Homework 2

Maryam Manzoor Amanullah

September 17, 2024

1 Introduction

The homework provided a foundational introduction to how computers work in terms of numerical data representation. I covered key concepts related to how computers store numbers using different data types, such as integers and floating-point numbers, and how these representations can lead to various types of computational errors. For instance, floating-point arithmetic can result in rounding errors due to the limited precision with which computers store real numbers. This gave me a deeper understanding of issues like overflow and underflow, as well as how small numerical inaccuracies can propagate and affect the outcome of computations, especially when performing many operations in sequence.

The homework allowed me to explore practical tools in NumPy, a Python library for numerical computing. I learned how to use a wider range of NumPy functions for matrix operations and data manipulation.

I saw the comparison between using NumPy's vectorized operations and traditional for loops in Python. NumPy's operations are optimized for performance, often running in parallel, especially for matrix computations (such as the one done for Q3). By leveraging parallel processing**, NumPy can handle operations on entire arrays or matrices simultaneously, leading to significantly lower computational time.

2 Methods

2.1 Question 1

There was a difference between the original number inputted into the computer and the floating point representation of the number. This difference arises because floating-point numbers can't precisely represent all real numbers due to the limited number of bits. In binary, numbers like '100.98763' can't be represented exactly and are represented as a fraction in the 32 bits of space because the binary representation has to be truncated after 23 bits in the mantissa. As a result, the number stored in memory was an approximation of the true value.

2.2 Question 2

In this homework question I found the smallest epsilon value by adding the epsilon to the 1 in 64 bit and 32 bit respectively and if the result was not one I would divide the epsilon by 2 and reiterate. I divided by 2 since the way numbers are stored is in binary.

In the next part I multiplied a large number by 2 to find the largest number before the computer reads the number as infinity. I did similar thing and divided a number by 2 until the computer read it as 0

2.3 Question 3

Solving for the Madelung constant with the meshgrid function in NumPy took 1/10 of the time it took using for loops. This is because of how I can use vectorized computations in meshgrid. Vectorized computations refer to performing operations on entire arrays or matrices (vectors) at once, instead of using loops to perform operations element by element.

2.4 Question 4

I used similar operations for this question as I did in Question 3 when using meshgrid.

2.5 Question 5

This question was a great example of recognizing how rounding errors that occur in calculations can give different answers and need to be considered and taken into account.

In creating the function that would be able to solve any quadratic, I was sure to include cases where the determinant was negative. Additionally, to prevent rounding errors I considered two cases, one where b would be greater and equal to zero and the other when it would be less than zero.