EE2703 Assignment 4

Author: Pranav Prasad Phatak , EE19B105

10^{th} March 2021

1 Abstract

We will fit two functions, e^x and $cos(cos(x))$ made periodic over their values in the interval $[0, 2\pi)$ using their computed Fourier series coefficients.

2 Introduction

The Fourier Series of a function $f(x)$ with period 2π is computed as follows:

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

where ,

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

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

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

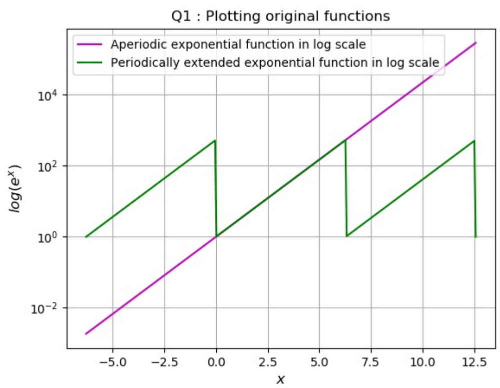
Since e^x doesn't have a period of 2π, we will make it periodic by defining the function as $e^{x\%(2\pi)}$

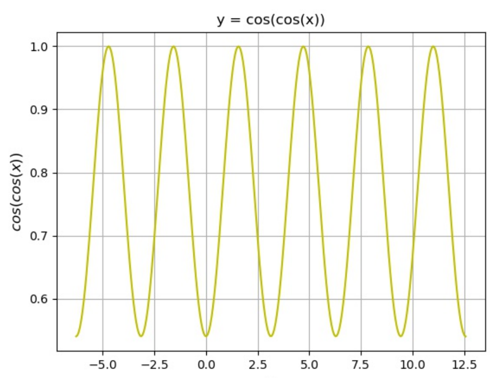
3 Assignment Questions

3.1 Creating the functions

$cos(cos(x))$ is a periodic function with a fundamental period of π, so $2 * \pi$ is also its period, whereas e^x is not a periodic function. The functions that will be generated from the Fourier series are $cos(cos(x))$ and $e^{x\%(2\pi)}$

```
f1 = lambda x: np.exp(x)                              #Defining the 2 given functions
f2 = lambda x: np.cos(np.cos(x))

f1_periodic = lambda x: np.exp(x%(2*math.pi))    #Defining the periodically extended ex
```





3.2 Generating Fourier Coefficients

The first 51 coefficients are generated using the scipy.integrate.quad and the equations mentioned in the introduction function.They are saved in the following form as required by part 3:

$$\begin{bmatrix} a_0 \\ a_1 \\ b_1 \\ \dots \\ a_{25} \\ b_{25} \end{bmatrix}$$

```
    #Returns the first k ak and (k-1) bk fourier series coefficients for a function f
def FSC(f,k):
    coeff=[]                    #List which stores all the coefficients
    a=[]                        #List of only an's
    b=[]                        #List of only bn's

    u = lambda x, n: f(x)*math.cos(n*x)
    v = lambda x, n: f(x)*math.sin(n*x)

    b.append(0)
#Since there is no b(0) will make it 0
    a.append((1/(2*math.pi))*integ.quad(u, 0, 2*math.pi, args=0)[0])
    coeff.append((1/(2*math.pi))*integ.quad(u, 0, 2*math.pi, args=0)[0])
#Solve for n=0
    for n in range(1,k):
        a.append((1/math.pi)*integ.quad(u, 0, 2*math.pi, args=n)[0])
        coeff.append((1/math.pi)*integ.quad(u, 0, 2*math.pi, args=n)[0])

        b.append((1/math.pi)*integ.quad(v, 0, 2*math.pi, args=n)[0])
        coeff.append((1/math.pi)*integ.quad(v, 0, 2*math.pi, args=n)[0])

    return coeff,a,b

coeff_f1,a_f1,b_f1 = FSC(f1,26)
coeff_f2,a_f2,b_f2 = FSC(f2,26)
```

3.3 Visualizing Fourier Coefficients

As expected, $\sum \|b_n\|$ for cos(cos(x)) = 2.09e-14 which is almost 0. The coefficients for e^x decay faster than that of The Log-log plot for Fourier coefficients of e^x is nearly linear because :

$$\int_0^{2\pi} e^x cos(kx)dx = \frac{(e^{2\pi} - 1)}{(k^2 + 1)} \tag{2}$$

and

$$\int_0^{2\pi} e^x sin(kx)dx = \frac{(-ke^{2\pi} + k)}{(k^2 + 1)} \tag{3}$$

The log-log plots of these functions are linear

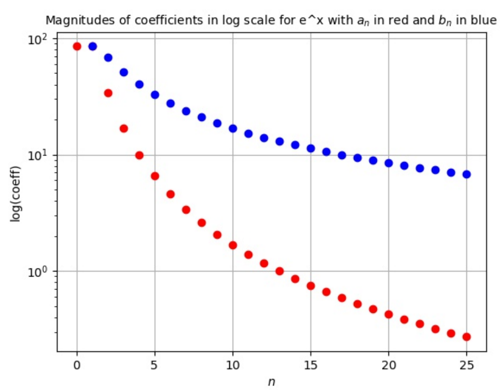
The semi-logy plot for Fourier Coefficients of $cos(cos(x))$ is linear as the integral converges to a Linear Combination of Bessel functions which are proportional to e^x.

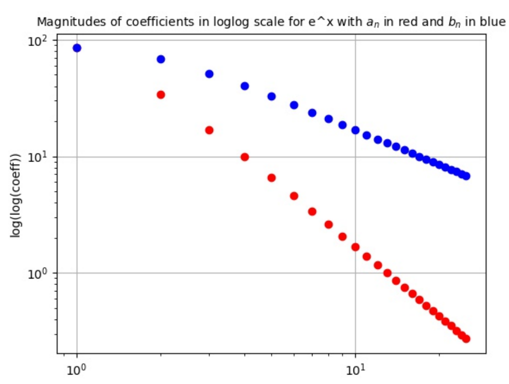
In all of the following graphs, we have been told to plot fourier coefficients vs n, so I will use list having all $a_i's$ and all $b_i's$ where $b_0 = 0$ and rest all $b_i's$ corresponding to actual fourier coefficient for that i, so that plots will have entry of b_i at i on the x axis

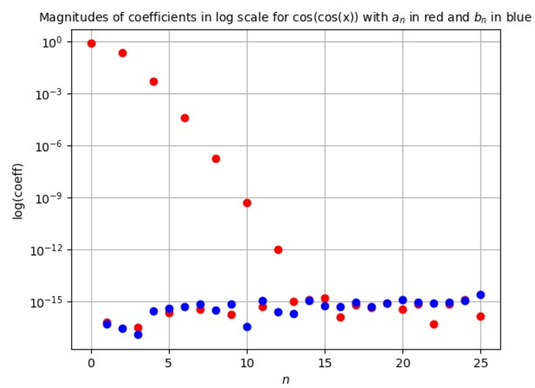
---

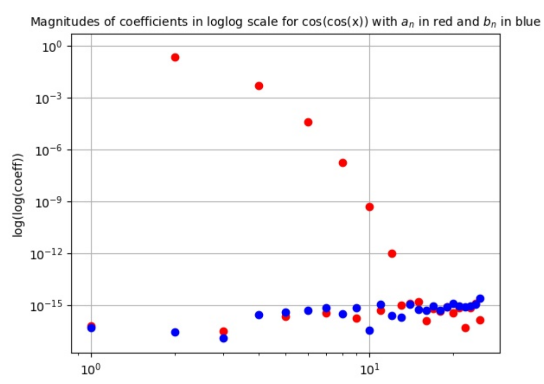
```
#Plots the coefficients in semilogy and loglog scale
def plotting_coefficients():
#In all the graphs b(0) will be 0
    figure(3)
    semilogy(np.abs(a_f1), 'ro')
    semilogy(np.abs(b_f1), 'bo')

    grid(True)
    title(r'Magnitudes of coefficients in log scale for e^x with $a_n$ in red and $b_n$ in blu
    ylabel(r'log(coeff)')
    xlabel(r'$n$')
    savefig("Ques3_1.jpg")
    close()


    figure(4)
    loglog(np.abs(a_f1), 'ro')
    loglog(np.abs(b_f1), 'bo')

    grid(True)
    title(r'Magnitudes of coefficients in loglog scale for e^x with $a_n$ in red and $b_n$ in 
    ylabel(r'log(log(coeff))')
    xlabel(r'$n$')
    savefig("Ques3_2.jpg")
    close()


    figure(5)
    semilogy(np.abs(a_f2), 'ro')
    semilogy(np.abs(b_f2), 'bo')
```

```
grid(True)
title(r'Magnitudes of coefficients in log scale for cos(cos(x)) with $a_n$ in red and $b_n
ylabel(r'log(coeff)')
xlabel(r'$n$')
savefig("Ques3_3.jpg")
close()


figure(6)
loglog(np.abs(a_f2), 'ro')
loglog(np.abs(b_f2), 'bo')

grid(True)
title(r'Magnitudes of coefficients in loglog scale for cos(cos(x)) with $a_n$ in red and $
ylabel(r'log(log(coeff))')
xlabel(r'$n$')
savefig("Ques3_4.jpg")
close()
```






Magnitudes of coefficients in log scale for cos(cos(x)) with a_n in red and b_n in blue
log(coeff)
n
10^{0}
10^{-3}
10^{-6}
10^{-9}
10^{-12}
10^{-15}
0
5
10
15
20
25


Magnitudes of coefficients in loglog scale for cos(cos(x)) with a_n in red and b_n in blue
log(log(coeff))
n
10^{0}
10^{-3}
10^{-6}
10^{-9}
10^{-12}
10^{-15}
10^{0}
10^{1}

3.4 A Least Squares Approach

We will also solve the same using least squares approach. We linearly choose 400 values of x in the range [0,2π). By using more values instead of 400, we can achieve better approximations. We try to solve Equation (1) By using regression on these 400 values

$$\begin{pmatrix} 1 & \cos(x_1) & \sin(x_1) & .... & \cos(25x_1) & \sin(25x_1) \\ 1 & \cos(x_2) & \sin(x_2) & .... & \cos(25x_2) & \sin(25x_2) \\ ... & ... & ... & .... & ... & ... \\ 1 & \cos(x_{400}) & \sin(x_{400}) & .... & \cos(25x_{400}) & \sin(25x_{400}) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ ... \\ a_{25} \\ b_{25} \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ ... \\ f(x_{400}) \end{pmatrix}$$

We create the matrix on the left side and call it A . We want to solve $Ac = b$ where c are the fourier coefficients.

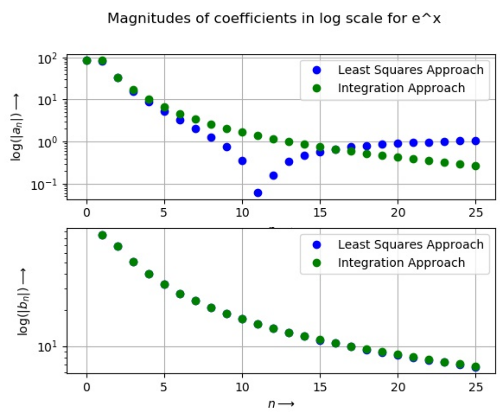
```
def matrix_method(A,f):                                    #Function which return all coefficients a
    coeff = scipy.linalg.lstsq(A,f(x))[0]                  #using lstsq for finding value of matrix
    coeff_a = []
    coeff_b = []

    coeff_a.append(coeff[0])
    coeff_b.append(0)
    for i in range(1,51,2):
        coeff_a.append(coeff[i])

    for i in range(2,51,2):
        coeff_b.append(coeff[i])

    return coeff, coeff_a, coeff_b


#Least Squares Approach
x = np.linspace(0,2*pi,401)
x=x[:-1]
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)
#Matrix A (51 x 400) has been defined

coeff1, coeff1_a , coeff1_b = matrix_method(A,f1)          #Calling above function
coeff2, coeff2_a , coeff2_b = matrix_method(A,f2)
```

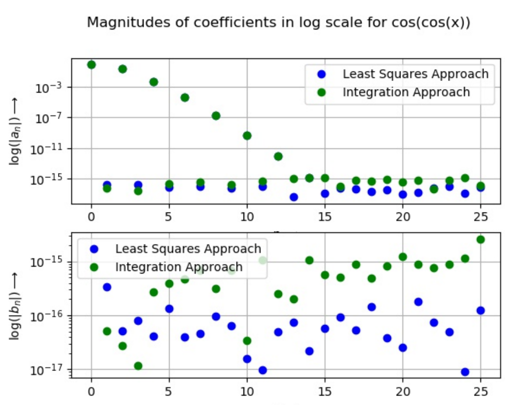
3.5 Visualizing output of the Least Squares Approach

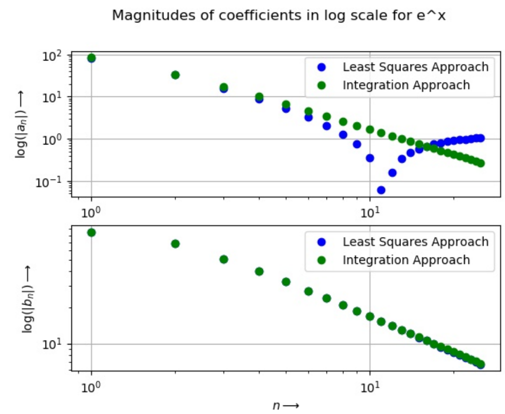
```
def plotting_comparing_coeff():
    figure(7)
    fig, axs = plt.subplots(2)                             #To divide axis into 2 to plot an on one an
```

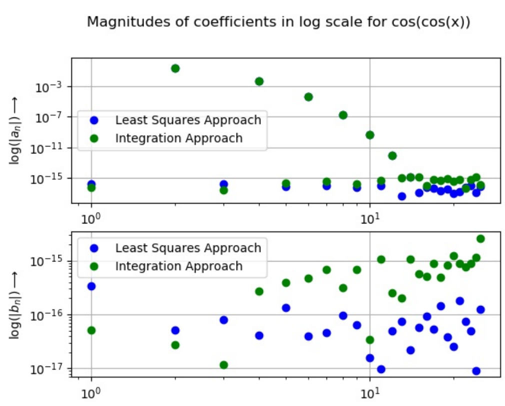
```
axs[0].semilogy(np.abs(coeff1_a), 'bo', label = 'Least Squares Approach')
axs[0].semilogy(np.abs(a_f1), 'go', label = 'Integration Approach')
axs[1].semilogy(np.abs(coeff1_b), 'bo', label = 'Least Squares Approach')
axs[1].semilogy(np.abs(b_f1), 'go', label = 'Integration Approach')
fig.suptitle(r'Magnitudes of coefficients in log scale for e^x')

axs[0].set(ylabel=r'log(|$a_n$|)$\longrightarrow$',xlabel=r'$n\longrightarrow$')
axs[1].set(ylabel=r'log(|$b_n$|)$\longrightarrow$',xlabel=r'$n\longrightarrow$')
axs[0].grid()
axs[1].grid()
axs[0].legend()
axs[1].legend()
savefig("Ques5_1.jpg")
close()


figure(8)
fig, axs = plt.subplots(2)

axs[0].semilogy(np.abs(coeff2_a), 'bo', label = 'Least Squares Approach')
axs[0].semilogy(np.abs(a_f2), 'go', label = 'Integration Approach')
axs[1].semilogy(np.abs(coeff2_b), 'bo', label = 'Least Squares Approach')
axs[1].semilogy(np.abs(b_f2), 'go', label = 'Integration Approach')
fig.suptitle(r'Magnitudes of coefficients in log scale for cos(cos(x))')

axs[0].set(ylabel=r'log(|$a_n$|)$\longrightarrow$',xlabel=r'$n\longrightarrow$')
axs[1].set(ylabel=r'log(|$b_n$|)$\longrightarrow$',xlabel=r'$n\longrightarrow$')
axs[0].grid()
axs[1].grid()
axs[0].legend()
axs[1].legend()
savefig("Ques5_2.jpg")
close()


figure(9)
fig, axs = plt.subplots(2)

axs[0].loglog(np.abs(coeff1_a), 'bo', label = 'Least Squares Approach')
axs[0].loglog(np.abs(a_f1), 'go', label = 'Integration Approach')
axs[1].loglog(np.abs(coeff1_b), 'bo', label = 'Least Squares Approach')
axs[1].loglog(np.abs(b_f1), 'go', label = 'Integration Approach')
fig.suptitle(r'Magnitudes of coefficients in log scale for e^x')

axs[0].set(ylabel=r'log(|$a_n$|)$\longrightarrow$',xlabel=r'$n\longrightarrow$')
axs[1].set(ylabel=r'log(|$b_n$|)$\longrightarrow$',xlabel=r'$n\longrightarrow$')
axs[0].grid()
axs[1].grid()
axs[0].legend()
axs[1].legend()
savefig("Ques5_3.jpg")
close()


figure(10)
fig, axs = plt.subplots(2)
```

```
axs[0].loglog(np.abs(coeff2_a), 'bo', label = 'Least Squares Approach')
axs[0].loglog(np.abs(a_f2), 'go', label = 'Integration Approach')
axs[1].loglog(np.abs(coeff2_b), 'bo', label = 'Least Squares Approach')
axs[1].loglog(np.abs(b_f2), 'go', label = 'Integration Approach')
fig.suptitle(r'Magnitudes of coefficients in log scale for cos(cos(x))')

axs[0].set(ylabel=r'log(|$a_n$|)$\longrightarrow$',xlabel=r'$n\longrightarrow$')
axs[1].set(ylabel=r'log(|$b_n$|)$\longrightarrow$',xlabel=r'$n\longrightarrow$')
axs[0].grid()
axs[1].grid()
axs[0].legend()
axs[1].legend()
savefig("Ques5_4.jpg")
close()
```






Magnitudes of coefficients in log scale for e^x
Least Squares Approach
Integration Approach
log(|a_n|) ⟶
log(|b_n|) ⟶
n ⟶


Magnitudes of coefficients in log scale for cos(cos(x))
Least Squares Approach
Integration Approach
log(|a_n|) ⟶
log(|b_n|) ⟶
n ⟶

3.6 Comparing Predictions

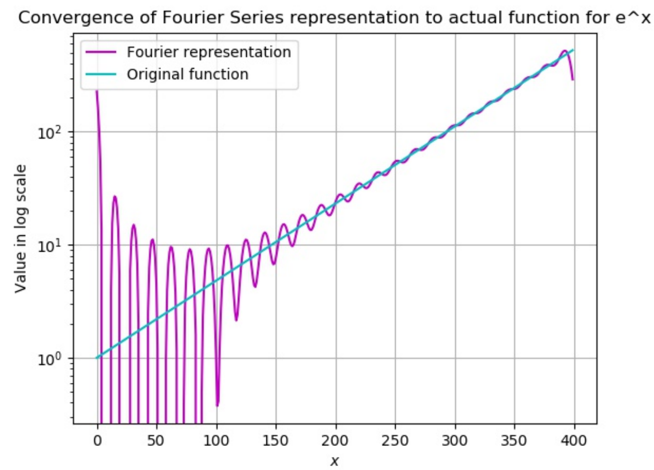
Maximum error for coefficients of e^x is 1.3327308703353395
Maximum error for coefficients of $cos(cos(x))$ is 2.7194358321802792e-15

Our Predictions for e^x are very poor compared to that of cos(cos(x)). This can be fixed by sampling at a larger number of points, but since sampling into higher number of points will increase the compilation time by a lot compared to the changes seen in the result its not ideal to do so for the sake of the assignment

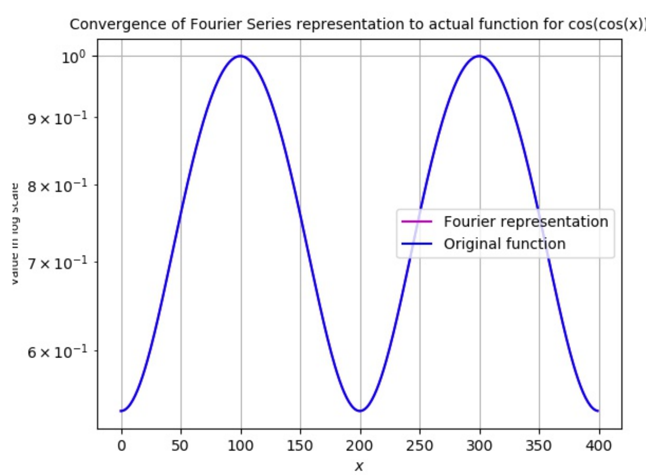
3.7 Plotting Results

```
def plotting_convergence():
    fourier_f1 = np.matmul(A, coeff1)
    fourier_f2 = np.matmul(A, coeff2)

    figure(11)
    semilogy(fourier_f1 , 'm', label = 'Fourier representation')
    semilogy(f1_periodic(x), 'c', label = 'Original function')

    grid(True)
    title(r'Convergence of Fourier Series representation to actual function for e^x')
    ylabel(r'Value in log scale')
    xlabel(r'$x$')
    legend()
    savefig("Ques7_1.jpg")
    close()


    figure(12)
    semilogy(fourier_f2 , 'm', label = 'Fourier representation')
    semilogy(f2(x), 'b', label = 'Original function')

    grid(True)
    title(r'Convergence of Fourier Series representation to actual function for cos(cos(x))',
    ylabel(r'Value in log scale', fontsize = 8)
    xlabel(r'$x$')
    legend()
    savefig("Ques7_2.jpg")
    close()
```

It should be noted that e^x is a non periodic function and Fourier series' exists only for periodic functions. Hence we have considered a variation of e^x with period 2π that has the actual value of e^x only in the range $[0,2\pi)$. Hence it is acceptable that there is a large discrepancy in the predicted value of e^x at these boundaries



Convergence of Fourier Series representation to actual function for e^x
Fourier representation
Original function
Value in log scale
x



Convergence of Fourier Series representation to actual function for cos(cos(x))
Fourier representation
Original function
x

4 Conclusion

We have examined the case of approximating functions using their Fourier coefficients upto a threshold. While doing so, we perform the same for two cases, one a continuous function, and the other a function with finite discontinuities.

The methods adopted in finding the respective Fourier coefficients have been direct evaluation of the Fourier series formula, as well an Least Square best fit. We notice close matching of the two methods in case of $\cos(\cos(x))$ while, there is a larger discrepancy in $\exp(x)$. This is because $\cos(\cos(x))$ is a continuous periodic function so the fourier series converges to true values at all points, whereas $\exp(x)$ is a function with discontinuities at both its ends of the period which causes the fourier series to converge to its mean at these points.