EE2703:
APPLIED PROGRAMMING LAB

Assignment 4

Fourier Approximations

Arjun R
EE20B016

March 10, 2022

Contents

1 Aim **1**

2 Introduction **1**

3 Finding Fourier Coefficients **1**
3.1 *Plotting the Functions* 1
3.2 *Finding the Coefficients through Direct Integration* 3
3.3 *Finding the Coefficients through* ***Least Squares*** *Method* 6
3.4 Calculating Deviation 6
3.5 Plotting the Approximate Functions 9

4 Conclusion **10**

1 Aim

This assignment mainly focuses on

- Fitting the functions e^x and $cos(cos(x))$ over the interval $[0, 2\pi)$ using the Fourier series expansion
- Finding the Fourier coefficients using least-squares method and analyzing their deviation from actual coefficients
- Drawing important inferences through a variety of plots

2 Introduction

Fourier series is a way of representing any periodic function as a sum of sinusoids:

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

where the coefficients are given by:

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

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

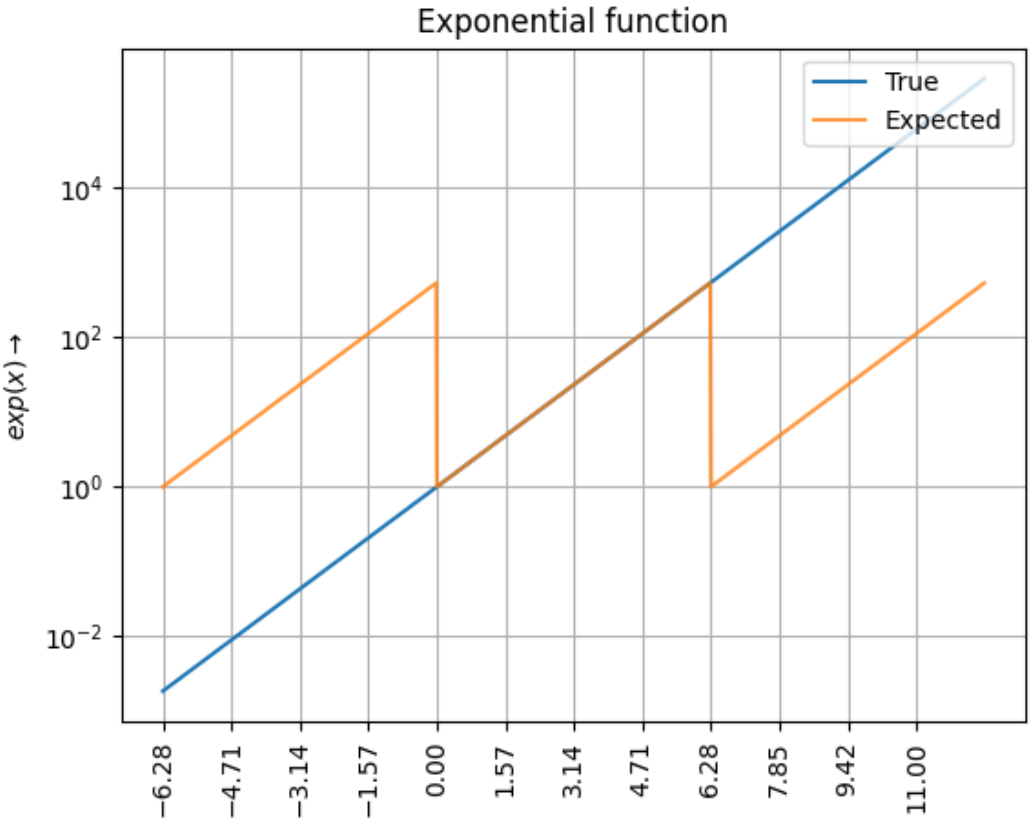
$$b_n = \frac{1}{\pi}\int_0^{2\pi} f(x)sin(nx)dx$$

In this assignment, we will dealing with the Fourier approximations of the two functions e^x and $cos(cos(x))$.

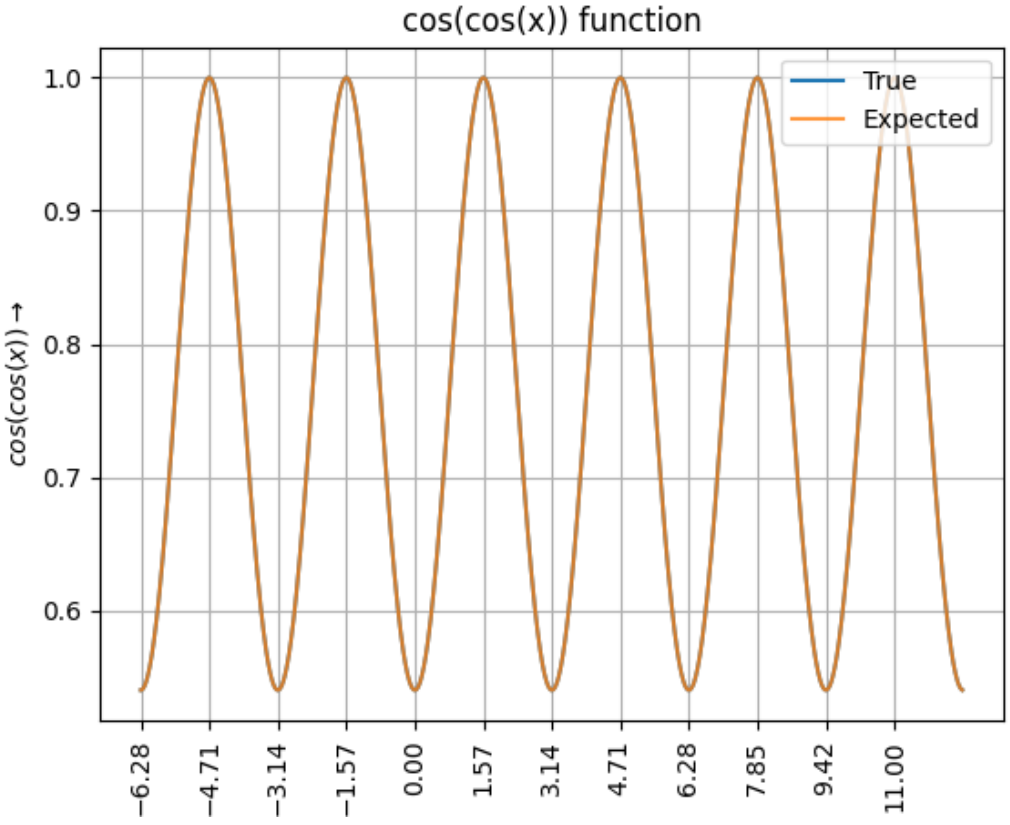
3 Finding Fourier Coefficients

3.1 *Plotting the Functions*

Clearly, $cos(cos(x)$ is a periodic function but e^x is not. So we have to convert e^x to a periodic function by repeating its $[0, 2\pi)$ nature over the entire domain.



(a) e^x(semi-logy)



(b) $cos(cos(x))$(log-log)

Figure 1: True plot and plot expected to be generated by Fourier series

3.2 *Finding the Coefficients through Direct Integration*

The first 51 coefficients can be obtained through the coefficient formulae mentioned before. Integration is performed by the function `quad()` under module `scipy.integrate`. It requires the creation of two additional functions to be integrated. This is done as follows:

```
def u(x, f, k):
    return f(x) * cos(k*x)

def v(x, f, k):
    return f(x) * sin(k*x)
```

These functions are passed to the `quad()` function and the required Fourier coefficients are generated as follows:

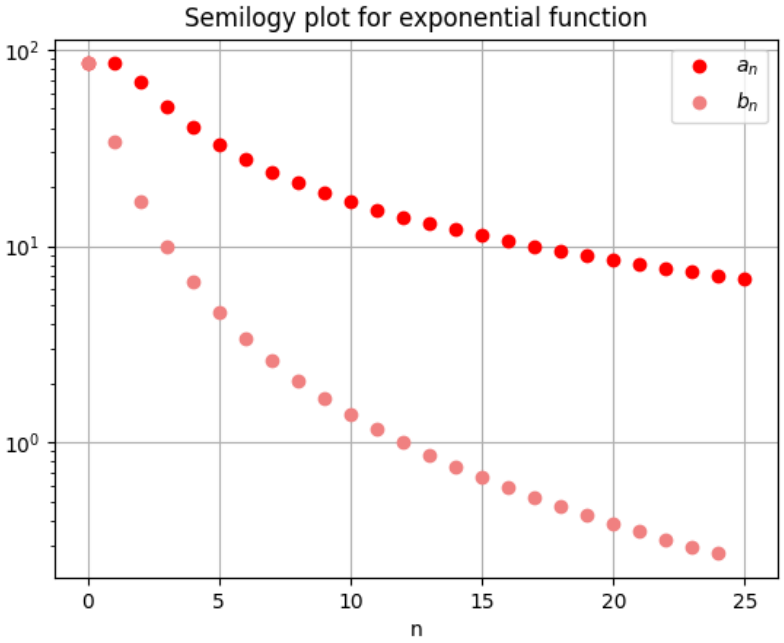
```
c_exp = np.zeros(51)
c_exp[0] = quad(u, 0, 2*pi, args=(exp, 0))[0]/(2*pi)
for k in range(1, 26):
    c_exp[2*k-1] = quad(u, 0, 2*pi, args=(exp, k))[0]/pi
    c_exp[2*k] = quad(v, 0, 2*pi, args=(exp, k))[0]/pi

c_coscos = np.zeros(51)
c_coscos[0] = quad(u, 0, 2*pi, args=(coscos, 0))[0]/(2*pi)
for k in range(1, 26):
    c_coscos[2*k-1] = quad(u, 0, 2*pi, args=(coscos, k))[0]/pi
    c_coscos[2*k] = quad(v, 0, 2*pi, args=(coscos, k))[0]/pi
```

Extra arguments can be passed to the function being integrated by `args` parameter inside `quad` function.

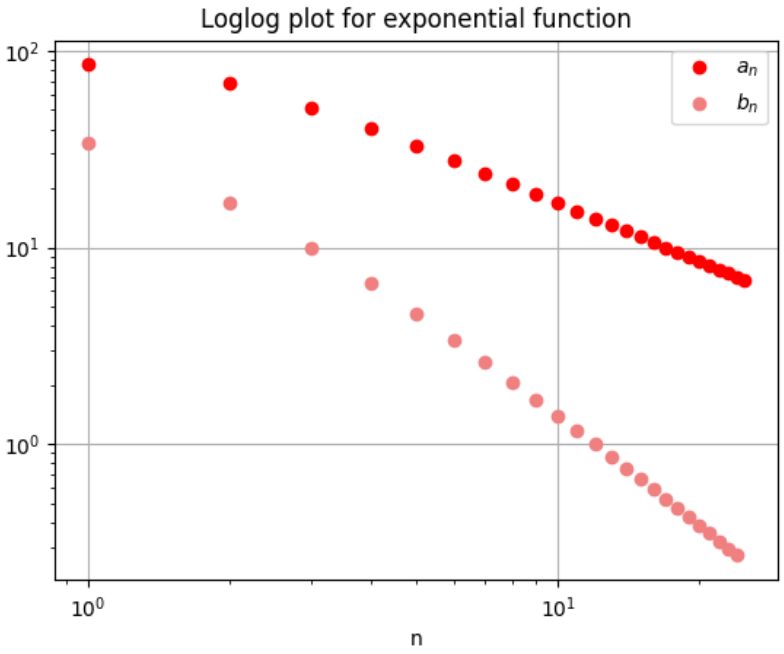
The generated coefficients are plotted in semi-logy and log-log plots for both e^x and $cos(cos(x))$.

From figures 2 and 3, we can draw some interesting observations:

- In the second case, b_n coefficients are nearly zero because the function $cos(cos(x))$ is primarily ***cosine*** in nature and hence the ***sine*** contribution in its expansion is close to zero.
- In the first case, e^x function isn't periodic but increases monotonically. Hence, its coefficients are spread out over a wide range of frequencies. But the function $cos(cos(x))$ is periodic and hence the coefficients are concentrated over the fundamental frequency of the function and decays after that.
- The log-log plot of coefficients of exponential function (figure 2b) is linear because the coefficient expressions are rational functions (functions of the form $\frac{p}{q}$ where both p and q are polynomials). The semi-log plot of $cos(cos(x))$ coefficients (figure 3a) is roughly linear because the coefficient expressions turn out to be some combinations of exponentials of x.

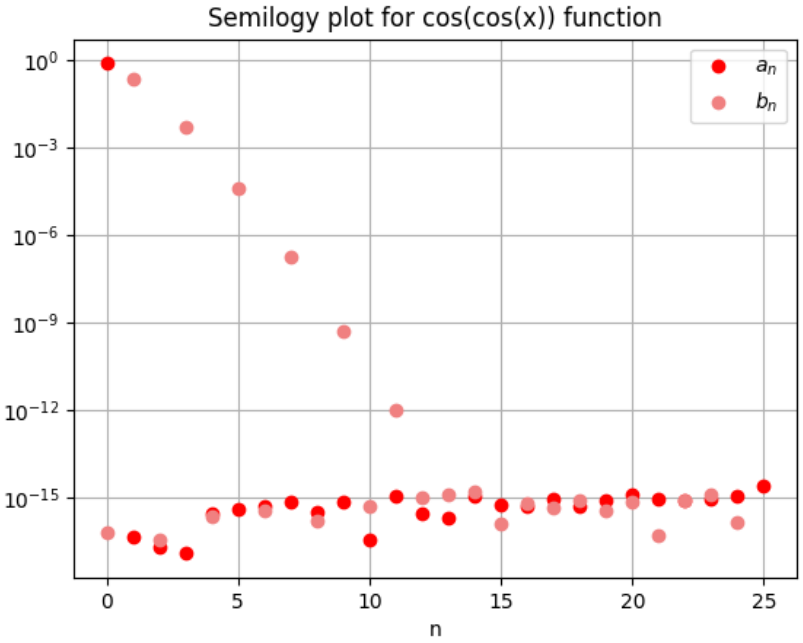


(a) Semi-logy

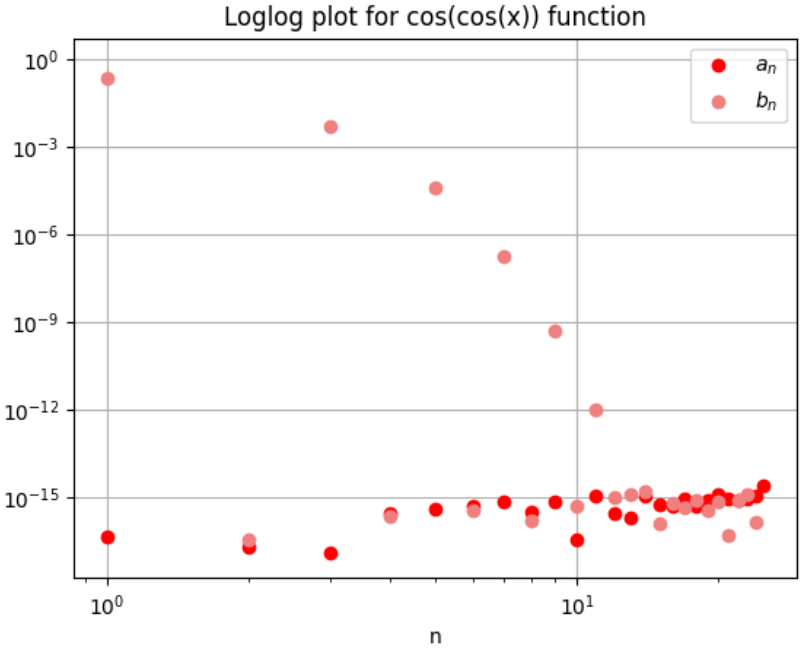


(b) Log-log

Figure 2: Coefficients of Fourier series of e^x



(a) Semi-logy



(b) Log-log

Figure 3: Coefficients of Fourier series of $cos(cos(x))$

3.3 *Finding the Coefficients through Least Squares Method*

In this method, we will define a vector x going from 0 to 2π in say, 400 steps, and calculate $f(x)$ for each x_i and call it b. This can be turned into a matrix problem as:

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

Let the matrix on the left side be A and let the coefficient matrix be c. Then we want to solve for c in

$$Ac = b$$

We can use the function `lstsq` from module `scipy.linalg` to do this. The implementation in code is given below:

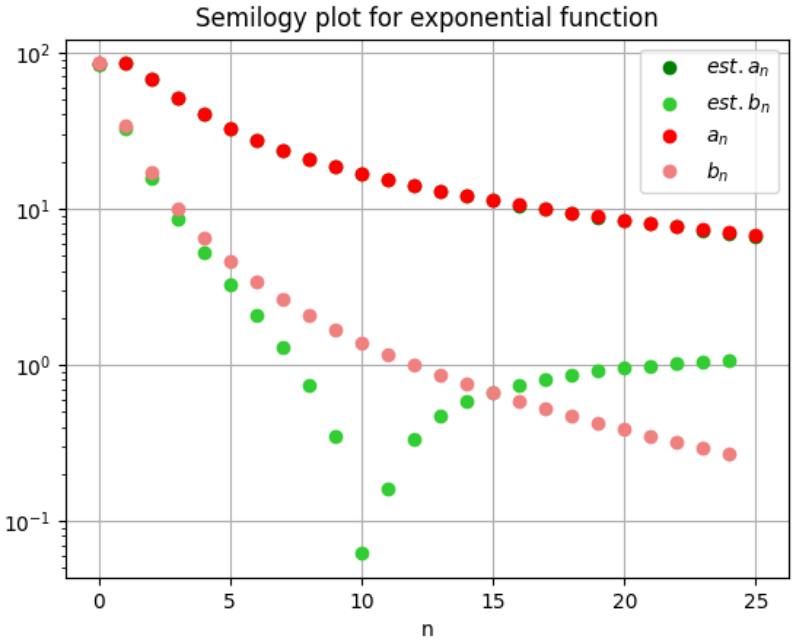
```
x = linspace(0, 2*pi, 401)
x = x[:-1]
b1 = exp(x)
A = zeros((400, 51)) # allocate space for A
A[:, 0] = 1 # col 1 is all ones
for k in range(1, 26):
    A[:, 2*k-1] = cos(k*x) # cos(kx) column
    A[:, 2*k] = sin(k*x) # sin(kx) column
    #endfor
c1 = lstsq(A, b1, rcond = None)[0] # the '[0]' is to pull out the
# best fit vector. lstsq returns a list.

# Similarly applying least-squares method for cos(cos(x))
b2 = coscos(x)
c2 = lstsq(A, b2, rcond = None)[0]
```

The coefficients estimated through ***least squares*** method are plotted in *green* in figures 4 and 5.

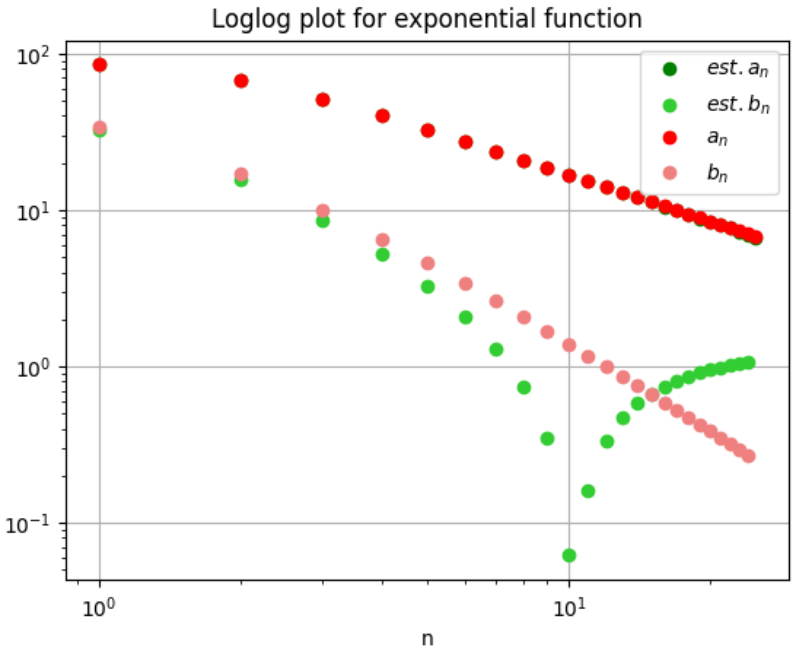
3.4 Calculating Deviation

The deviation between the coefficients obtained through both the methods can be calculated by directly subtracting the coefficient vectors in *Python*:

```
c_exp_dev = np.abs(c_exp-c1)
c_coscos_dev = np.abs(c_coscos-c2)
print("Max. absolute deviation between coefficients of exp(x)
generated through both methods: ", np.max(c_exp_dev))
print("Max. absolute deviation between coefficients of cos(cos(x))
generated through both methods: ", np.max(c_coscos_dev))
```

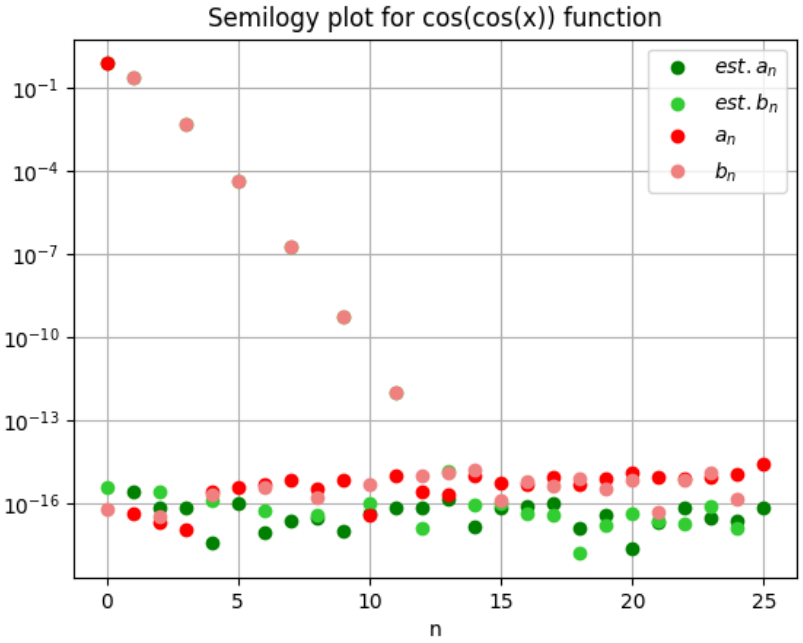


(a) Semi-logy

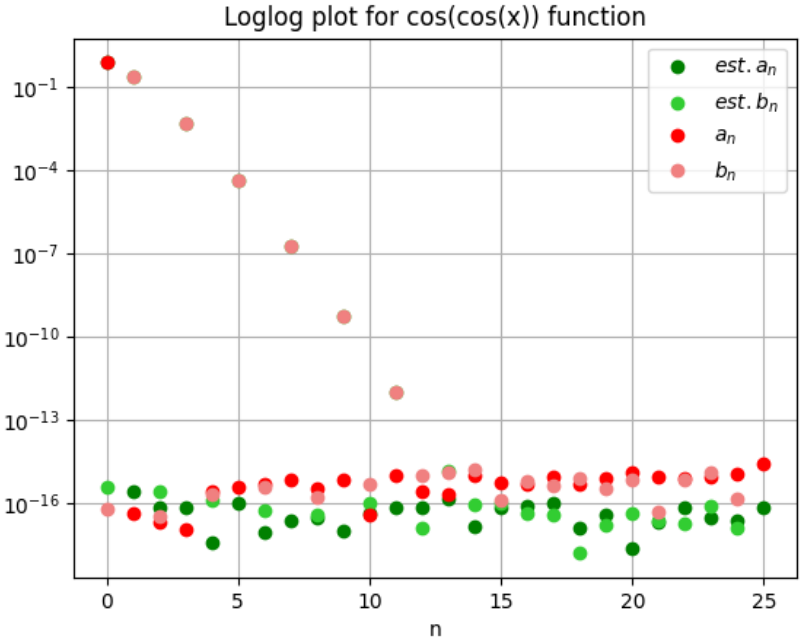


(b) Log-log

Figure 4: Coefficients of Fourier series of e^x



(a) Semi-logy



(b) Log-log

Figure 5: Coefficients of Fourier series of $cos(cos(x))$

OUTPUT:

```
Max. absolute deviation between coefficients of exp(x)
generated through both methods:  1.332730870335439
Max. absolute deviation between coefficients of cos(cos(x))
generated through both methods:  2.6603535900433973e-15
```

The plots and the maximum absolute deviations between the coefficients gives the following conclusions:

- Since e^x is a not a periodic function, the proper Fourier series expansion of the function will require the whole spectrum of frequency components to define it completely, and cannot be accurately modeled by a paltry 51 terms. This is why the least squares method fails to give the best fit of coefficients .
- In contrast, $cos(cos(x))$ is a periodic function, and thus can be accurately modeled by a smaller number of frequency components closer to the fundamental frequency. Hence, the least squares method works perfectly here in giving the best fit.

3.5 Plotting the Approximate Functions

Approximate functions are calculated by computing Ac from the estimated values of c. These functions values are plotted in figures 6 and 7 along with the true function values.

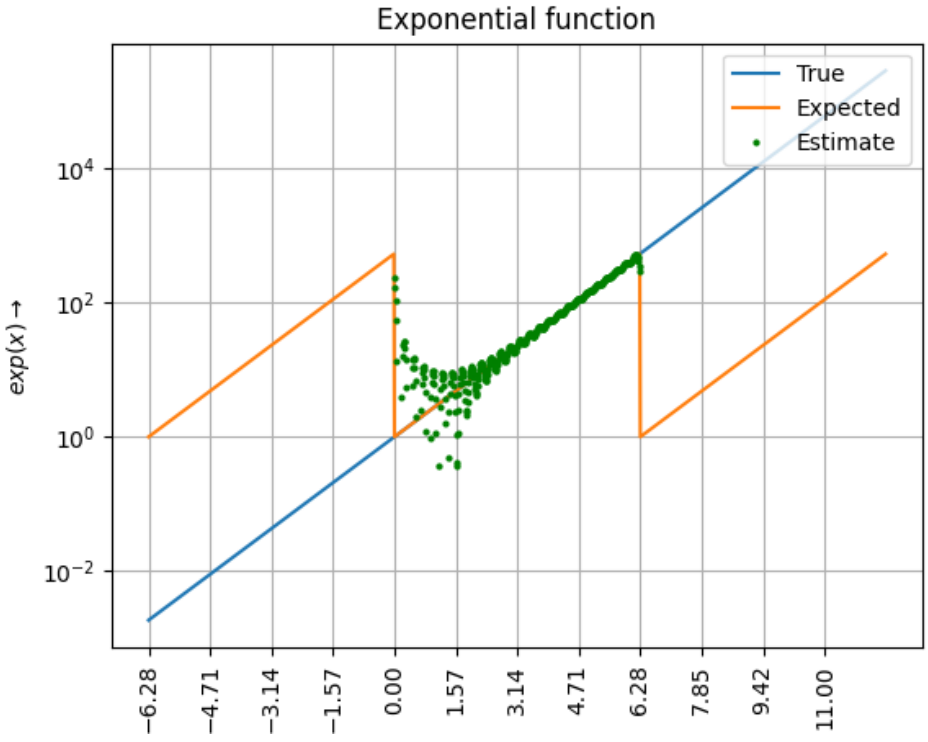


Figure 6: True and Approximate plots of e^x

The conclusions previously drawn from the maximum absolute deviations can also be clearly seen in these figures. It is intuitive yet exciting to see that the approximate function closely overlaps the true function in the case of $cos(cos(x))$.

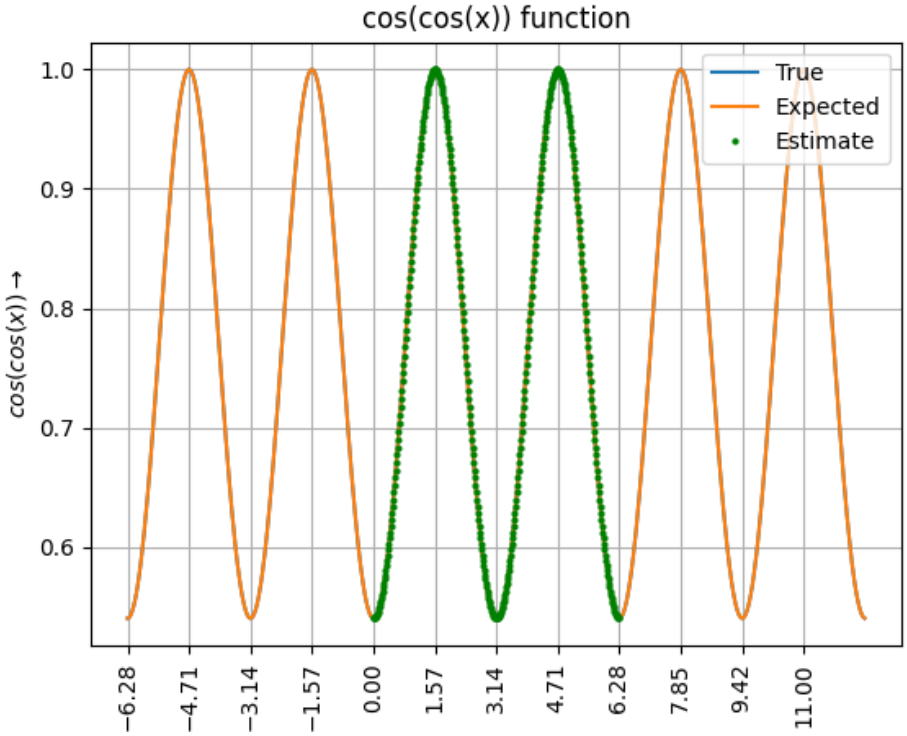


Figure 7: True and Approximate plots of $cos(cos(x))$

4 Conclusion

- We can see that the Fourier coefficients obtained using direct integration and the ones obtained using the **Least Squares** approach are different. This is because the **Least Squares** method finds the best fit for only a finite number of data samples.
- We can see considerable deviation between the true and approximate functions in the case of e^x which is not a periodic function in contrast to an almost perfect overlap in the case of $cos(cos(x))$ which is a periodic function. Thus finding the best fit coefficients using the **Least Squares** approach is not recommended for an aperiodic function.
- For proper periodic functions (not the ones which are made periodic), using the **Least Squares** approach is an excellent alternative which can give Fourier coefficients with a high accuracy.