Homework 1

Report :

Question 1:

I have developed three program using Fortran 90. Respectively, they are trap.f90, simpson.f90 ad mc.f90.

In my calculation, I picked up x as 1, and changed the partition number N for the purpose of comparing the efficiency using these three different methods.
For the evaluated function is even, so I only integrated from 0 to 1, and multiply 2 at last.
I took the evaluated function as a independent modulo, named function f(t).
I integrated with the partition number 10, 100, 100, 1000, 10000 in all three program, and display the result each time.

In trap.f90, I used the formula :

$$N_e = \sum_{i=1}^{N} n_i \quad \text{where} \quad n_i = h \cdot \frac{\rho_{i-1} + \rho_i}{2}$$
$$= \sum_{i=1}^{N} \frac{h}{2}(\rho_{i-1} + \rho_i)$$
$$= \boxed{h(\frac{1}{2}\rho_0 + \rho_1 + \rho_2 + \cdots + \rho_{N-1} + \frac{1}{2}\rho_N)}$$

while in simpson.f90, I used these :

$$S_d(N) = f_0 + f_N$$
$$S_o(N) = f_1 + f_3 + \cdots + f_{N-1}$$
$$S_e(N) = f_2 + f_4 + \cdots + f_{N-2}$$

$$I_{[a,b]}(N) = \frac{h}{3}[S_d(N) + 4S_0(N) + 2S_e(N)].$$

In Monte-Carlo Method, I randomly generated N different number, then calculated their values in the function. At last I added them together and divided the sum by N.

The following are my results :

```
➜ myFortran ./trap
 The trapezoidal rule : n =          10 the integral is :   0.595390022
 The trapezoidal rule : n =         100 the integral is :   0.595874250
 The trapezoidal rule : n =        1000 the integral is :   0.595878899
 The trapezoidal rule : n =       10000 the integral is :   0.595879734
 The trapezoidal rule : n =      100000 the integral is :   0.595879555
```

```
➜ myFortran ./simpson
 The simpson's rule : n =          10 the integral is :   0.595880091
 The simpson's rule : n =         100 the integral is :   0.595879316
 The simpson's rule : n =        1000 the integral is :   0.595879614
 The simpson's rule : n =       10000 the integral is :   0.595879853
 The simpson's rule : n =      100000 the integral is :   0.595878899
```

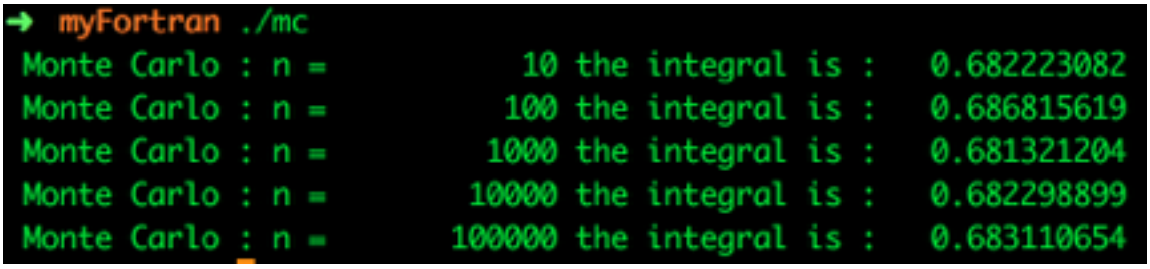
```
➜ myFortran ./mc
 Monte Carlo : n =          10 the integral is :   0.598238528
 Monte Carlo : n =         100 the integral is :   0.591974854
 Monte Carlo : n =        1000 the integral is :   0.592685997
 Monte Carlo : n =       10000 the integral is :   0.595177412
 Monte Carlo : n =      100000 the integral is :   0.596547902
```

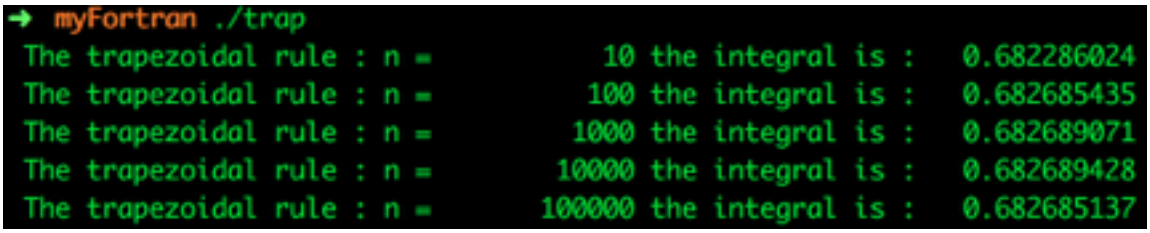
Since we don't know the accurate answer to this integral, I altered the function to the "true" normal distribution, i.e.

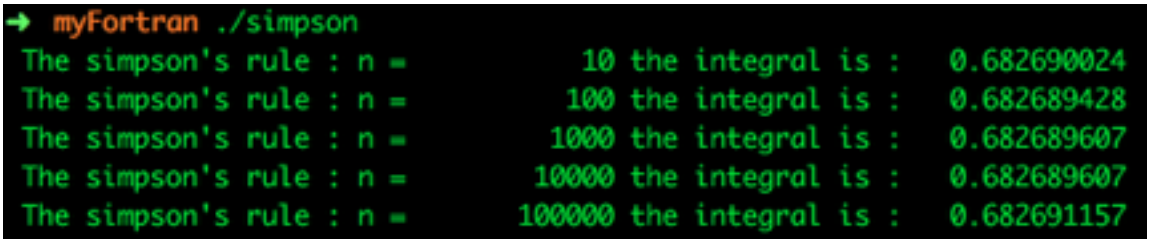
$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

to analyze their efficiency and accuracy.

The following is the altered results :







The accurate answer is 0.68268949. As we can see from the results, the Simpson's rule is the most efficient method, and the trapezoidal method is the second one. This is easy to understand since the Simpson's rule used more information in its calculation. One seemly strange phenomena from these result is that only the trapezoidal method increased its accuracy as n grows. The reason for that may is n is chosen too large to show the trend of increased accuracy.

After I finished this question, I learned that when I developed the program it is better to develop it one functionality at one time. I firstly wrote the situation n = 10, then I added the do-loop to realize multiple n. This thinking is similar to develop a program modulo by modulo, which keeps our program clearly and easy to read and debug.

Question 2:

card.f90

I separated subroutine geneRand and check from the main program for clarity.

I used a two dimensional array to store the card. I generated the card by firstly generating a float number between 0 and 1, then I multiply the some integer to convert it to a integer between 1-4 and 1-13.

I changed the chosen card as 1, and the others as default 0. In check subroutine, if the same card detected, I changed the flag to 1, and generated one more card until the new card is different from the previous ones.

Then I count the number in each row(each row stands for one color). If one row has more than three cards, I increased M by 1.

When I chose N = 10000, my program shows M = 3538.

After I finished this small program, I felt more confident in using Fortran to write program as I like. It is very beneficial for me.