EE2703 : Applied Programming Lab
Assignment 4
Fouier Approximations

T.M.V.S GANESH
EE19B124

March 12, 2021

Introduction

The assignment is about

- Fitting the functions exp(x) and cos(cos(x)) over the interval $[0, 2\pi]$
- Finding the first 51 coefficients using integration and least squares method
- plotting graphs and comparing coefficients

Q1

Defining the Required Functions to calculate the coefficients of exp(x) and cos(cos(x))

Python code:

```
    # Required Functions
def f(x):
    return np.exp(x)
def g(x):
    return np.cos(np.cos(x))
def u(x,k,f):
    return f(x)*np.cos(k*x)
def v(x,k,f):
    return f(x)*np.sin(k*x)
```

Graph:
This code plots the original function from the interval[-2π,4π] and the function obtained from fourier analysis.Fourier analysis gives the function in the region [0,2π] and varies periodically with period 2π

Python Code:

```
#Original and expected functions from fourier series
x = np.arange(-2*np.pi,4*np.pi,0.01)
a = np.arange(-2*np.pi,4*np.pi,0.01)
for i in range(len(a)):
    if a[i] > 2*np.pi or a[i] < 0:
        a[i] = a[i] - (2*pi)*(a[i]//(2*np.pi))
figure(1,figsize=(10,10))    #plot of exp(x) in semilogy axis
semilogy(x,f(x),label = 'Actual Graph')

figure(2,figsize=(10,10)) #plot of cos(cos(x))
plot(x,g(x),label = 'Actual Graph')

```

```
figure(1)
semilogy(x,f(a),label = 'Graph obtained by fourier Analysis')#expected plot from
    fourier series
title("Fig 1:Actaul vs expected on semilog scale of exp(x)")
xlabel("x",fontsize = 15)
ylabel("y(logscale)",fontsize = 15)
legend()
grid()

figure(2)
plot(x,g(a),label = 'Graph obtained by fourier Analysis') #expected plot from
    fourier series
legend(loc = 'upper right')
title("Fig 2:Actaul vs expected of cos(cos(x))")
xlabel("x",fontsize = 15)
ylabel("y",fontsize = 15)
grid()
```

"a" in the above program returns the value corresponding to that in region $[0,2\pi]$ with period 2π by adding or subtracting multiples of 2π

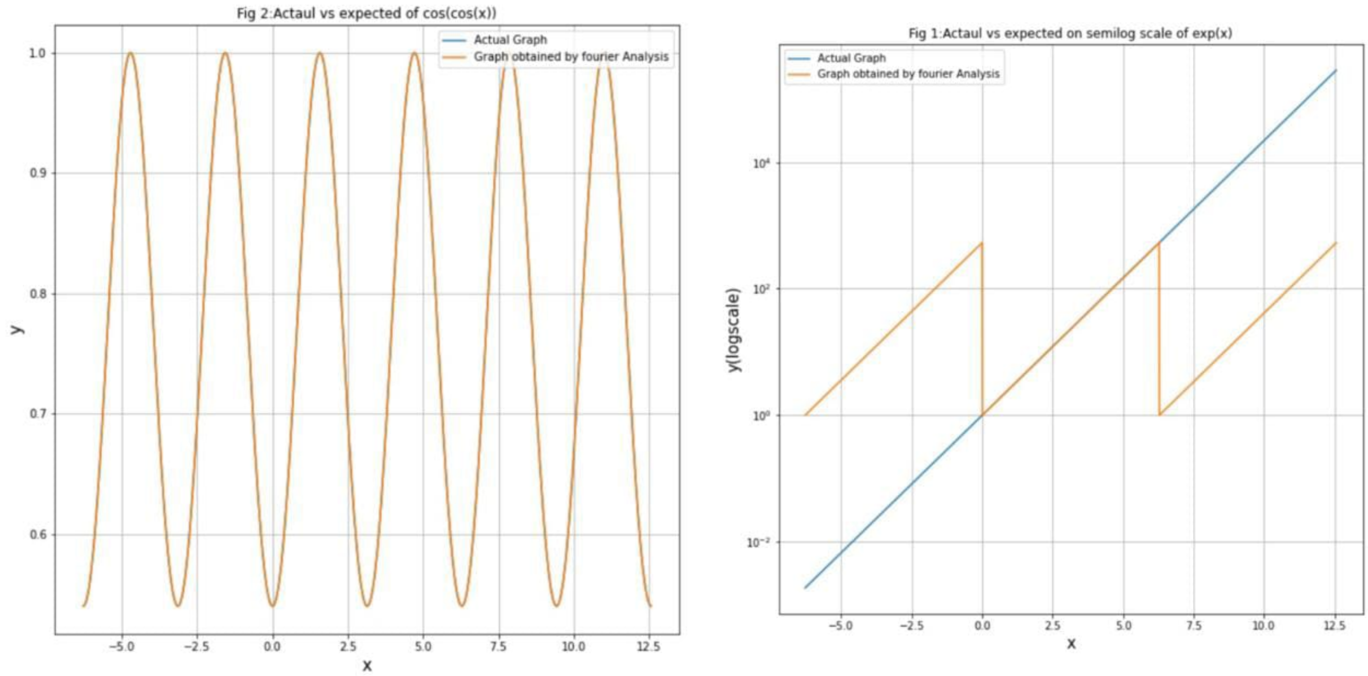

Figure 1: Actual and expected graphs of functions

Q2

The fourier series of any function can be represented 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 which are used to find the Fourier coefficients are as follows:

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

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

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

Computing the first 51 coefficients of the functions using the quad function and storing the a_n and b_n of the functions separately

Python code:

```
    #Q2 Computing the Coefficients

bf_0 = 0
af_0 = integral.quad(f,0,2*np.pi)[0] / (2*np.pi)
coefficients_f = [af_0,bf_0] #first 51 coefficients of exp(x)
for i in range(1,26):
    a_n = integral.quad(u,0,2*pi,args = (i,f))[0] / np.pi
    b_n = integral.quad(v,0,2*pi,args = (i,f))[0] / np.pi
    temp = [a_n,b_n]
    coefficients_f = coefficients_f + temp
bg_0 = 0
ag_0 = integral.quad(g,0,2*np.pi)[0] / (2*np.pi)
coefficients_g = [ag_0,bg_0]#first 51 coefficients of cos(cos(x))
for i in range(1,26):
    a_n = integral.quad(u,0,2*pi,args = (i,g))[0] / np.pi
    b_n = integral.quad(v,0,2*pi,args = (i,g))[0] / np.pi
    temp = [a_n,b_n]
    coefficients_g = coefficients_g + temp
af_n = []
bf_n = []
for i in range(52):
    if i%2 == 0:
        af_n = af_n +[coefficients_f[i]]
    else:
```

```
        bf_n = bf_n + [coefficients_f[i]]
ag_n = []
bg_n = []
for i in range(52):
    if i%2 == 0:
        ag_n = ag_n +[coefficients_g[i]]
    else:
        bg_n = bg_n + [coefficients_g[i]]
coefficients_f.remove(coefficients_f[1])
coefficients_g.remove(coefficients_g[1])
```

Q3

Plotting the magnitude of Coefficients in loglog and smilog y axis the coefficients include first 26 a_n and first 25 b_n terms

Python code:

```
#Q3 Plotting magnitude of coefficients vs n

figure(3,figsize=(10,10))       #plot of coefficients  of exp(x) in semilog axis
semilogy(range(26),af_n,'ro',label='a_n of exp(x)',markersize=4)
semilogy(range(1,26),np.abs(bf_n[1:]),'bo',label='b_n of exp(x)',markersize=6)
figure(4,figsize=(10,10))    #plot of coefficients  of exp(x) in loglog axis
loglog(range(26),af_n,'ro',label='a_n of exp(x)',markersize=4)
loglog(range(1,26),np.abs(bf_n[1:]),'bo',label='b_n of exp(x)',markersize=6)
figure(5,figsize=(10,10))  #plot of coefficients  of cos(cos(x)) in semilog axis
semilogy(range(26),np.abs(ag_n),'ro',label='a_n of cos(cos(x))',markersize=4)
semilogy(range(1,26),np.abs(bg_n[1:]),'bo',label='b_n of  cos(cos(x))',markersize=4)
figure(6,figsize=(10,10))  #plot of coefficients  of cos(cos(x)) in loglog axis
loglog(range(26),np.abs(ag_n),'ro',label='a_n of cos(cos(x))',markersize=4)
loglog(range(1,26),np.abs(bg_n[1:]),'bo',label='b_n of cos(cos(x))',markersize=4)
```

Answers:
*(a)*The b_n coefficients of cos(cos(x)) are nearly zero in this case since g(x) is a even function but the quad function considers them as very low value which is why there is a deviation from zero
*(b)*The Coefficients of cos(cos(x)) die out quickly as the contribution from the larger frequencies is quite low.This can be observed from the expansion of cos(x) by replacing x with cos(x) as the power increases the coefficients in expansion varies as (1/n!) and cos^{2n}(x) can be represented in terms of sum of cos(kx),k ¡ 2n where as coefficients of exp(x) are given below and are multiplied by a large constant

*(c)*The Coefficients of exp(x) computed from the integral are

$$a_n = \frac{e^{2\pi} - 1}{\pi(n^2+1)} \tag{5}$$

$$b_n = \frac{(e^{2\pi} - 1)n}{\pi(n^2+1)} \tag{6}$$

for $n >> 1$ the variations of a_n and b_n can be treated as $1/n^2$ and $1/n$
so in loglog axis the plot is against log(coefficients) vs log(n) which gives a approximate linear plot with negative slope

Q4

Computing the Coefficients using the Least Squares approach by fitting the function at 400 points in [0,2π]

$$a_0 + \sum_{n=1}^{25} a_n \cos(nx_i) + b_n \sin(nx_i) \approx f(x_i) \tag{7}$$

Python code:

```
#Q4 Computing the coefficients using the least squares approach

x2=linspace(0,2*pi,401)
x2=x2[:-1] # drop last term to have a proper periodic integral
b_f=f(x2)# f has been written to take a vector
b_g = g(x2)
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*x2) # cos(kx) column
    A[:,2*k]=sin(k*x2) # sin(kx) column
#endfor
c1=lstsq(A,b_f,rcond = None)[0]# the [0] is to pull out the best fit vector.
    lstsq returns a list.
c2 = lstsq(A,b_g,rcond = None)[0]
pred_af = [c1[0]] #predicted coefficients
pred_bf = []
for i in range(1,51):
    if i%2 == 0:
        pred_bf = pred_bf + [c1[i]]
    else:
        pred_af = pred_af + [c1[i]]
pred_ag = [c2[0]]  #predicted coefficients
pred_bg = []
for i in range(1,51):
    if i%2 == 0:
```

```
            pred_bg = pred_bg + [c2[i]]
        else:
            pred_ag = pred_ag + [c2[i]]
```

c1 and c2 represent the predicted coefficient matrix of exp(x) and cos(cos(x)) respectively
pred_af,pred_bf represent lists of a_n of f(x) and g(x) similarly for b_n

Q5

Plotting the Coefficients from Least Squares along with that of Calculated coefficients
Python code:

```

#Q5 PLotting the coefficients of f(x),g(x) in semilog,loglog axis

figure(3)
semilogy(range(26),np.abs(pred_af),'go',label='predicted_an of exp(x)',markersize
    =4)
semilogy(range(1,26),np.abs(pred_bf),'yo',label='predicted_bn of exp(x)',
    markersize=4)
legend()
title("Fig 3:Calculated vs predicted coeff of exp(x)")
xlabel("n",fontsize = 15)
ylabel("Coefficients(semilog)",fontsize = 15)
grid()
```

In the Same Way remaining plots are plotted
Graphs:

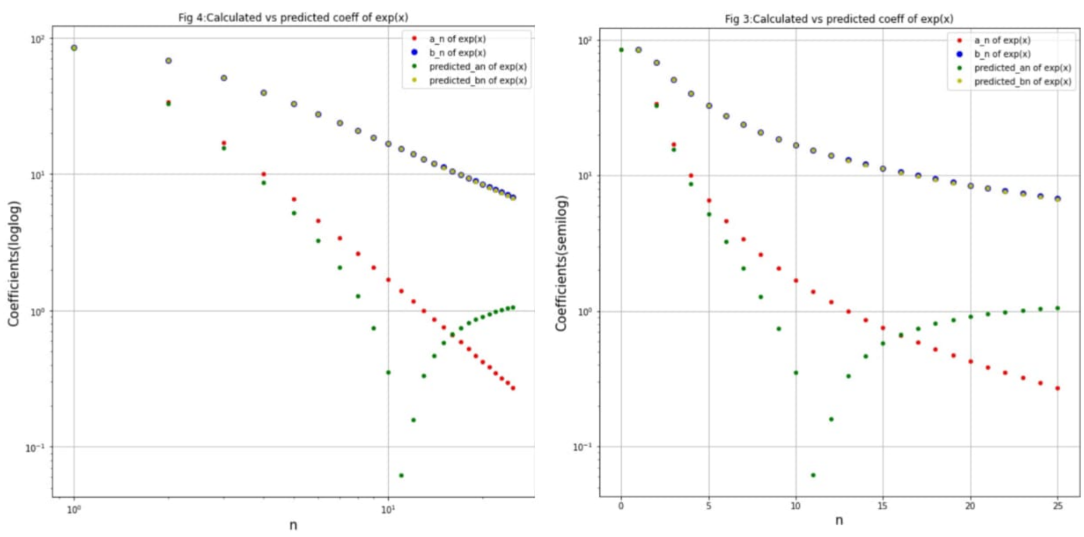

Figure 2: Graph of coefficients of exp(x) in loglog,semilog axis

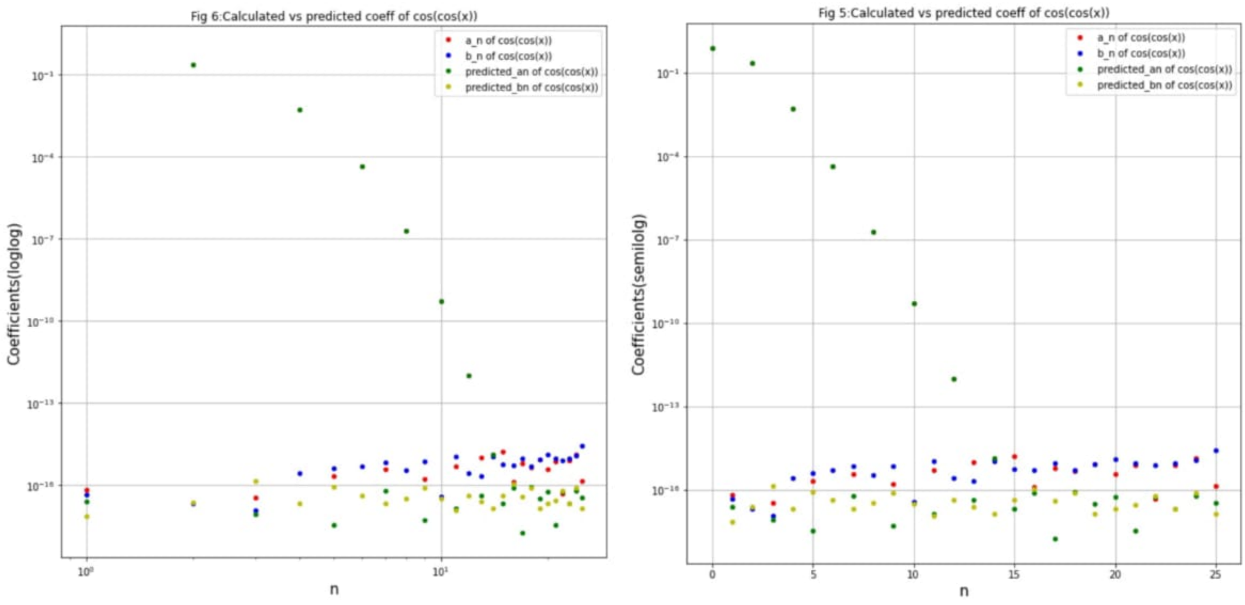

Figure 3: Graph of coefficients of cos(cos(x)) in loglog,semilog axis

Q6

Comparing the coefficients obtained from Least Squares approach and fourier analysis
Clearly from the Graph, both the results donot match as in the least squares approach we have considered only 400 points and the values are such that error at that points are minimal where as fourier takes the integarl
Caluculating deviation:

```
#Q6

deviation_f = max(np.abs(coefficients_f - c1))
deviation_g = max(np.abs(coefficients_g - c2))
print("The Coefficients obtained from both the methods are Different from the
    Graph")
print("The maximum deviation betwwen the coefficients of exp(x) is {}".format(
    deviation_f))
print("The maximum deviation betwwen the coefficients of cos(cos(x)) is {}".
    format(deviation_g))
```

Q7

Computing the Estimated function from Fourier and Estimated coefficients and plotting:
Python Code:

```
#Q7

figure(7,figsize=(10,10))
semilogy(x2,f(x2),'bo',label='exp(x)',markersize=2)
semilogy(x2,(np.array(A) @ np.array(coefficients_f)),'ro',markersize=4,label = '
    Fourier')
semilogy(x2,(np.array(A) @ np.array(c1)),'go',markersize=4,label = 'lstsq')
legend()
title("Calculated vs predicted functions of exp(x)")
xlabel("x",fontsize = 15)
ylabel("y(semilog)",fontsize = 15)
grid()

figure(8,figsize=(10,10))
semilogy(x2,g(x2),'yo',label='cos(cos(x))',markersize=6)
semilogy(x2,(np.array(A) @ np.array(coefficients_g)),'ro',markersize=5,label = '
    Fourier')
semilogy(x2,(np.array(A) @ np.array(c2)),'go',markersize=3.5,label = 'lstsq')
legend()
title("Calculated vs predicted fittings of cos(cos(x))")
xlabel("x",fontsize = 15)
ylabel("y(semilog)",fontsize = 15)
grid()
```

The @ command gives the matrix multiplication of the two matrices

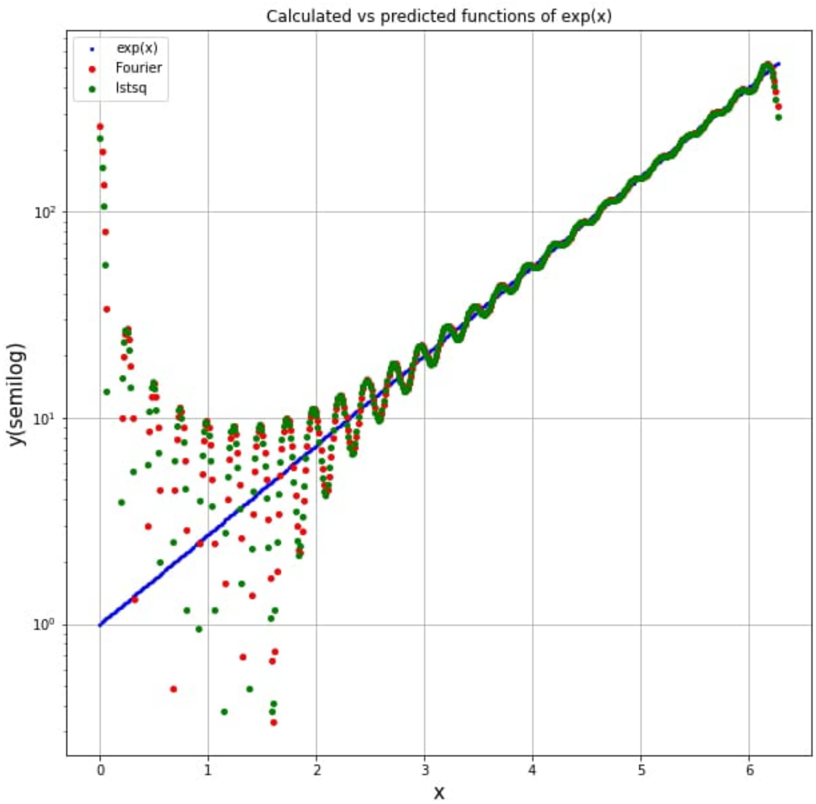

Figure 4: Estimated and actual plots of exp(x)

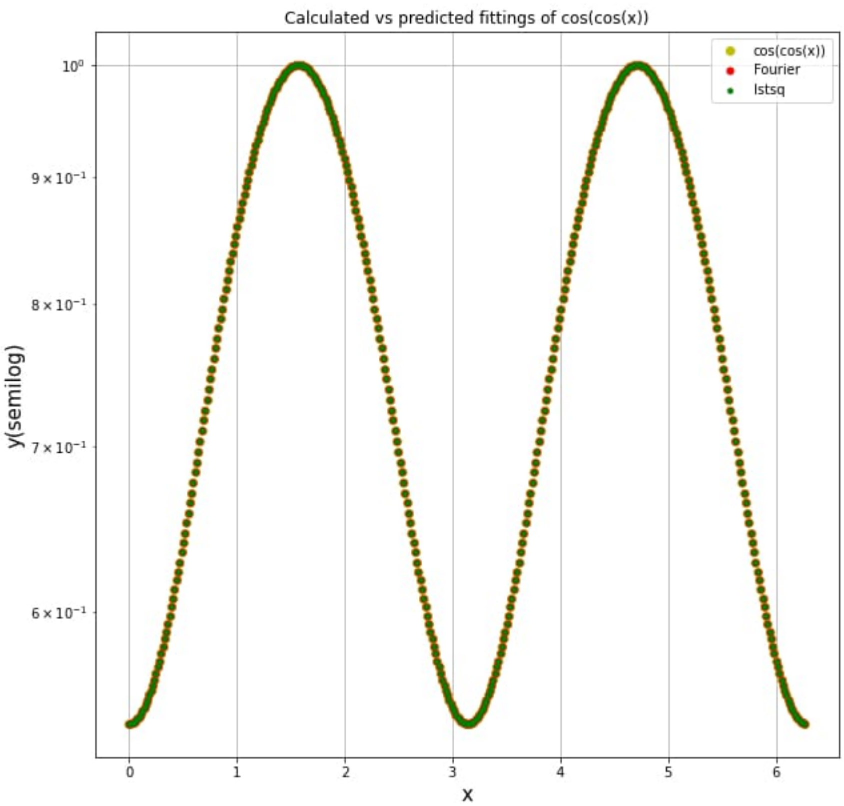

Figure 5: Estimated and actual plots of cos(cos(x))

Reason: There is a lot of deviation in the figure of exp(x) where as it is nearly agreed in the case of cos(cos(x)) as the functions expected from fourier analysis of exp(x) is discontinuous at the ends 0,2π where as cos(cos(x)) is continuous and from the graphs the coefficients of cos(cos(x)) decay very quick and the effect of higher coefficients is nearly negligable where as that of exp(x) coefficients donot tend to.As we have considered only first 51 coefficients the error in calculating exp(x) is high and in cos(cos(x)) is very low