EE2703-Assignment4

EE19B094

March 2021

Abstract

The purpose of this assignment is to find fourier series coefficients of two curves 1) e^x and 2) $cos(cos(x))$. We will use two different methods to calculate the fourier coefficients and find the error between the methods.

1 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 ,

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

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

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

Now, as e^x doesn't have a period of 2π, We will change its definition in a way that will make it periodic i.e. we extend the function from [0,2π) to $(-\infty, \infty)$.

2 Generating and plotting raw functions

We can generate the functions using the *Numpy* library. The x values range from [-4π,2π]. Remember that we extended the e^x function as it was not periodic and thus it is calculated as $e^{x\%(2\pi)}$ thus making it periodic with period 2π.

```
generate_exp = lambda x: np.exp(x) #Function to generate aperodic
    e^x function
generate_coscos = lambda x: np.cos(np.cos(x)) #Function to generate
    cos(cos(x)) function
generate_perodic_exp = lambda x: np.exp(x%(2*math.pi)) #Function to
    generate periodic e^x function


#Function to plot both kind of exponential functions
def plot_raw_exponential():
```

```
    x = np.linspace(-2*math.pi,4*math.pi,300)
    figure(1)
    semilogy(x,generate_exp(x),'m',label=r'Aperiodic $e^x$ in log
        scale')
    semilogy(x,generate_perodic_exp(x),'g',label=r'Periodically
        extended $e^x$ in log scale')
    title("Q1 : Plotting original functions")
    xlabel(r'$x\longrightarrow$',fontsize=12)
    ylabel(r'$log(e^{x})\longrightarrow$',fontsize=12)
    grid()
    legend()
    savefig("Q1-1.png")
    close()

#Function to plot cos(cos(x)) function
def plot_raw_coscosx():
    x = np.linspace(-2*np.pi,4*np.pi,300)
    figure(2)
    plot(x,generate_coscos(x))
    title(r'y = cos(cos(x))')
    xlabel(r'$x\longrightarrow$', fontsize = 12)
    ylabel(r'$cos(cos(x))\longrightarrow$', fontsize = 12)
    grid()
    savefig("Q1-2.png")
    close()

plot_raw_exponential()  #Q1 plots function call
plot_raw_coscosx()
```

Graphs below show the aperiodic and periodic versions of e^x in log scale and $cos(cos(x))$ in linear scale.

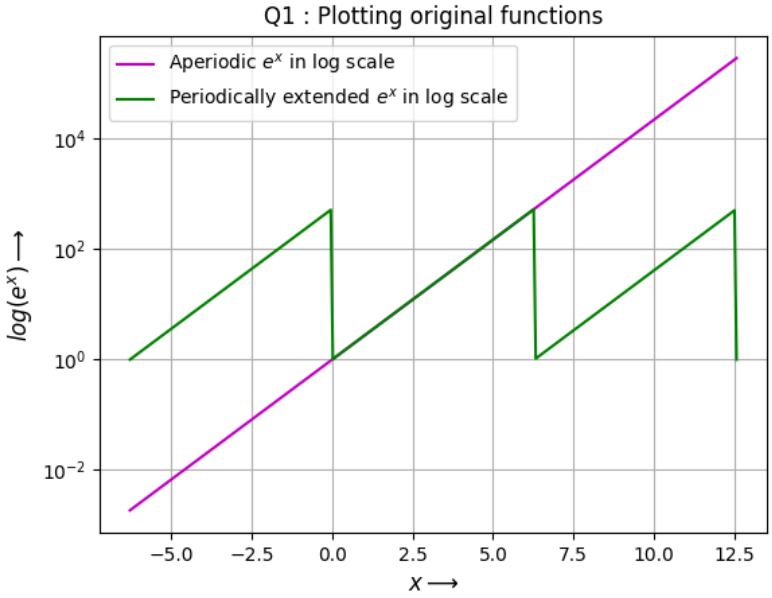

Figure 1: e^x

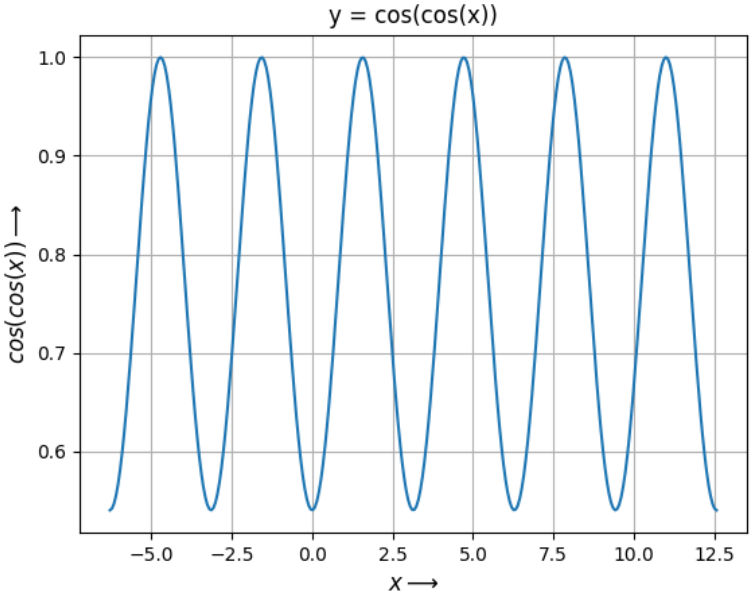

Figure 2: $cos(cos(x))$

3 Fourier Series Coefficients calculation

We create two new functions to be integrated, namely

```
u(x,k)=f(x)*cos(kx)
```

and

```
v(x,k)=f(x)*sin(kx)
```

We obtain the first 25 co-efficients for both e^x and $cos(cos(x))$ by using the equations given in the introduction, scipy's built in integrator and the quad function to pass extra arguments to the function being integrated. They are stored in a vector in the order

$$\begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ ... \\ a25 \\ b25 \end{pmatrix}$$

```
#Function to calculate 'k' Fourier Coefficients for given function f
def calculate_fourier_coefficients(f,k):
    coeffs=[]  #List storing combined coefficients
    a_n=[]     #List storing a_n coefficients
    b_n=[]     #List storing b_n coefficients

    u = lambda x, n: f(x)*math.cos(n*x) #Functions to be integrated
    v = lambda x, n: f(x)*math.sin(n*x)

    a_n.append((1/(2*math.pi))*integ.quad(u, 0, 2*math.pi,
        args=0)[0]) #a_0
    coeffs.append((1/(2*math.pi))*integ.quad(u, 0, 2*math.pi,
        args=0)[0])
    b_n.append(0)

    for n in range(1,k):
        a_n.append((1/math.pi)*integ.quad(u, 0, 2*math.pi,
            args=n)[0])
        coeffs.append((1/math.pi)*integ.quad(u, 0, 2*math.pi,
            args=n)[0])

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

    return coeffs,a_n,b_n
```

```
#Calculating fourier coefficients using integration method
coeffs_exp,a_n_exp,b_n_exp =
    calculate_fourier_coefficients(generate_exp,COEFF_COUNT+1)
coeffs_coscos,a_n_coscos,b_n_coscos =
    calculate_fourier_coefficients(generate_coscos,COEFF_COUNT+1)
```

4 Plotting the Fourier Coefficients

We plot all the fourier coefficients for both the functions on two scales i.e. loglog and semilog.

```
    #Function to plot the coefficients
def plot_coefficients():
    #Coefficients of e^x on a log scale
    figure(3)
    semilogy(np.abs(a_n_exp),'ro',label=r'$a_n$ of $e^x$')
    semilogy(np.abs(b_n_exp),'bo',label=r'$b_n$ of $e^x$')
    grid(True)
    title(r'Magnitudes of coefficients in log scale', fontsize = 10)
    xlabel(r'n$\longrightarrow$')
    ylabel(r'log(coeffs)$\longrightarrow$')
    legend()
    savefig("Q3-1.png")
    close()

    #Coefficients of cos(cos(x)) on a log scale
    figure(4)
    semilogy(np.abs(a_n_coscos),'ro',label=r'$a_n$ of
        $cos(cos(x))$')
    semilogy(np.abs(b_n_coscos),'bo',label=r'$b_n$ of
        $cos(cos(x))$')
    grid(True)
    title(r'Magnitudes of coefficients in log scale', fontsize = 10)
    xlabel(r'n$\longrightarrow$')
    ylabel(r'log(coeff)$\longrightarrow$')
    legend()
    savefig("Q3-2.png")
    close()

    #Coefficients of e^x on a loglog scale
    figure(5)
    loglog(np.abs(a_n_exp),'ro',label=r'$a_n$ of $e^x$')
    loglog(np.abs(b_n_exp),'bo',label=r'$b_n$ of $e^x$')
    grid(True)
    title(r'Magnitudes of coefficients in loglog scale', fontsize =
        10)
```

```
    xlabel(r'n$\longrightarrow$')
    ylabel(r'log(log(coeff))$\longrightarrow$')
    legend()
    savefig("Q3-3.png")
    close()

    #Coefficients of cos(cos(x)) on a loglog scale
    figure(6)
    loglog(np.abs(a_n_coscos),'ro',label=r'$a_n$ of $cos(cos(x))$')
    loglog(np.abs(b_n_coscos),'bo',label=r'$b_n$ of $cos(cos(x))$')
    grid(True)
    title(r'Magnitudes of coefficients in loglog scale', fontsize =
        10)
    xlabel(r'n$\longrightarrow$')
    ylabel(r'log(log(coeff))$\longrightarrow$')
    legend()
    savefig("Q3-4.png")
    close()

plot_coefficients()
```

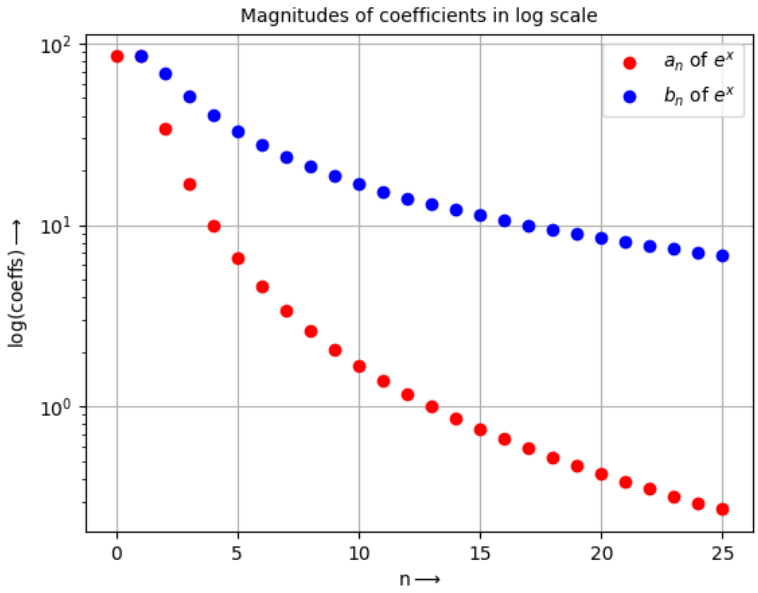

Figure 3: e^x *in log scale*

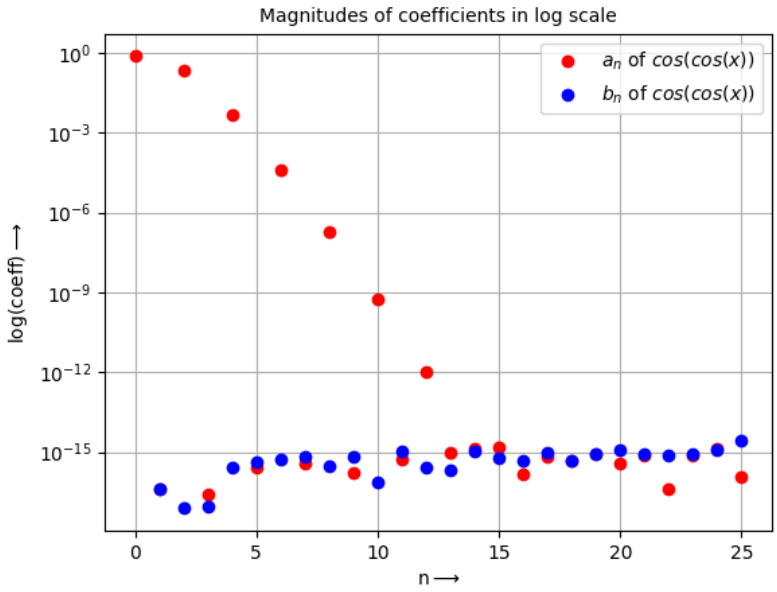

Figure 4: *cos(cos(x)) in log scale*

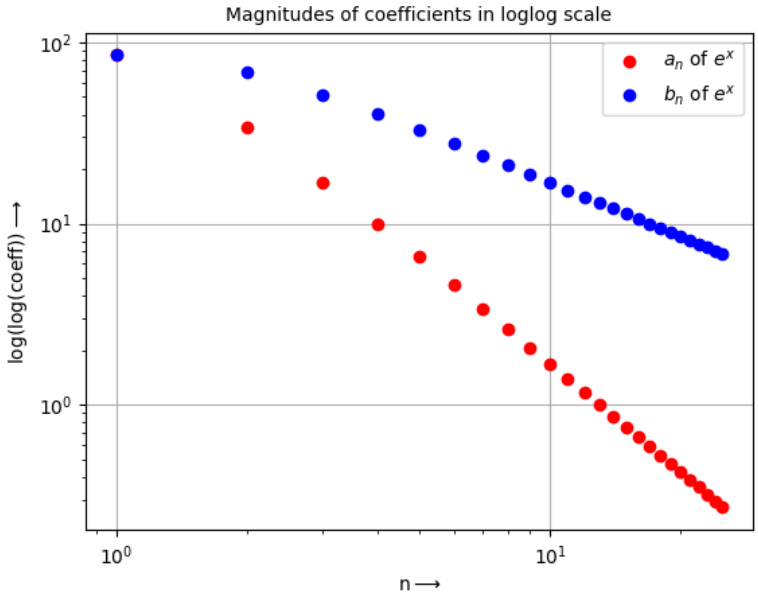

Figure 5: e^x *in loglog scale*

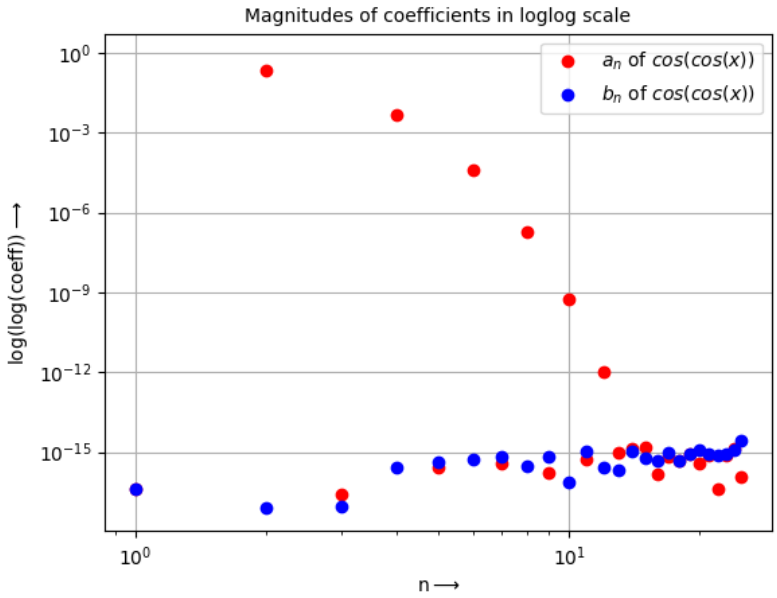

Figure 6: *cos(cos(x)) in loglog scale*

5 Observations

Q : The b_n coefficients for $cos(cos(x))$ are nearly approaching zero in Figure 4. What is the reason ?
A : As $cos(cos(x))$ is an even function, its fourier series will be a pure cosine series with no sine terms i.e. $b_n = 0$.

Q : The coefficients for $cos(cos(x))$ decrease rapidly as compared to e^x for higher frequencies. Why ?
A : $cos(cos(x))$ is a sinusoid type of function and thus most of the contribution is from the lower frequencies. And therefore the values of coefficients decreases rapidly for higher frequencies. This is evident from Figure 4. Whereas e^x has contributions from many harmonics due to the discontinuities in our periodic definition of e^x and thus it's coefficients decrease less rapidly. This can be seen in Figure 3.

Q : Why does the loglog plot of Figure 5 look linear while the log plot of Figure 4 look linear ?
A : The coefficients of e^x decay with n as

$$a_n \propto 1/n^2$$
$$b_n \propto 1/n$$

hence, taking log, $\log a_n$ and $\log b_n$ are almost proportional to log(n). So the loglog scale features linear behaviour. For cos(cos(x)) the FSC's decay approximately exponentially with n i.e

$$a_n, b_n \propto e^{-n}$$

, and hence the log plot looks linear.

6 Least Square Approach

We will now try and predict the values of fourier coefficients using the least square approach. We will create two matrices A, x satisfying the equation

$$A \begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ ... \\ a25 \\ b25 \end{pmatrix} = x$$

$$\begin{pmatrix} 1 & cos(x_1) & sin(x_1) & ... & cos(25x_1) & sin(25x_1) \\ 1 & cos(x_2) & sin(x_2) & ... & cos(25x_2) & sin(25x_2) \\ ... & ... & ... & ... & ... & ... \\ 1 & cos(x_{400}) & sin(x_{400}) & ... & cos(25x_{400}) & sin(25x_{400}) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ ... \\ a25 \\ b25 \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ ... \\ f(x_{400}) \end{pmatrix}$$

Now, using *scipy.linalg.lstsq()* function, we find the coefficient matrix.

```
#Defining A,x arrays for lstsq method
x = np.linspace(0,2*pi,401)
x=x[:-1]
A = np.zeros((400,2*COEFF_COUNT+1))
A[:,0]=1
for k in range(1,COEFF_COUNT+1):
    A[:,2*k-1] = np.cos(k*x)
    A[:,2*k] = np.sin(k*x)


#Calculating coefficients for e^x using lstsq method
coeffs_exp_lstsq = scipy.linalg.lstsq(A,generate_exp(x))[0] #List
    storing combined coefficients
a_n_exp_lstsq = [] #List storing a_n coefficients
b_n_exp_lstsq = [] #List storing b_n coefficients
```

```
#Deriving a_n and b_n from combined coefficients
a_n_exp_lstsq.append(coeffs_exp_lstsq[0])
b_n_exp_lstsq.append(0)
for i in range(1,2*COEFF_COUNT+1,2):
    a_n_exp_lstsq.append(coeffs_exp_lstsq[i])
for i in range(2,2*COEFF_COUNT+1,2):
    b_n_exp_lstsq.append(coeffs_exp_lstsq[i])


#Calculating coefficients for cos(cos(x)) using lstsq method
coeffs_coscos_lstsq = scipy.linalg.lstsq(A,generate_coscos(x))[0]
    #List storing combined coefficients
a_n_coscos_lstsq = [] #List storing a_n coefficients
b_n_coscos_lstsq = [] #List storing b_n coefficients

#Deriving a_n and b_n from combined coefficients
a_n_coscos_lstsq.append(coeffs_coscos_lstsq[0])
b_n_coscos_lstsq.append(0)
for i in range(1,2*COEFF_COUNT+1,2):
    a_n_coscos_lstsq.append(coeffs_coscos_lstsq[i])
for i in range(2,2*COEFF_COUNT+1,2):
    b_n_coscos_lstsq.append(coeffs_coscos_lstsq[i])
```

The below shown graphs show the difference in values of coefficients calculated using the two methods.

```
    #Plot coefficients obtained from integration and lstsq methods
def plot_comparing_coeff():
    #Coefficients of e^x on a log scale
    figure(7)
    fig, axs = plt.subplots(2)
    axs[0].semilogy(np.abs(a_n_exp_lstsq), 'bo', label = 'Least
        Squares Approach')
    axs[0].semilogy(np.abs(a_n_exp), 'go', label = 'Integration
        Approach')
    axs[1].semilogy(np.abs(b_n_exp_lstsq), 'bo', label = 'Least
        Squares Approach')
    axs[1].semilogy(np.abs(b_n_exp), '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("Q5-1.png")
    close()
```

```
#Coefficients of cos(cos(x)) on a log scale
figure(8)
fig, axs = plt.subplots(2)
axs[0].semilogy(np.abs(a_n_coscos_lstsq), 'bo', label = 'Least
    Squares Approach')
axs[0].semilogy(np.abs(a_n_coscos), 'go', label = 'Integration
    Approach')
axs[1].semilogy(np.abs(b_n_coscos_lstsq), 'bo', label = 'Least
    Squares Approach')
axs[1].semilogy(np.abs(b_n_coscos), '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("Q5-2.png")
close()

#Coefficients of e^x on a loglog scale
figure(9)
fig, axs = plt.subplots(2)
axs[0].loglog(np.abs(a_n_exp_lstsq), 'bo', label = 'Least
    Squares Approach')
axs[0].loglog(np.abs(a_n_exp), 'go', label = 'Integration
    Approach')
axs[1].loglog(np.abs(b_n_exp_lstsq), 'bo', label = 'Least
    Squares Approach')
axs[1].loglog(np.abs(b_n_exp), '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("Q5-3.png")
close()

#Coefficients of cos(cos(x)) on a loglog scale
figure(10)
fig, axs = plt.subplots(2)
axs[0].loglog(np.abs(a_n_coscos_lstsq), 'bo', label = 'Least
    Squares Approach')
axs[0].loglog(np.abs(a_n_coscos), 'go', label = 'Integration
```

```
        Approach')
    axs[1].loglog(np.abs(b_n_coscos_lstsq), 'bo', label = 'Least
        Squares Approach')
    axs[1].loglog(np.abs(b_n_coscos), '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("Q5-4.png")
    close()

plot_comparing_coeff() #Q5 plots function call
```

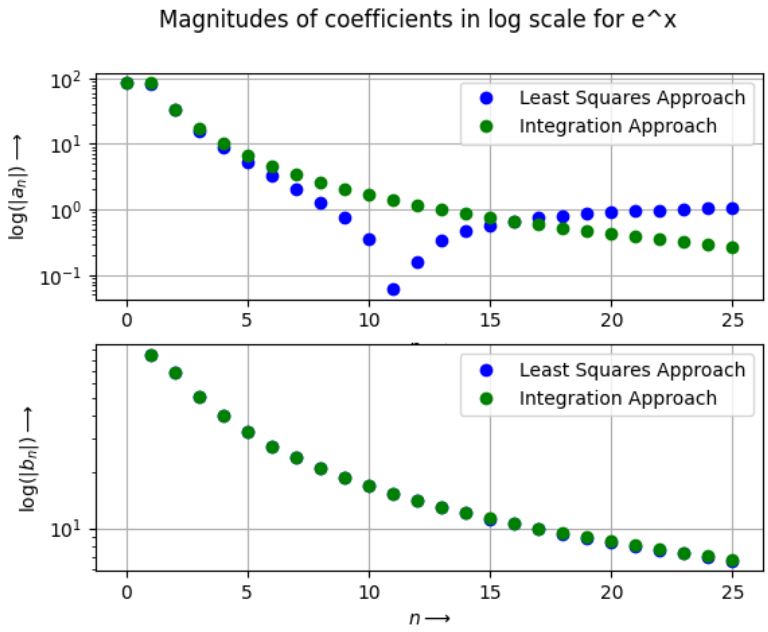

Figure 7: e^x *in log scale*

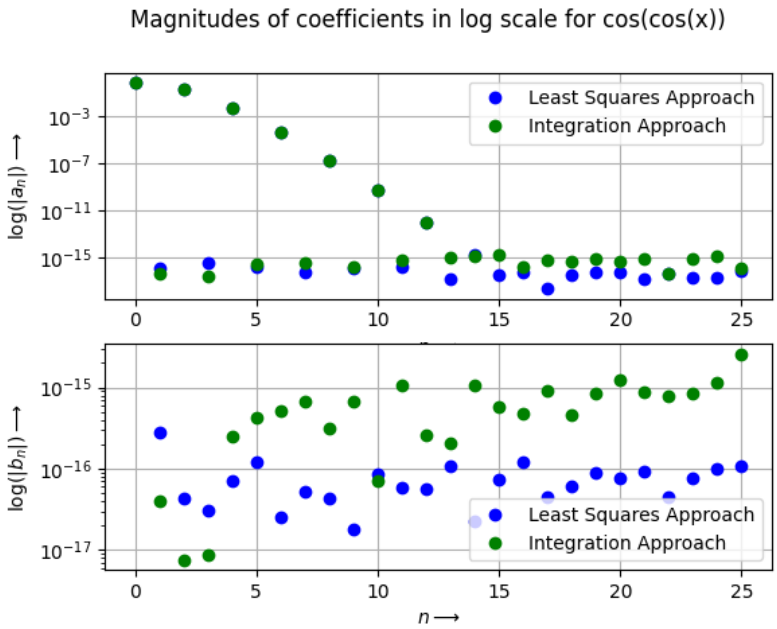

Figure 8: *cos(cos(x)) in log scale*

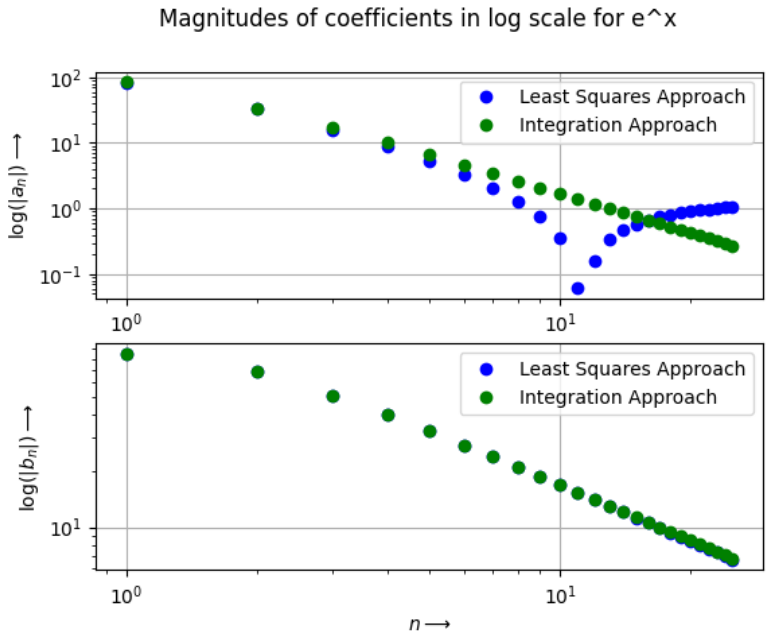

Figure 9: e^x *in loglog scale*

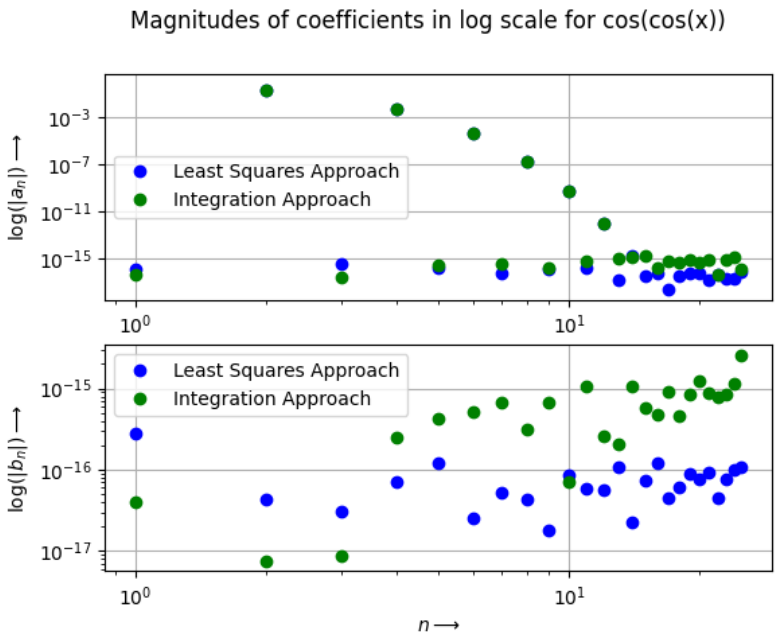

Figure 10: *cos(cos(x)) in loglog scale*

7 Difference between the two methods

We can clearly see that the difference in coefficients between the two methods is much more in e^x than in $cos(cos(x))$. This is due to the fact that our definition of e^x is discontinious and thus this error is much more accountable in integration method than in least square method.

If we find the maximum error between the two methods, we get

```
#Calculating max error in the two methods
error_exp = np.abs(coeffs_exp - coeffs_exp_lstsq)
error_coscos = np.abs(coeffs_coscos - coeffs_coscos_lstsq)
max_error_exp = np.max(error_exp)
max_error_coscos = np.max(error_coscos)

print("Maximum error between the two methods for e^x is
    ",max_error_exp)
print("Maximum error between the two methods for cos(cos(x)) is
    ",max_error_coscos)
```

Maximum error between the two methods for e^x is 1.332730870335368
Maximum error between the two methods for cos(cos(x)) is 2.68042108650032e-15

8 Convergence to actual function

Using the predicted values of the fourier coefficients, we can calculate the functional values for both e^x and $cos(cos(x))$ by multiplying our coefficient vector with the earlier defined matrix A.

```
#Plotting graph obtained from fourier coefficients alongside the
    original graph
def plotting_convergence():
    fourier_exp = np.matmul(A,coeffs_exp_lstsq)
    fourier_coscos = np.matmul(A,coeffs_coscos_lstsq)

    #e^x
    figure(11)
    semilogy(fourier_exp, 'm', label = 'Fourier representation')
    semilogy(generate_perodic_exp(x), 'c', label = 'Original
        function')
    grid(True)
    title(r'Convergence of Fourier Series representation to actual
        function for e^x')
    xlabel(r'$x\longrightarrow$')
    ylabel(r'Value in log scale$\longrightarrow$')
    legend()
    savefig("Q7-1.png")
    close()

    #cos(cos(x))
    figure(12)
    semilogy(fourier_coscos, 'm', label = 'Fourier representation')
    semilogy(generate_coscos(x), 'b', label = 'Original function')
    grid(True)
    title(r'Convergence of Fourier Series representation to actual
        function for cos(cos(x))', fontsize = 10)
    xlabel(r'$x\longrightarrow$')
    ylabel(r'Value in log scale$\longrightarrow$', fontsize = 8)
    legend()
    savefig("Q7-2.png")
    close()

plotting_convergence()    #Q7 plots function call
```

We can clearly see from Figure 12 as well that the fourier representation of $cos(cos(x))$ is perfectly overlapping with the actual curve. However, from Figure 11, we see that there is a huge deviation in the case of e^x. The main cause of this is that, to perfectly account for the discontinuities in e^x we need infinite harmonics whereas we are only considering 51 coefficients. However this is not the case for $cos(cos(x))$, which because it is sinusodial in nature, is very nicely fit by even just 51 fourier coefficients.

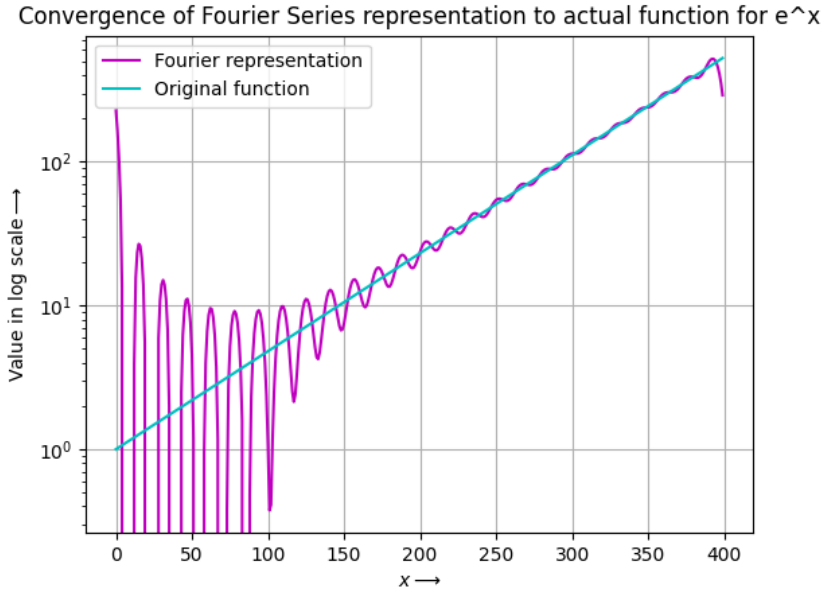

Figure 11: e^x

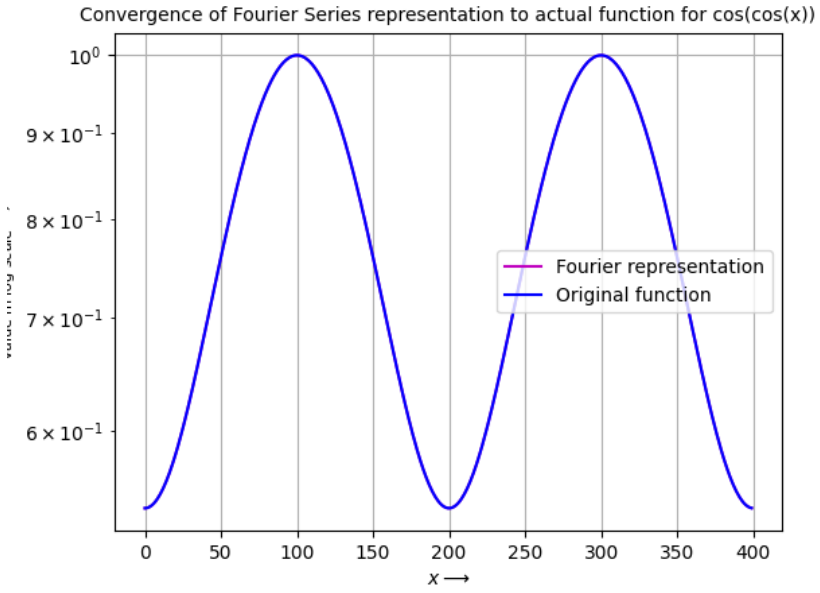

Figure 12: $cos(cos(x))$

9 Conclusion

In this assignment, we learnt how to calculate the fourier series coefficients for a perodic function by two methods, namely integration and least square.

Also, we saw that as our e^x was discontinious, there was significant error between the curve we predicted and the actual curve. Whereas that was not the case for $cos(cos(x))$.