of "Fay's function", also known as the butterfly curve, in the range `0 <= θ <= 24π`.

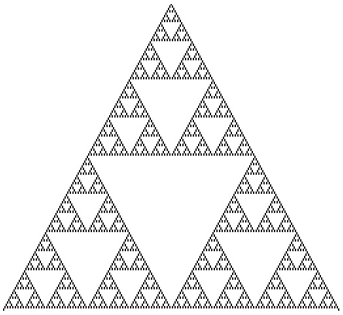
$$r = e^{\sin\theta} - 2\cos 4\theta + \sin^5\frac{\theta}{12}$$

For full credit turn in a printout of your three programs and plots.

3. Scanning Tunneling Microscope: Using the wget command obtain the file: `http://hadron.physics.fsu.edu/~eugenio/comphy/mat/stm.txt`. This file contains a grid of values from scanning tunneling microscope measurements of the (111) surface of silicon. A scanning tunneling microscope (STM) is a device that measures the surface of a solid at the atomic level by tracking a sharp tip over the surface and measuring quantum tunneling current as a function of position. The end result is a grid of values that represent the height of the surface and the file stm.txt contains just such a grid of values. Write a program that reads the data contained in the file and makes a density plot of the values. Use the various options and variants you have learned about to make a picture that shows the structure of the silicon surface clearly.

For full credit turn in a printout of your program and printout of your plot.

4. Fractal Patterns: The Sierpinski triangle is a fractal with the overall shape of an equilateral triangle, subdivided recursively into smaller equilateral triangles. Originally constructed as a curve, this is one of the basic examples of self-similar sets, i.e. it is a mathematically generated pattern that can be reproducible at any magnification or reduction. It can be constructed using a random rather than a deterministic algorithm. It is named after the Polish mathematician Wacław Sierpiński.



One can generate the fractal pattern by following the prescription below:

[1] Use the three (x,y) vertex points: (-1,0), (0, $\sqrt{3}$), and (1,0) in a plane to form a triangle.
[2] Starting from the origin (0,0) plot a point.
[3] Randomly select any one of the 3 vertex points.
[4] Move half the distance from your current position to the selected vertex.
[5] Plot the new current position.
[6] Repeat from step 3.

Write a program that generates 200,000 data points and make a scatter plot of the values. Use a small enough plotting point size so that you can clearly observe a detailed pattern. Demonstrate that the pattern is indeed a fractal. For randomly selecting a vertex point, use of the `numpy.random` module's function `randint(n)` which return a random integer `r` such that `0 <= r < n`.

For full credit turn in a printout of your final program, scatter plot(s), and a discussion of the fractal pattern demonstration.