EE2703 : Applied Programming Lab
Assignment 4

Rakshit Pandey
EE20B106

February 26, 2022

1. AIM OF THE ASSIGNMENT

In week 4 assignment we were supposed to the following things:

- Generate the semilog plot of e^x and plot of cos(cos(x))in the interval [-2π, 4π].
- Obtain the first 51 fourier coefficients by direct integration method, for both the functions and plot them against n (x-axis).
- Use the least square approach for finding the approximated fourier coefficients for both the functions.
- Compare the results obtained by direct integration approach vs the least square method.
- Compute the functional values with the help of best fit fourier coefficients and plot them against the True value curve.
- Do the rigorous theoretical analysis of the obtained fourier coefficients.

2. INTRODUCTION TO THE ASSISGNMENT

In this week's assignment we will generate the first 51 fourier coefficients for our 2 functions, by direct integration method, then will use the least square approach to get the best fit values for the fourier coeffients of the two functions. All this will be done in python .

All the corresponding analysis will be visualized with the help of graphs and some theoretical analysis will be done for better understanding of our observations.

We will fit two functions e^x and cos(cos(x)) over the interval [0,2π] using the fourier series

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

Where;

$$a_0 = \frac{1}{2\pi}\int_0^{2\pi} f(x)dx$$
$$a_n = \frac{1}{\pi}\int_0^{2\pi} f(x)cos(nx)dx$$
$$b_n = \frac{1}{\pi}\int_0^{2\pi} f(x)sin(nx)dx$$

3. ASSIGNMENT QUESTIONS

1.Importing important libraries

In order to this assignment we have to import some important python libraries, which will be useful when we work with matrices, or we want to perform definite integration, or for plotting curves.

Code

```
        import numpy as np
        from matplotlib.pyplot import *
        import matplotlib.pyplot as plt
        from scipy.linalg import lstsq
        from scipy.integrate import quad
        from math import sin,cos,pi,exp

```

2. Generating curves for e^x and cos(cos(x))

We will plot the two curves using python library matplotlib.pyplot, such that the two functions

$$f_1(x) = e^x$$
$$f_2(x) = cos(cos(x))$$

will be plotted in the interval [-2π,4π], where e^x is plotted on the semilog scale and cos(cos(x)) is plotted on linear scale. This will be done in python as follows:

Code

```
        def f1(x):
          return exp(x)

        def f2(x):
          return cos(cos(x))

        t = np.arange(-2.0*pi,4.0*pi,0.01)

        y1 = []
        y2 = []

        for i in range(len(t)):
```

```
        y1.append(f1(t[i]))
        y2.append(f2(t[i]))

```

The follwing two curves will be obtained as a result

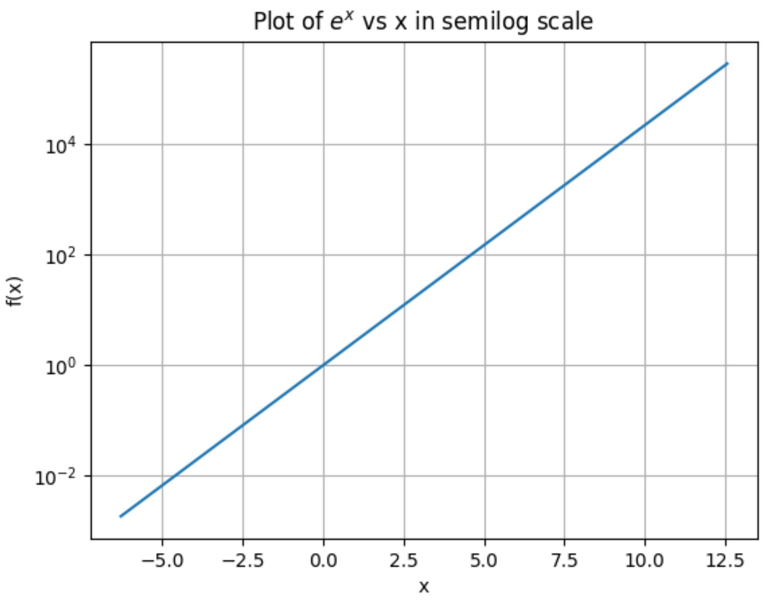
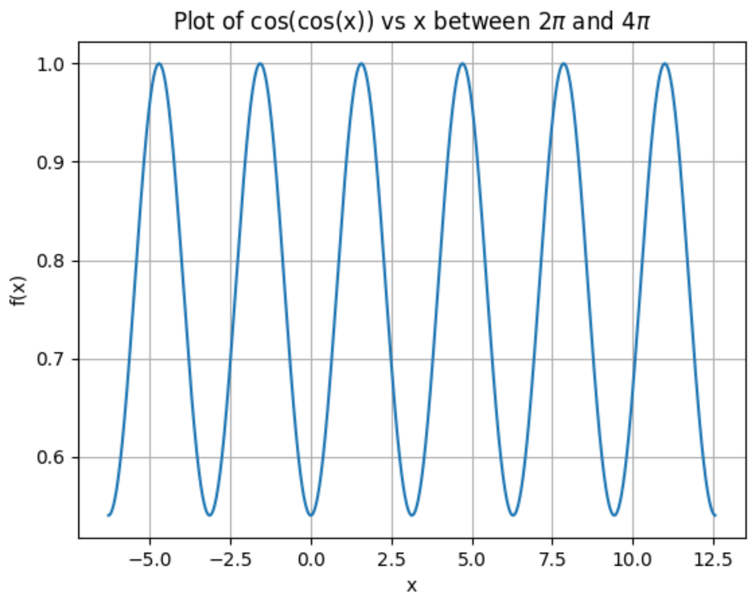


Figure 1: Figure 1



It is clear from the two figures, that e^x is non-periodic, whereas cos(cos(x)) is a periodic function with a fundamental period of π.

$$cos(cos(\pi + x)) = cos(-cos(x)) = cos(cos(x))$$

It is expected that a very precise curve should be generated using fourier series for cos(cos(x)) while the deviation from the true figure should be quite considerable in case of e^x.

3. Obtaining the first 51 fourier coefficients for both functions

In order to obtain first 51 fourier coefficients we first create two functions for $f_1(x)$ and $f_2(x)$ each namely, $u_1(x)$ and $v_1(x)$ for first function and $u_2(x)$ and $v_2(x)$ for second function. This is implemented in python as follows:

Code

```
    def u1(t,k):
      return f1(t)*cos(k*t)
    def u2(t,k):
      return f2(t)*cos(k*t)

    def v1(t,k):
      return f1(t)*sin(k*t)
    def v2(t,k):
      return f2(t)*sin(k*t)


```

In order to perform definite integral from 0 to 2π we have imported quad function from scipy.integrate python library, the integral will be perfomed and the 51 coefficients will be generated for each of the two functions. These coefficients will then be stored in the two arrays, this is will be implemented in python as follows:

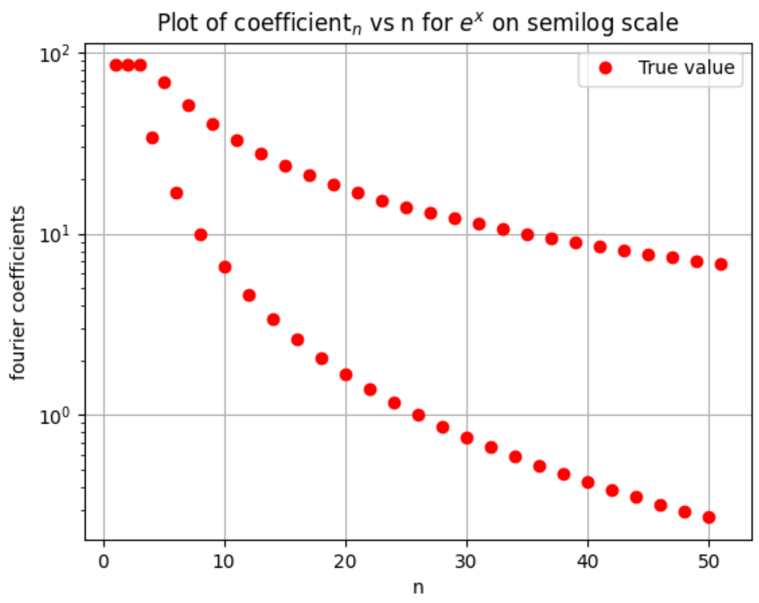
Code

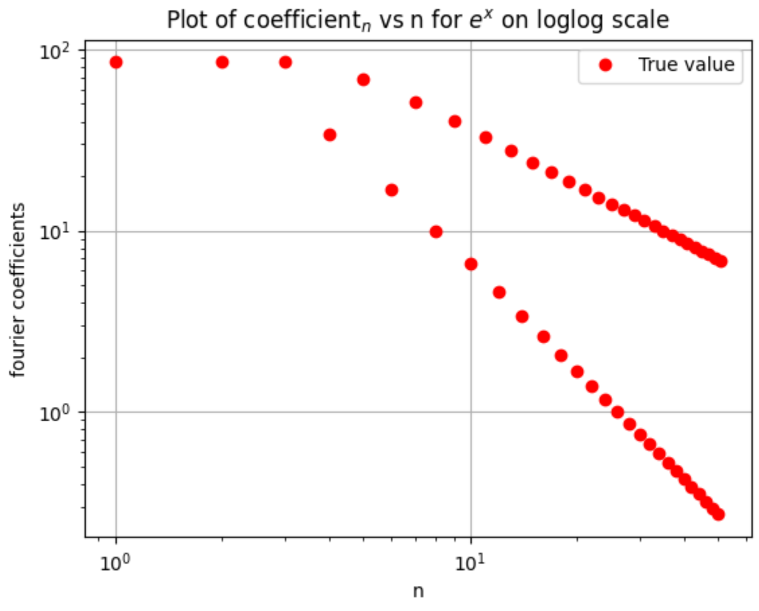
```
    a0_f1 = (1/(2.0*pi))*quad(f1,0,2.0*pi)[0]
    a0_f2 = (1/(2.0*pi))*quad(f2,0,2.0*pi)[0]

    c1_f1 = []
    c2_f2 = []

    c1_f1.append(a0_f1)
    c2_f2.append(a0_f2)

    for k in range(1,26):
      c1_f1.append((1/pi)*quad(u1,0,2.0*pi,args=(k))[0])
      c1_f1.append((1/pi)*quad(v1,0,2.0*pi,args=(k))[0])

      c2_f2.append((1/pi)*quad(u2,0,2.0*pi,args=(k))[0])
      c2_f2.append((1/pi)*quad(v2,0,2.0*pi,args=(k))[0])

```

4.Plotting the obtained fourier coefficients

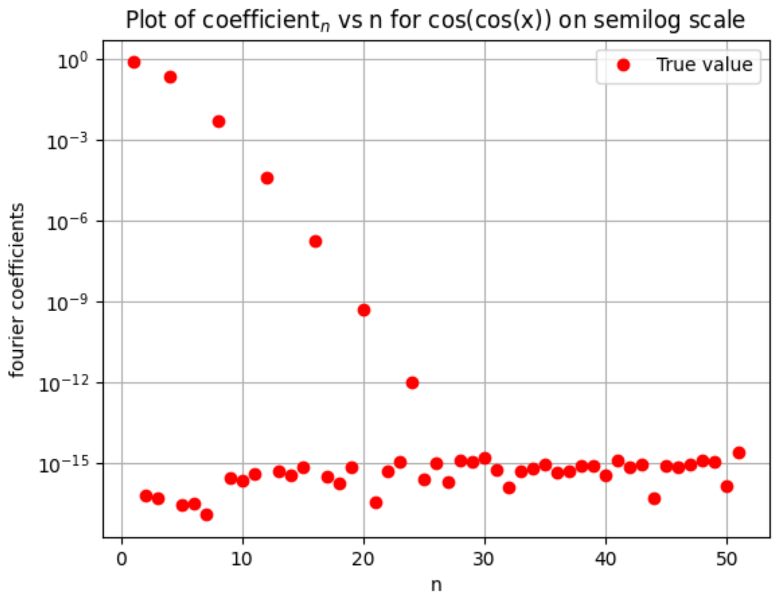
Now, we will generate the plots for the obtained 51 fourier coefficients. Two sets of plots will be generated, one for ^{x} and other for cos(cos(x)), Each set will have a semilog plot and a loglog plot for obtained coefficients vs n as n varies from 0 to 50.

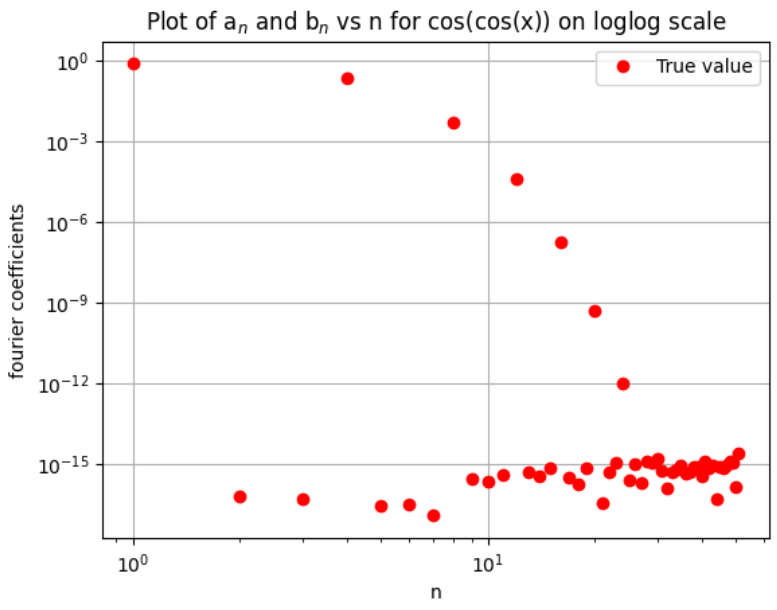
For e^x




Plot of coefficient$_n$ vs n for e^x on loglog scale
True value
fourier coefficients
10^2
10^1
10^0
n

For cos(cos(x))


Plot of coefficient$_n$ vs n for cos(cos(x)) on semilog scale
True value
fourier coefficients
10^0
10^{-3}
10^{-6}
10^{-9}
10^{-12}
10^{-15}
0
10
20
30
40
50
n



It is observed that the b_n coefficients in second case are nearly zero , this is because cos(cos(x)) is an even function

$$cos(cos(-x)) = cos(cos(x))$$

, and for even functions the b_n coefficients are always zero, but it is not coming out to be exactly zero because of π not being absolutely precise in our computation.

Also it is observed that in first case the coefficients do not decay as rapidly as second case. In second case the frequency of consideration is $\frac{1}{\pi}$ while the exponential function has wide range frequencies in it's fourier series expansion, hence the coefficients decay at a much faster rate in second case when compared to the first case.

We also observe that the loglog plot in coefficients of first function vs n looks almost linear because if we calculate and find the fourier series coefficients for e^x it turns out that

$$|a_n|, |b_n| \propto \frac{n}{1+n^2}$$

Hence in the loglog scale

$$log(|a_n|), log(|b_n|) \propto log(\frac{n}{1+n^2}) \approx -log(n)$$

Since, the plot is in loglog scale so the curve is almost linear even for little large n.
Also it turns out that semilog plot in case of second function coefficients looks linear, which can be similarly explained as above.

5.Least square approach

In the least square approach we a vector x is defined which is going from 0 to 2π in 400 steps, now we create a column vector b which contains the functional values at these points of x, the function can then be approximated as

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

The problem turns into a simple matrix equation of the form

$$Ac = b$$

which is

$$A = \begin{pmatrix} 1 & cosx_1 & sinx_1 \cdots & cos25x_1 & sin25x_1 \\ 1 & cosx_2 & sinx_2 \cdots & cos25x_2 & sin25x_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & cosx_{400} & sinx_{400} \cdots & cos25x_{400} & sin25x_{400} \end{pmatrix}$$

$$b = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \cdots \\ f(x_{400}) \end{pmatrix}$$

$$c = \begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ \cdots \\ a_{25} \\ b_{25} \end{pmatrix}$$

Here c matrix is the column vector of 51 coefficients which is obtained as a result of direct integration

The A matrix is constructed in python as follows:

Code

```python
        A = np.zeros((400,51))
        A[:,0] = 1.0

        for k in range(1,26):
          row = 0
          while row<400:
            A[row][2*k-1] += cos(k*x[row])
            A[row][2*k] += sin(k*x[row])
            row += 1

```

the b matrix is constructed as shown:

Code

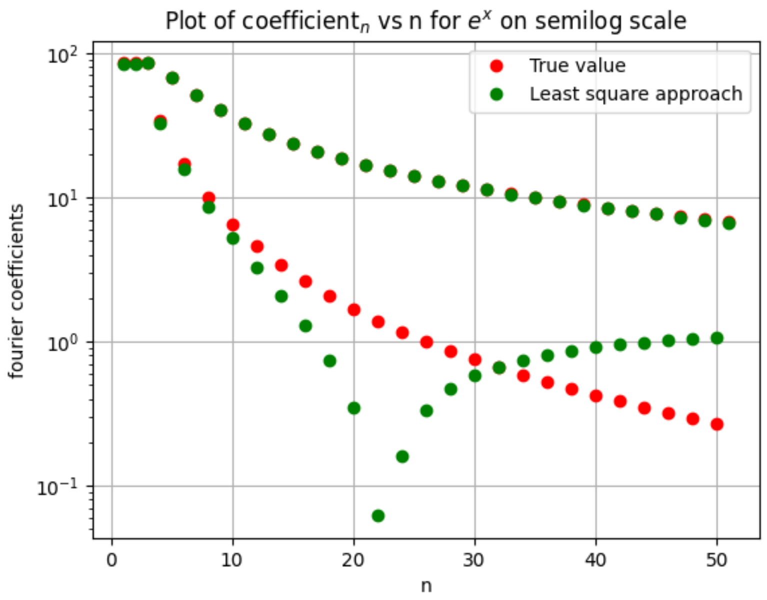
```python
        b1 = np.zeros((400,1))
        b2 = np.zeros((400,1))

        for i in range(400):
          b1[i][0] += exp(x[i])
          b2[i][0] += cos(cos((x[i])))

```

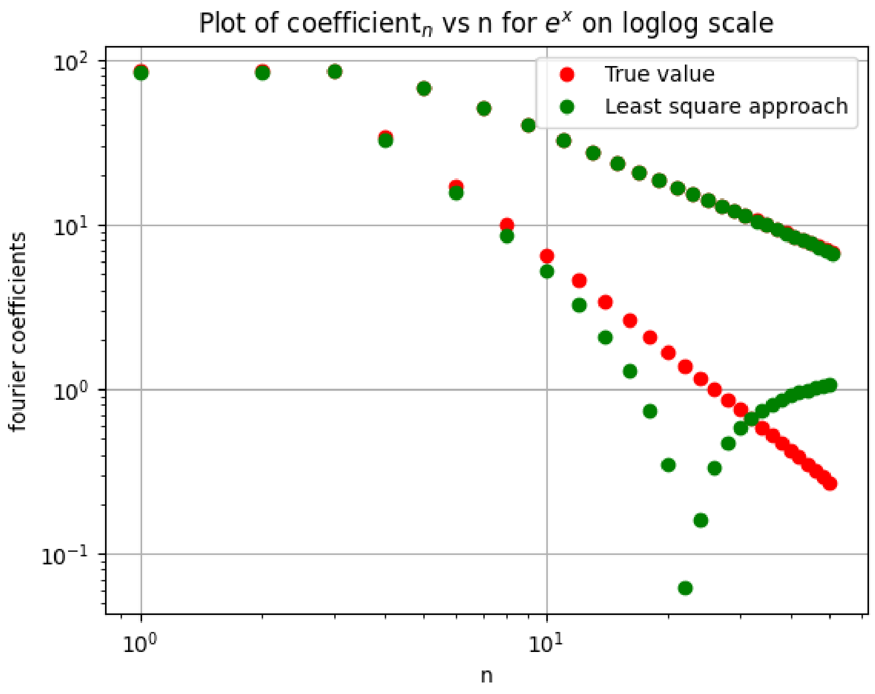
The process of generating best fit c matrix involves the use of least square approach. We have imported lstsq function from scipy.linalg. The When we use lstsq function on matrices A and b, the resulting matrix is the matrix with the best fit value of coefficients.This is done in python as follows:

Code

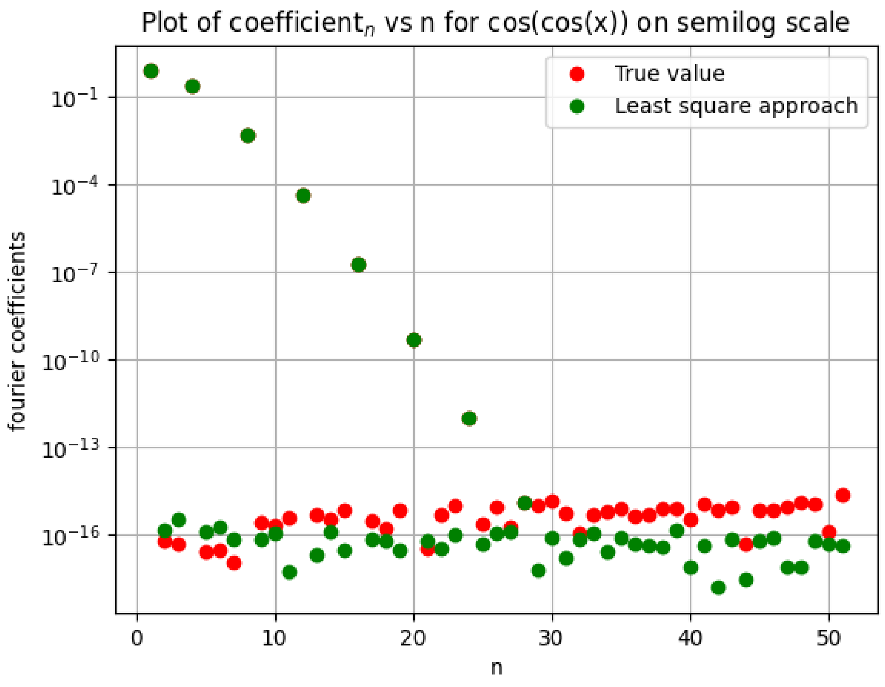
```python
        from scipy.linalg import lstsq
        c1 = lstsq(A,b1)[0]
        c2 = lstsq(A,b2)[0]

```

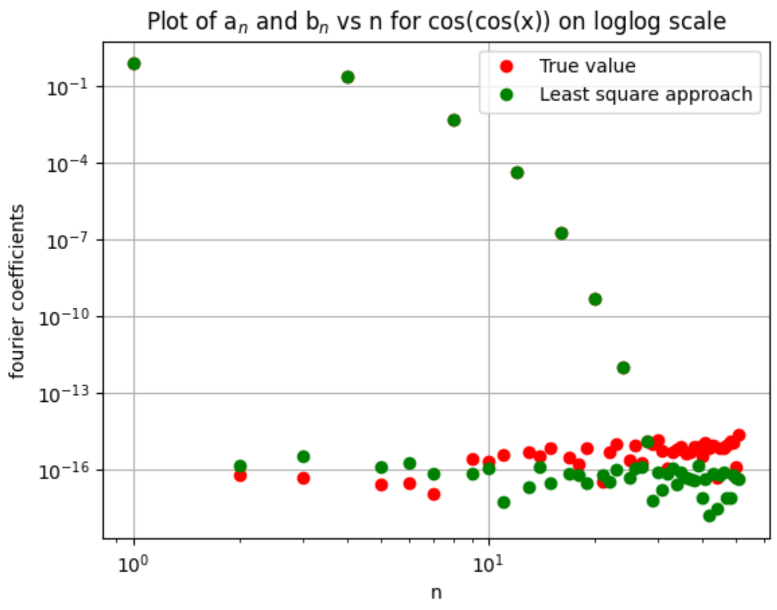
where c_1 is the best fit fourier coefficients matrix for $f_1(x)$, and c_2 is the best fit fourier coefficients matrix for $f_2(x)$
Having obtained the best fit matrices c, we now plot the newly generated fourier coefficients along with the fourier coefficients that were obtained as a result of direct integration method. For e^x




Plot of coefficient$_n$ vs n for e^x on loglog scale
True value
Least square approach
fourier coefficients
n

For cos(cos(x))


Plot of coefficient$_n$ vs n for cos(cos(x)) on semilog scale
True value
Least square approach
fourier coefficients
n



6. Least square approach vs Direct integration method

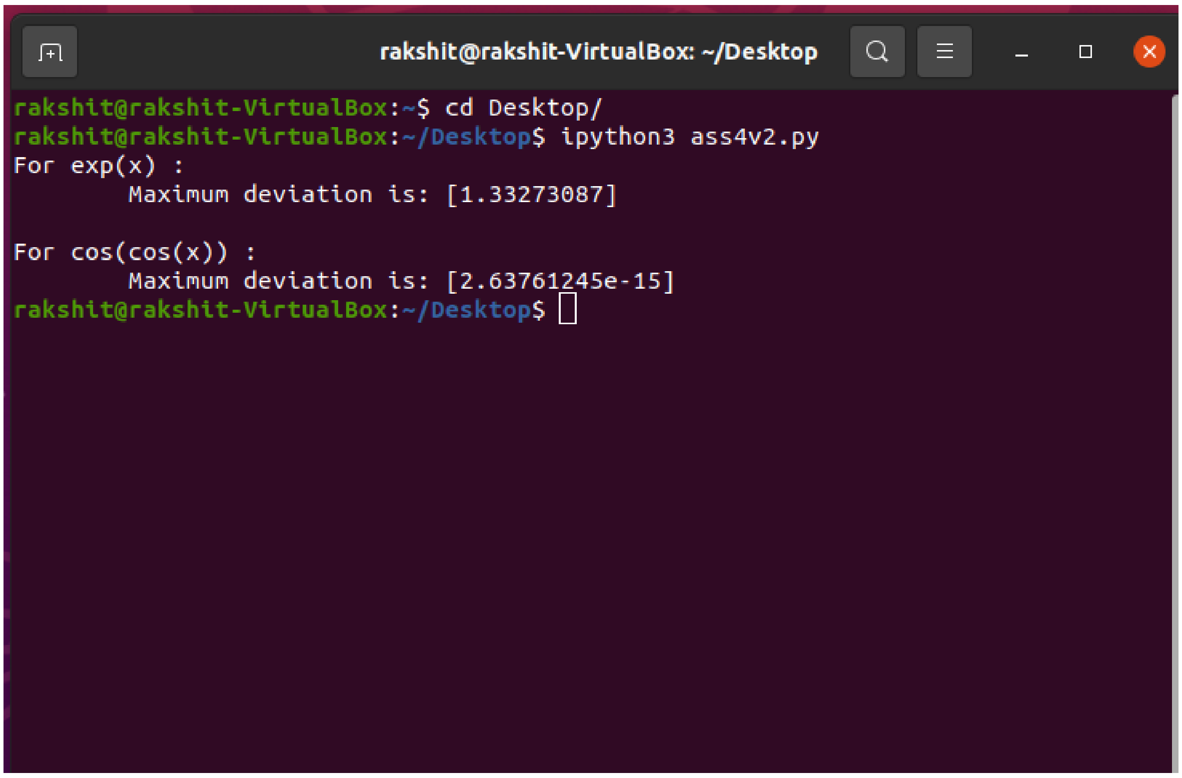
As we can see from the above plots there is a significant deviation in case of e^x while the deviation in case of cos(cos(x)) is very less, infact negligible at times. We will further elaborate this point and solidify our observation with statistcal approach.

We will calculate the absolute deviation value of fourier coefficients obtained by direct integration method vs the fourier coefficients obtained by least square approach. We will then take the maxinmum value of this absoulte deviation to understand what is the maximum extent to which the two results differ.This is executed with the help of python as follows:

Code

```
f1_dev = []
f2_dev = []

for i in range(51):
  f1_dev.append(abs(c1[i] - c1_f1[i]))
  f2_dev.append(abs(c2[i] - c2_f2[i]))

print("For exp(x) :")
print(" Maximum deviation is: {}".format(max(f1_dev)))
print(" ")
print("For cos(cos(x)) :")
print(" Maximum deviation is: {}".format(max(f2_dev)))

```

The following result is obtained when we run the code:



```
rakshit@rakshit-VirtualBox:~$ cd Desktop/
rakshit@rakshit-VirtualBox:~/Desktop$ ipython3 ass4v2.py
For exp(x) :
        Maximum deviation is: [1.33273087]

For cos(cos(x)) :
        Maximum deviation is: [2.63761245e-15]
rakshit@rakshit-VirtualBox:~/Desktop$ 
```

Clearly the answers obtained from direct integration method and least square method do not agree, and they shouldn't agree, but as we can see the maximum deviation in the values obtained for cos(cos(x)) is of the order of magnitude of 10^{-15}, which can be considered negligible, whereas the deviation is quite significant in case of e^x.

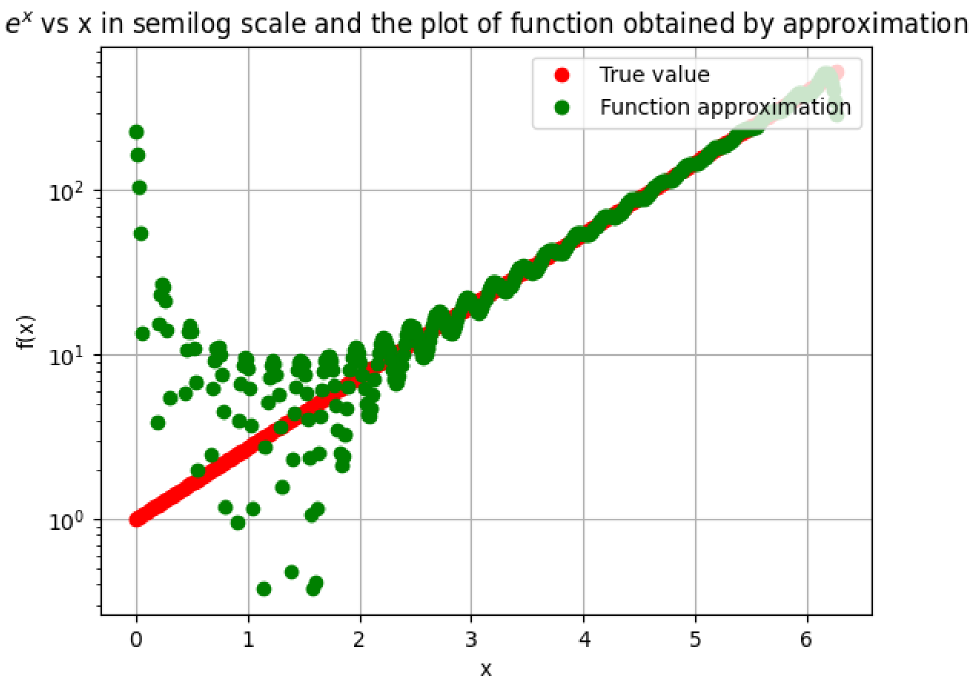
7. Functional values for best fit matrix

The column vector for functional values at points x_i can be obtained as the matrix product of Matrix A and column vector c which is obtained as a result of least square method. The obtained functional values are then plotted alongside the true functional values, and the accuracy of the method is visulized by analyzing the obtained curves.

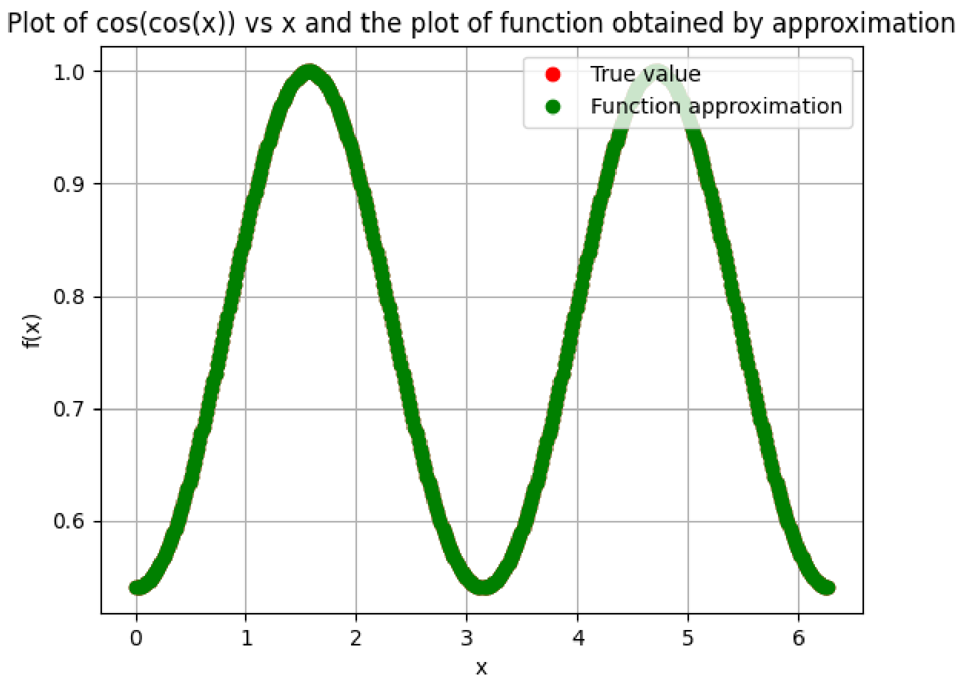
Code

```
    f1_approx = np.dot(A,c1)
    f2_approx = np.dot(A,c2)

    y1_new = []
    y2_new = []

    for i in range(len(x)):
      y1_new.append(f1(x[i]))
      y2_new.append(f2(x[i]))

    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.title("Plot of $e^{x}$ vs x in semilog scale and the plot of function
   obtained by approximation")
    plt.semilogy(x,y1_new,'ro',label="True value")
    plt.semilogy(x,f1_approx,'go',label="Function approximation")
    plt.grid()
    plt.legend(loc="upper right")
    plt.figure()

    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.title("Plot of cos(cos(x)) vs x and the plot of function obtained by
   approximation")
    plot(x,y2_new,'ro',label="True value")
    plot(x,f2_approx,'go',label="Function approximation")
    plt.grid()
    plt.legend(loc="upper right")

```

For e^x



For cos(cos(x))



In order to get better estimate of e^x we should use very high frequency sinusoids, whereas in case of cos(cos(x)) which is a fundamental period of π, it is very easily represented by 2π periodic sinusoids. Hence there is near perfect agreement in case of second function, whereas we see significant amount of deviation in case of first function.

Results

In this assignment we plotted e^x and cos(cos(x)) by using fourier's theorem, we used both direct integration method and least square method for obtaining the fourier series coefficients of two functions. By doing this assignment it is verified that the b_n coefficients for even function are always zero. Also we observed that the deviation in obtained result was more in case of e^x, whereas cos(cos(x)) was almost perfectly obtained with the help of best fit fourier coefficients. This may be explained by using the fact that cos(cos(x)) is a perfectly periodic function whereas e^x is not a periodic function but a monotonically increasing function, hence the coefficients obtained yielded very precise values for cos(cos(x)). Also the use of semilog and loglog scales for plotting the curves made visualization much easier and better.