Numerical Analysis and Applications—HW2

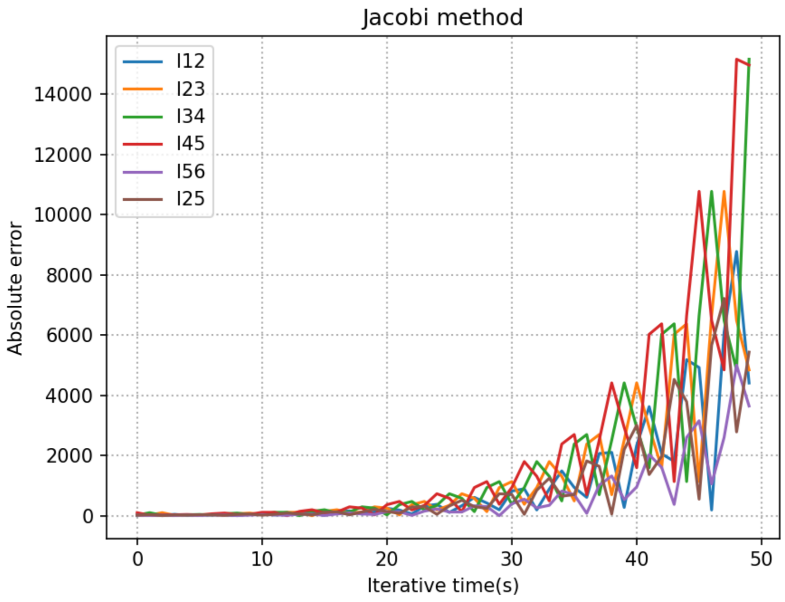
B083040041 張嵩禾

Contents

☐ Question1

Jacobi method relaxation for Figure 1 Circuit

jacobi.py

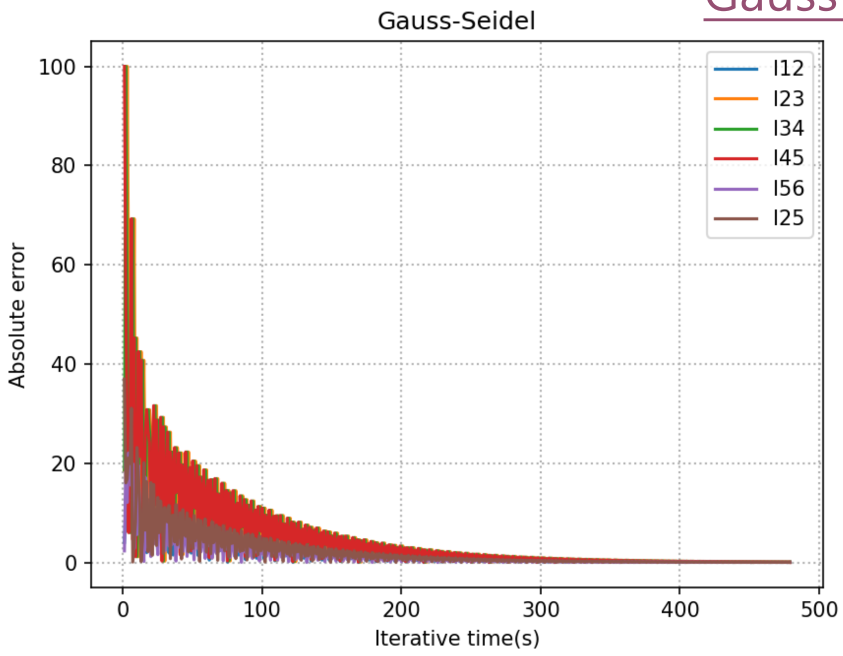


50 iterations.

Through the Jacobi method, it can be observed that the divergence phenomenon becomes evident after 50 iterations, and therefore, convergence cannot be achieved.

Gauss-Seidel method for Figure 1 Circuit

Gauss-Seidel.py



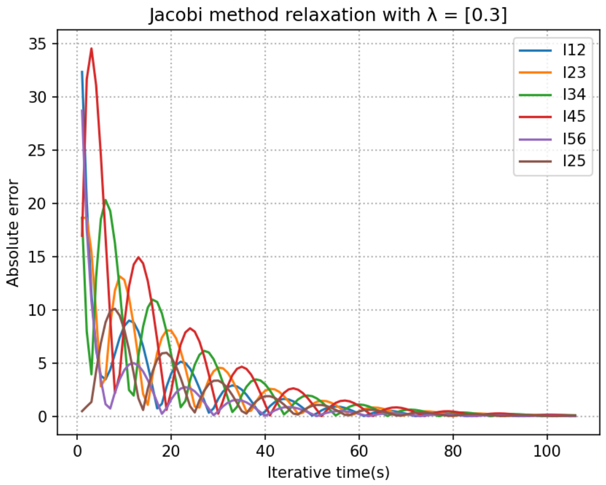
479 iterations.

However, by using the Gauss-Seidel method, convergence can be achieved after 479 iterations.

□ Question1

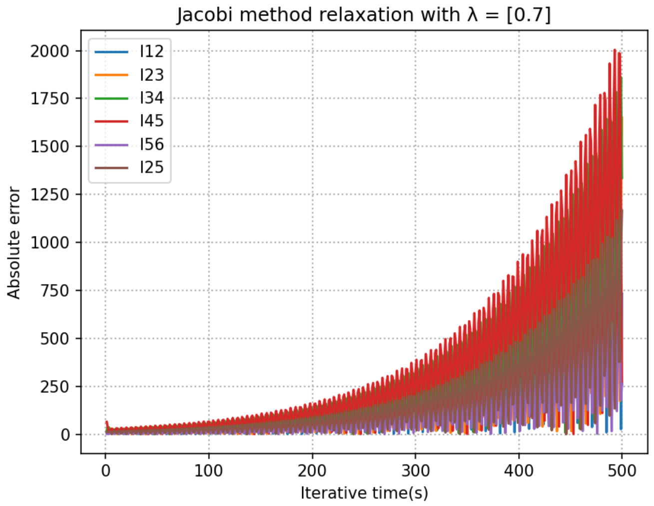
[jacobiwithld.py]

Jacobi method relaxation with λ = [0.3]



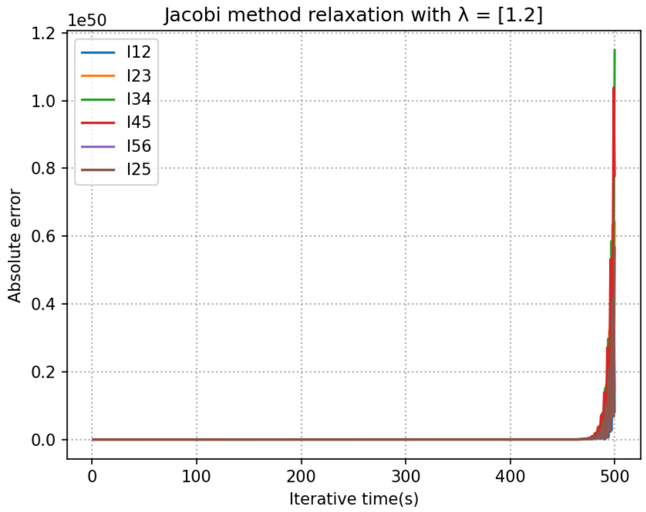
106 iterations.

Jacobi method relaxation with λ = [0.7]



500 iterations.

Jacobi method relaxation with λ = [1.2]

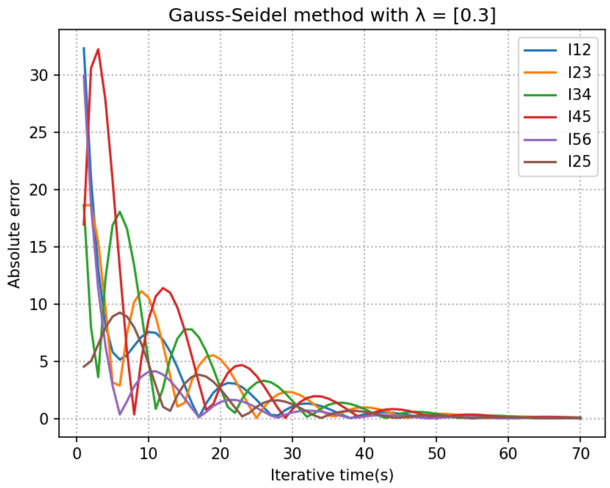


500 iterations.

□ Question1

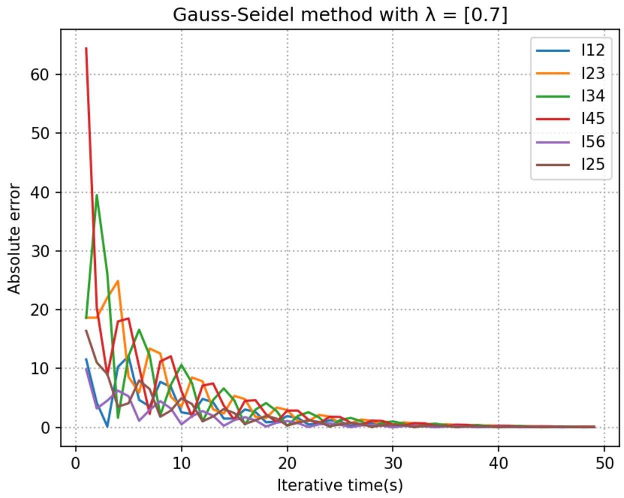
[Gauss-Seidelwithld.py]

Gauss-Seidel method relaxation with λ = [0.3]



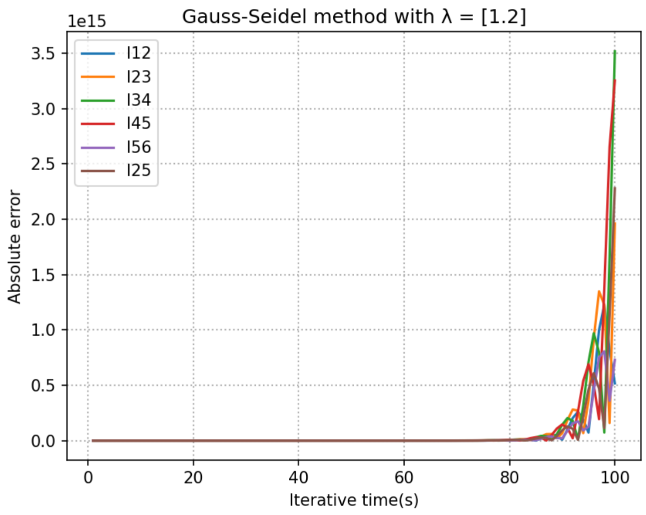
70 iterations.

Gauss-Seidel method relaxation with λ = [0.7]



49 iterations.

Gauss-Seidel method relaxation with λ = [1.2]



100 iterations.

□ Question1

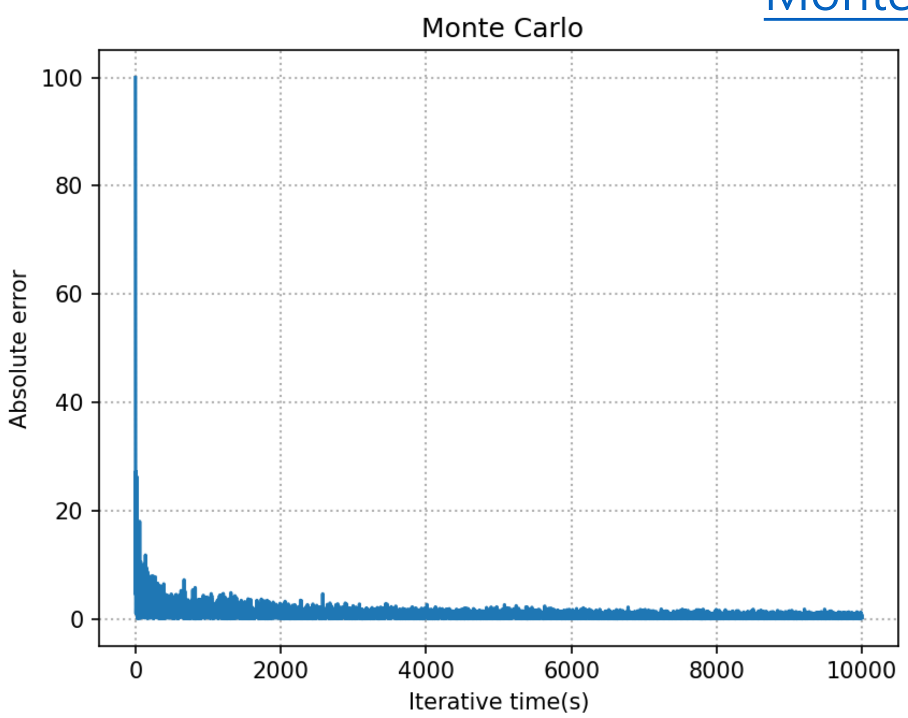
"A discussion on the impact of weights on convergence"

From the results, it can be inferred that under-relaxation with λ = 0.3 is necessary for effective convergence when using the **Jacobi method**, as it slows down the convergence speed to control the error.

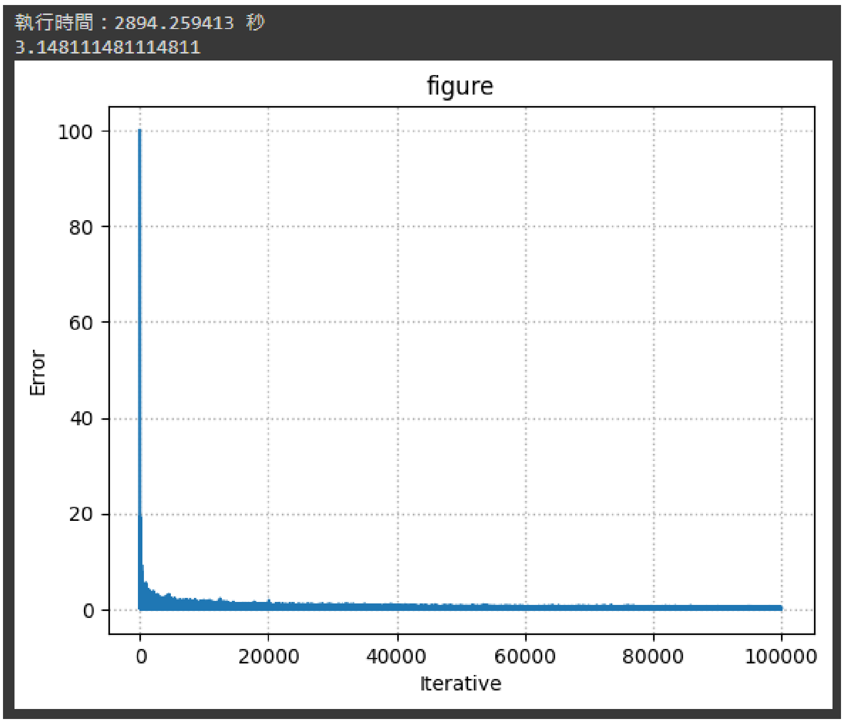
However, when using the **Gauss-Seidel method**, a value of λ = 0.7 results in faster convergence compared to λ = 0.3. Therefore, even though it is still under-relaxation, a slightly larger step size in the iteration process can be used to achieve faster convergence

◻ Question2

Monte-Carlo.py



10000 iterations.



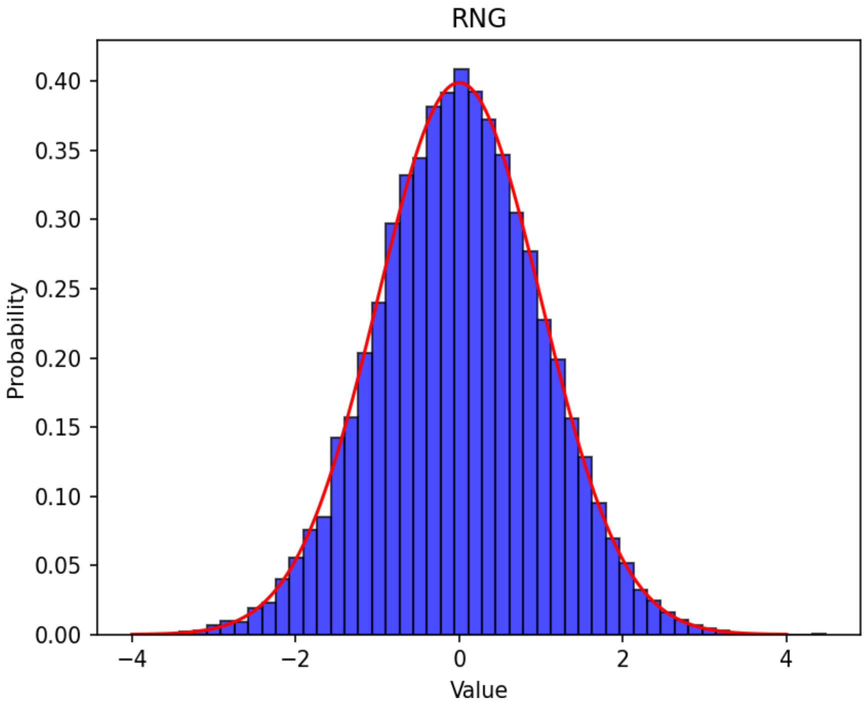
100000 iterations.

Using the **Monte Carlo method** with **10,000 random numbers**, I obtained a value of **PI equal to 3.1355**. Although this is not the exact value of PI, we can observe that taking more random samples can lead to more accurate results. After computing **100,000 random numbers**, I was able to obtain a more precise value of **PI equal to 3.1481114.** However, this method requires a significant amount of time to perform the iterations, and may not be the most efficient way to obtain PI.

☐ Question3

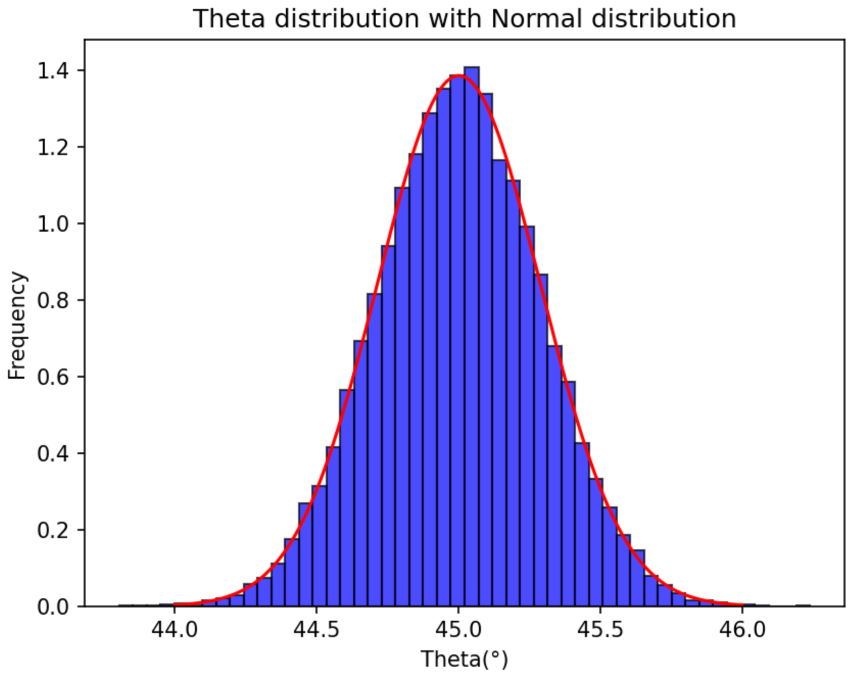
a.

3a.py



b.

3b.py



Mean: 45.001

Standard Deviation: 0.2877

c.

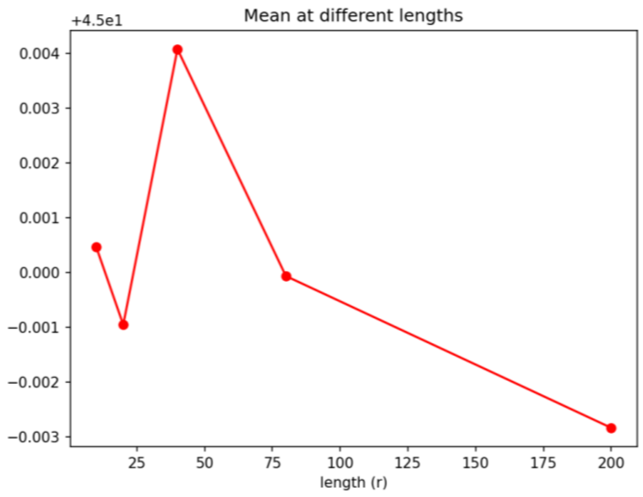
Question3

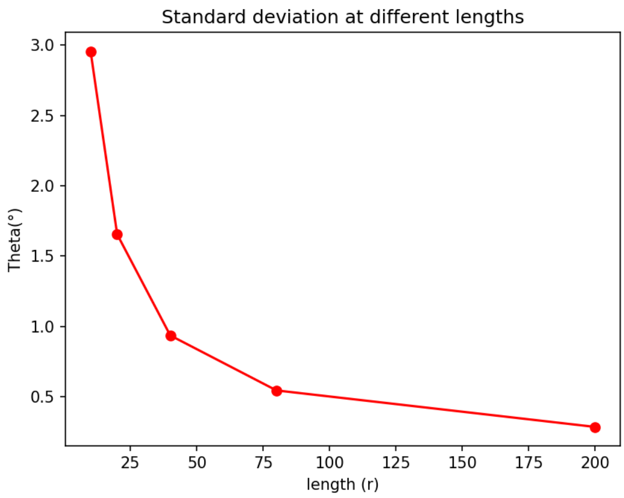
Mean at different lengths

Standard deviation at different lengths

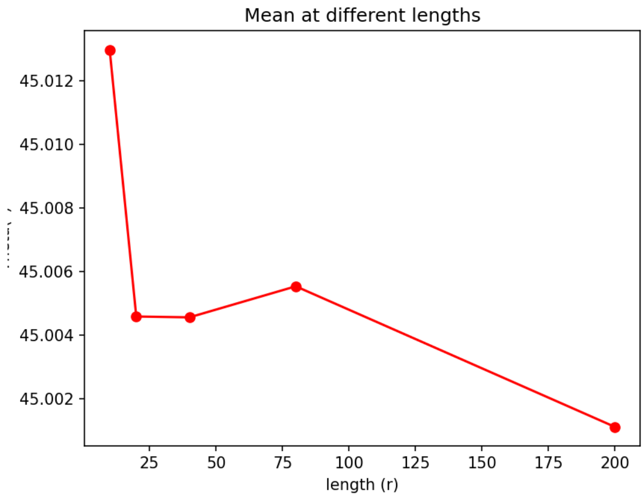
[3c.py](3c.py)

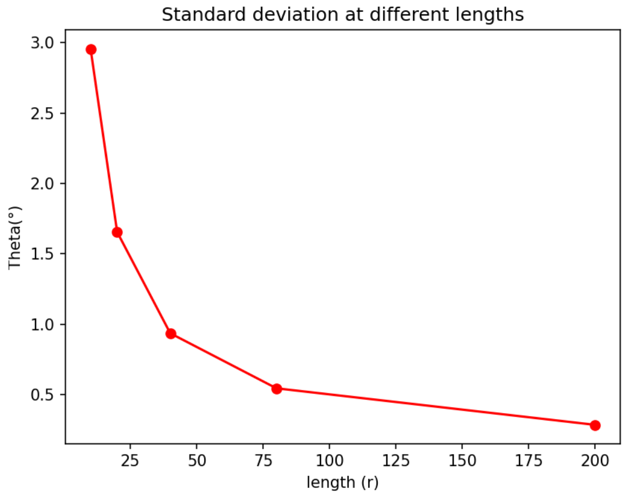
1st test



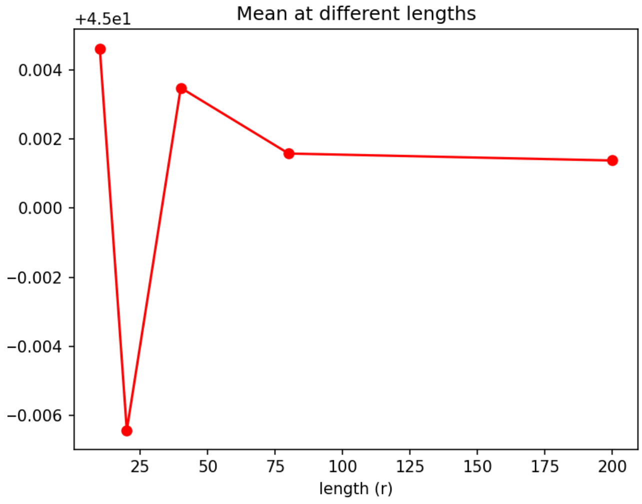


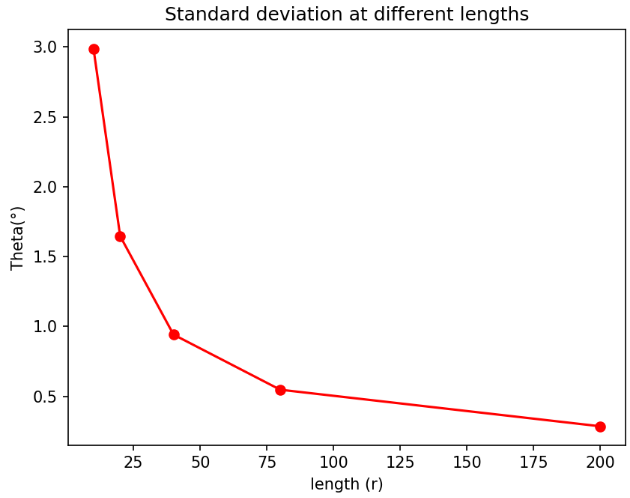
2nd test





3rd test





□ Question3

c.

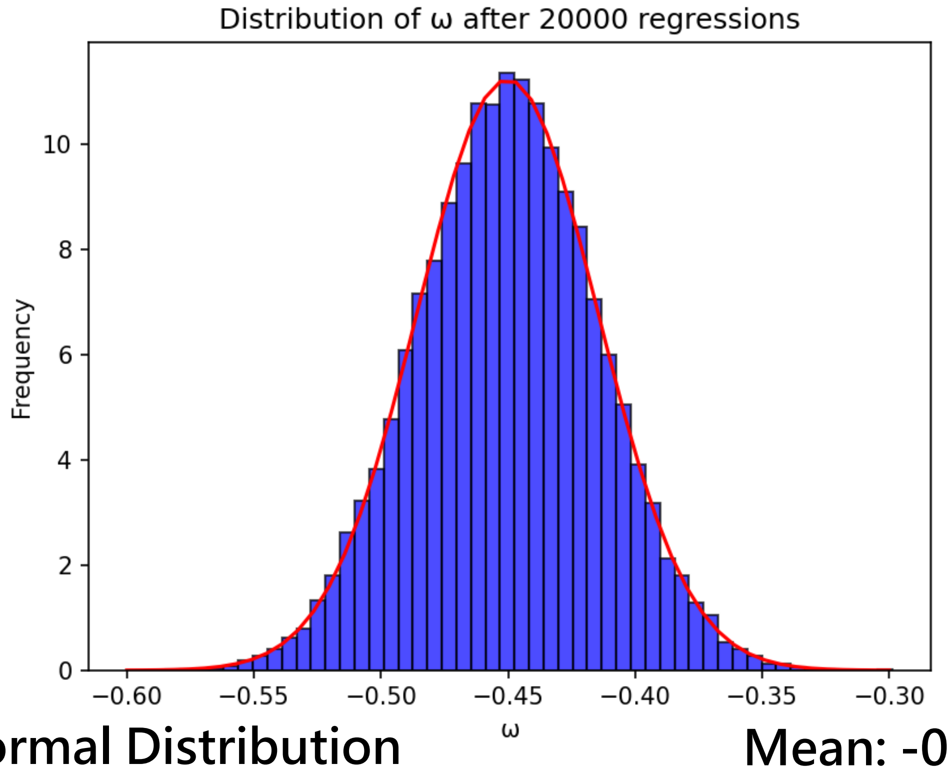
From various tests, we can observe that the results of the **mean are relatively independent of the length**, whereas **the standard deviation decreases as the length increases**.

d.

The conclusion observed from the result plot is that the longer the length of r, the smaller the standard deviation. I believe this is because the noise is added to the x and y axes separately. **If r is longer, it is harder to change its angle.**

e. 3e.py

Distribution of ω after 20000 regressions



Normal Distribution

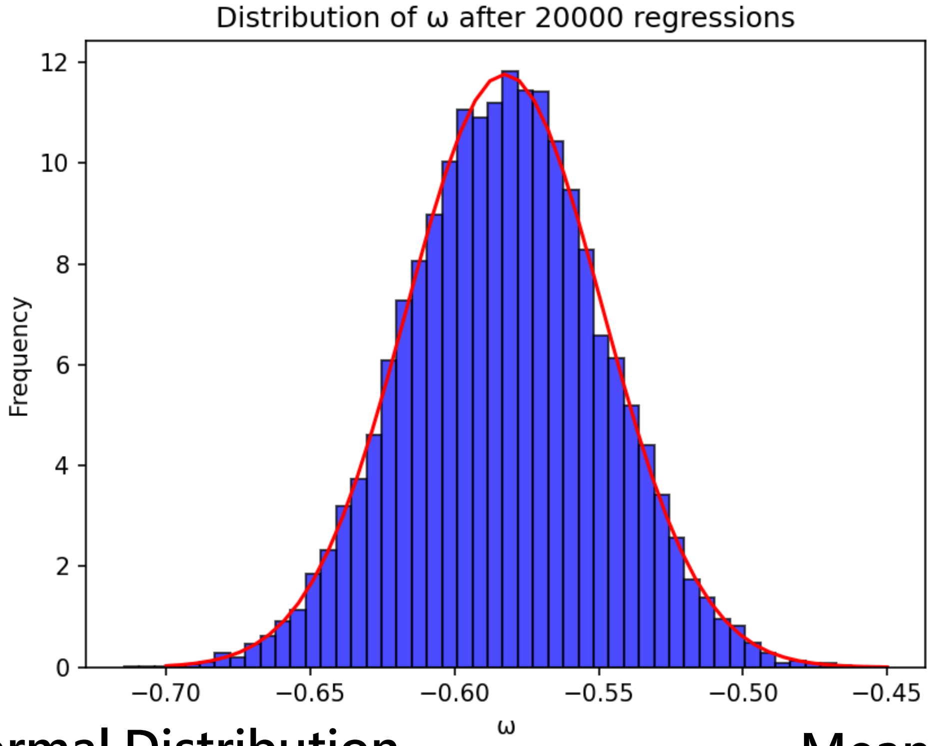
Mean: -0.4502

Standard Deviation: 0.0355

☐ Question3

f. 3f.py

Distribution of ω after 20000 regressions



Normal Distribution

Mean -0.5826
Standard Deviation: 0.0339

Weighting Function:

$$\frac{abs(Si)}{\sum abs(S)}$$

\# Each Signal magnitude

\# The sum of Signal magnitude

From the question 3d, we learned that **a larger length result in a lower standard deviation.** Therefore, I believe that if we assign **larger weights to signals with larger magnitudes**, it can reduce the standard deviation. The results did confirm this.