Homework 1

Erica Kotta

September 15, 2016

1 Problem 3.7

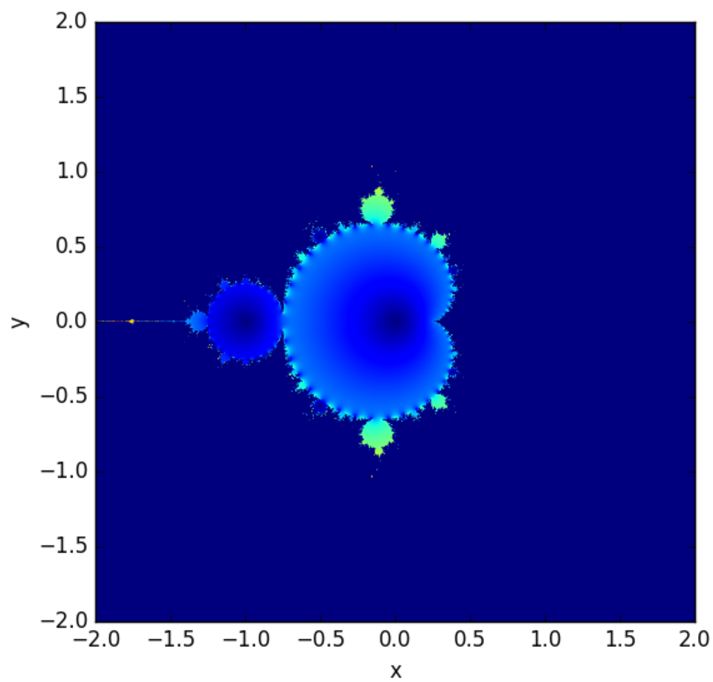

Figure 1: Mandelbrot plot with 50 iterations, 1000x1000 resolution.

This problem asks us to plot the Mandelbrot set. This means we want to ultimately get square matrix and plot it using "imshow." I defined a function "Mandelbrot" that, for each element of the matrix (size of matrix defined by "points" value in code), it will first convert the matrix indices of the element into x and y values ($-2 \leq x, y \leq +2$). Then, for each (x,y), it will perform the iteration $z' = z^2 + c$, where c is a complex number (x+iy). The number of times this iteration is performed is defined earlier in the code (I used N=50 in this case). Then there is an "if/else" statement which asks the computer to calculate the final value of z'. If the absolute value of z' is less than 2, the code assigns that matrix element with this value. If it is greater than 2, the matrix element is assigned 0.

2 Problem 3.8

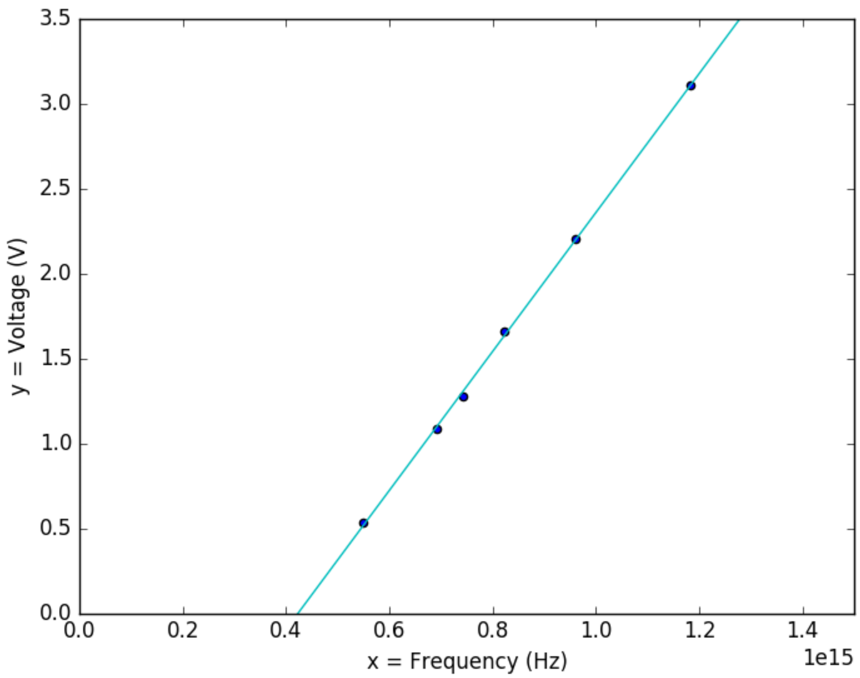

Figure 2: Scatter plot of data points with calculated best-fit line.

This problem asks us to plot some values from a pre-existing text file "millikan.txt" and to find the best-fit line from the data points.

Using the method of least squares as given in the problem, I calculated the slope and y-intercept of the best-fit line for the data points. This gave the blue line shown on the plot (amidst the plot of the data points).

The calculated values for m (slope) and c (y-intercept) as given by the Python code (included in repo as "Problem3-8abcd.py") are

$$\begin{aligned} m &= 4.08822735852e - 15 \\ c &= -1.73123580398 \end{aligned} \tag{1}$$

Comparing $y = mx + b$ to the photoelectric effect equation:

$$V = (\frac{h}{e})\nu - \phi \tag{2}$$

where h is Planck's constant, e is the electron charge, ν is the frequency of light (in Hz), and ϕ is the work function, we see that m is equivalent to $\frac{h}{e}$ and c is equivalent to ϕ. We can solve for h using our calculated m:

$$h = \frac{e(V + \phi)}{\nu} = 6.54934022835e - 34 \tag{3}$$

Comparing this to the accepted value 6.6207004e-34 gives us a percent error (calculated by same Python file) of 1.08%.