Assignment 4: Fourier Approximations

SHEKHAR DIWAKAR [EE20B123]

March 11, 2022

Abstract

In this assignment we aim to :
A Fourier series is an expansion of a periodic function f(x) in terms of an infinite sum of sines and cosines. Fourier series make use of the orthogonality relationships of the sine and cosine functions. The computation and study of Fourier series is known as harmonic analysis and is extremely useful as a way to break up an arbitrary periodic function into a set of simple terms that can be plugged in, solved individually, and then recombined to obtain the solution to the original problem or an approximation to it to whatever accuracy is desired or practical.

1 The functions e^x and $cos(cos(x))$

The following python snippet is used to declare the functions e^x and $cos(cos(x))$. The x values are also declared from -2π to 4π.

```
#Defining the Functions exponential and cos(cos(x))
def f(x):
    return np.exp(x)
def g(x):
    return np.cos(np.cos(x))


x = np.linspace(-2*pi,4*pi,100)  #Dividing the 0 to 2*pi into 100 equal intervals
```

The following code is used to plot the graphs of e^x and $cos(cos(x))$.

```
#Plot of g(x) vs x
def plot_fig1():
    plt.grid()
    plt.plot(x,g(x))
    plt.title('Plot of g(x)')
    plt.xlabel('x')
```

```
    plt.ylabel('g(x)')
    plt.show()

def plot_fig2():
    plt.grid()
    plt.semilogy(x,f(x))
    plt.title('Plot of f(x) in semilog scale')
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.show()

#plot of f(x) vs x
figure(3)
def plot_fig3():
    plt.grid()
    plt.plot(x,f(x))
    plt.title('Plot of f(x)')
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.show()
```

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

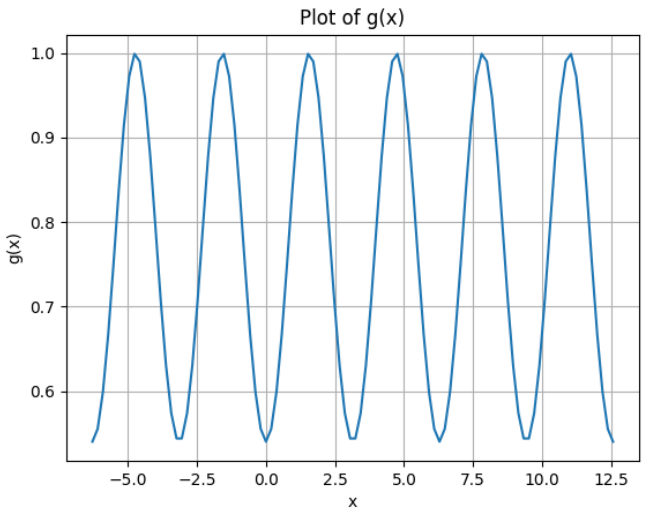

Figure 1: Data plot

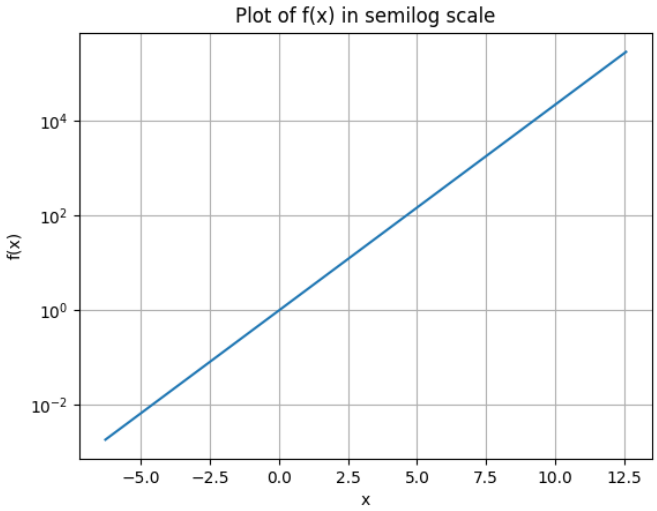

Figure 2: Data plot

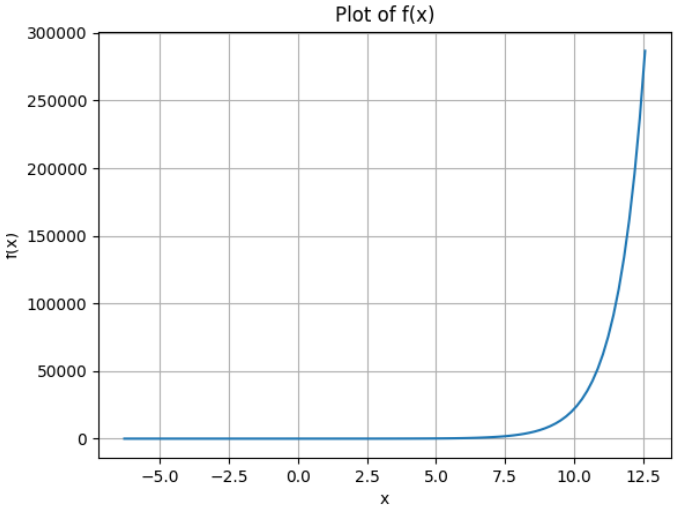

Figure 3: Data plot

2 The Fourier coefficients

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:

```
def m(x,k):
    return f(x)*np.cos(k*x)
def n(x,k):
    return f(x)*np.sin(k*x)
def o(x,k):
    return g(x)*np.cos(k*x)
def p(x,k):
    return g(x)*np.sin(k*x)
```

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

```
#Defining the empty matrixes for the fourier coefficients
a_coeff_of_f = np.zeros(26,)
b_coeff_of_f = np.zeros(25,)
a_coeff_of_g = np.zeros(26,)
b_coeff_of_g = np.zeros(25,)

#Integrating using the Quad integrate method
for i in range(26):
    fa[i]  = sp.integrate.quad(u,0,2*pi,(i,))[0]/pi
    ga[i] = sp.integrate.quad(w,0,2*pi,(i,))[0]/pi
for i in range(25):
    fb[i] = sp.integrate.quad(v,0,2*pi,(i+1,))[0]/pi
    gb[i] = sp.integrate.quad(z,0,2*pi,(i+1,)) [0]/pi

fa[0] /= 2
ga[0] /= 2
```

3 The plots of Fourier coefficients

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

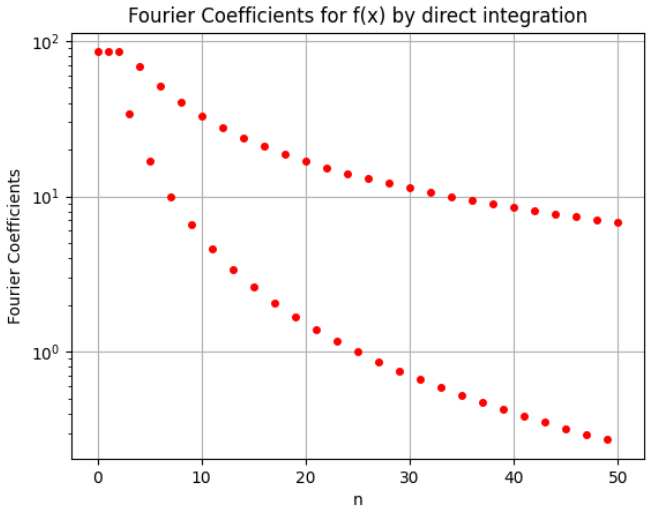

Figure 4: Semilog plot of the fourier coefficients of e^x

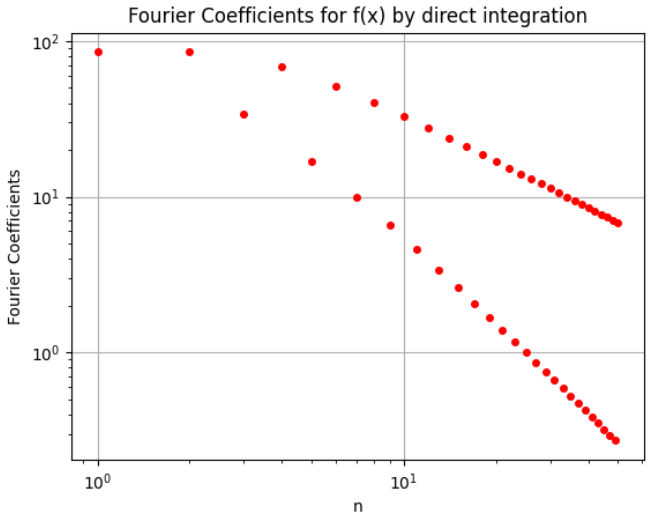

Figure 5: Log plot of the fourier coefficients of e^x

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 expan-

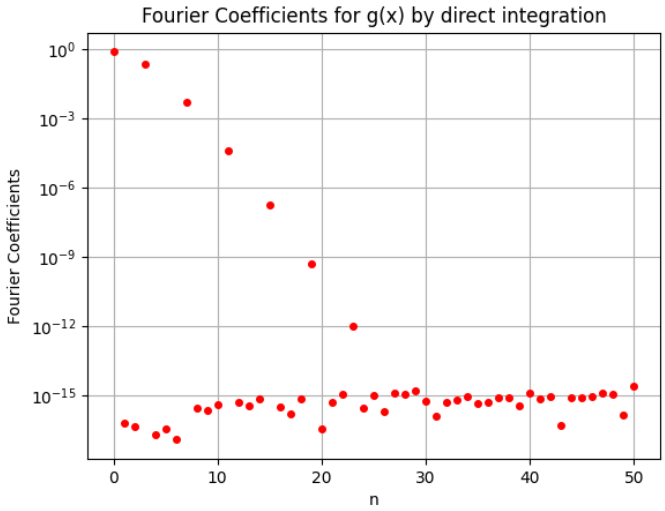

Figure 6: Semilog plot of the fourier coefficients of $cos(cos(x))$

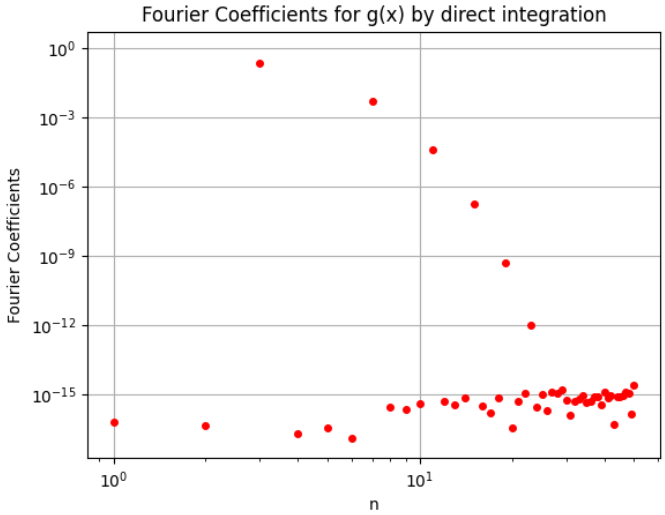

Figure 7: Log plot of the fourier coefficients of $cos(cos(x))$

sion, 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. $cos(cos(t))$ does not have very many frequencies of harmonics, so it dies out quickly. However, since the periodic extension of e^t 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^t since Fourier coefficients of e^t decay with $1/n$ or $1/n^2$. The semilog plot seems linear in the $cos(cos(t))$ case as its fourier coefficients decay exponentially with n.

4 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 = np.linspace(0,2*pi,401)
X = X[:-1]
b1 = f(X)
b2 = g(X)
A = np.zeros((400,51))
A[:,0] = 1
for k in range(1,26):
    A[:,2*k-1] = np.cos(k*X)
    A[:,2*k] = np.sin(k*X)

c1 = np.linalg.lstsq(A,b1,rcond = -1)[0]
c2 = np.linalg.lstsq(A,b2,rcond = -1)[0]
```

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

5 Deviation from Actual Values

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

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^t. 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.

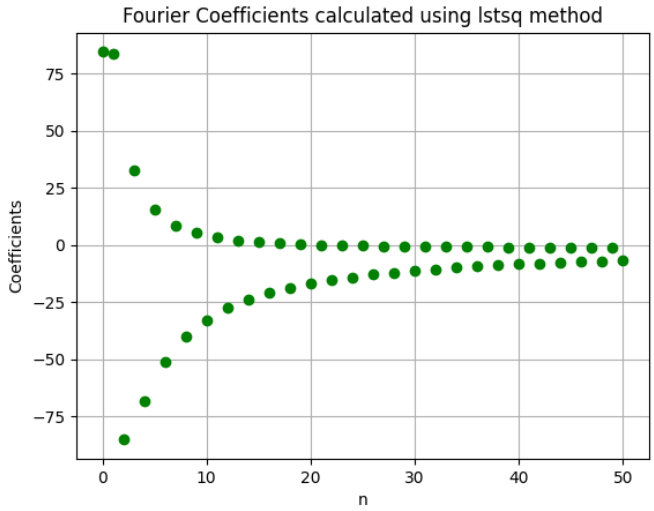

Figure 8: DATA PLOT

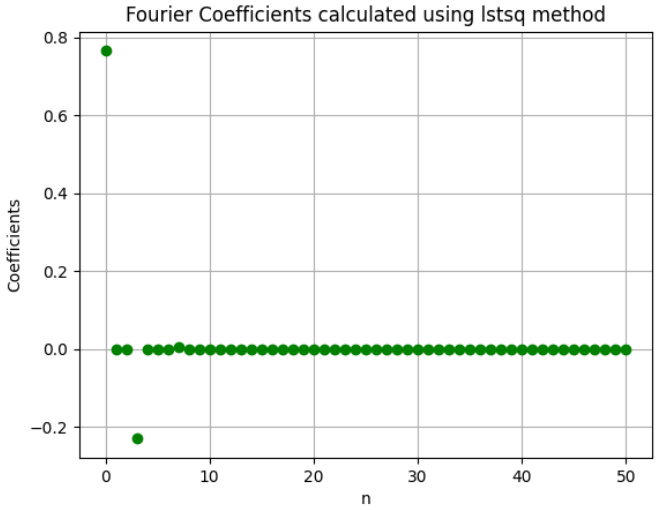

Figure 9: DATA PLOT

6 Estimated Functions

Using the predicted values of the fourier coefficients, we can calculate the functional values for both e^x and $cos(cos(x))$.
The plots showing both the actual and predicted functional values are as shown below:

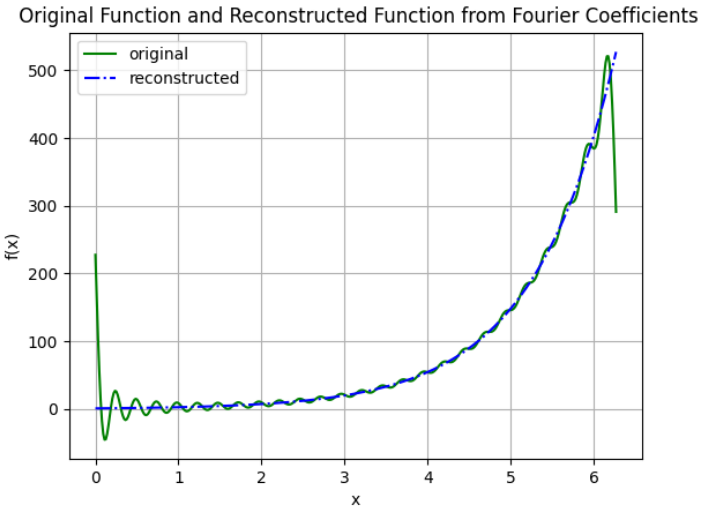

Figure 10: Actual and predicted values for e^x

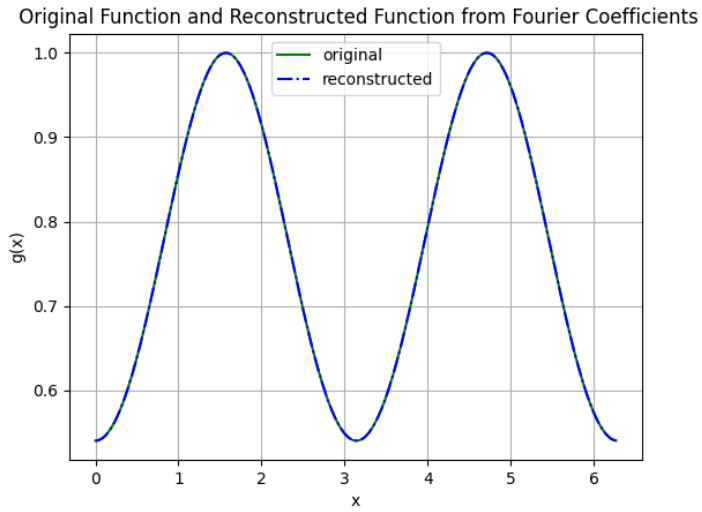

Figure 11: Actual and predicted values for $cos(cos(x))$

Conclusions

- We saw two different ways to calculate the Fourier series of a periodic signal.
- We saw how least squares fitting can be used to simplify the process of calculating the Fourier Series.

- We observed Gibbs phenomenon at the discontinuity in the Fourier approximation of e^t.