Assignment 4: Fourier Approximations

Siddharth D P
EE18B072

February 16, 2020

1 Introduction

The goal of the this assignment is to do the following:

- To fit two functions e^x and $cos(cos(x))$ using the Fourier series.
- To use least squares fitting to simplify the process of calculating Fourier series.
- To plot and compare the *loglog* and *semilog* plots with the true Fourier coefficients obtained by integration and the ones obtained by the least squares method on the same graph.
- Understand the deviations in the values obtained and analyse them.

2 Assignment

2.1 Part 1

Importing the standard libraries

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

Defining the given functions e^x and $cos(cos(x))$

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

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

A function to plot all function whenever required by passing the necessary arguments is created.

```
def plotter(fig_no,plot_x,plot_y,label_x,label_y,type=None,kind='b-',title=''):
    plt.figure(fig_no)
    plt.grid(True)
    if type =="semilogy":
        plt.semilogy(plot_x,plot_y,kind)
    elif type =='ll':
        plt.loglog(plot_x,plot_y,kind)
    elif type ==None:
        plt.plot(plot_x,plot_y,kind)
    plt.xlabel(label_x,size =18)
    plt.ylabel(label_y,size =18)
    plt.title(title)
```

Each of the functions is plotted only between -2π to 4π and periodic extensions are also shown.

```
t = np.linspace(-2*np.pi,4*np.pi,1200)
fr_length = np.arange(1,52) #Fourier Coefficient Number

plotter(1,t,exp(t),r"$t$",r"exp(t)","semilogy",title ="Plotting an exponential
semilog plot")
plotter(2,t,coscos(t),r"$t$",r"cos(cos(t))","semilogy",title="The cos(cos(.))
function on a log plot")
plotter(1,t,np.concatenate((exp(t)[400:800],exp(t)[400:800],exp(t)[400:800])),
r"$t",r"exp(t)","semilogy",'r-')
plt.legend(("The actual function","periodic extension"))
plotter(2,t,np.concatenate((coscos(t)[400:800],coscos(t)[400:800],coscos(t)
[400:80])),r"$t$",r"cos(cos(t)","semilogy",'r-')
plt.legend(("The actual function","Its periodic extension"))
```

2.2 Part 2

Using the scipy integrate function *quad* to calculate the Fourier coefficient integrals in a for loop. Two new functions **u** and **v** are defined.

```
def u1(x,k):
    return(coscos(x)*np.cos(k*x))

def v1(x,k):
    return(coscos(x)*np.sin(k*x))

def u2(x,k):
    return(exp(x)*np.cos(k*x))

def v2(x,k):
```

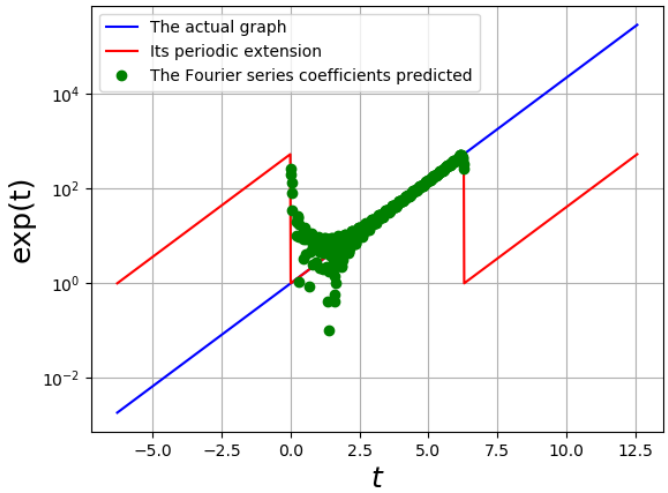

Figure 1: e^t vs t on a linear plot

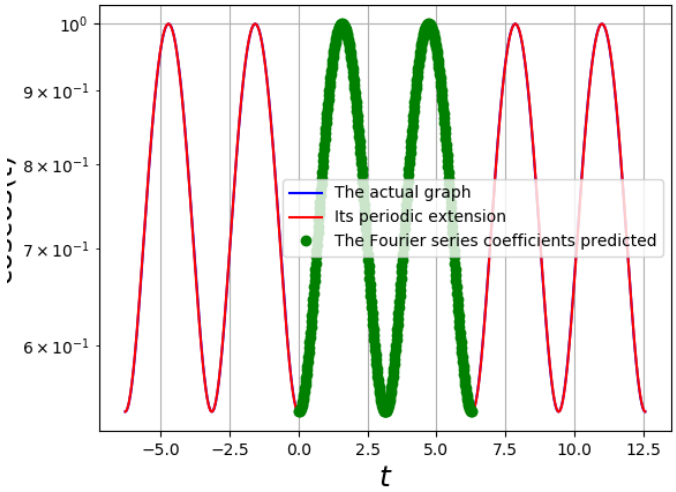

Figure 2: $cos(cos(t))$ vs t on a linear plot

```
    return(exp(x)*np.sin(k*x))

#Defining the Fourier coefficient integrating function
def integrate():
    a = np.zeros(51)
    b = np.zeros(51)
    a[0] =  scipy.integrate.quad(exp,0,2*np.pi)[0]/(2*np.pi)
```

```
    b[0] =  scipy.integrate.quad(coscos,0,2*np.pi)[0]/(2*np.pi)
    for i in range(1,51,2):
        a[i] = scipy.integrate.quad(u2,0,2*np.pi,args=(i//2+1))[0]/(np.pi)
        b[i] = scipy.integrate.quad(u1,0,2*np.pi,args=(i//2+1))[0]/(np.pi)
        a[i+1] = scipy.integrate.quad(v2,0,2*np.pi,args=(i//2+1))[0]/(np.pi)
        b[i+1] = scipy.integrate.quad(v1,0,2*np.pi,args=(i//2+1))[0]/(np.pi)
    return a,b

frexp,frcos = integrate()
```

2.3 Part 3

The Fourier coefficients are now plotted on the same graphs.

```
plotter(3,fr_length,np.absolute(frexp),"n","Magnitude","semilogy",'ro',
title="Semilog Fourier Coefficients for exp(t)")
plotter(4,fr_length,np.absolute(frexp),"n","Magnitude","ll",'ro',
title="Loglog Fourier Coefficients for exp(t)")
plotter(5,fr_length,np.absolute(frcos),"n","Magnitude","semilogy",'ro',
title="Semilog Fourier Coefficients for cos(cos(t))")
plotter(6,fr_length,np.absolute(frcos),"n","Magnitude","ll",'ro',
title="Loglog Fourier Coefficients for cos(cos(t))")
```

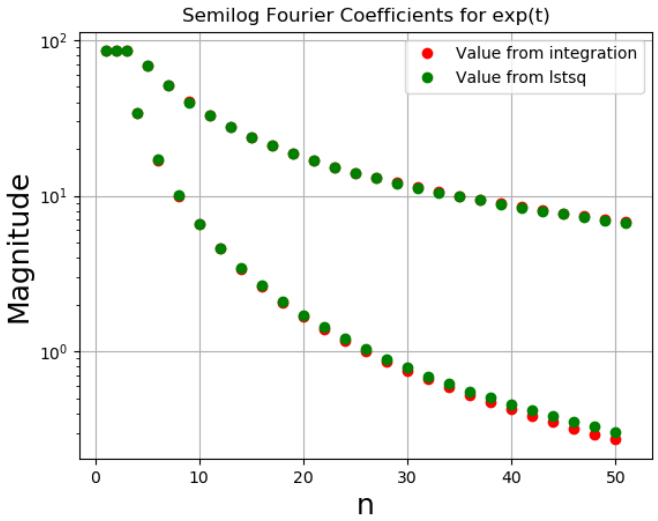

Figure 3: Semilog plot of magnitude of Fourier coefficients of e^t vs n

(*a*) The b_n coefficients are nearly zero for $cos(cos(t))$ since it is an even function, and hence does not have an odd component. All the b_n components are equal to 0.

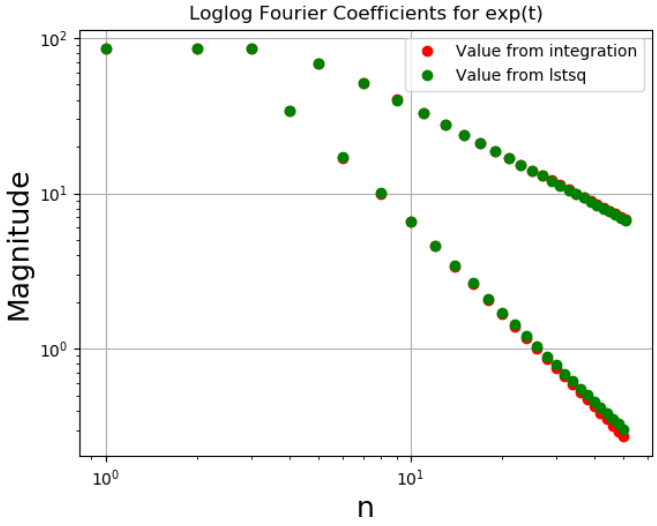

Figure 4: Loglog plot of magnitude of Fourier coefficients of e^t vs n

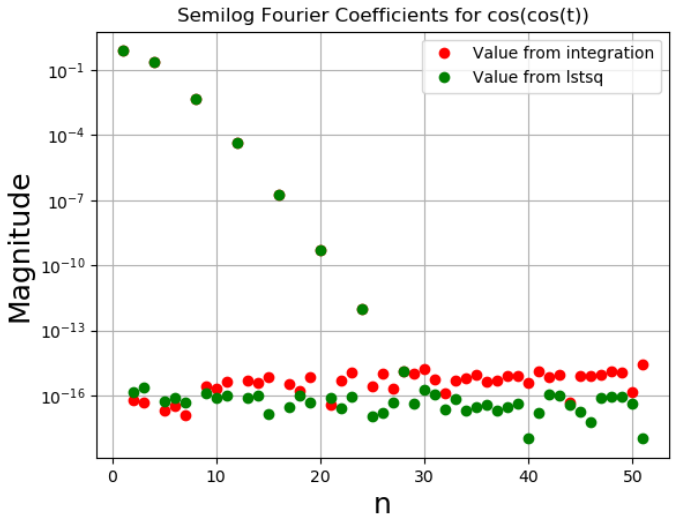

Figure 5: Semilog plot of Fourier coefficients of $cos(cos(t))$ vs n

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

(c) The *loglog* plot is linear for e^t since the Fourier coefficients of e^t decay

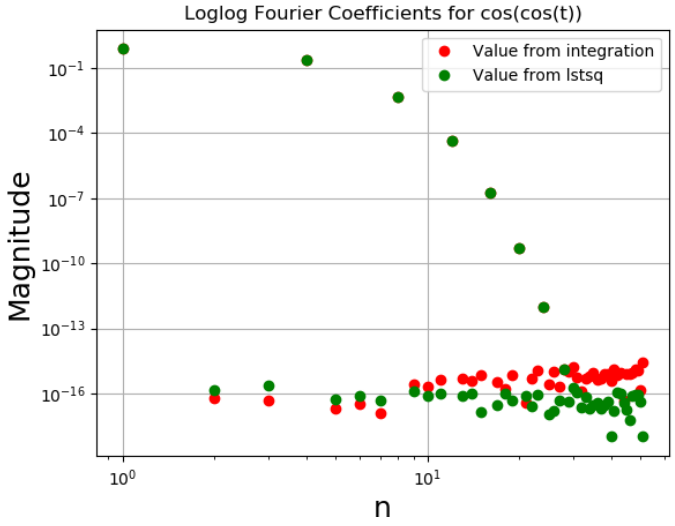

Figure 6: Loglog plot of of Fourier coefficients of $cos(cos(t))$ vs n

with $1/n$ or $1/n^2$ whereas the *semilog* plot seems linear in $cos(cos(t))$ as its Fourier coefficients decay exponentially with n.

2.4 Parts 4 and 5

The least squares approach is now used to calculate the Fourier series.

```
x =np.linspace(0,2*np.pi,400,endpoint =True)
bexp = exp(x)
bcoscos =coscos(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)
cexp = np.linalg.lstsq(A,bexp,rcond=None)[0]
ccoscos = np.linalg.lstsq(A,bcoscos,rcond=None)[0]
print('\nThe length of Coefficient matrix of:\n1. Least Squares: exponential is {},
cos(cos(.)) is {}\n2. Integration: exponential is {}, cos(cos(.)) is {}'.format(len
plotter(3,fr_length,np.abs(cexp),"n","Magnitude","semilogy",'go',title="Semilog
Fourier Coefficients for exp(t)")
plt.legend(("Value from integration","Value from lstsq"))
plotter(4,fr_length,np.abs(cexp),"n","Magnitude","ll",'go',title="Loglog Fourier
Coefficients for exp(t)")
plt.legend(("Value from integration","Value from lstsq"))
plotter(5,fr_length,np.abs(ccoscos),"n","Magnitude","semilogy",'go',title="Semilog
```

```
Fourier Coefficients for cos(cos(t))")
plt.legend(("Value from integration","Value from lstsq"))
plotter(6,fr_length,np.abs(ccoscos),"n","Magnitude","ll",'go',title="Loglog Fourier
Coefficients for cos(cos(t))")
plt.legend(("Value from integration","Value from lstsq"))
```

2.5 Part 6

The absolute difference between the two answers obtained by integration and least-squares approach is calculated in vector form the maximum value is obtained.

```
diffexp = np.absolute(cexp-frexp)
diffcos = np.absolute(ccoscos-frcos)
print('The maximum difference in exponential is {} and cos(cos(.)) is {}'.
format(np.amax(diffexp),np.amax(diffcos)))
print('\n')
print(np.around(diffexp,5),'\n',np.around(diffcos,20),'\n')
```

The max deviation in e^t is e0.0881216977876722 whereas for cos(cos(t)), it is 2.5466388061776397e-15

There is a very good agreement in values in the case of $cos(cos(t))$, 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 say, 10^6, the maximum deviation would possibly reduce, but it would have to reach infinity to vanish. The effect of this lack of samples is felt more near the discontinuity of the signal, which can be attributed to the Gibbs Phenomenon.

2.6 Part 7

The function is now computed using the estimated values of c from the least squares method.

```
Acexp = A@cexp
Accos = A@ccoscos
plotter(1,t,np.concatenate((np.zeros(400),Acexp,np.zeros(400))),r"$t$",r"exp(t)",
"semilogy",'go')
plt.legend(("The actual graph","Its periodic extension","The Fourier series
coefficients predicted"))
plotter(2,t,np.concatenate((np.zeros(400),Accos,np.zeros(400))),r"$t$",
r"coscos(t)",None,'go')
plt.legend(("The actual graph","Its periodic extension","The Fourier series
coefficients predicted"))
```

```
plt.show()
```

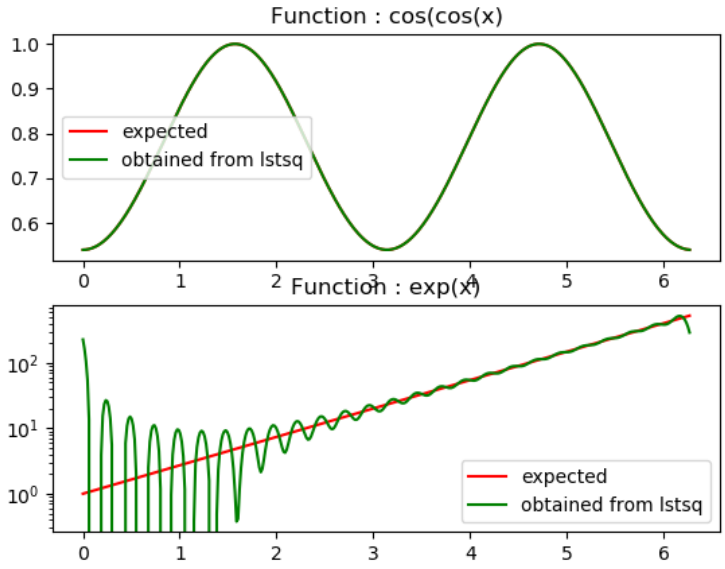

Figure 7: Expected curves and curves fitted using *lstsq* function

There is so much agreement in Figure 2 because the $cos(cos(t))$ function agrees beyond the scope of the precision of the least squares fitter.

On the other hand, there is considerable deviation in Figure 7 because the Fourier approximation of e^t presents a discontinuity. The cause for this is the Gibbs Phenomenon, which can be stated as-

The partial sums of the Fourier series have large oscillations near the discontinuity of the function. These oscillations do not die out as n increases, but approach a finite limit. This results in the occurrence of a "ringing" near the discontinuities. Plotting the output on a linear graph would make this ringing much more apparent.

3 Conclusion

- We saw two different ways to calculate the Fourier coefficients of a periodic and aperiodic (converted to its periodic extension!) using the usual integration approach and Scipy's in-built least-squares function.
- We saw how least-squares fitting can be used to obtain the magnitude of the Fourier coefficients.

- As the discontinuity in the Fourier approximation of e^t was very prominent, it resulted in large deviations compared to $cos(cos(t))$.