Problem Set 2

By Hayden Orth; GitHub: haydenorth

September 19, 2023

Abstract

This article contains my solutions to Problem Set 2 for the graduate Computational Physics course. Problem Set 2 investigates topics such as binary representations and errors, truncation and precision errors, arrays, array arithmetic, and plotting.

1 Problem 1: Binary Representations

A computer has different ways of representing different numbers in binary. NumPy's 32-bit floating point representation represents the number 100.98763 as the following 32-bit sequence: 01000010110010011111100110101011. The bits are broken down as follows. Sign bit: 0. Exponent: 10000101. Mantissa: 10010011111100110101011. The way that Python natively represents the value is different than the 32-bit floating point representation. The two representations differ by a total -2.75E-6. It's pretty close, depending on who you ask.

2 Problem 2: Madelung Constant

Problem 2 investigates a comparison between the calculation of the Madelung constant using for loops versus using 3D Numpy arrays. Both methods produced a value of M = -1.74 using L = 50. Using nested for loops, the calculation took around 3.1 seconds. The method using Numpy arrays worked much faster, taking only 0.07 seconds. This discrepency in speed will only grow as L is increased.

3 Problem 3: Mandelbrot Set

In Problem 3, I wrote a script that makes an image of the Mandelbrot set for complex values c = x + iy, where x and y are between -2 and 2. Values in the set are colored black while values that are not in the set are white. The image is included in Figure 1 on the next page.

4 Problem 4: Quadratic Function

In Problem 4, I implemented code to find the zeros of a quadratic function in two different ways. Here, I experienced differences in the calculated values for the zeros due to truncation error. In this case, one of the floating point values involved in the final division operation is to far less precision than the other. When the computer changes the exponent of the floating point representation of the divisor to that of the dividend, the precision of the divisor value is lost if it is far more precise than the dividend. Thus, I needed to ensure that the value with greater precision is always the dividend in order to mitigate truncation errors. My implementation of these concepts passed the unit test.

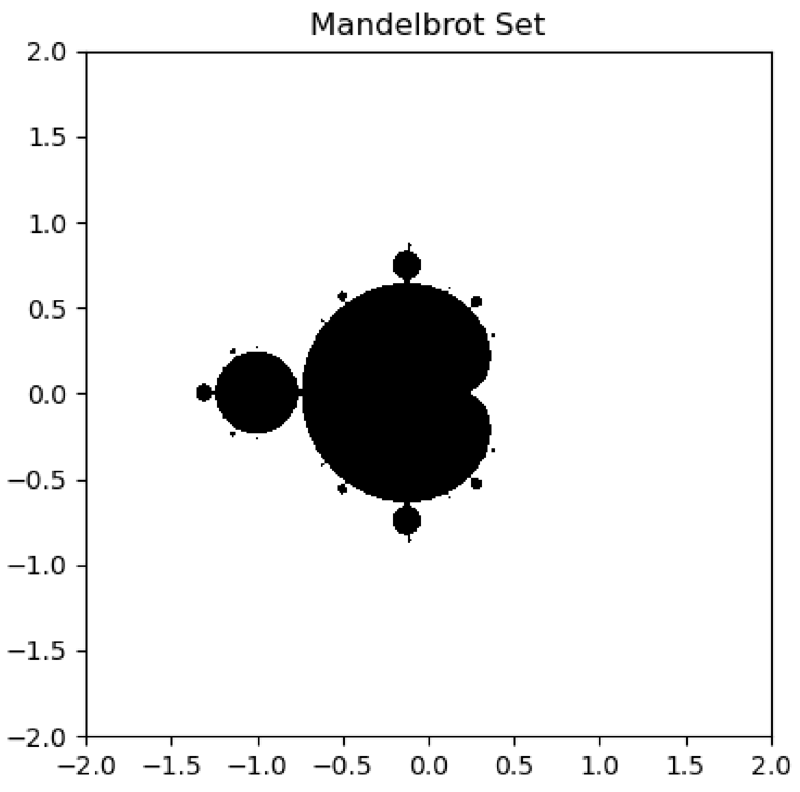

Figure 1: Image of the Mandelbrot Set created in Problem 3.