PHYS-GA2000-PS2

Ahmet Koral Aykın

September 19, 2023

1 Introduction

In this homework set, we are particularly interested in understanding how the numbers are represented and stored in computer and, what kind of errors are associated with that. In addition, we are asked to develop algorithms to visualize two specific problems in physics, namely Mendelbrot set in complex plane and Madelung constant for NaCl.

First of all, the numbers are stored in a computer in a similar way to scientific notation. To put it another way, the significant digits (mantissa) associated with the number are multiplied by the base which is to the power of exponent. Then, the exponent and the mantissa (as well as the sign) are represented by a special arrangement of bits. The way how the numbers are presented and stored in a computer can cause errors during numerical operations. One of the most common sources of error in a calculation is the truncation error which is commonly encountered in subtraction of nearly equal terms. Besides, an error can be completely due to the precision used to represent numbers. It is inevitable to get an error when one tries to store a number with significant digits that contain more information than the precision can represents.

Moreover, it is of great importance to mention how NumPy handles array. As will be demonstrated on the results part, it takes significantly higher amount of time to perform element-wise operations using ***for*** loop.

2 Methods

The methodology used in Q-2 and -1 are essentially the same, which is performing element-wise calculations through multiple ***for*** loops. However, it is also possible to make use of the property of NumPy which enables to perform element-wise operations at once. We are also asked in Q-2 to find Madelung constant in that way in addition to the one using ***for*** loops. To be able to use that property, it is required to first form the matrix which contains integer coordinate positions. For that purpose, **numpy.meshgrid** function is used, which generates coordinate arrays from given one dimensional arrays. It should be noted that although the coordinate points are integer values, the format is set to floating point numbers since integer values are not allowed in exponent.

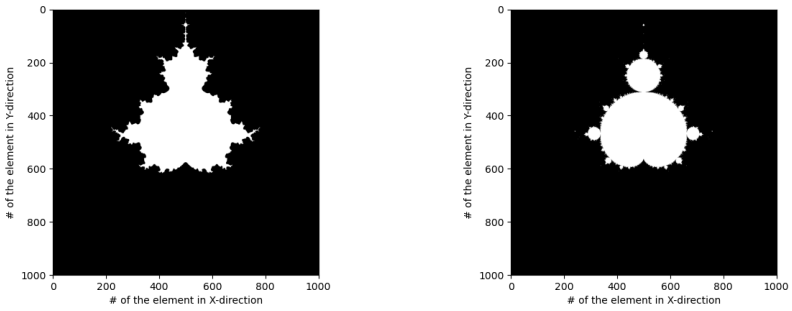
The method utilized to obtain the image of the Mendelbrot set is basically to form a matrix whose elements consisting of 0s and 1s corresponds to a point in discretized complex plane. This matrix is visualized using **matplotlib.pyplot.imshow** which displays 2D data as an image. After defining the function which assigns 1s and 0s according to whether an element in complex plane is in the set or not, respectively.
The mathematical approach adopted in Q-4 to determine the quadratic solver that returns correct solutions is to avoid the subtraction of two nearly equal terms.

3 Results

The answer to the first question is a number of order 10^{-6}, which is very small. The reason why the difference of the same numbers represented in different precision settings is non-zero is that the 32bit floating point representation does not have enough bit compared to the default one which is 64bit. Since the mantissa and the exponent must be an integer, it is more probable for a number stored in 32bit to lose information related to the number.

According to the results of Q3, it can be concluded that multiple for loops has a significant drawback in terms of the time that it takes to perform iterations.

The output image of the Q-3 is presented in Figure 1. As seen, some of the points which are included in Mendelbrot set for the case of N = 10, go out of the set with increasing number of iteration. The color convention used here is opposite to the one in the problem statement. But it is only a matter of replacing the 1s with the 0s.



(a) N = 10 (b) N = 100

Figure 1: The images of Mendelbrot set in the complex plane for two different iteration numbers, N.