Computational Physics / PHYS-GA 2000 / Problem Set 2

Sally Liang

September 17, 2024

Problem 1

The NumPy's 32-bit floating point representation of the number 100.98763 was obtained through the program introduced in the class notes. And for calculating what that 32-bit number is and how much differs from its 32-bit floating point representation Python struct was introduced.

```
PS C:\Users\Sally\Documents\GitHub\phys-ga2000\ps-2> python problem1.py
100.98763 decimal ->
       sign = 0
       exponent = [1, 0, 0, 0, 0, 1, 0, 1]
       mantissa = [1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1]
Reconstructed value from 32-bit representation: 100.98763275146484
Difference between original and 32-bit representation: 2.7514648479609605e-06
```

Problem 2

```
PS C:\Users\Sally\Documents\GitHub\phys-ga2000\ps-2> python problem2.py
Smallest value that can be added to 1 (32-bit): 1.1920928955078125e-07
Smallest value that can be added to 1 (64-bit): 2.220446049250313e-16
```

Problem 3

```
PS C:\Users\Sally\Documents\GitHub\phys-ga2000\ps-2> python problem3.py
By using a for loop:
The value of Madelung constant is 1.7418198158396654
Runtime for using a loop (s): 8.242798699997365

Without using a for loop:
The value of Madelung constant is 1.7418198158362386
Runtime without using a loop (s): 0.20104449999780627

Program without a for-loop is faster.
```

Problem 4

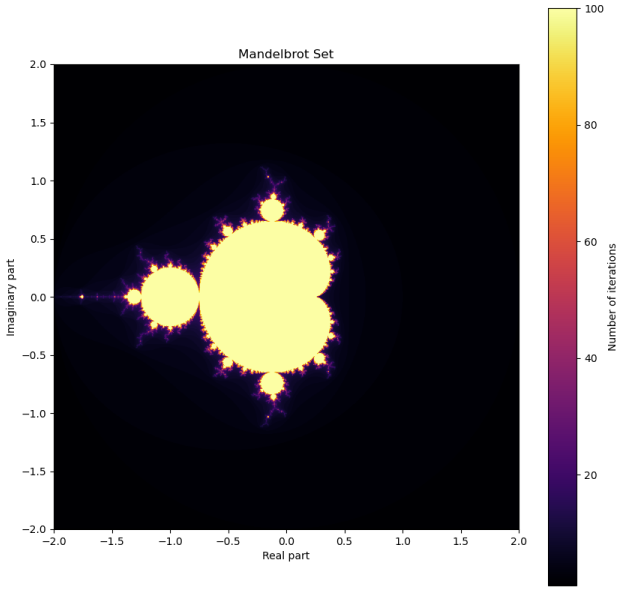

Figure 1: The Mandelbrot Set

Problem 5

Given the quadratic equation in the form:

$$ax^2 + bx + c = 0.$$

This has the solution

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Multiplying by the conjugate of the numerator divided by itself, we get

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \cdot \frac{-b \mp \sqrt{b^2 - 4ac}}{-b \mp \sqrt{b^2 - 4ac}} = \frac{\frac{1}{2a} \cdot \left(b^2 - (-b)^2 + 4ac\right)}{-b \mp \sqrt{b^2 - 4ac}}.$$

Simplifying, this becomes

$$x = \frac{2c}{-b \mp \sqrt{b^2 - 4ac}}.$$

```
PS C:\Users\Sally\Documents\GitHub\phys-ga2000\ps-2> python quadratic.py
(a) Standard formula solutions: x1 = -9.999894245993346e-07, x2 = -999999.999999
(b) Alternative formula solutions: x1 = -1.000000000001e-06, x2 = -1000010.5755125057
```

The solutions computed by different methods might not be the same due to limitations in floating-point arithmetic. The subtraction of similar order numbers can have substantial round off error. In the case where $|b| \gg |ac|$, in either of these methods one of the roots is accurate, but the other will have round off errors.

For part (c) We can obtain more accurate solutions by combining half of each method above, the modified solutions are (-1000010.5755125057, -9.999894245993346e-07). I assumed that $|b| \gg |ac|$, so the size of the square root term and the b term can be compared directly. Following my thought that the signs of the two terms should be the same, I separated my solution into two cases: $b > 0$ and $b < 0$.

In the positive case, we have

$$x_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

and

$$x_2 = \frac{2c}{-b - \sqrt{b^2 - 4ac}}.$$

In the negative case, we instead have

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

and

$$x_2 = \frac{2c}{-b + \sqrt{b^2 - 4ac}}.$$

```
PS C:\Users\Sally\Documents\GitHub\phys-ga2000\ps-2> pytest test_quadratic.py
========================================================= test session starts =========================================================
platform win32 -- Python 3.12.4, pytest-7.4.4, pluggy-1.0.0
rootdir: C:\Users\Sally\Documents\GitHub\phys-ga2000\ps-2
plugins: anyio-4.2.0
collected 1 item

test_quadratic.py .                                                                                                          [100%]

========================================================== 1 passed in 0.20s ==========================================================
```

Figure 2: Proof of quadratic code passing unit test