

# Computational Astrophysics HW2

Yi-Hsiang Kuo, 110022506

(Dated: Oct. 13, 2022)

## I. EXERCISE 2

(1) I have finished my python code as *pi5.py* and the screen shot of my result entitled *result\_pi5.jpg*, the git hub link is **here**

(2) In addition to do the  $\log_{10}|\epsilon|$  v.s.  $\log_{10}N$ , since I want to see the relative error scales as  $\frac{1}{\sqrt{N}}$  more explicitly, I plot 2 tables as FIG. 1 (or *result\_pi5\_error.jpg* in git hub):

(3) The modified code adding up a line, we need to do  $max\_y \times np.random.rand$  and use this to compute the total area. see FIG. 2 and FIG. 3, and the **git hub link**

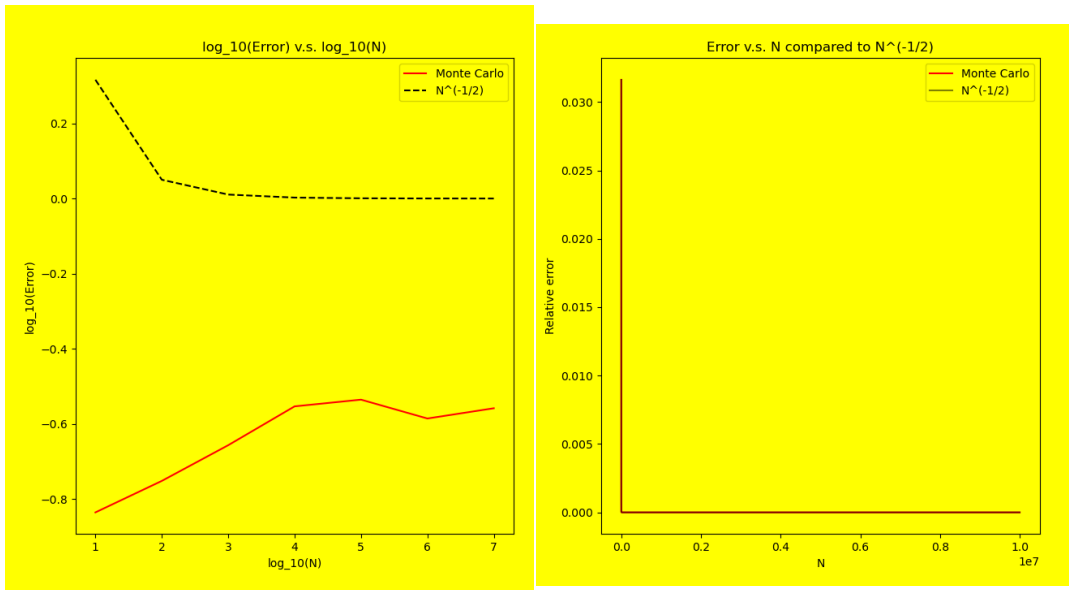


FIG. 1. Exercise2-(2) Relative Error figure

```
(compAstro) [yhkuo@fomalhaut HW2]$ python Exercise2-3.py
2.6719416297003695
The relative error:N=10 Error=0.10759669790585968
The relative error:N=100 Error=0.034349962111602794
The relative error:N=1000 Error=0.03363560621300549
The relative error:N=10000 Error=0.019919982168311312
The relative error:N=100000 Error=0.0013414901143203872
The relative error:N=1000000 Error=0.0019781111376386273
The relative error:N=10000000 Error=0.00030429938847936766
```

FIG. 2. Exercise2-(3) Relative Error

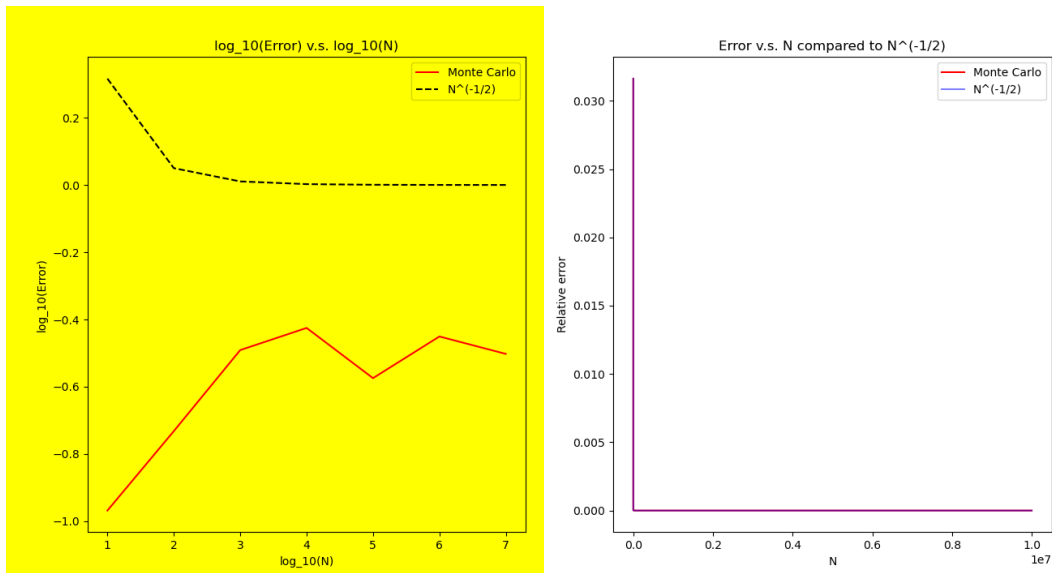


FIG. 3. Exercise2-(3) Relative Error figure

## II. EXERCISE 3

(1) Find the optimal  $N$  where the minimum total error occurs:

$$\epsilon_{tot} = \epsilon_{tr} + \epsilon_{ro} \approx \frac{1}{N^2} + \epsilon_m \sqrt{N} \quad (1)$$

by doing differentiation, we can find that  $N = \epsilon_m^{-\frac{2}{5}}$  has extreme value, hence the optimal  $N$  for 32-bit and 64-bit are  $1.58 \times 10^3$  and  $1.0 \times 10^6$  respectively.

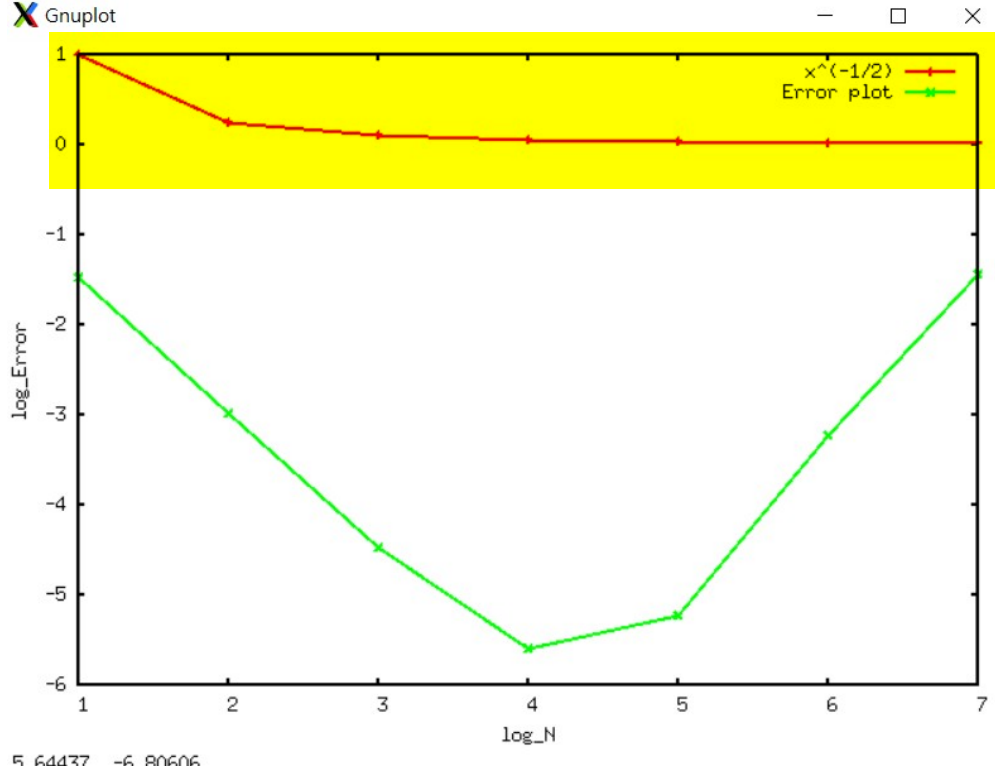


FIG. 4. Exercise4-1 Relative Error figure

(2) For Simpson's rule:

$$\epsilon_{tot} = \epsilon_{tr} + \epsilon_{ro} \approx \frac{1}{N^4} + \epsilon_m \sqrt{N} \quad (2)$$

by doing differentiation, we can find that  $N \approx 1.59 \times \epsilon_m^{-\frac{2}{9}}$  has extreme value, hence the optimal  $N$  for 32-bit and 64-bit are  $5.71 \times 10^1$  and  $3.43 \times 10^3$  respectively.

(3) Putting the optimal  $N$  back to the total error, for midpoint and trapezoid method, the minimal total error for 32-bit and 64-bit are  $4.38 \times 10^{-6}$  and  $2.00 \times 10^{-12}$  respectively; for Simpson's rule, the minimal total error for 32-bit and 64-bit are  $8.50 \times 10^{-7}$  and  $6.58 \times 10^{-14}$  respectively.

### III. EXERCISE 4

(1) I have completed the *pi4.f90* and the **git hub link here** and the **output file here**, with another figure seeing that whether the error scales like  $\frac{1}{N^2}$ , and FIG. 4. is the result.

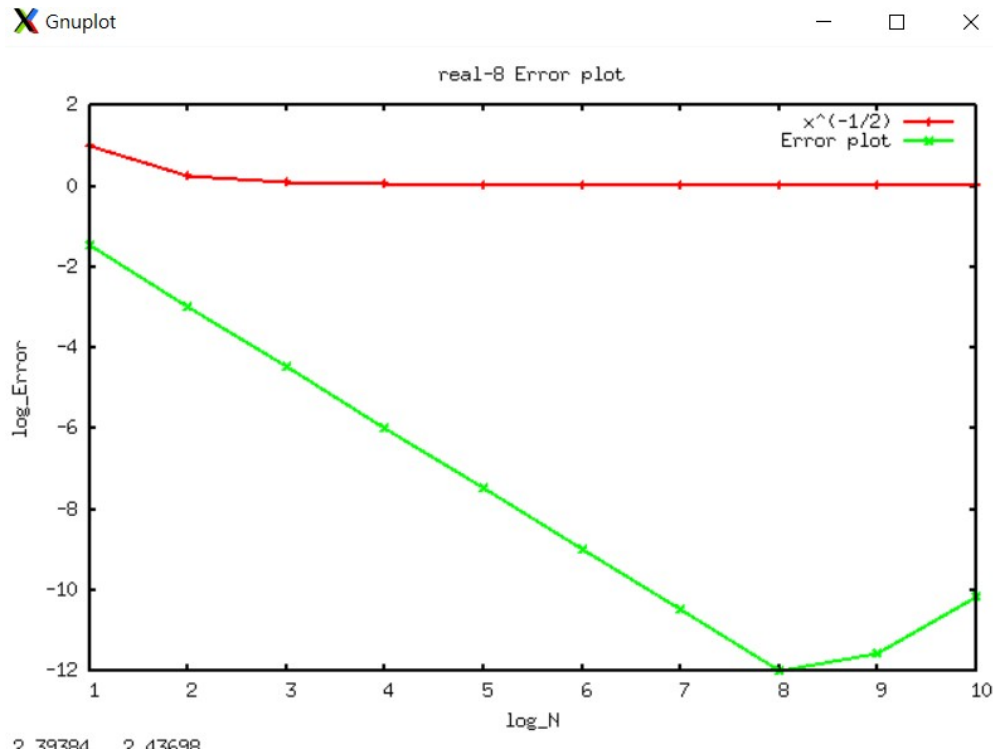


FIG. 5. Exercise4-2 real(8) Relative Error figure

(2) Since we want to see the difference between real(4) and real(8), we need to set our compiler using **-fdefault-real-8** while doing the gfortran command. On top of that, according to my answer to Exercise3, the optimal  $N \approx 10^6$  and the minimal error  $\epsilon_{tot} \approx 2.00 \times 10^{-12}$ , this properties can be seen in FIG.5, where  $N \approx 10^8$ (?) and minimal error indeed  $\epsilon_{tot} \approx 10^{-12}$ .

(3) for trapezoid method, do the same thing but change the original code a little bit. **The code link is here**, just alter the **calculate integral part to trapezoid form**. The results are following figures FIG. 6. and FIG. 7., similar to midpoint method.

(4) for Simpson's method, do the same thing but change the original code a little bit. **The code link is here**, just alter the **calculate integral part to Simpson's form**. The results are following figures FIG. 8. and FIG. 9., according to my answer to Exercise3, **(32-bit)** the optimal  $N \approx 5.71 \times 10^1$  and the minimal error  $\epsilon_{tot} \approx 8.50 \times 10^{-7}$ , **(64-bit)** the optimal  $N \approx 3.43 \times 10^3$  and the minimal error  $\epsilon_{tot} \approx 6.58 \times 10^{-14}$ . **(result weird?)**

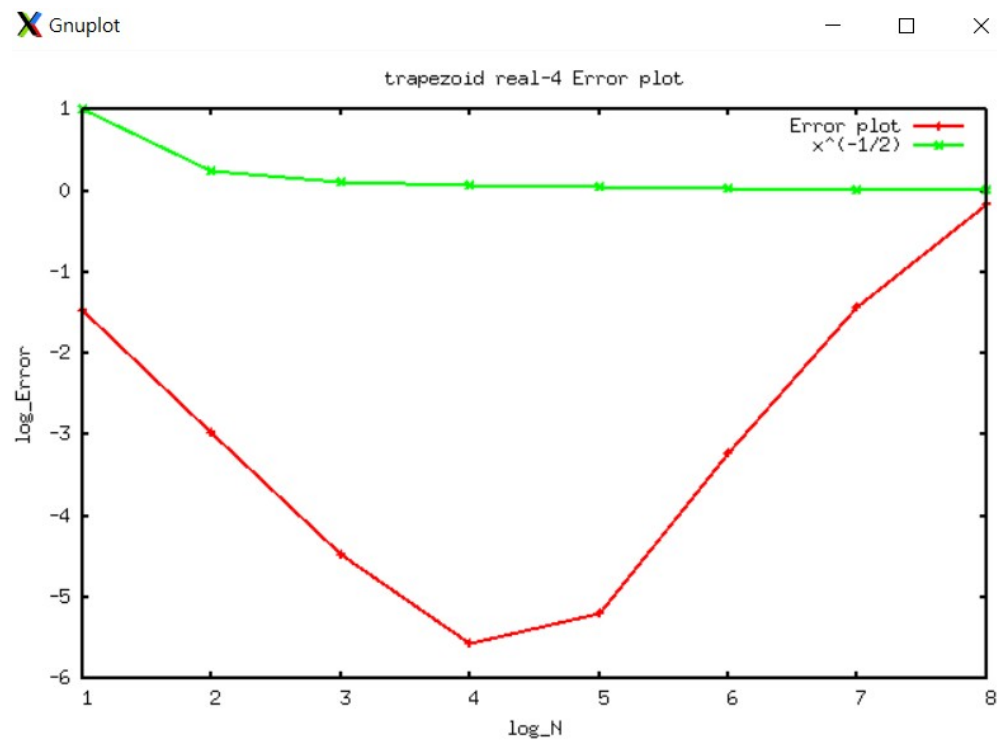


FIG. 6. Exercise4-3 real(4) Relative Error figure

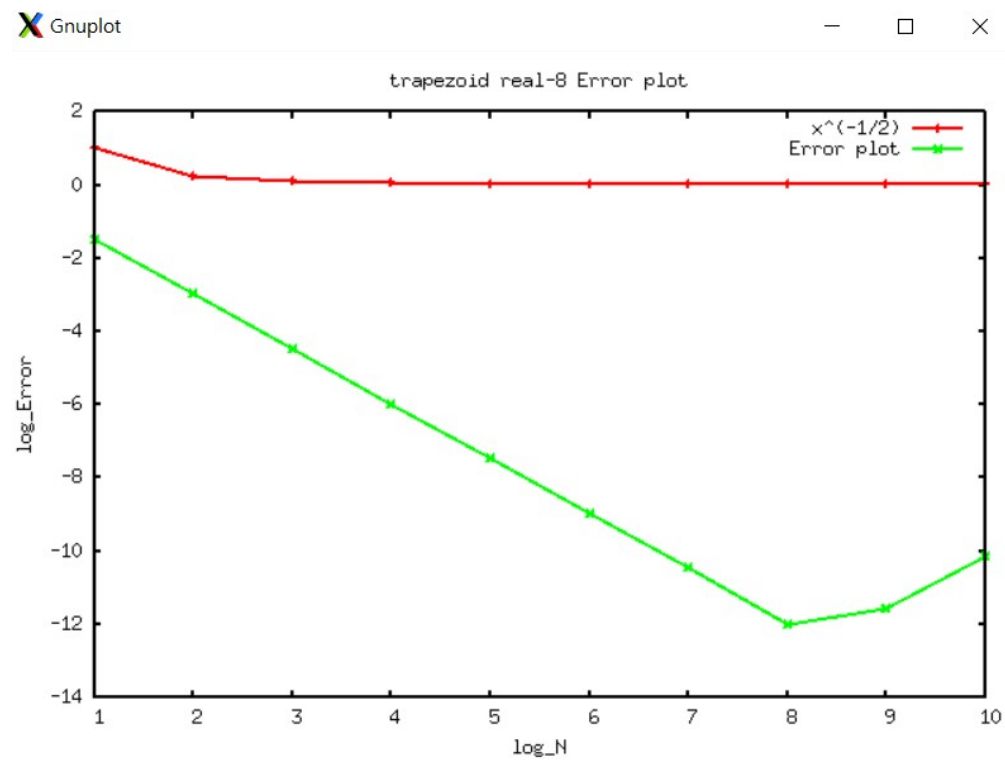


FIG. 7. Exercise4-3 real(8) Relative Error figure

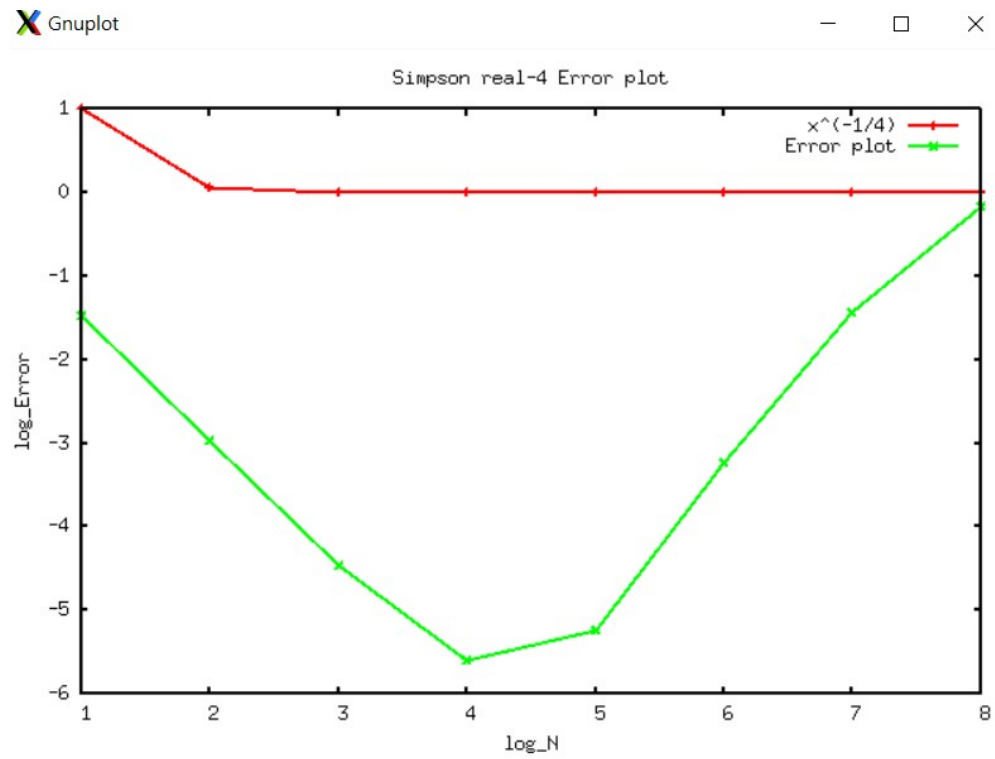


FIG. 8. Exercise4-4 real(4) Relative Error figure

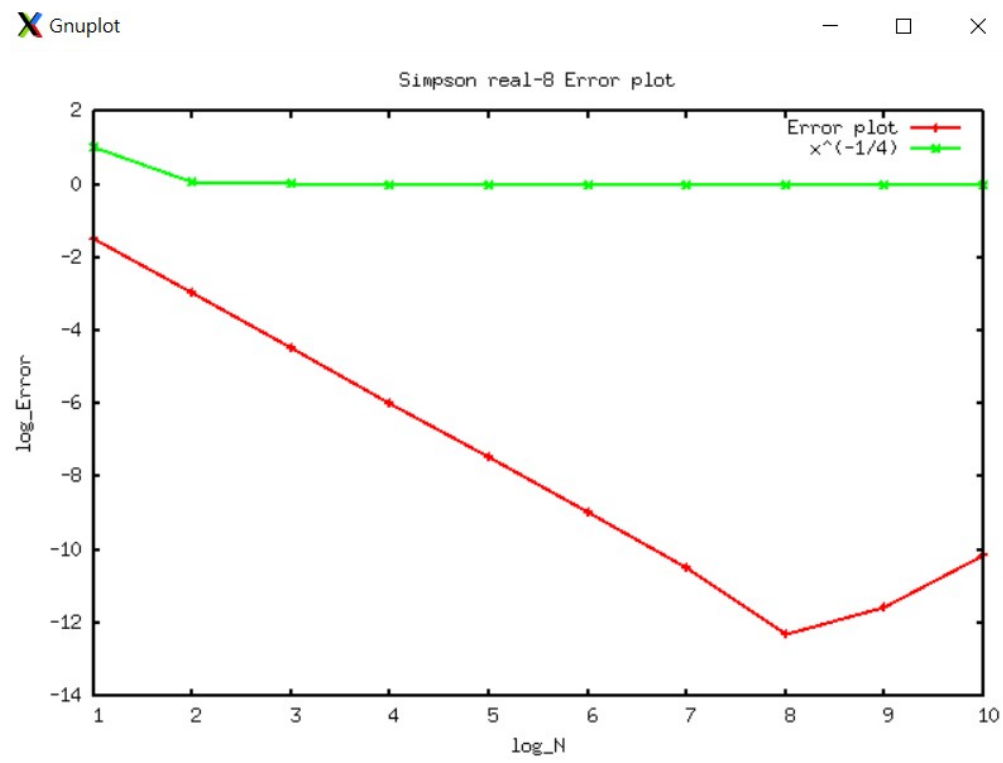


FIG. 9. Exercise4-4 real(8) Relative Error figure

#### IV. EXERCISE 5

(1) To find the optimal  $h$  which minimize the total error (64-bit):

$$\epsilon_{tot} = \epsilon_{tr} + \epsilon_{ro} \approx \frac{f^{(2)}}{2}h + \frac{\epsilon_m}{h} \quad (3)$$

Since  $f^{(2)} \approx 1$ , so after differentiation we find that **when**  $h \approx 4.47 \times 10^{-8}$

(2) To find the optimal  $h$  which minimize the total error of (64-bit) of **the central-difference algorithm**:

$$\epsilon_{tot} = \epsilon_{tr} + \epsilon_{ro} \approx \frac{f^{(3)}}{24}h^2 + \frac{\epsilon_m}{h} \quad (4)$$

Since  $f^{(3)} \approx 1$ , so after differentiation we find that **when**  $h \approx 2.29 \times 10^{-5}$

(3) Put the optimal  $h$  back, we can get the minimal total error for both method,  $4.47 \times 10^{-8}$  and  $6.55 \times 10^{-11}$  for method 1 and method 2 respectively, **so method 2 is better behaved**, also the step size  $h$  is smaller for method 2.

#### V. EXERCISE 6