Computational Astrophysics HW1

Yi-Hsiang Kuo, 110022506

(Dated: Sep. 29, 2022)

I. EXERCISE 2

For 250 billion stars, each star has 10 **double-precision** variable needed to be stored.

Since double-precision variable is **64-bit** system, so the memory required is simply 10 variable each times 250 billion stars times 64-bits, approximately 1.6×10^{14} bits needed.

II. EXERCISE 3

(1) Since $s = 0$, means the sign is positive, $e = 00001110(2) = 14(10)$ means the exponent is 14, and the mantissa $f = 10100000000000000000000(2) = 5242880(10)$

(2) The full value of A is determined by the coefficient above apply into the formula:

$$x_{float} = (-1)^s \times 1.f \times 2^{e-bias} \tag{1}$$

where $bias = 127$ **for single-precision**, after calculation we can get the value of A is about 1.47×10^{-34}

III. EXERCISE 4

The code and result will be shown in the following figure1&2, or the link **here**

IV. EXERCISE 5

Consider the power series for the exponential of a negative argument:

$$e^{-x} = 1 - x + \frac{x^2}{2} - \frac{x^3}{3} + ... \approx \sum_{n=0}^{N} \frac{(-x^n)}{n} \tag{2}$$

(1)By using python to do the calculation and plot, see FIG. 3 containing the information of e^{-x} and error respect to N, and we can see that the error stop decreasing when N reaches

```
 GNU nano 2.3.1

#!/bin/bash

echo -e "Hello, please type the path: \c"
read path

i=0
j=0

for something in $path/*; do
    if [ -f "$something" ]
    then
        echo "$something is a file"
        i=$((i+1))
    elif [ -d "$something" ]
    then
        echo "$something is a directory"
        j=$((j+1))
    fi
done

echo "There are $i files and $j dirctories"
```

FIG. 1. Exercise 4 code

```
[yhkuo@fomalhaut ~]$ ./hw1.sh
Hello, please type the path: /data/hyang/shared/astro660CompAstro/hw1/
/data/hyang/shared/astro660CompAstro/hw1//dir is a directory
/data/hyang/shared/astro660CompAstro/hw1//file1.dat is a file
/data/hyang/shared/astro660CompAstro/hw1//file2.dat is a file
/data/hyang/shared/astro660CompAstro/hw1//file3.dat is a file
There are 3 files and 1 dirctories
```

FIG. 2. Exercise 4 result

about 20, the reason that error stop varying at $N \approx 20$ is $\frac{1}{20!}$ over the double-precision decimal places of significance.

(I put python code also in **here**, named Method1.py)

(2) For method 2, by exchange the equation to:

$$nth\ term = \frac{-x}{n} \times (n-1)th\ term \tag{3}$$

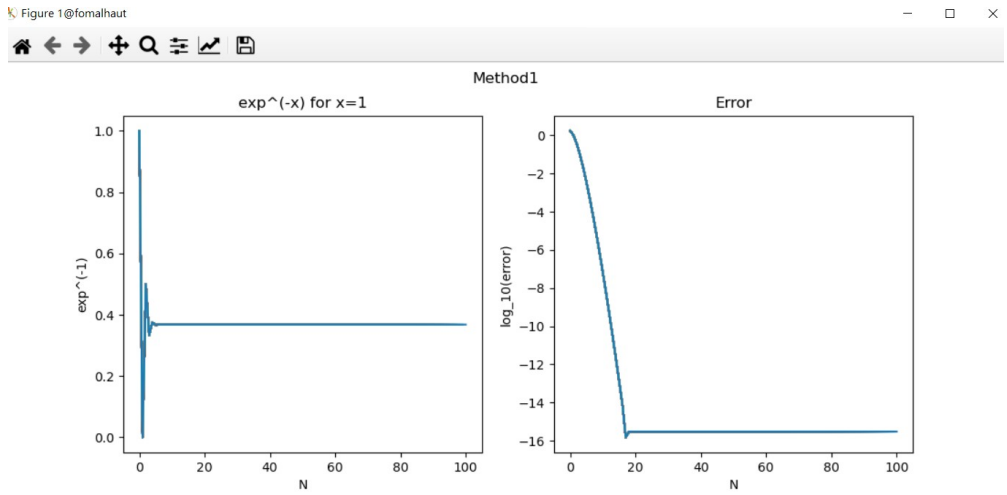

FIG. 3. Exercise 5 Method1

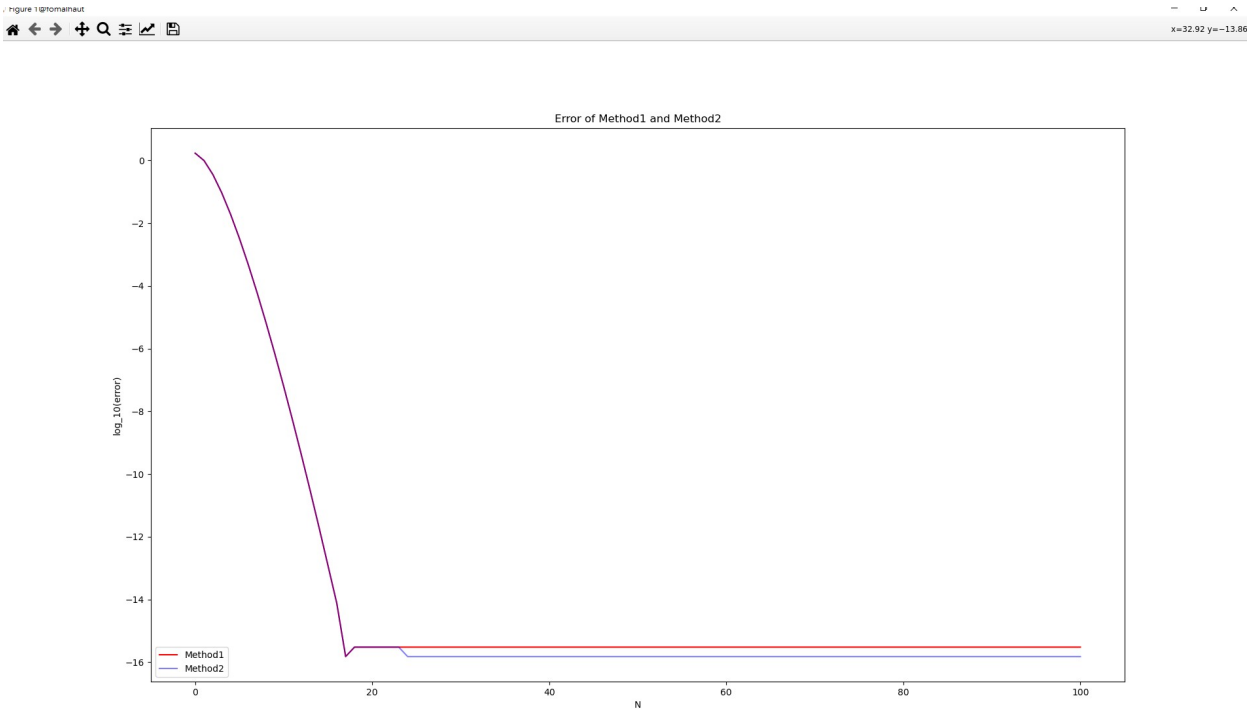

FIG. 4. Exercise 5 Method1&2 comparison

we can see that the behavior of the FIG. 4 does not change a lot before $N > 20$, after that, exists a little deviation which I do not really know the reason. (I put python code also in **here**, named Method2.py)

(3) Last, compute e^x and then do the inverse, repeat it for $x = 1, x = 10, x = 100$, then FIG. 6 is the result.

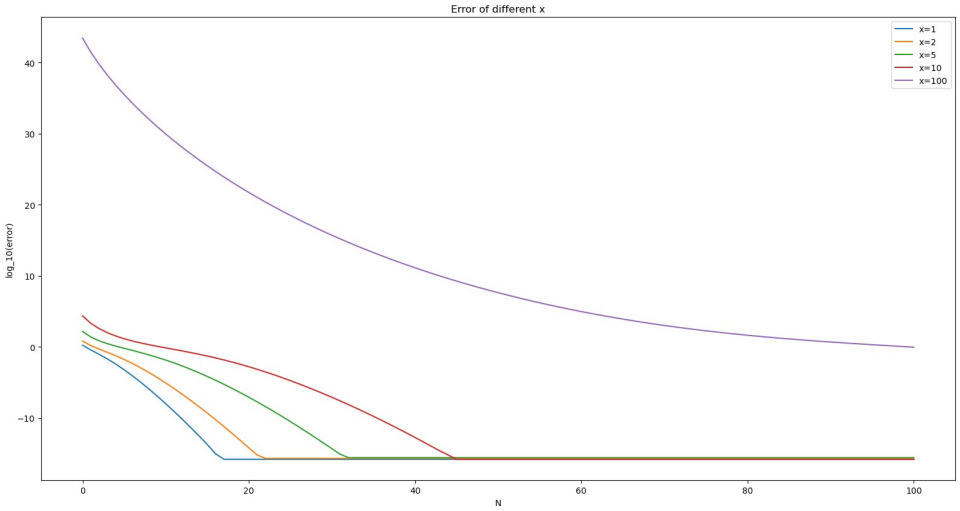

FIG. 5. Exercise 5 Method3 comparison

It seems that Method 3 is the best choice among these 3 methods, and the reason that Method1&2 failed to converge when x is large is we do the calculation with one term positive and then one term negative repeatedly, the summation can not converge to the real answer we want before the factorial term blows up.