EE2703 Assignment 4 : Fourier Approximations

Aman Kumar EE19B066

March 10, 2021

1 The assignment

In this assignment we do the following-

- We will fit two functions **exp(x)** and **cos (cos(x))** in the interval [0,2π) using the fourier series.
- Find fourier coefficients using integration.
- Find fourier coefficients using the method of least sqaures
- Compare the two results.

1.1 Introduction

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

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

Where, a_n and b_n are given by:

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

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

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

1.2 Things to do

Mainly the following things are asked in this assignment:

- Plot the functions over the interval [-2π,4π) and the expected functions generated by fourier series.
- Obtain the first 51 coefficients for the two functions above.
- Make different plots using "semilogy" and "loglog" and plot the magnitude of the coefficients vs n. We have to plot this matrix:

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

- Find the same coefficients using the "Least squares approach".
- Compare the answers got by least squares and by the direct integration.
- Plot the function corresponding to fourier coefficients by least squares approach.

2 Assignment questions

2.1 The given functions

The functions **exp(x)** and **cos(cos(x))** are defined using *numpy.exp(x)* and *numpy.cos()* respectively so that they can take vectors and input and return vectors as output. They are defined as:

```
def f1(x):
    return(np.exp(x))
def f2(x):
    return(np.cos(np.cos(x)))
```

The plot of the given functions, their expected fourier approximation plot and their plot by approximation by the method of least squares are given in Figure 1 and Figure 2. Since $exp(x)$ is not a periodic function, by fourier method we will get a periodic function with period 2π. It's graph over $[0,2\pi)$ will be repeated as shown in Figure 1. But for $cos(cos(x))$, no such problem exists as it is a periodic function.

2.2 Obtaining Fourier coefficients through Integration

We are asked to compute the first 51 coefficients of the fourier series. To compute a_n and b_n we need to define two more functions $u(x,k,f) = f(x)cos(kx)$ and $v(x,k,f) = f(x)sin(kx)$. These function has a function as one of the arguments, this function is multiplied by $cos(nx)$ or $sin(nx)$ as required. They are defined as:

```
def u(x,f,k):
    return(f(x)*cos(k*x))
def v(x,f,k):
    return(f(x)*sin(k*x))
```

To perform integration **quad()** method is used of the module **scipy.integrate**. Since *u(x,f,k)* and *v(x,f,k)* both require 3 arguments, remaining two arguments i.e. f and k are passed by doing *args = (f,k)*. The function to find coefficients is defined as:

```
def cal_coeff(f,coeffs):
    coeffs[0] += (1/(2*Pi))*quad(f,0,2*Pi)[0]

    for i in range(1,26):
        coeffs[2*i - 1] += (quad(u,0,2*Pi,args=(f,i))[0])/Pi
        coeffs[2*i] += (quad(v,0,2*Pi,args=(f,i))[0])/Pi
```

semilogy plots of magnitude of coefficients are plotted in Figure 3 and Figure 5, and **loglog** are plotted in Figure 4 and Figure 6 for **exp(x)** and **cos(cos(x))** in that order with red circles.

2.3 Obtaining Fourier coefficients through Least squares approach

We define a vector x(called as $\mathbf{x}\_$) going from 0 to 2π in 400 steps and evaluate the function $f(x)$ at those x values and call it b. Now we get the coefficients from the following matrix problem by **Least squares** approach.

$$\begin{bmatrix} 1 & cosx_1 & sinx_1 & ... & cos25x_1 & sin25x_1 \\ 1 & cosx_2 & sinx_1 & ... & cos25x_2 & sin25x_2 \\ ... & ... & ... & ... & ... & ... \\ 1 & cosx_{400} & sinx_{400} & ... & cos25x_{400} & sin25x_{400} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ b_1 \\ ... \\ a_{25} \\ b_{25} \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ ... \\ f(x_{400}) \end{bmatrix}$$

We call the matrix on the left **A**. Then the coefficient matrix **c** is given by:

```
c=lstsq(A,b)[0]
```

The matrix **A** is calculated as follows:

```
A = np.zeros((400,51))
A[:,0] = 1
for i in range(1,26):
    A[:,2*i - 1] = np.cos(i*x_)
    A[:,2*i] = np.sin(i*x_)
```

In my code the coefficients through this approach is found as follows:

```
def cal_coeff_lsq(A,f):
    if f == f1:
        b = f1(x_)
    elif f == f2:
        b = f2(x_)
    return(pl.lstsq(A,b,rcond=None)[0])
```

semilogy plots of magnitude of coefficients obtained through this approach are plotted in Figure 3 and Figure 5, and **loglog** plot are plotted in Figure 4 and Figure 6 for **exp(x)** and **cos(cos(x))** in that order with green circles.

2.4 Comparing the two answers

We compare the answers got by least squares and by the direct integration by finding the absolute difference between the two sets of coefficients and finding the largest deviation. I did this by:

```
def Compare():
    return(max(abs(coeffs_f1 - coeffs_f1_ls)),max(abs(coeffs_f2 - coeffs_f2_ls)))
```

Max deviation in the coefficients of $exp(x)$: 1.3327308703352259
Max deviation in the coefficients of $cos(cos(x))$: 2.5428380512387945e-15

Clearly, Our Predictions for $exp(x)$ are very poor compared to that of $cos(cos(x))$. This can be fixed by sampling at a larger number of points. But this will be a small change in accuracy at a very large computational cost. Hence it isn't worth it as far as this assignment is concerned.

3 Plots and Results

3.1 The functions

We plot the functions $exp(x)$ and $cos(cos(x))$ in Figure 1 and Figure 2 respectively, along with the expected fourier apprixamation and approximation through least squares.

Here, we can see that there are large deviations in the approximation by least squares, this because $exp(x)$ is not a periodic function and to determine it's fourier series we periodically extended the function, thus creating discontinuities at intervals of 2π. Therefore at the first discontinuity of these intervals we see large deviations. We can also reduce the deviation by sampling at a very large number of points(as mentioned earlier).

However, there is almost no deviation from expected plot in the case of $cos(cos(x))$ as it is a periodic and continuous function. Hence, fourier series converges everywhere to the function. Code snippet used to plot the above two plots:

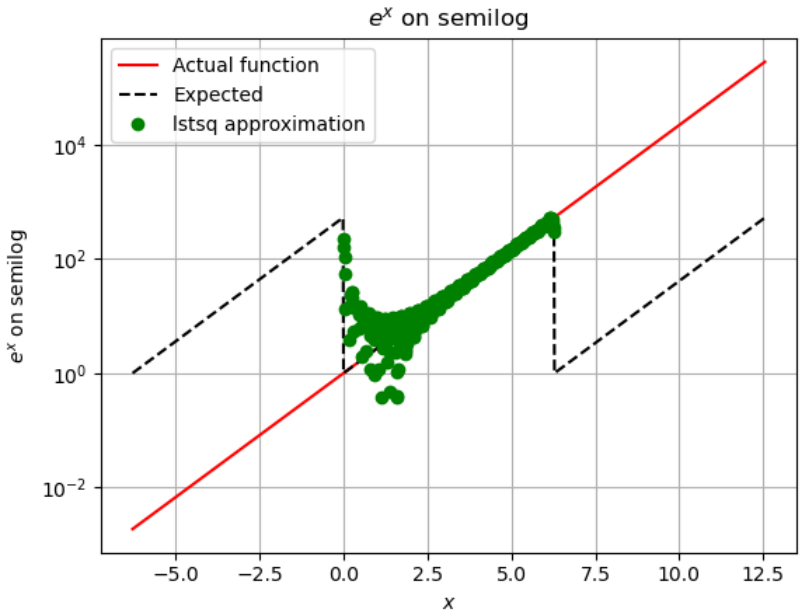

Figure 1: $exp(x)$

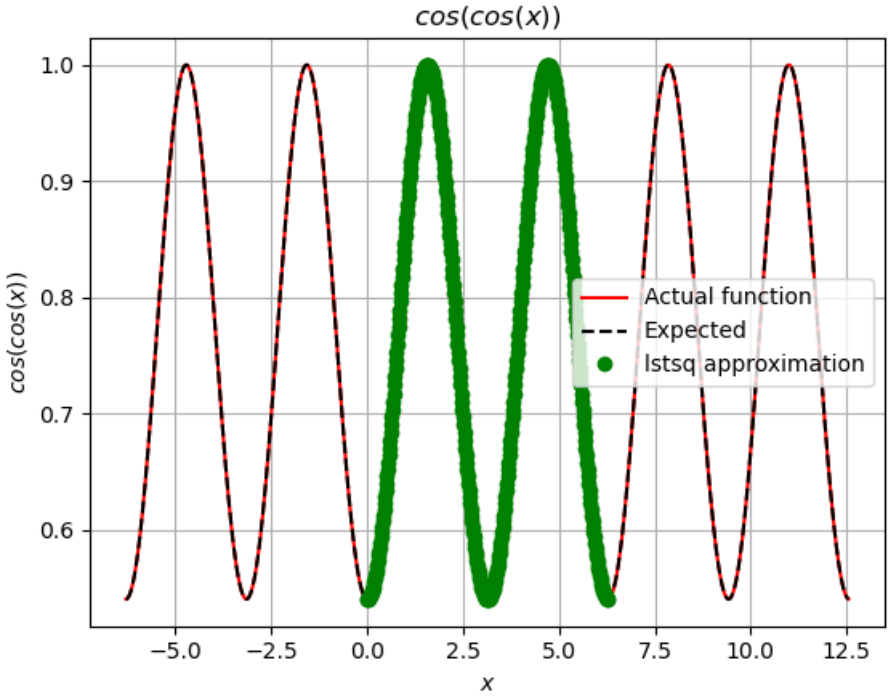

Figure 2: $cos(cos(x))$

```
def Plot_fn(f):
    y1 = f(x)
    y3 = f(x_)
    y3 = np.concatenate((y3,y3,y3))
    if f == f1:                          #If function is exp(x)
        Title = "$e^x$ on semilog"
        y2 = np.dot(A,coeffs_f1_ls)
        pl.semilogy(x,y1,label="Actual function",color='red')
        pl.semilogy(x,y3.transpose(),label="Expected",color='black',linestyle='dashed')
        pl.semilogy(x_,y2,'go',label="lstsq approximation")
```

```
elif f == f2:                    #If function is cos(cos(x))
    Title = "$cos(cos(x))$"
    y2 = np.dot(A,coeffs_f2_ls)
    pl.plot(x,y1,label="Actual function",color='red')
    pl.plot(x,y3.transpose(),label="Expected",color='black',linestyle='dashed')
    pl.plot(x_,y2,'go',label="lstsq approximation")
```

3.2 Coefficients

We plot the magnitude of coefficients for both the functions on **semilogy** and **loglog**. The coefficients obtained through both methods are plotted in the same plot. Code snippet used to plot the **semilog** and **loglog** plots(I've omitted non-important stuff):

```
pl.semilogy(n,abs(coeffs_f1),'ro',label="$Integration$",linestyle='dashed',linewidth=1)
pl.semilogy(n,abs(coeffs_f1_ls),'go',label="$Least sqaures$",linestyle='dashed',linewidth=1)

pl.loglog(n,abs(coeffs_f1),'ro',label="$Integration$",linestyle='dashed',linewidth=1)
pl.loglog(n,abs(coeffs_f1_ls),'go',label="$Least sqaures$",linestyle='dashed',linewidth=1)
```

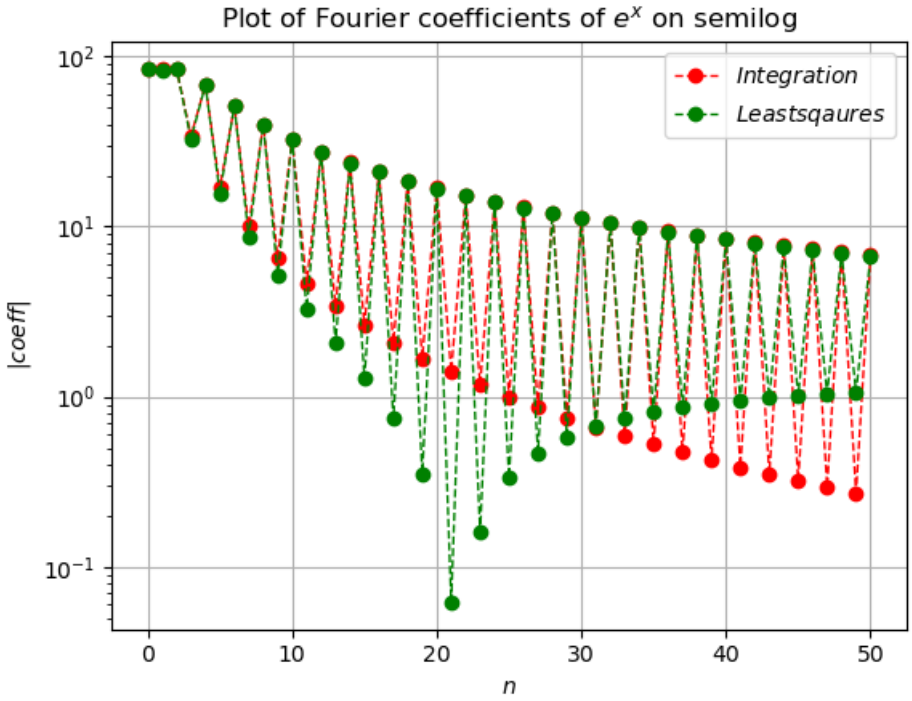

Figure 3: Coefficients of $exp(x)$ on semilogy

We can see here that this plot(semilogy) is not linear, while the plot in loglog is linear. Also we can see that value of coefficients(b_n) found by the two methods are differing in some range.

We observe that the coefficients for $exp(x)$ do not decay as quickly as the coefficients for $cos(cos(x))$, which means $exp(x)$ has a lot(and higher) of frequencies in it's spectrum, while $cos(cos(x))$ is made up of very less number of frequencies.

Also, since $cos(cos(x))$ is an **even function** therefore its fourier sine series is zero. This is reflected in the plots as the coefficients b_n are nearly zero($< 10^{-15}$) while this is not the case for $exp(x)$ as it is not an even function.

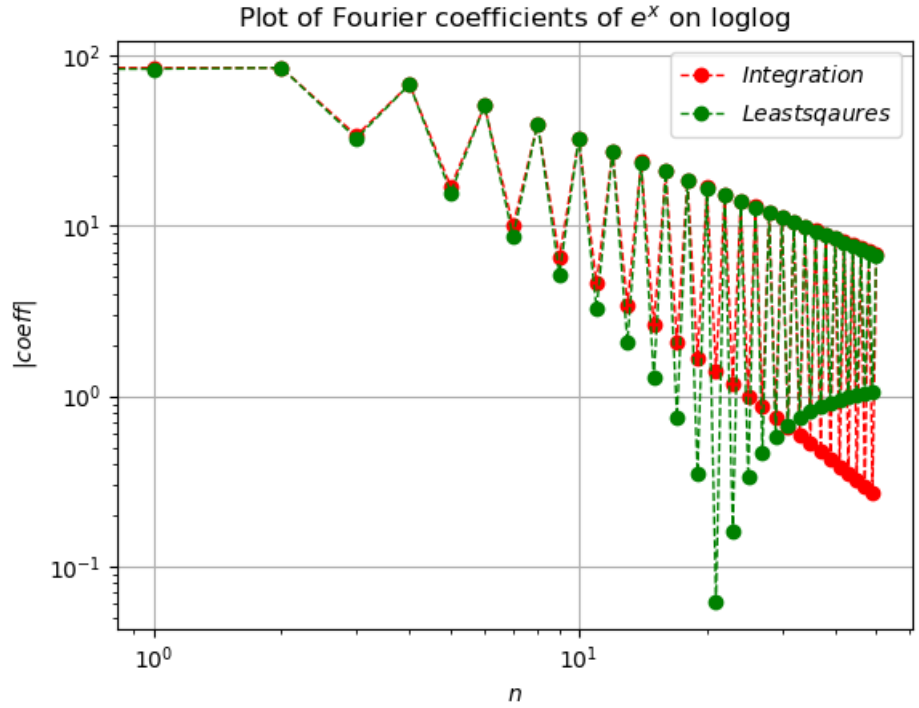

Figure 4: Coefficients of $exp(x)$ on loglog

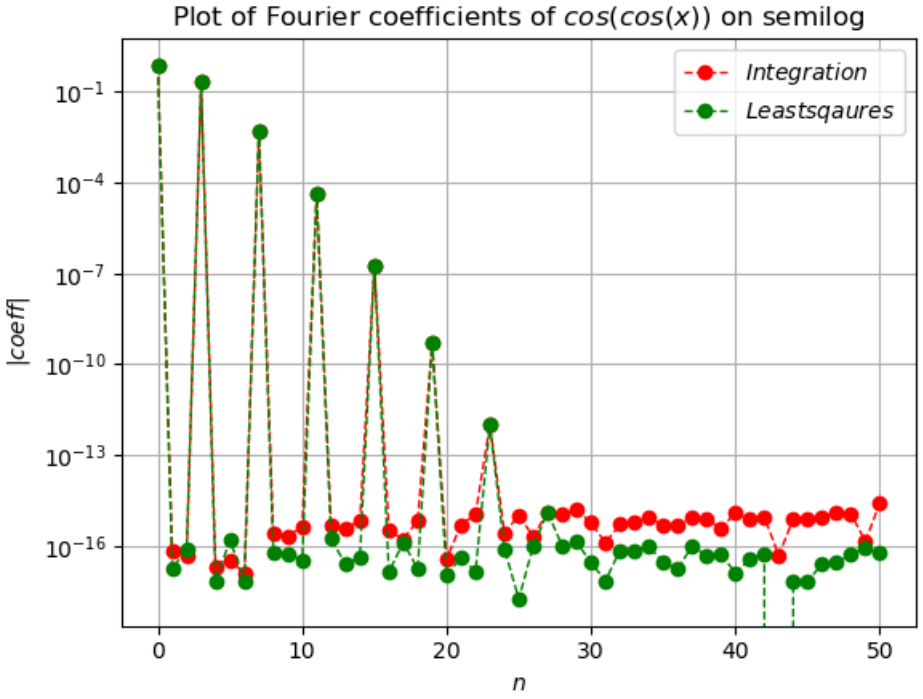

Figure 5: Coefficients of $cos(cos(x))$ on semilogy

The coefficients for $cos(cos(x))$ are linear in semilogy whille non-linear in loglog. For $exp(x)$ a_n and b_n decay as $1/n^2$ and $1/n$, so $log(a_n)$ and $log(b_n)$ will be approximately proportional to $log(n)$. Thus loglog is linear. While in $cos(cos(x))$ the coefficients decay exponentially w.r.t n, hence semilog plot looks linear.

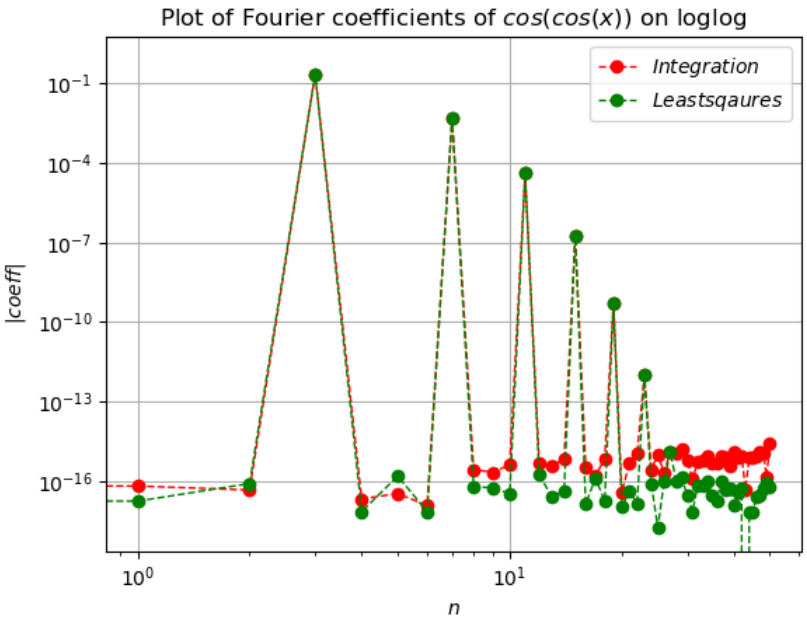

Figure 6: Coefficients of $cos(cos(x))$ on loglog

4 Concluions

We have examined the case of approximating functions using their Fourier coefficients upto a threshold. We perform the same for two cases, one a continuous function, and the other a function with finite discontinuities. In the latter case, we found that the error in apprximation is more than that of continuous case.

The methods adopted in finding the Fourier coefficients have been evaluation through Euler's formulae(integration), as well an Least Square best fit approach. We notice close matching of the two methods in case of $cos(cos(x))$ while, there is a larger discrepancy in $exp(x)$.