EE2703: Assignment 4

KORRA SRIKANTH
EE19B033

April 17, 2021

Abstract

In this assignment, an analysis of fourier series approximations on two functions is done, namely e^x and $\cos(\cos(x))$.
The fourier coefficients for the above two functions are estimated by numerically evaluating the integrals.
Another approach for estimating the fourier coefficients is also done, namely least squares estimation.
The results of the two approaches are compared.
The differences in convergence between the two test functions are also noted.

Tasks:

The goal of this assignment is the following.

- To fit two functions e^x and $cos(cos(x))$ using the Fourier series.
- Obtain the first 51 coefficients i.e., a_0, a_1, b_1,... for e^x and $cos(cos(x))$ using `quad` function.
- Use a *Least Squares* approach to find the Fourier coefficients of the functions, using `scipy.linalg.lstsq`.
- Two different plots for each function using `semilogy` and `loglog` and plot the magnitude of the coefficients versus n.
- Find the maximum deviation between the coefficients obtained in the two methods.
- Computing $A \cdot c$ from the estimated values of c by *Least Squares Method* and plotting them.

Steps:

1 Question 1

- Define Python functions for the two functions e^x and $cos(cos(x))$ which return a vector (or scalar) value.
- Plot the functions over the interval $[-2\pi, 4\pi)$.
- Discuss periodicity of both functions.
- Plot the expected functions from Fourier series.

1.0.1 Code

```
def periodic(m, n):
    interval = n - m
    return lambda f: lambda x: f((x - m) % interval + m)
periodic_e =  lambda x : np.e**(np.remainder(x,2*pi))

e = lambda x : np.e**x
f = lambda x : np.cos(np.cos(x))

#Question: 1
#1200 points in interval of -2pi to 4pi
x0 = linspace(-2*pi,4*pi,1200)
y0 = list(map(f, x0))
y1 = list(map(e, x0))

plt.figure(1)
plt.grid(True)
plt.semilogy(x0,y1)
plt.semilogy(x0,periodic_e(x0),color='orange')
plt.title("$\cos(\cos(x))$ and its periodically extended version")
plt.ylabel(r"$y$ ")
plt.xlabel(r"$x$ (linear)")
plt.legend([r"$\cos(\cos(x))$",r"periodic version of $\cos(\cos(x))$"], loc=1)
plt.show()

plt.figure(2)
plt.grid(True)
plt.semilogy(x0,y0,color = 'green')
plt.semilogy(x0,cos(cos(x0)),color='orange')
plt.title("$e^x$ and its periodically extended version")
plt.ylabel(r"$y$ (log)")
plt.xlabel(r"$x$ (linear)")
plt.legend([r"$e^x$",r"periodic version of $e^x$"], loc='upper right')
plt.show()
```

The plots of e^x and $cos(cos(x))$ are as shown below:

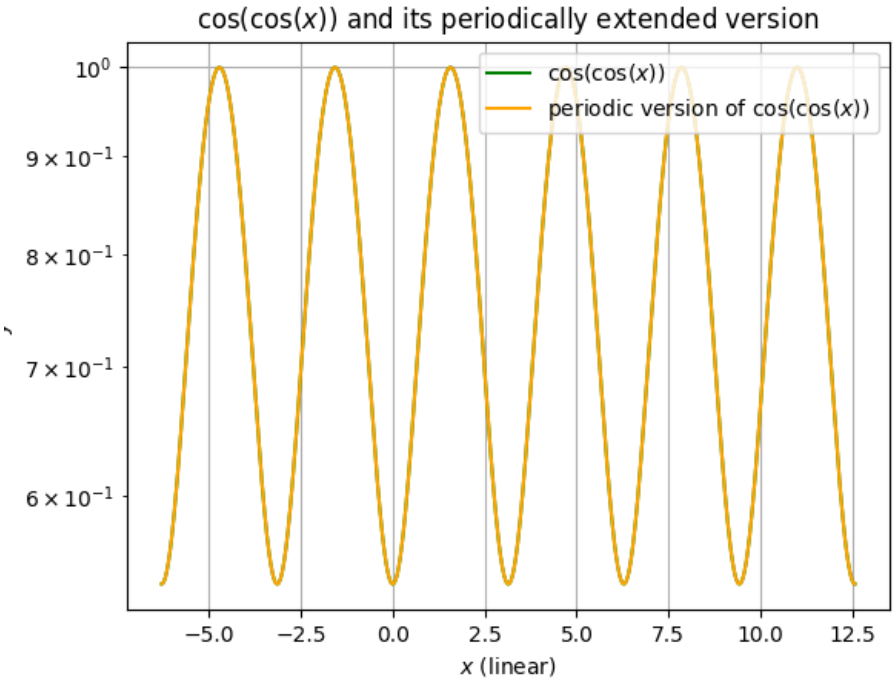

Figure 1: Data plot

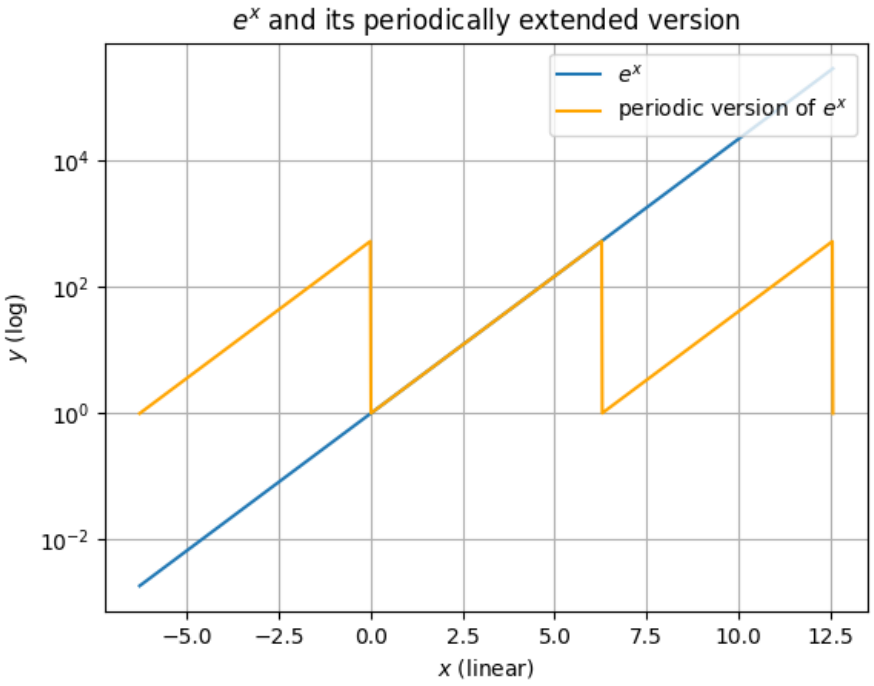

Figure 2: Data plot

1.0.2 Discussions

1. From the above plots, you can easily see that while $cos(cos(x))$ is periodic with period 2π, the latter, e^x isn't periodic and rises monotonically.

2. From the Fourier series, we expect that the Fourier approximation of the function would be repetitions of the function between $[0, 2\pi)$. We can thus say that the Fourier approximation of the function $f(x)$ is $f(x \mod 2\pi)$[1].

2 Question 2 &3

- Obtain the first 51 coefficients i.e., a_0, a_1, b_1,... for e^x and $cos(cos(x))$ using `quad` function.
- Calculate the function using those coefficients and compare with original functions graphically.
- Two different plots for each function using `semilogy` and `loglog` and plot the magnitude of the coefficients versus n.

‘

2.1 The Fourier coefficient

The fourier series used to approximate a function is as follows:

$$a_0 + \sum_{n=1}^{\infty} a_n \cos(nx_i) + b_n \sin(nx_i) \approx f(x_i) \tag{1}$$

The equations used here to find the Fourier coefficients are as follows:

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

$$a_n = \frac{1}{\pi} \int_0^{2\pi} f(x)\cos(nx)dx \tag{3}$$

$$b_n = \frac{1}{\pi} \int_0^{2\pi} f(x)\sin(nx)dx \tag{4}$$

Hence, in python we will use the *quad()* function to perform an intergration function. First we'll have to create functions which contains the variable k also. The python code snippet for declaring the functions with an additional variable k is as follows:

```
u1 = lambda x,k : e(x)*np.cos(k*x)
v1 = lambda x,k : e(x)*np.sin(k*x)
u2 = lambda x,k : f(x)*np.cos(k*x)
v2 = lambda x,k : f(x)*np.cos(k*x)
```

The python code snippet for finding the fourier coefficients is as follows:

```
a1 = [0]*26
b1 = [0]*26
a2 = [0]*26
b2 = [0]*26
a1[0] = quad(u1,0,2*pi,args=(0))
a2[0] = quad(u2,0,2*pi,args=(0))
F1 = [(1/(2*pi))*a1[0][0]]
F2 = [(1/(2*pi))*a2[0][0]]
```

[1]It is actually $f(x \mod p)$, where p the minimum among the period of the periodic function and 2π.

```
for i in range(1,26):
    a1[i] = quad(u1,0,2*pi,args=(i))
    F1.append((1/(pi))*a1[i][0])
    b1[i] = quad(v1,0,2*pi,args=(i))
    F1.append((1/(pi))*b1[i][0])
    a2[i] = quad(u2,0,2*pi,args=(i))
    F2.append((1/(pi))*a2[i][0])
    b2[i] = quad(v2,0,2*pi,args=(i))
    F2.append((1/(pi))*b2[i][0])
```

2.2 The plots of Fourier coefficients

The semilog and log plots of the Fourier coefficients of e^x is as shown:

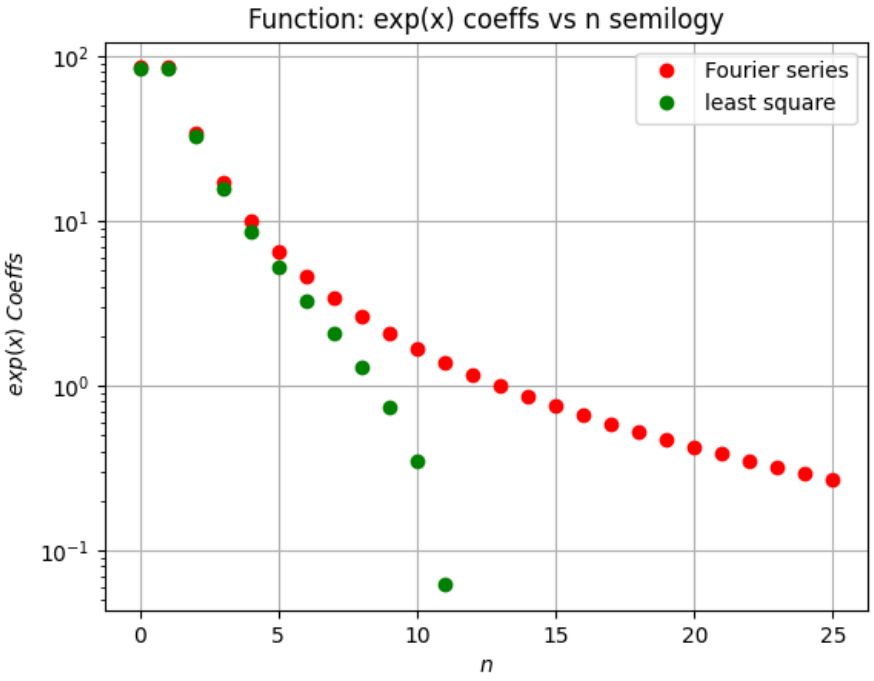

Figure 3: Semilog plot of the fourier coefficients of e^x

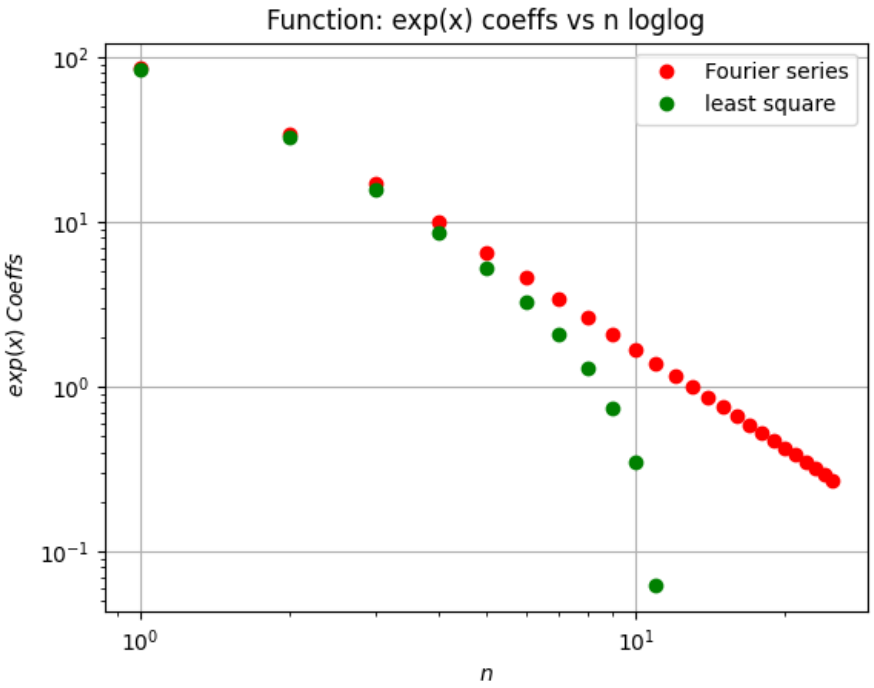

Figure 4: Log plot of the fourier coefficients of e^x

2.3 The plots of Fourier coefficients

The semilog and log plots of the Fourier coefficients of $cos(cos(x))$ is as shown:

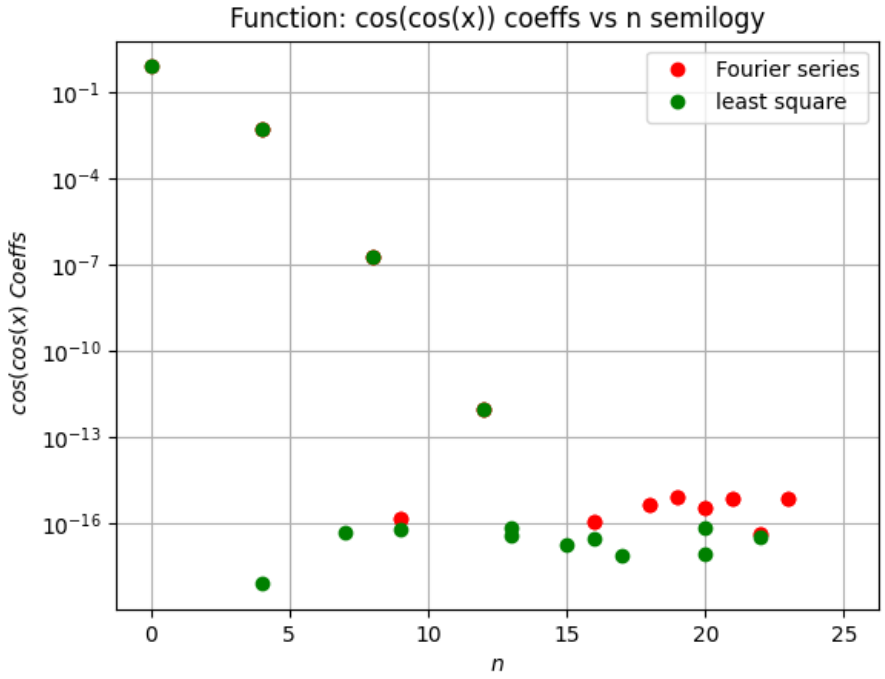

Figure 5: Semilog plot of the fourier coefficients of $cos(cos(x))$

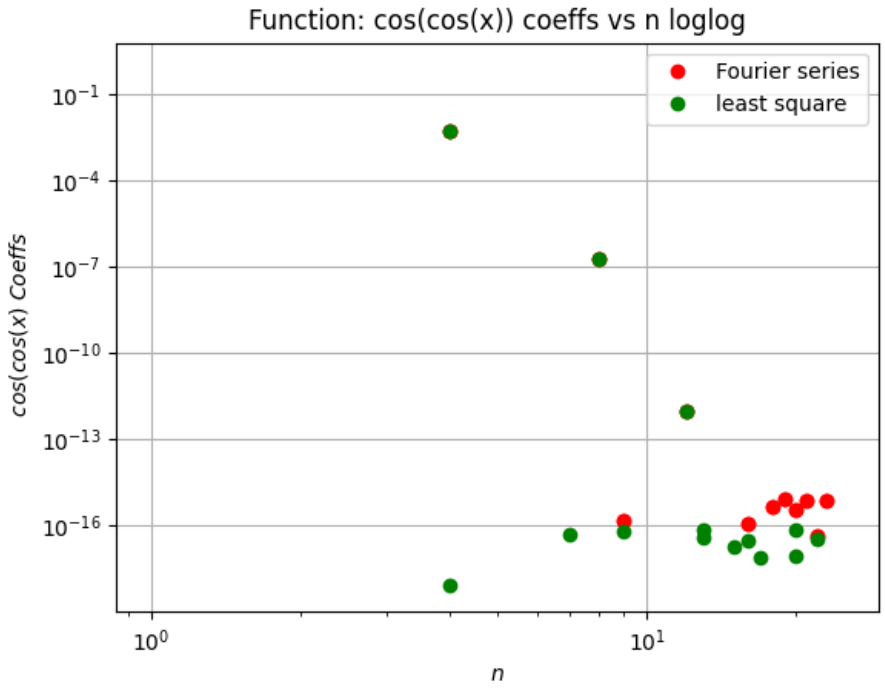

Figure 6: Log plot of the fourier coefficients of $cos(cos(x))$

2.3.1 Discussions

a. As it is evident from the plots, b_n is nearly zero for $cos(cos(x))$. This is because $cos(cos(x))$ is an even function, hence in the fourier series expansion, all the b_n terms should be zero for the series to be an even function.

b. The magnitude of the coefficients would represent how much of certain frequencies happen to be in the output. $cos(cos(x))$ does not have very many frequencies of harmonics, so it dies out quickly. However, since the periodic extension of e^x is discontinuous. To represent this discontinuity as a sum of continuous sinusoids, we would need high frequency components, hence coefficients do not decay as quickly.

c. The loglog plot is linear for e^x since Fourier coefficients of e^x decay with $1/n$ or $1/n^2$. The semilog plot seems linear in the $cos(cos(x))$ case as its fourier coefficients decay exponentially with n.

3 Question 4 &5

- Use a *Least Squares* approach to find the Fourier coefficients of the functions, using `scipy.linalg.lstsq`.
- Build the coefficient matrix A and the constant matrix b.

3.1 The Least Squares Approach

For the least squares approach, we'll have to create matrices and then use *lstsq()* function inorder to get the most approximate values of the fourier coefficients.
The python code snippet to create the matrices and to get the least squared value of the coefficients is as follows:

```
x = linspace(0,2*pi,401)
x = x[:-1]
a = zeros((400,51)) # Initializing empty matrix
```

```
a[:,0] = 1 # Making first column 1
for i in range(1,26):
    a[:,2*i-1] = cos(i*x) # Constructing the required matrix
    a[:,2*i] = sin(i*x)

#Question: 5
# Solving for the coefficients which give least square error
Y = linalg.lstsq(a,e(x),rcond=-1)[0]
Z = linalg.lstsq(a,f(x),rcond=-1)[0]
```

The plots in order to show the differences between the actual and predicted values of the fourier coefficients are shown below

4 Question 6

- Compare the coefficients obtained through the *least squares method* and the *direct integration* method.

- Find the maximum deviation between the coefficients obtained in the two methods.

4.1 Deviation from Actual Values

The least squares approach is still an approximate method and will definitely have a slight deviation from the actual value.

```
print("The max deviation in exp(x) is " + str(max(F1 - Y)))
print("The max deviation in cos(cos(x) is " + str(max(F2 - Z)))
```

The deviation in the case of e^x is 1.33273087
The deviation in the case of $cos(cos(x))$ is 0.0049539779

4.1.1 Discussions

There is very good agreement in values in the case of $cos(cos(x))$ but a significant amount of difference in the case of e^x. The reason for this is that the periodic extension of the exponential function is discontinuous, and hence would require a lot more samples to accurately determine its Fourier coefficients. If we increased the number of samples to 10^6, the maximum deviation would reduce, but not vanish. The effect of this lack of samples is felt more near the discontinuity of the signal.

5 Question 7

- Computing $A \cdot c$ from the estimated values of c by *Least Squares Method* and plotting them.

5.1 Estimated Functions

Using the predicted values of the fourier coefficients, we can calculate the functional values for both e^x and $cos(cos(x))$.

5.2 code

The following python code snippet is used to calculate the functional values:

```
plt.figure(1)
plt.title('Function : cos(cos(x)')
plt.scatter(x,np.dot(a,Z),color = 'green')
plt.grid(True)
plt.xlabel(r'$x$')
plt.ylabel(r'$cos(cos(x))$')
plt.plot(x0,y0,label = 'Original graph')
plt.plot(x,np.dot(a,Z),'g',label = 'Computed graph')
plt.legend(loc = 'upper right')
plt.show()

# plotting the original and computed graph of e(x)
plt.figure(2)
plt.title('Function : exp(x)')
plt.grid(True)
plt.xlabel(r'$x$')
plt.ylabel(r'$e(x)$')
plt.semilogy(x0,y1,label = 'Original graph')
plt.semilogy(x,np.dot(a,Y),'g',label = 'Computed graph')
plt.legend(loc = 'upper right')
plt.show()
```

The plots showing both the actual and predicted functional values are as shown below:

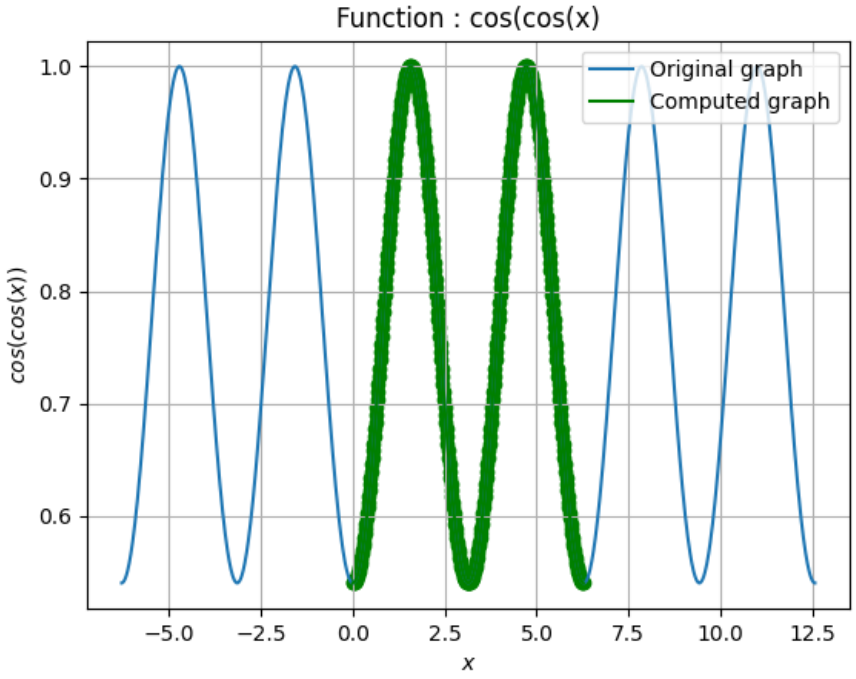

Figure 7: Actual and predicted values for e^x

5.2.1 Discussion

- As we observe that there is a significant deviation for e^x as it has discontinuities at $2n\pi$ which can be observed in the figure, and so there will be **Gibbs phenomenon** i.e, there will be

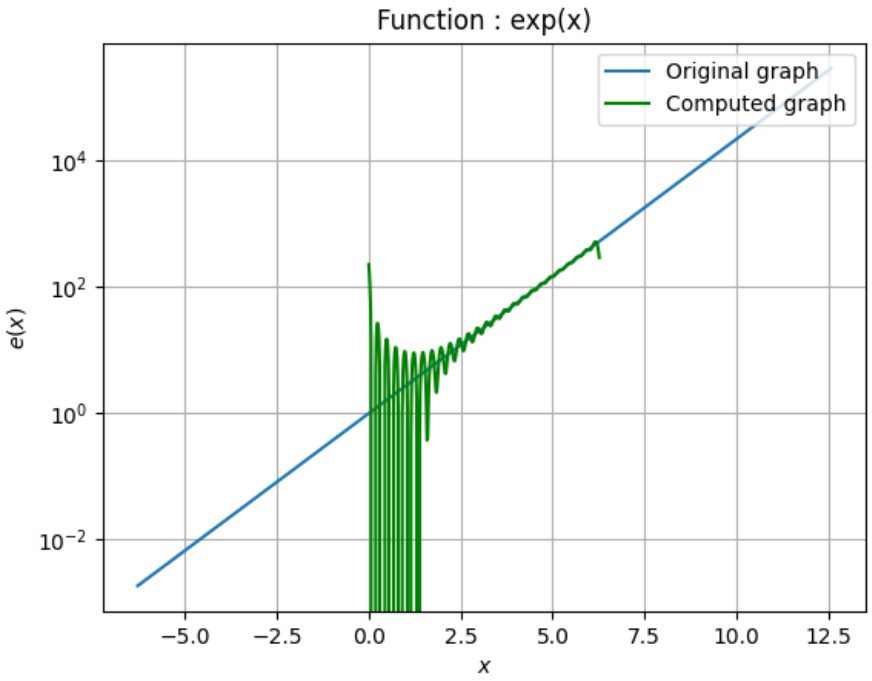

Figure 8: Actual and predicted values for $cos(cos(x))$

oscillations around the discontinuity points and their ripple amplitude will decrease as we go close to discontinuity. In this case it is at 2π for e^x.

- As we observe that ripples are high initially and reduces and oscillate with more frequency as we go towards 2π. This phenomenon is called **Gibbs Phenomenon**
- Due to this, the original function and one which is reconstructed using least squares will not fit exactly.
- And as we know that Fourier series is used to define periodic signals in frequency domain and e^x is a aperiodic signal so you can't define an aperiodic signal on an interval of finite length (if you try, you'll lose information about the signal), so one must use the Fourier transform for such a signal.
- That is why there are significant deviations for e^x from original function.
- Forcos(cos(x)) the curves fit almost perfectly because the function itself is a periodic function and it is a continuous function everywhere, so we get very negligible deviation and able to reconstruct the signal with just the Fourier coefficients.

5.2.2 Conclusion

We see that the Fourier estimation of e^x does not match significantly with the function close to 0, but matches near perfectly in the case of cos(cos(x)). This is due to the presence of a discontinuity at $x = 0$ for the periodic extension of e^x. This discontinuity leads to non-uniform convergence of the Fourier series, with different rates for both the functions.

The difference in the rates of convergence leads to the **Gibb's phenomenon**, which is the ringing observed at discontinuities in the Fourier estimation of a discontinuous function. This explains the mismatch in the Fourier approximation for e^x.

Thus we can conclude that the Fourier Series Approximation Method works extremely well for smooth periodic functions, but gives bad results for discontinuous periodic functions