EE2703: Applied Programming Lab

Week 4: Fourier Approximations

Author: Surya Prasad S, EE19B121

March 10, 2021

1 Abstract:

In this week's assignment, we shall be fitting two functions, exp(x) and cos(cos(x)) over the interval [0,2π) using the Fourier Series. We compute the Fourier Series using scipy.integrate and "Least Squares approach". We expect the coefficients got through integration to be more accurate and we take them as the true coefficients. We shall also compare and contrast Least Squares approach's coefficients with the true coefficients and plot various graphs for each of the two functions.

2 Theory:

The Fourier Series is given by

$$a_0 + \sum_{n=1}^{+\infty} \{a_n cos(nx) + b_n sin(nx)\} \tag{1}$$

The coefficients a_n and b_n are computed by

$$a_0 = \frac{1}{2\pi} \int_0^{2\pi} f(x)dx$$

$$a_n = \frac{1}{\pi} \int_0^{2\pi} f(x)cos(nx)dx$$

$$b_n = \frac{1}{\pi} \int_0^{2\pi} f(x)sin(nx)dx$$

We use scipy.integrate to compute the integrals and then we store it in a vector of the form

$$\begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ \dots \\ a_{25} \\ b_{25} \end{pmatrix}$$

Previously, the coefficients for the Fourier Series were calculated using scipy's integrate which is based on Reimann Integration and it involves a lot of computations. A more efficient approach would be to attempt to get reasonable set of coefficients using the "Least Squares approach". Define a vector x going from 0 to 2π in 400 steps. Evaluate the function f(x) at those values and call it b. Now this function is to be approximated by Eq. (1). So for each x_i we want

$$a_0 + \sum_{n=1}^{25} a_n cosnx_i + \sum_{n=1}^{25} b_n sinnx_i \approx f(x_i) \tag{2}$$

Now, turn this into a matrix problem:

$$\begin{pmatrix} 1 & cosx_1 & sinx_1 & \dots & cos25x_1 & sin25x_1 \\ 1 & cosx_2 & sinx_2 & \dots & cos25x_2 & sin25x_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & cosx_{400} & sinx_{400} & \dots & cos25x_{400} & sin25x_{400} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ \dots \\ a_{25} \\ b_{25} \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \dots \\ f(x_{400}) \end{pmatrix}$$

Let's call the matrix on the left side as A. We want to solve

$$Ac = b$$

where c is the vector of Fourier coefficients and b is the vector of the actual functional values.

3 Plotting and Code Analysis:

3.1 Libraries imported:

The following libraries are imported:

```
import numpy as np
import scipy
import scipy.integrate as integrate
import matplotlib.pyplot as plt
import math
```

3.2 Constants used in the code:

The following are the constants used in the code:

```
PI = np.pi                          # Pi value
N = 100000                          # Number of elements in a vector to be plotted
THRESHOLD = float('1e-12')          # Used to control the error due to integration
```

3.3 Creating the function:

We define functions which compute exp(x) and cos(cos(x)) and also plot them. Code snippet for the function definition is:

```
def e(x):
        return np.exp(x)

def cos_cos(x):
    return np.cos(np.cos(x))
```

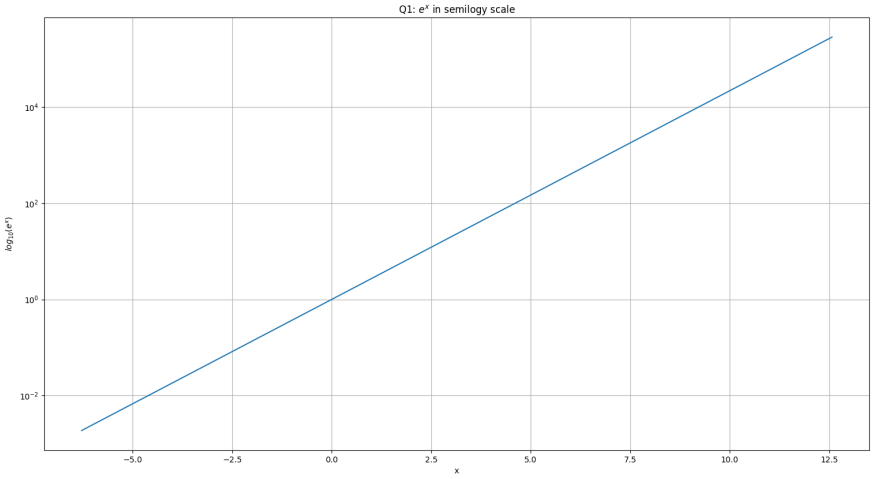

Figure 1: Semilogy plot of exp(x)

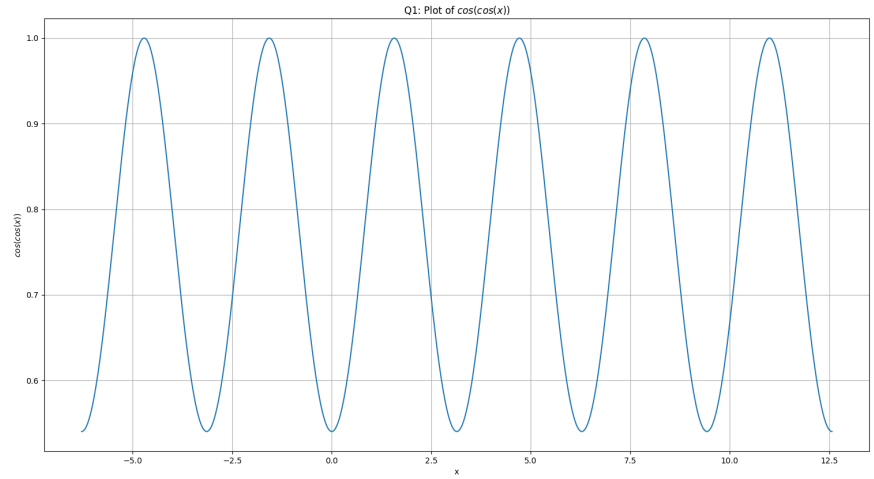

Figure 2: Plot of cos(cos(x))

Since exp(x) grows rapidly, we plot it in semilogy scale. As we can see in Figure 1, exp(x) is not a periodic function whereas cos(cos(x)) is a periodic function.

We shall define another function to convert a given function into a piece-wise periodic function with the default principle interval [0,2π]. Here we use the built-in function numpy.remainder to convert all the numbers to the corresponding value in the principle interval. Code snippet for the function to compute for periodic form of a function:

```
def periodic(func, x, lower_limit = 0, upper_limit = 2 * PI):
    return func(np.remainder(x - lower_limit, upper_limit - lower_limit))
```

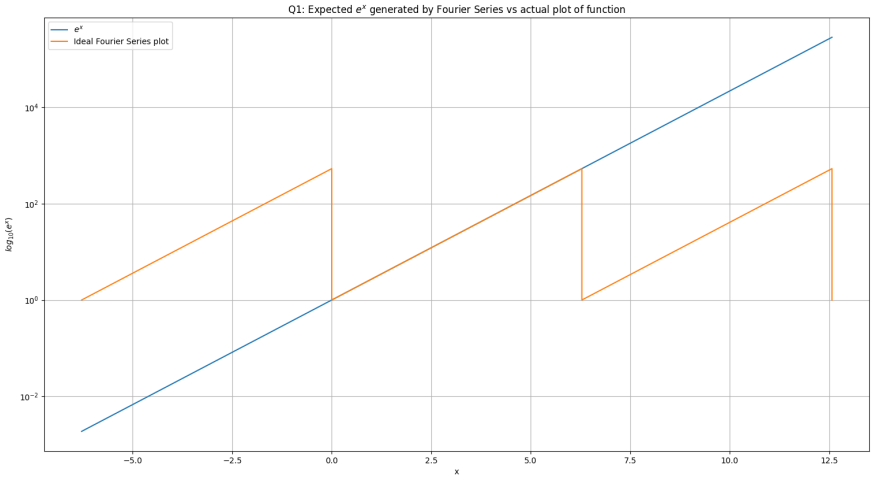

Figure 3: Semilogy plot of periodic(exp(x)) and exp(x):

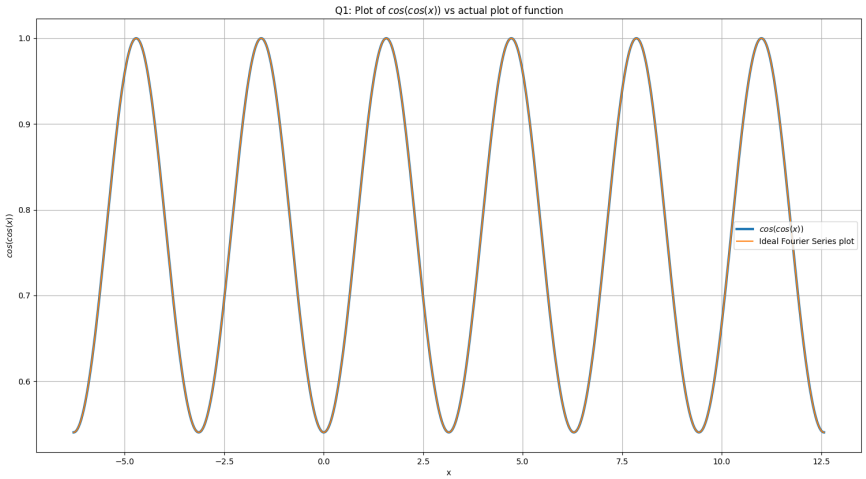

Figure 4: Plot of periodic(cos(cos(x))) and cos(cos(x))

These are the expected plots of the Fourier Series. As we can see, there is no difference in the plot of periodic(cos(cos(x)) and cos(cos(x)). This is because the period of cos(cos(x)) is π periodic.

3.4 Obtaining the Fourier Series coefficients:

First, we define a function to compute Fourier Series coefficient. Code snippet for the function definition to compute Fourier Series Coefficients:

```
def Fourier_Series(n, func, lower_limit = 0, upper_limit = 2 * PI):
        a = np.zeros(n)

        def fcos(x, n, f):
                return f(x) * np.cos(n*x) * (1/PI)
        def fsin(x, n, f):
                return f(x) * np.sin(n*x) * (1/PI)

        a[0] = integrate.quad(func, lower_limit, upper_limit, epsabs =
           THRESHOLD)[0] * (1/(2 * PI))

        for i in range(1, n):
        if i%2 == 1:
            a[i] = integrate.quad(fcos, lower_limit, upper_limit, args = ((
                i//2 + 1), func), epsabs = THRESHOLD)[0]
        else:
            a[i] = integrate.quad(fsin, lower_limit, upper_limit, args = (i
                //2, func), epsabs = THRESHOLD)[0]
    return a
```

We use scipy's integrate function to compute the coefficient of the Fourier Series. It does integration by using Reimann's sum. The integral function has the following parameters:

1. function - The function to be integrated
2. lower limit - Lower limit of integration
3. upper limit - Upper limit of integration
4. args - If the passed function has other parameters then they can be given here
5. epsabs - Magnitude of error in computing the integral can be controlled with this. By default the value is 1.49e-8.

The values are stored in a vector of the form

$$\begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ \dots \\ a_{25} \\ b_{25} \end{pmatrix}$$

We shall plot these values in semilogy and loglog scale for both the functions.

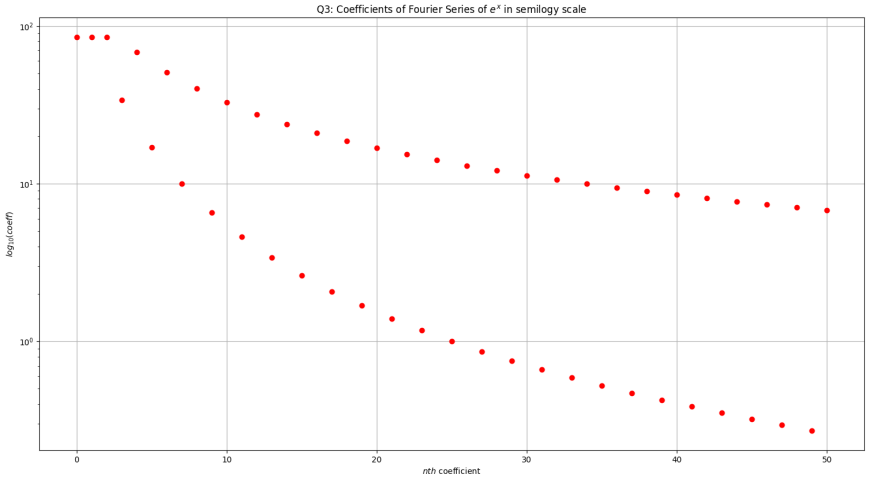

Figure 5: Coefficients of Fourier Series of exp(x) in semilogy scale

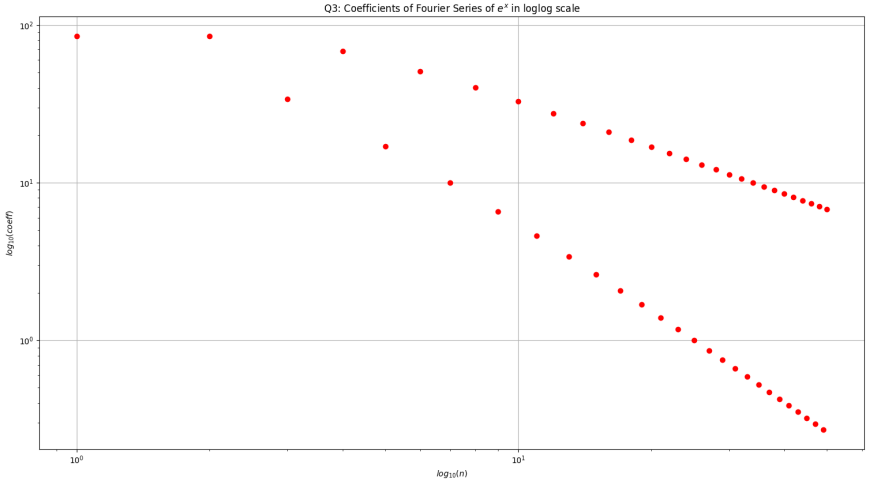

Figure 6: Coefficients of Fourier Series of exp(x) in loglog scale

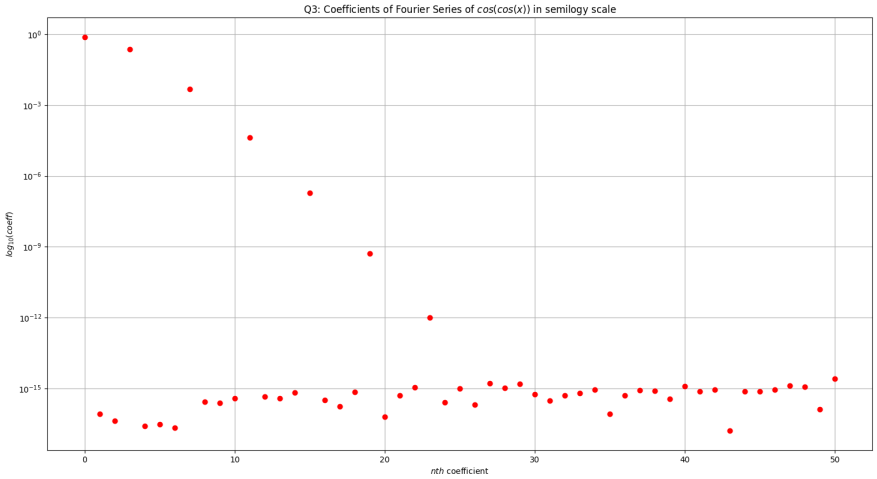

Figure 7: Coefficients of Fourier Series of cos(cos(x)) in semilogy scale

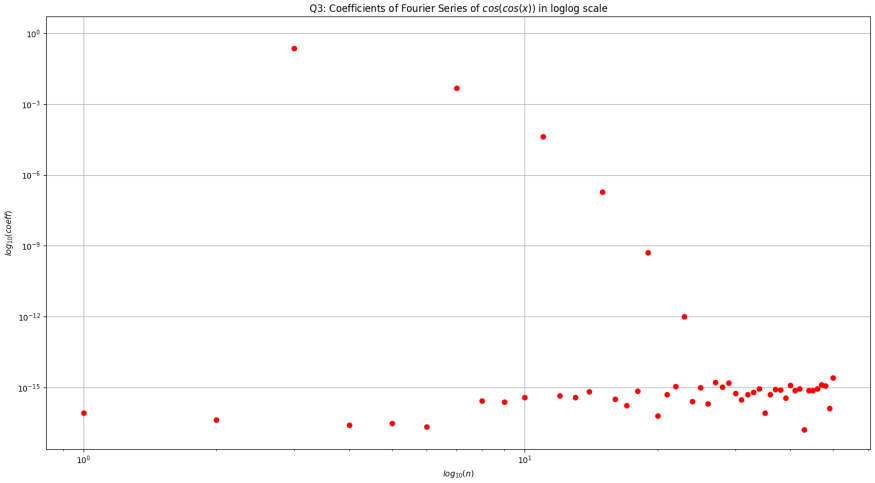

Figure 8: Coefficients of Fourier Series of cos(cos(x)) in loglog scale

The semilogy plot of cos(cos(x))'s coefficients are approximately linear. b_n coefficients of cos(cos(x)) are nearly zero as cos(cos(x)) is an even function. Here they have very small values because of computational errors and can be seen on an almost horizontal line.
From the plot of cos(cos(x)), we can see that it is approximately sinusoidal with period of 2π. Hence the value converges more quickly as compared to e^x.
The Fourier coefficients of e^x are given by:

$$a_n = \int_0^{2\pi} e^x cos(kx)dx = \frac{(e^{2\pi} - 1)}{(k^2 + 1)}$$

and

$$b_n = \int_0^{2\pi} e^x sin(kx) dx = \frac{(-ke^{2\pi} + k)}{(k^2 + 1)}$$

So the *log* of the coefficients is approximately proportional to $-logk$ and $-2logk$. Hence there are two straight lines in the log-log plot. The semi-logy plot for Fourier Coefficients of cos(cos(x)) is linear as the integral converges to a Linear Combination of Bessel functions which are proportional to e^x .
Throughout this code, we shall treat these values as the true Fourier Series coefficients.

3.5 Obtaining Fourier Series Coefficients using Least Squares Approach:

We shall first define a vector x going from 0 to 2π in 400 steps. For this we shall use numpy.linspace. In the previous week's assignment, we did curve fitting using scipy's lstsq function and on similar grounds we try to estimate the Fourier Series' coefficients. Code snippet for function to compute Fourier coefficients using Least Square approach:

```
def Least_Square_Fitting(func):
    x = np.linspace(0, 2 * PI, 401)
    x = x[:-1]

    A = np.zeros((x.shape[0], 51))
    A[:, 0] = 1

    for i in range(1, 26):
        A[:, 2 * i - 1] = np.cos(i * x)
        A[:, 2 * i] = np.sin(i * x)

    B = func(x)

    best_fit = scipy.linalg.lstsq(A, B)[0]

    return best_fit, A

e_LS_coeff, e_LS_A = Least_Square_Fitting(e)
cos_cos_LS_coeff, cos_cos_LS_A = Least_Square_Fitting(cos_cos)
```

We shall plot these values in semilogy and loglog scale for both the functions.

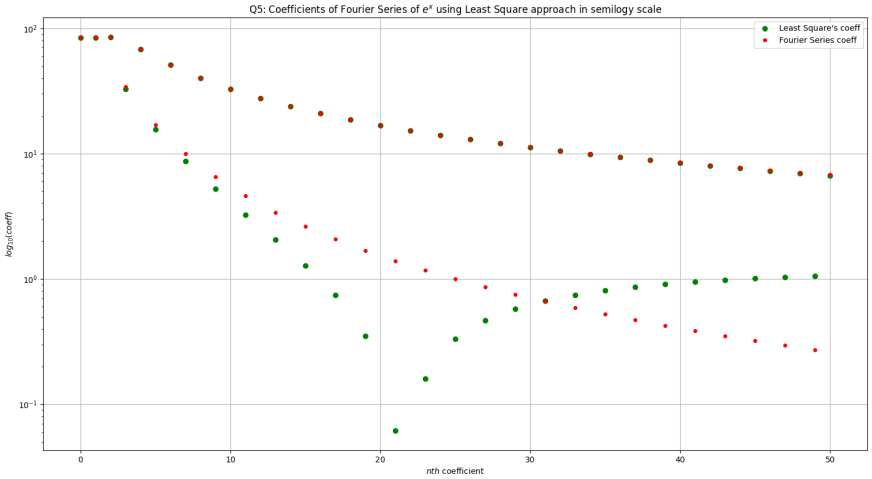

Figure 9: Coefficients of Fourier Series of exp(x) in semilogy scale

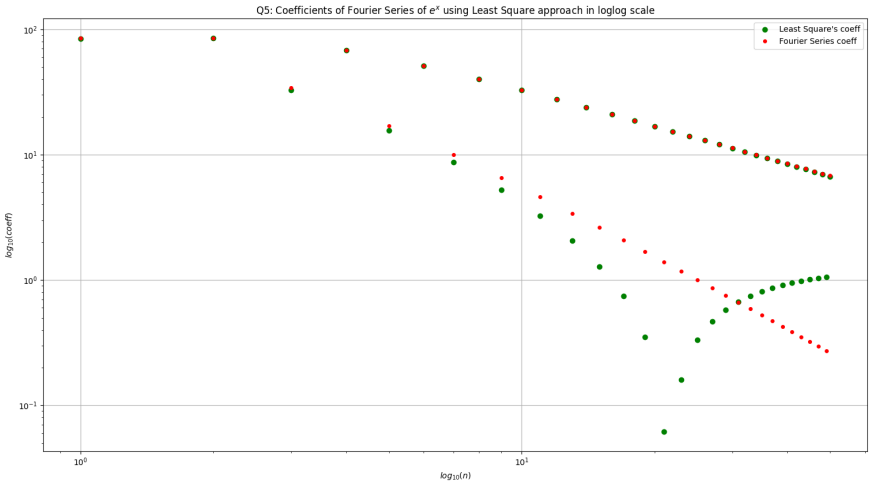

Figure 10: Coefficients of Fourier Series of exp(x) in loglog scale

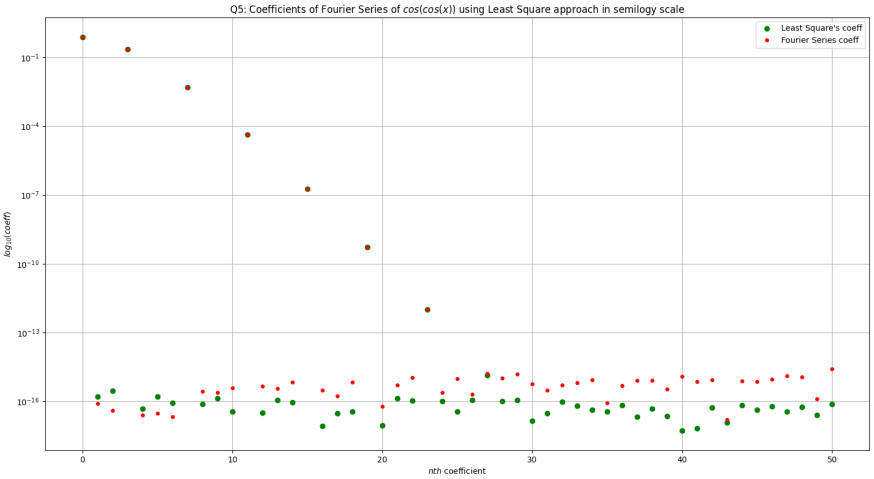

Figure 11: Coefficients of Fourier Series of coscos(x) in semilogy scale

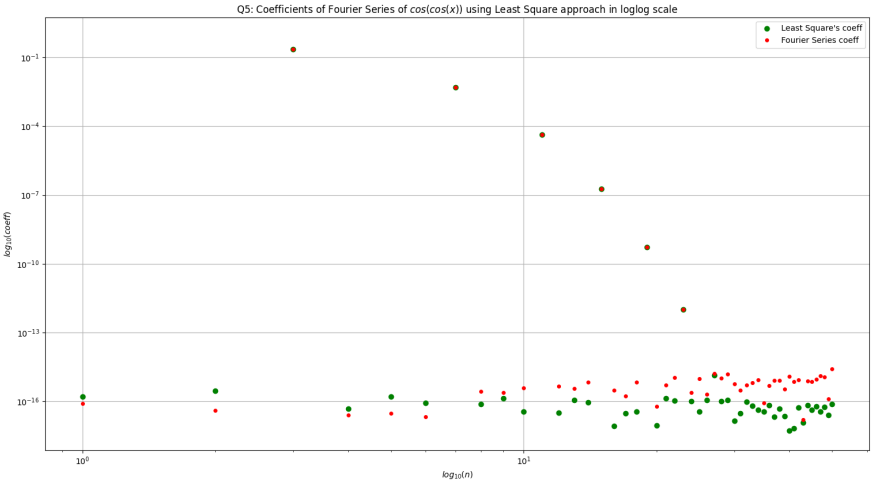

Figure 12: Coefficients of Fourier Series of coscos(x) in loglog scale

3.6 Comparison with true Fourier Series coefficients:

Previously we had computed Fourier Series coefficients using scipy's integrate function. Now, we shall compare these values against the values got through Least Squares approach. As we can see from the plots, the coefficients did not match exactly in the case of e^x but was very accurate for cos(cos(x)). Code snippet to find largest deviation:

```
Error_e = np.amax(np.abs(e_FS - e_LS_coeff))
Error_cos_cos = np.amax(np.abs(cos_cos_FS - cos_cos_LS_coeff))

print("The maximum deviation in Least Square coefficients of exp(x)
    compared to True values", Error_e)
print("The maximum deviation in Least Square coefficients of cos(cos())
    compared to True values", Error_cos_cos)
```

We use numpy.amax which returns the maximum value in an array along an axis. This is the output we get:

```
The maximum deviation in Least Square coefficients of exp(x) compared to
    True values is 1.3327308703353538
The maximum deviation in Least Square coefficients of cos(cos(x)) compared
    to True values is 2.683856629375929e-15
```

The maximum deviation is very significant for the exponential function as compared to the neglible deviation for cos(cos(x)). For plotting the generated functions using the coefficients from Least Squares approach we shall define the following function:

```
def Fourier_Series_Values(x, FS_coeff):
    n = np.size(FS_coeff,0)
    result = np.zeros((np.size(x, 0), 1))
    result += FS_coeff[0]

        for i in range(n):
        if i%2 == 0:
            result += FS_coeff[i] * np.sin(i//2 * x);
        else:
            result += FS_coeff[i] * np.cos((i//2 + 1) * x);
    return result
```

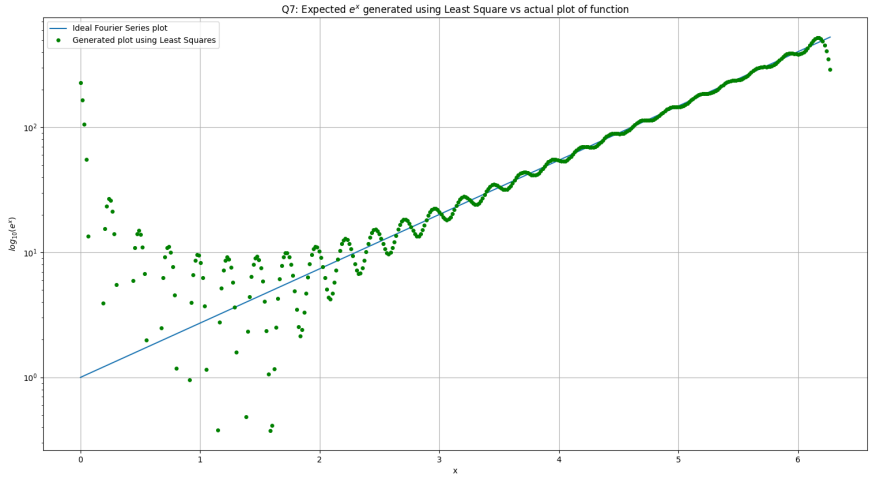

Figure 13: Plot of exp(x) generated using Least Squares approach's coefficients as compared to ideal Fourier Series plot

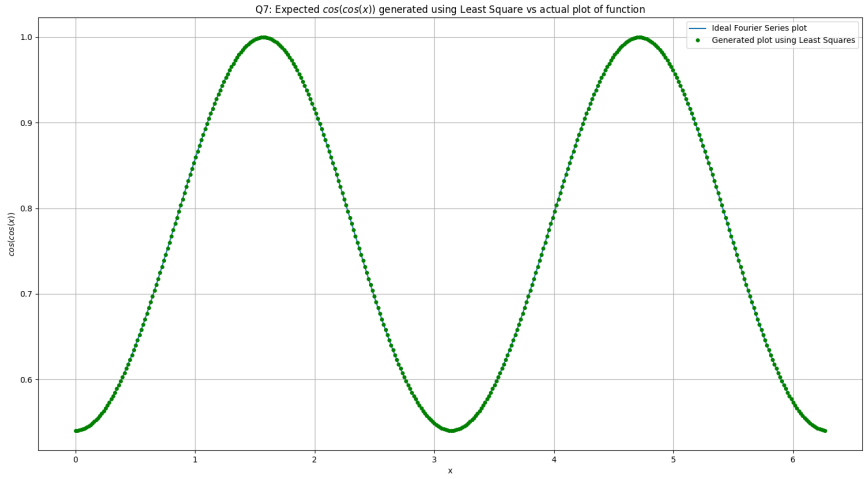

Figure 14: Plot of cos(cos(x)) generated using Least Squares approach's coefficients

As expected the deviation is significant in case of exp(x) as there are discontinous points present in the ideal plot generated by Fourier Series. When there are discontinous points, the Fourier Series does not converge to the required value and also has a lot of ripples. This is explained using Gibb's phenomena. In the case of cos(cos(x)), there are no discontinuos points and from the plots we can see that it is almost sinusoidal.

4 Conclusion:

In this assignment, we have plotted and examined the two ways of generating Fourier coefficients for the given two functions e^x and cos(cos(x)). The methods adopted in finding the respective Fourier coefficients have been the direct evaluation of the Fourier Series formula, as well as the Least Square approach's best fit. The Least Square approach's values fit well for cos(cos(x)) but diverge for e^x.