Homework 1, Computational Physics

Milad Noorikuhani (N17735678)

September 18, 2017

Abstract

In this homework we use python to write programs for two problems. One is density plots of all complex numbers in an $N \times N$ grid system that some of them belong to Mandelbrot set (ones for which a function including them remains bounded under a definite iteration process) and others fall outside the set. Different programs are written for separating them in different ways and beautiful fractal structures will appear in this way. The other program is related to applying least square method to find the best line that fits a set of data. At this part the slope and intersect of the best line are found. Surprisingly, these data are related to the famous photoelectric experiment of Robert Millikan. From the slope of the best line, the Planck's constant is calculated as $h = 6.54934022834904e - 34 J.s$. It is shown that the relative difference of the resulting number with the accepted value is about 1.2%.

1 Introduction and methods

First I introduce the programs written for problem (3.7) and then ones written for problem (3.8) will be introduced. Mandelbrot set, named after French Mathematician Benoît Mandelbrot, is a set of complex numbers c that function $z' = z^2 + c$ remains bounded under iteration. In other words, if we start from $z = 0$ and put it in the function and iterate the resulting value z' into the same function and repeat this process for infinite times, the function will never diverge. Practically, to check whether a complex number is in the Mandelbrot set or not, we can repeat the iteration for a relatively large number of times say more than 100 times and check the value of the resulting number for each iteration. If, in any step, this value is larger than 2, we exclude c from Mandelbrot set. For plane of complex numbers, we choose values of x and y generating the complex number $c = x + iy$. These two values are chosen so that $-2 \leq x \leq 2$ and $-2 \leq y \leq 2$. Then, we divide these ranges into N equal parts. This creates an $N \times N$ grid system. I have written 6 different programs for this problem. I briefly introduce them here. Results are shown in the next section. In the first file (37p1.py), I have written a program for a 100×100 grid system with 150 iterations. For this program and similarly other programs, first, x and y are defined as arrays containing N values between -2 and 2 (including the endpoints). A function is defined that takes two variables as input (the complex number and the number of iterations) and performs iteration based on the definition of Mandelbrot set. The output of this function for different values determine the fractal structure of Mandelbrot set in the complex plane. For plotting the Mandelbrot set, I used an array to fill its elements with the outcomes of the function for each complex number $(x[i] + iy[j])$ and plotted that with imshow(). The resulting figures are all in the complex plane $(x + iy)$ with $-2 \leq x, y \leq 2$. Figure 1 is the result of the program 37p1.py with $N = 100$ and 150 iterations. Mandelbrot set is formed from the black points. Figure 2 from the file 37p2.py shows a higher quality image with $N = 1000$ and 300 iterations. Figure 3 from the file 37p3.py shows a colored image of the Mandelbrot set with $N = 500$ and 200 iterations. Figure 4 from the file 37p4.py uses "jet" scheme to produce a colored image for $N = 500$ and maximum 200 iterations. For numbers outside the set the color is based on the number of iterations after which they fall out of the set. Figure 5 from the file 37p5.py uses "jet" scheme to produce an image similar to the previous time but this time the color of outside numbers are determined based on the logarithm of the number of iterations after which the they fall out of the set.

Figure 6 from the file 37p6.py is similar to the previous image except it uses "hot" scheme to create the figure. We see how beautiful ramified fractal structures arise from Mandelbrot set.

For the second problem (3.8) we use the least square method to find the best linear feat to a set of data related to Millikan's photoelectric experiment. The horizontal and vertical axes are frequency and voltage respectively. figure 7 from the file 38a.py shows the data points for (frequency, voltage) as black circles. This file is related to the part (a) of the problem. For part (b), I have written a code that calculates slope (m) and intersect (c) of the best line fitted on the data using the least square method described in the problem sheet. This code can be found in the file 38b.py. This code also plots the data again. In the file 38cdplot.py, which is related to the parts (c) and (d) of the problem, a code is written to generate figure 8 and calculate the Planck's constant and its accuracy which is the relative difference with the accepted value.

2 Plots and results

In this section, results of our programs is shown. Descriptions can be found in the previous section.

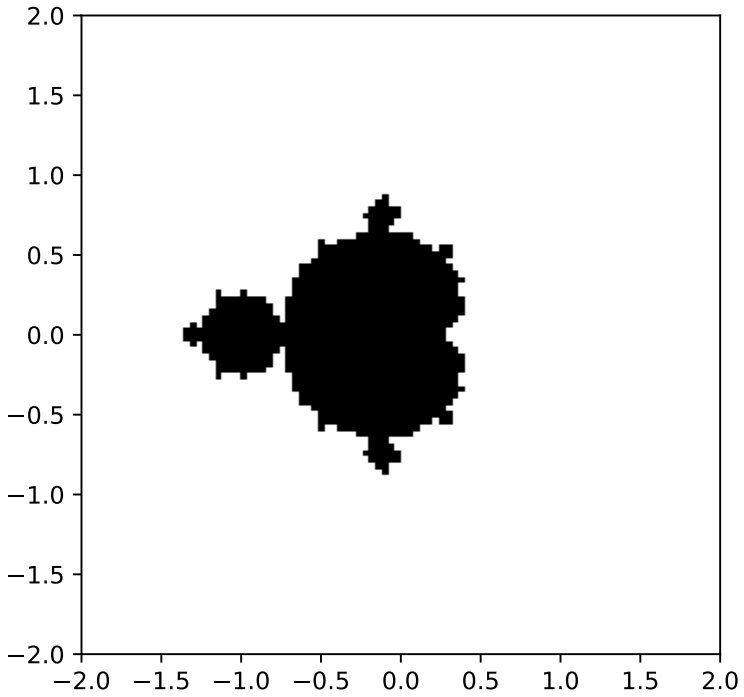

Figure 1: Mandlbrot set in "gray" scheme for N=200, max iterations=150.

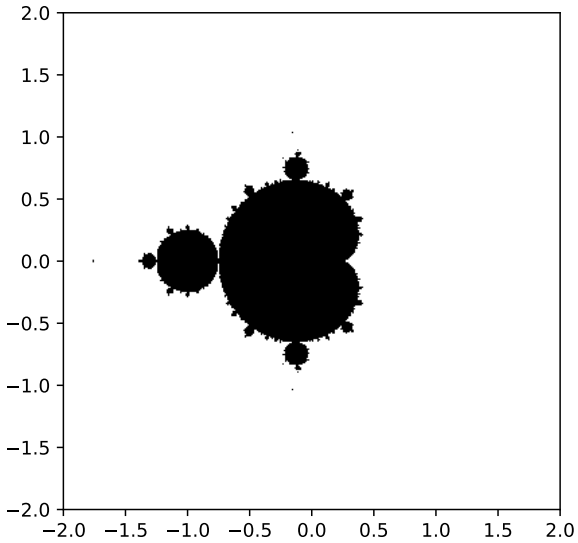

Figure 2: Mandlbrot set in "gray" scheme for N=1000, max iterations=300.

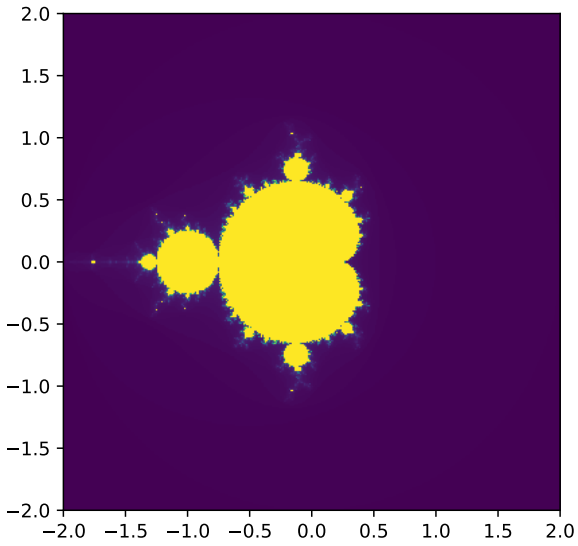

Figure 3: Colored Mandelbrot set with N=500 and max iterations = 200.

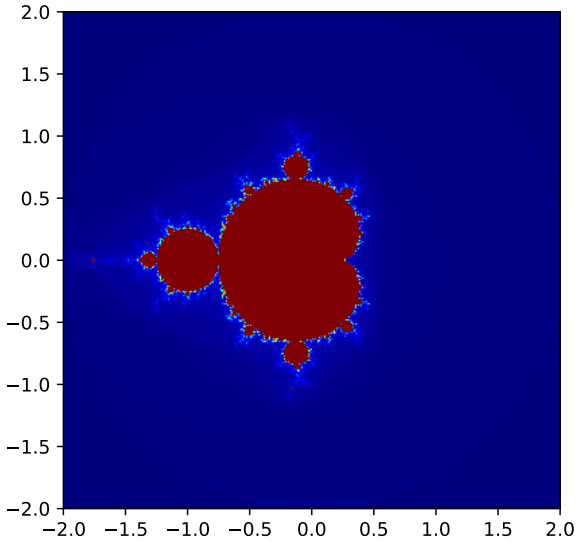

Figure 4: Colored Mandelbrot set using "jet" scheme with N=500 and max iterations = 200. The complex numbers out of the set are colored based on the number of iterations after which they fall out of the set.

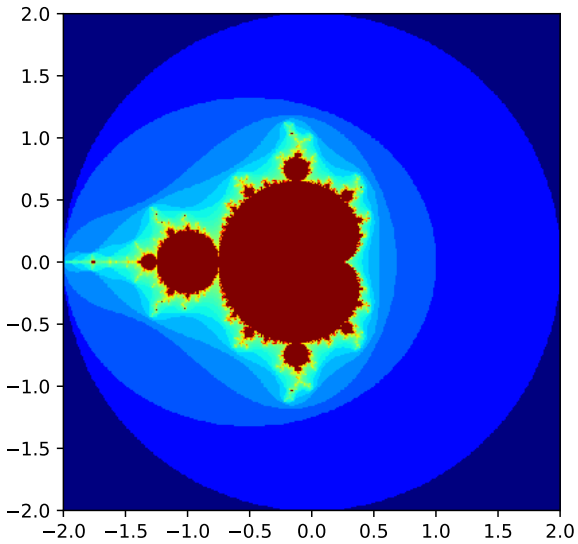

Figure 5: Colored Mandelbrot set using "jet" scheme with N=500 and max iterations = 200. The complex numbers out of the set are colored based on the logarithm of the number of iterations after which they fall out of the set. Substructures can be seen in this way.

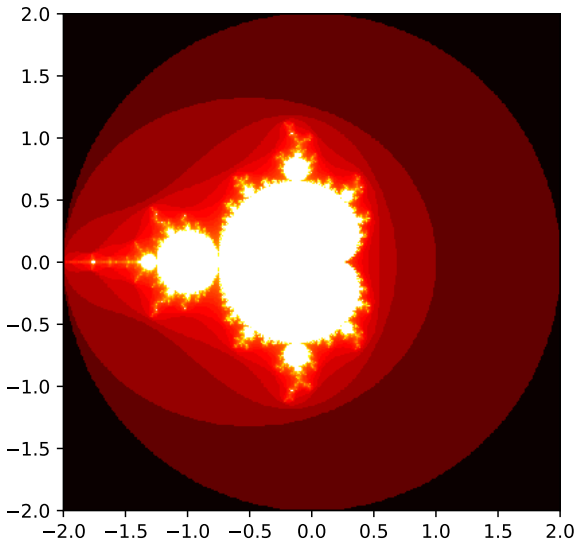

Figure 6: Colored Mandelbrot set using "hot" scheme with N=500 and max iterations = 200. The complex numbers out of the set are colored based on the logarithm of the number of iterations after which they fall out of the set. Substructures can be seen in this way.

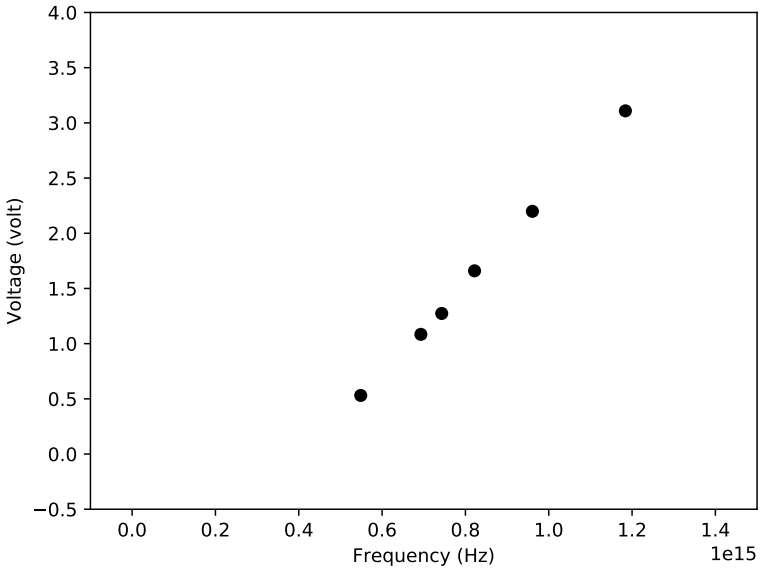

Figure 7: In this plot, black circles are the data points (frequency, voltage) from the file millikan.text

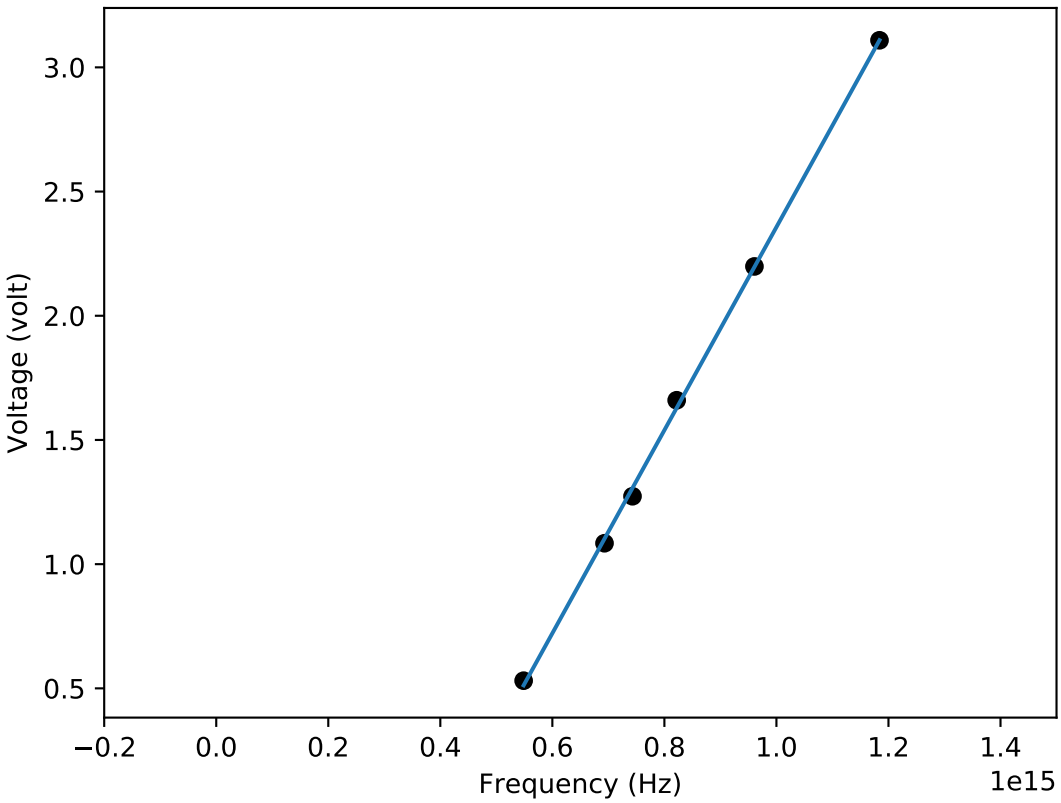

Figure 8: Using the least square method in my code in the file 38b.py, I found the slope and intersect of the best fit line to be m= 4.08822735852e-15 and c= -1.73123580398 respectively. This plot is created from the file 38cdplot.py and shows the best fitting line. In addition, using the slope of the best line, the Planck's constant has been calculated as h= 6.54934022834904e-34 J.s. The accuracy, which is the relative difference with the accepted value of the Planck's constant, is about 1.2%.