Problem Set #2

Sarah Odeh - odesmodes

1 Question 1

1. Floating point representation: 01000010110010011111100110101011
2. Actual number represented by the floating point: 100.98763275146484
3. Difference between the original number and it's representation: 2.7514648479609605e-06

2 Question 2

1. The smallest number after 1 in 32-bit representation is 1.0000001 while the smallest number in 64-bit representation is 1.0000000000000002
2. Positive minimums and maximums for each floating point representation:
   (a) 32-bit maximum: 3.4028235e+38
   (b) 32-bit minimum: 1.1754944e-38
   (c) 64-bit maximum: 1.7976931348623157e+308
   (d) 64-bit minimum: 2.2250738585072014e-308

3 Question 3

With the for loop method we get that the Madelung constant is 1.7475645950377328. I used time in linux (ie time python3 Newman_Exercise_2.9.py) to time how long it took to run instead of %time and got approximately 17 seconds on average. Without the for loop method we get that the Madelung constant is 1.7418198158362386 and it takes 0.5 seconds on average to run. So we can see that using the for loop method slows down the process.

4 Question 4

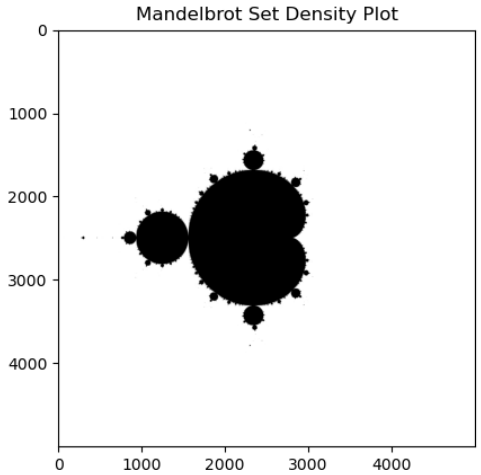

Figure 1: Mandelbrot plot generated in question 4

5 Question 5

1. Part a: (x1, x2) = (-9.999894245993346e-07, -999999.999999)
2. Part b: (x1, x2) = (-1.000000000001e-06, -1000010.5755125057) Note that the answers are different. This is because of the float precision error that happens when we are dividing in the quadratic equation.
3. Part c: (x1, x2) = (-9.999894245993346e-07, -1000010.5755125057) The accurate solutions to the quadratic.