Assignment4

Lakshya J ee19b035

March 10th, 2021

Introduction

In this assignment we aim to understand the different methods by which Fourier Series coefficients can be obtained for a series and which method is more suitable for a given function.
I start the code by declaring the essential libraries required for the code to run. They include:

```python
import numpy as np
import scipy.integrate as integrate
import scipy.special as special
import matplotlib.pyplot as plt
```

After this I explicitly define the functions $f_1(x) = exp(x)$ and $f_2(x) = cos(cos(x))$.
After this I also define the functions required to find the a_0, a_n and b_n coefficients for both the functions using the integration method.

Question 1

In this question I plot both the functions whose images obtained are as below. The first plot is of $exp(x)$ on the semi-log axis and the second one is $cos(cos(x))$ from -2π to 4π.

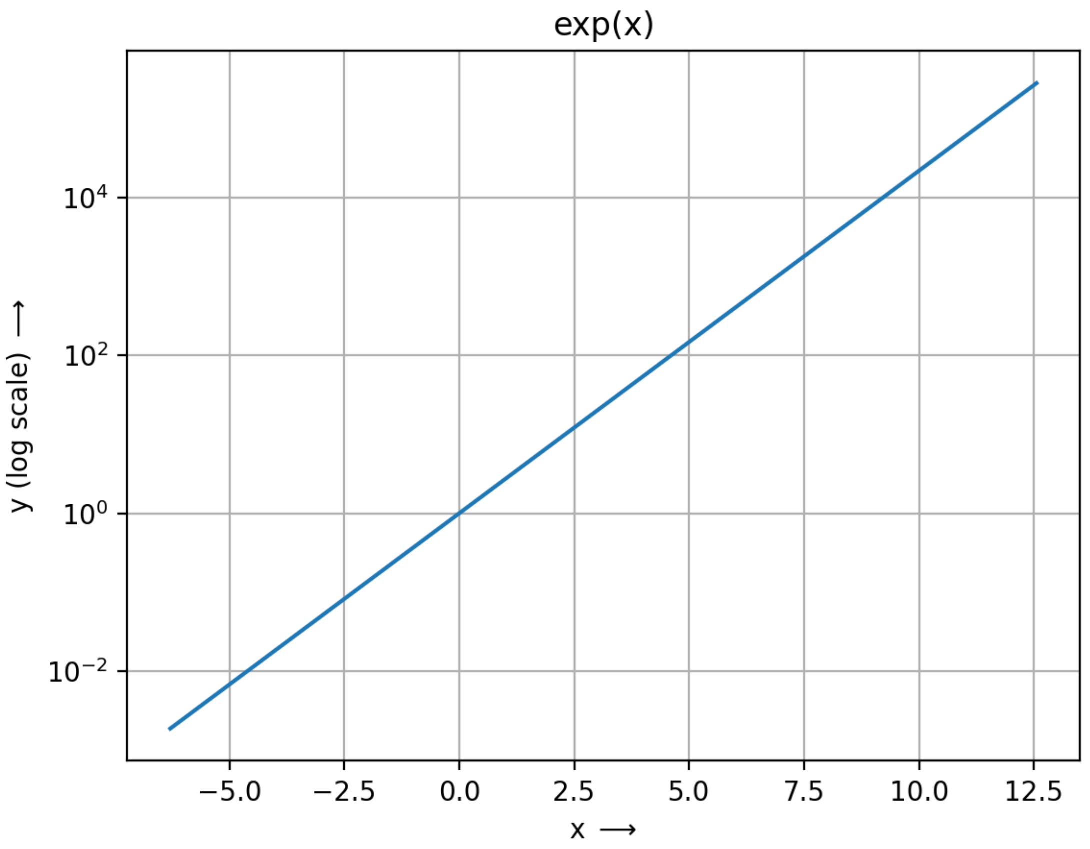

Figure 1: Plot of $exp(x)$ in semi-log scale

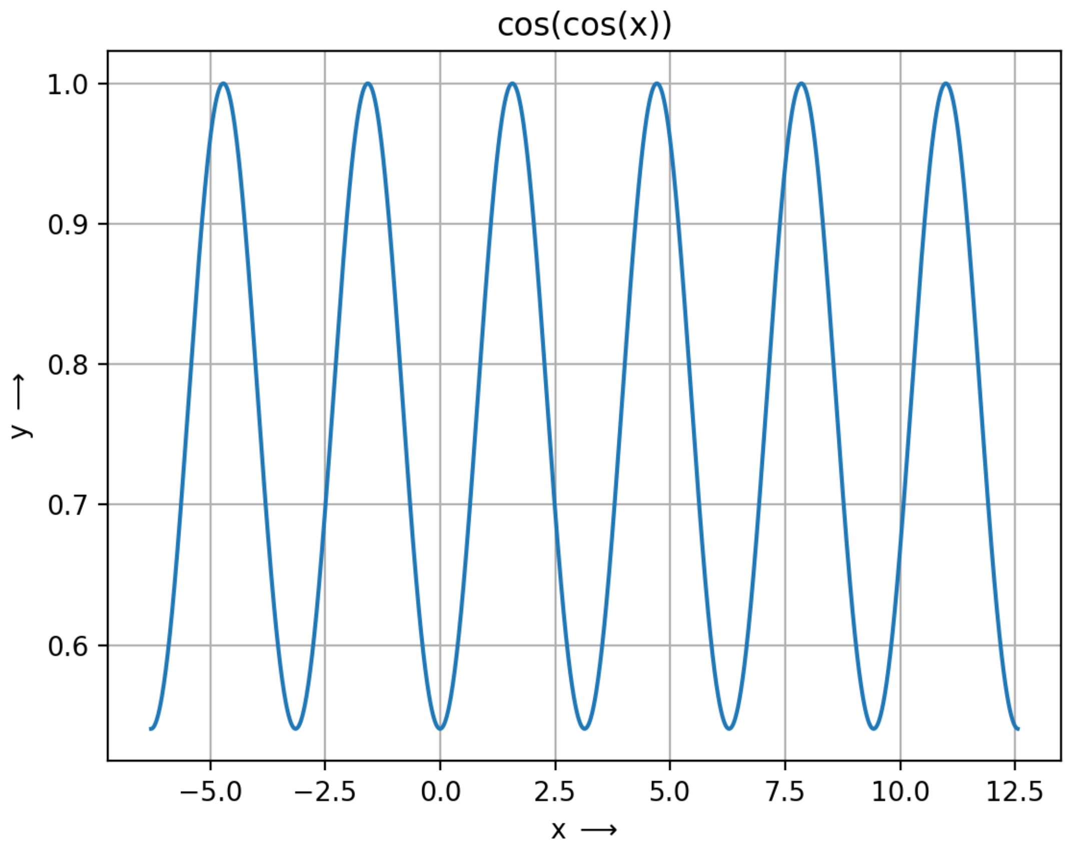

Figure 2: Plot of $cos(cos(x))$ in sem-log plot

Question 2

In this question I find the first 51 coefficients of both the functions. As an example below I find the zeroth coefficients of both the functions using the function $ao()$. Since it returns a list with multiple parameters. Since I need the first value, I extract it as follows:

```
ao1_, ao2_ = ao()
ao1 = ao1_[0]
ao2 = ao2_[0]
```

Question 3

In this question I plot the first 51 coefficients of the functions. Since it has been asked to plot the values against n, I plot a_n values with red circles and b_n values with green circles. I do it for each in both the semi-log and log-log plots.

0.1

$cos(cos(x))$ is an even function. Since sine coefficients of a Fourier transform introduces odd parts, the value of b_n is observed to be almost zero.

0.2

$cos(cos(x))$ is a composed of sines. It can be expressed within a few harmonics of the fundamental frequency, therefore, as n increases, the magnitude of its coefficients tends to zero. Since $exp(x)$ is a monotonically increasing function, the magnitude of its Fourier coefficients are sufficiently high. It would need more harmonics to express the function properly. Hence the decay in the magnitude of $exp(x)$ is slower than that of $cos(cos(x))$.

0.3

log-log plot for $exp(x)$ looks linear since the coefficients and n have an exponential dependence on each other. Similarly in $cos(cos(x))$ we can say that the coefficients have an exponential dependence on n.

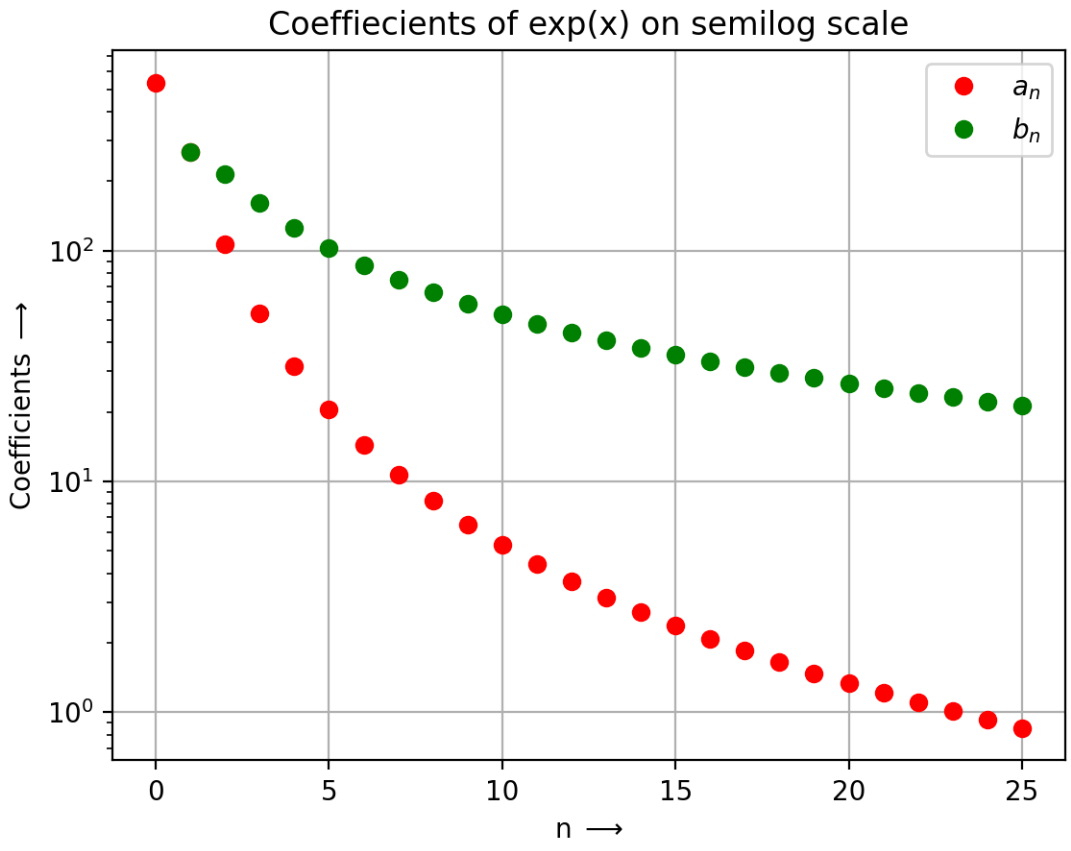

Figure 3: Plot of coefficients of $exp(x)$ vs n in semi-log scale

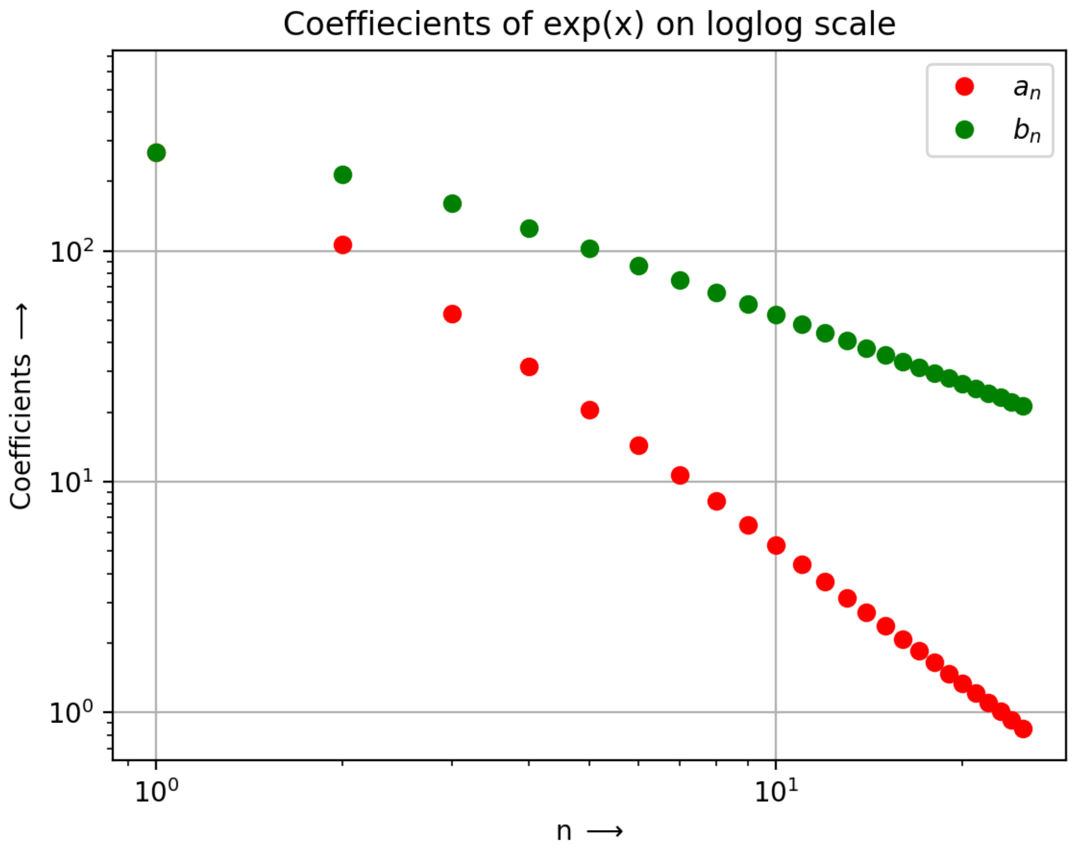

Figure 4: Plot of coefficients of $exp(x)$ vs n in log-log scale

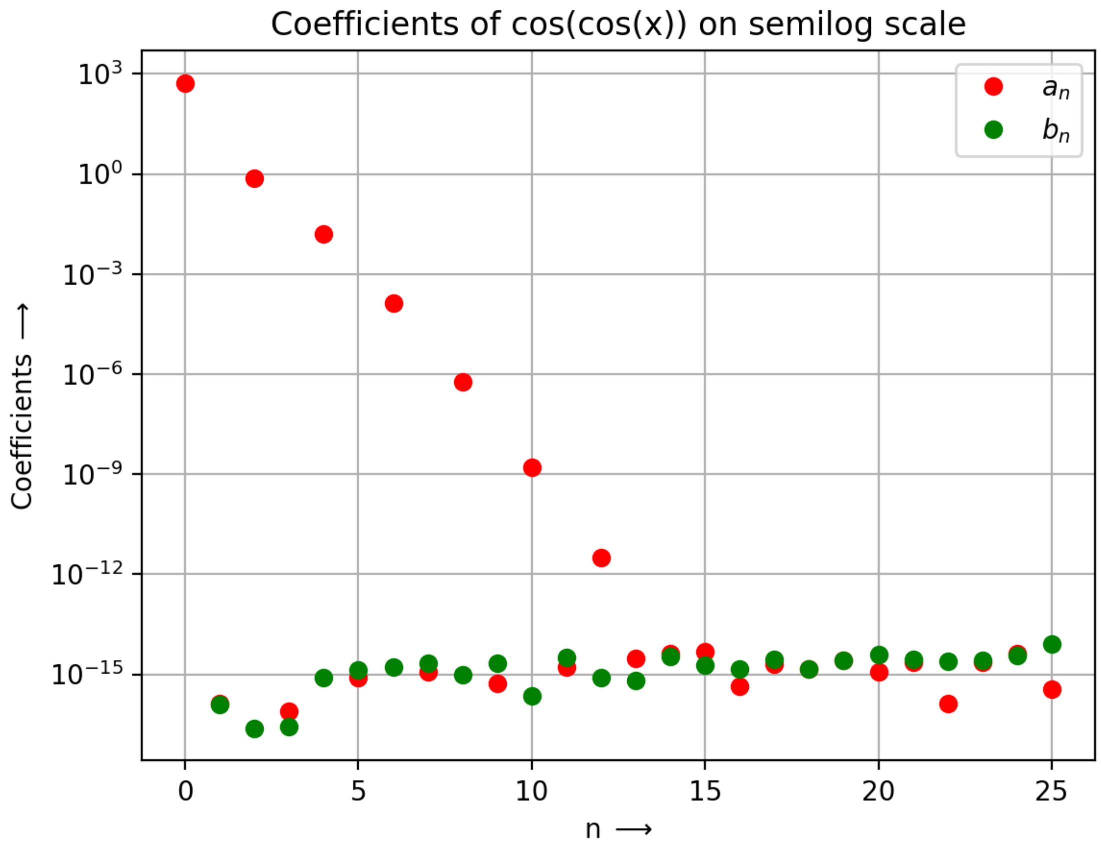

Figure 5: Plot of coefficients of $cos(cos(x))$ vs n in semi-log scale

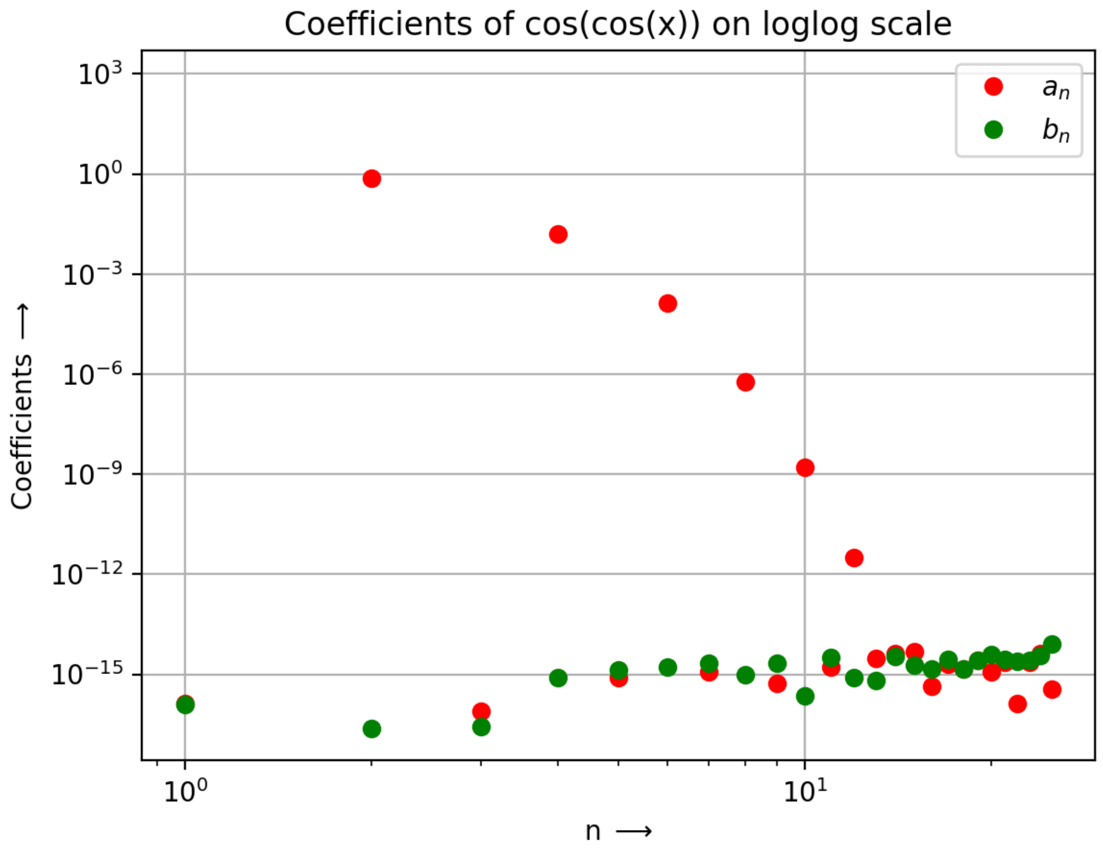

Figure 6: Plot of coefficients of $cos(cos(x))$ vs n in log-log scale

Question 4

In this question I set up the matrix A required for the Least Squares Approximation of the Fourier Coefficients of both the functions.

Question 5

In this question I plot the coefficients obtained by integration and the coefficients obtained by the integration method. I plot them in Figure 7 for $exp(x)$ and Figure 8 for $cos(cos(x))$.
I show the integration coefficients in red dots and least squares coefficients in green dots for both the functions. I plot the $exp(x)$ and $cos(cos(x))$ in a semi-log plot.
Here is the code I used.

```
x = np.linspace(0, 2*np.pi, 401)
x = x[:-1]
A = np.zeros((400, 51))
b1 = f1(x)
b2 = f2(x)
A[:,0] = 1
# Generating the A matrix
for k in range(1, 26):
  A[:,2*k-1] = np.cos(k*x)
  A[:, 2*k] = np.sin(k*x)

ls_co1 = np.linalg.lstsq(A, b1, rcond=None)[0]
ls_co2 = np.linalg.lstsq(A, b2, rcond=None)[0]
```

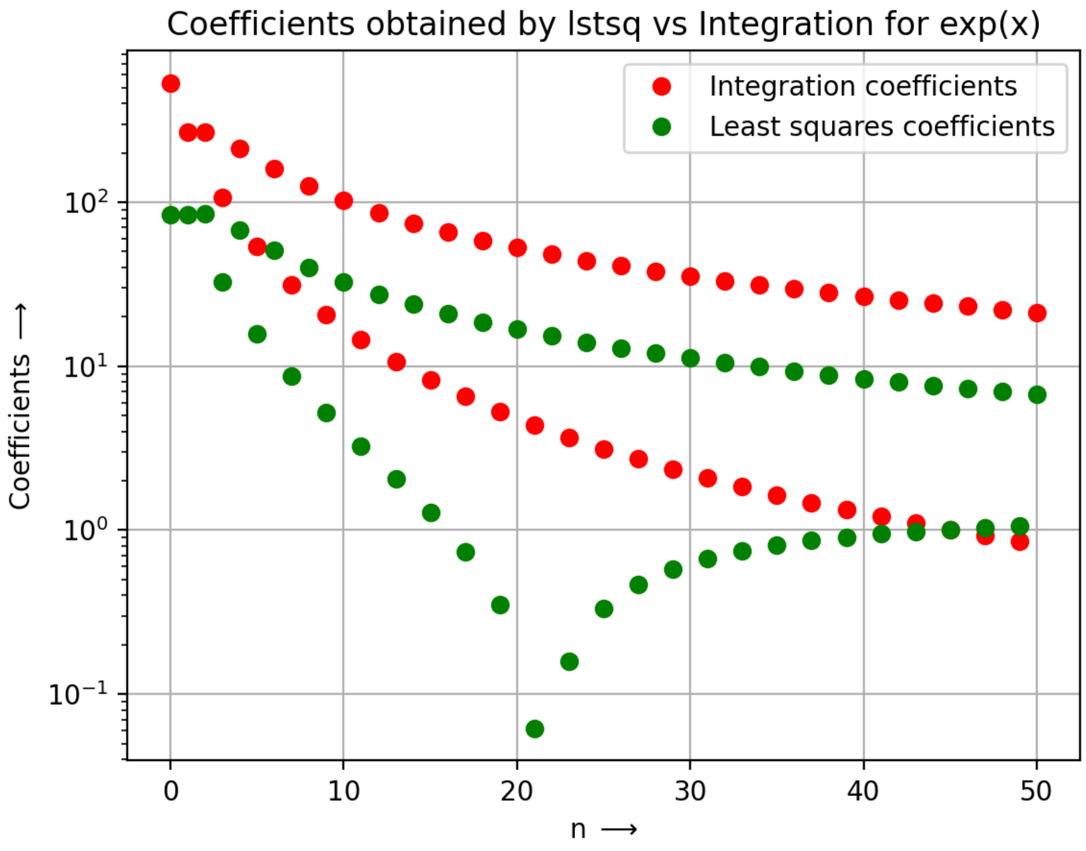

Figure 7: Plot of coefficients by least squares and integration

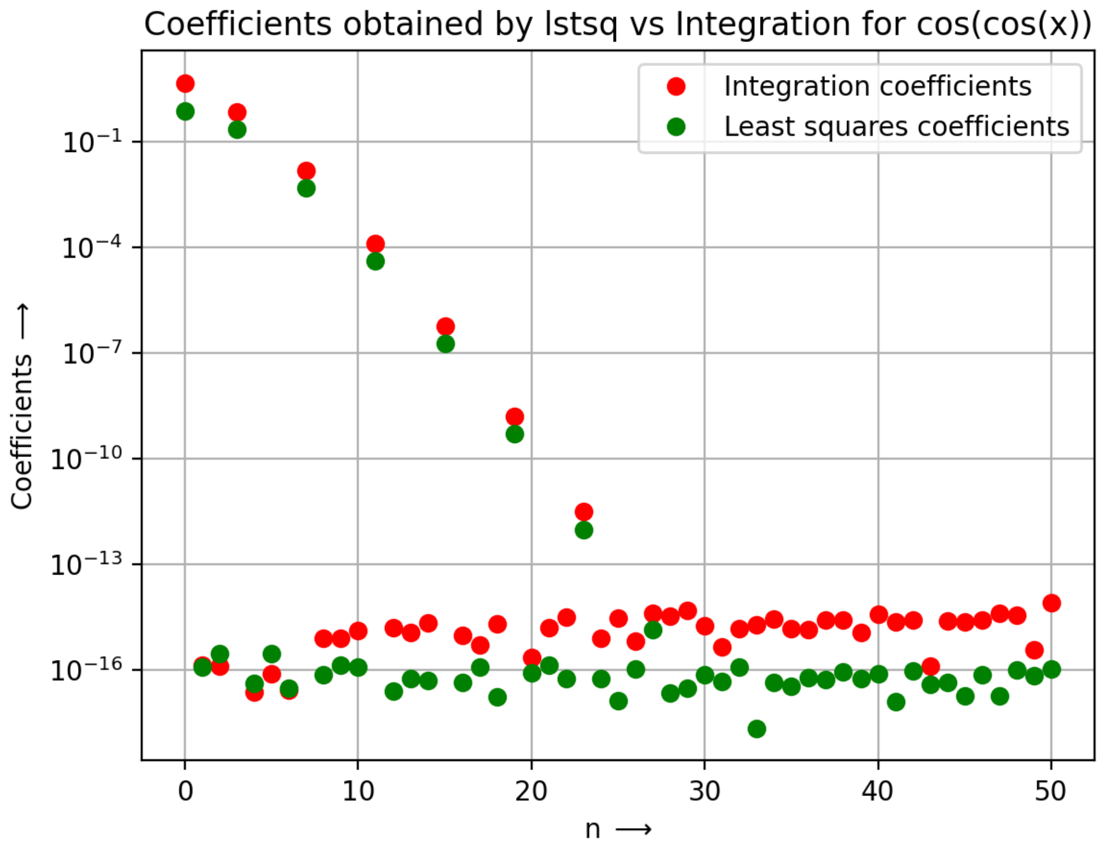

Figure 8: Plot of coefficients by least squares and integration

Question 6

In this question I find the errors between the least squares and integration approximations and find the largest deviation.
I print the output in the terminal which is as follows:
Largest deviation in coefficients of $exp(x)$: 450.091032
Largest deviation in coefficients of $cos(cos(x))$: 4.042681
This is the code I used:

```
err1 = np.abs(f1_co-ls_co1)
print('Largest deviation in coefficients of exp(x): %f' %(np.amax(
    err1)))
```

The coefficients obtained by least squares approximation and integration agree well in the case of $cos(cos(x))$, not so much in the case of $exp(x)$.
In the $exp(x)$ case, they do not agree very well since value of exponent varies rapidly for even a small change in x. Hence the least squares approximation won't match the values obtained by integration.
They needn't agree since the least squares method provides the overall rationale for the placement of the line of best fit among the data points being studied. The integration method is obviously more accurate that way.

Question 7

Since least squares is an approximation based on errors, it is not expected for them to agree well. $cos(cos(x))$ itself is composed of cosines. Hence we see a good agreement of the function with the output produced from the least squares method. Within the data points being studied, as periodicity is being observed, the least squares method is able to produce a closer approximation.
As 401 data points is a small number for a rapidly increasing function like $exp(x)$, we do not find much agreement.
In the plot, it is seen that the values diverge for smaller values of x which are small and then converge for larger values of x.

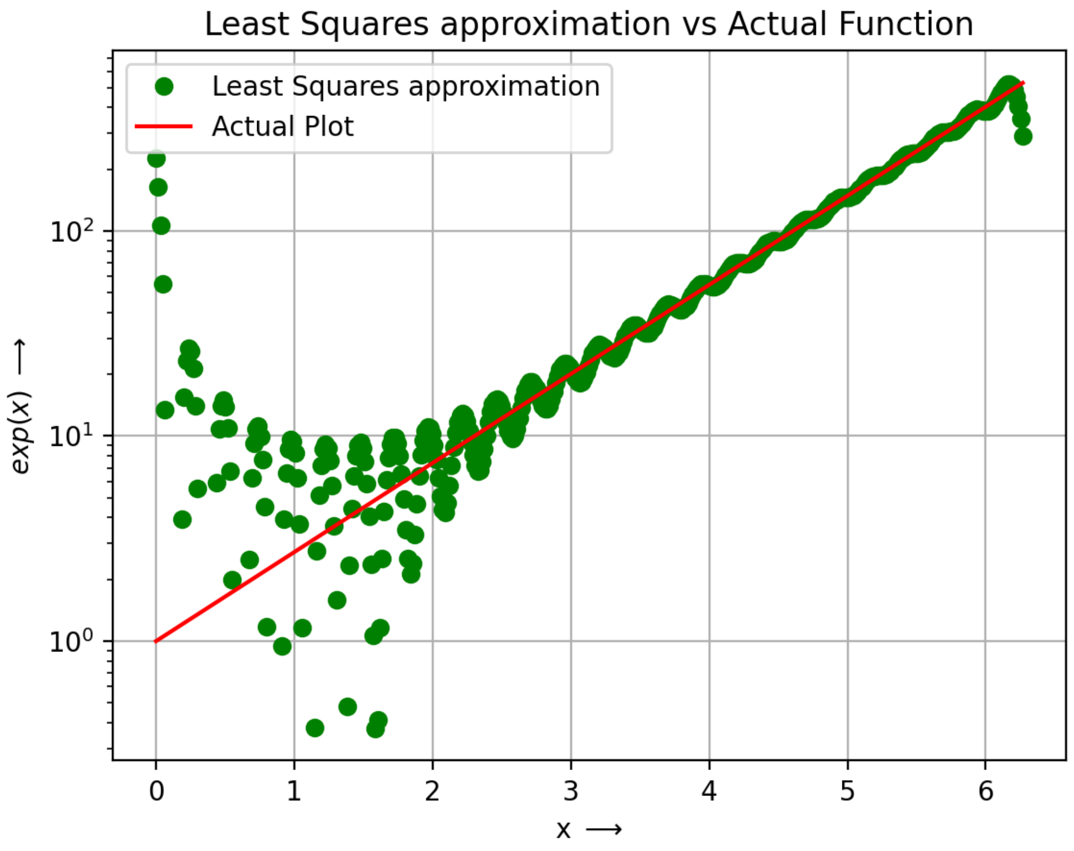

Figure 9: Plot of $exp(x)$ approximation by least squares and actual function plot

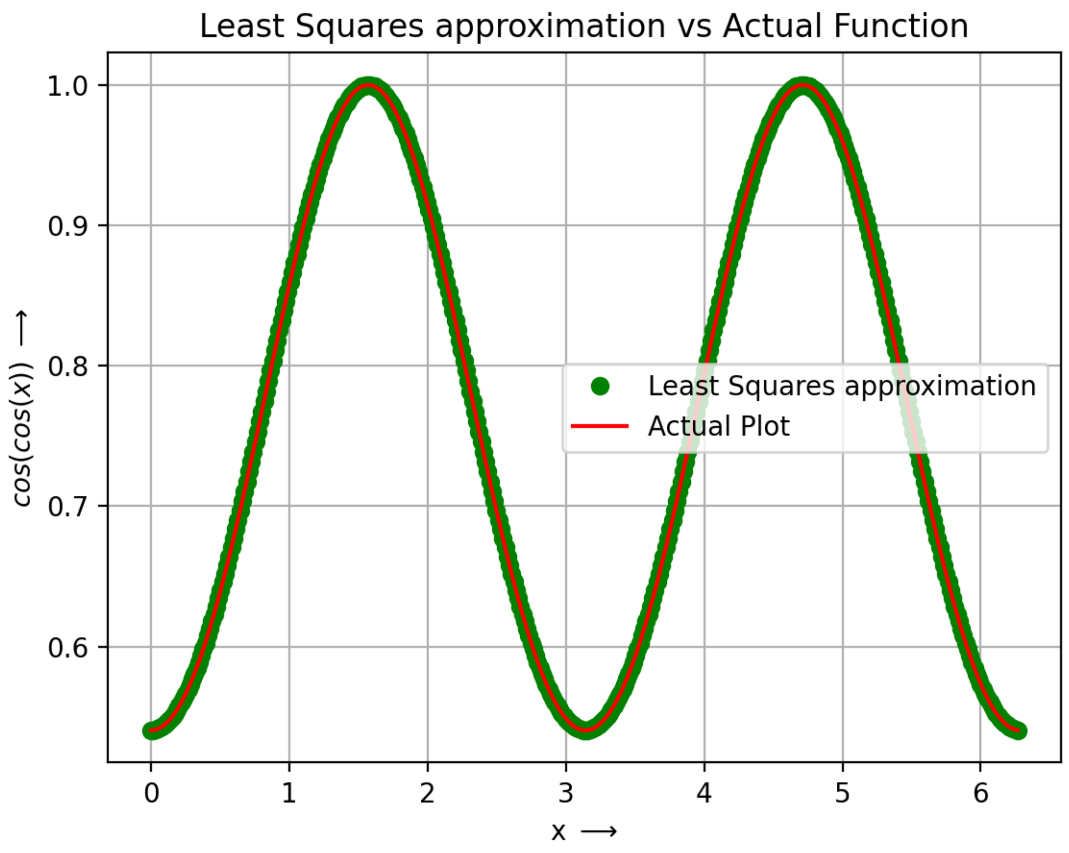

Figure 10: Plot of $cos(cos(x))$ by least squares and actual function plot