Assignment-4 : Fourier Approximations

Sai Gautham Ravipati - EE19B053

March 11, 2021

Abstract

Fourier series of a periodic function provides a convenient way of representing them as an infinite series of cosine and sine terms. Through this assignment, several insights about the same are analysed particularly through the functions $cos(cos(x))$ and the periodic extension of e^x with the period being [0,2π). The Fourier series representation of a 2π periodic function is given by,

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

where the coefficients a_n and b_n are given by,

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

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

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

Furthermore, an approach of computing the coefficients using least squares method is also foreseen.

1 Functions and Periodic Extension

Throughout the assignment only two particular functions e^x and $cos(cos(x))$ shall be referred. Clearly e^x is a aperiodic function for $x \in \mathbb{R}$. However considering $x \in [0, 2\pi)$, and repeating the same values for 2π length intervals gives the periodic extension of e^x and Fourier series can be computed. While this was the case of e^x, $cos(cos(x))$ is a 2π periodic function, and Fourier series can be computed directly. Following are plots of the functions and there extensions in the interval [-2π,4π).

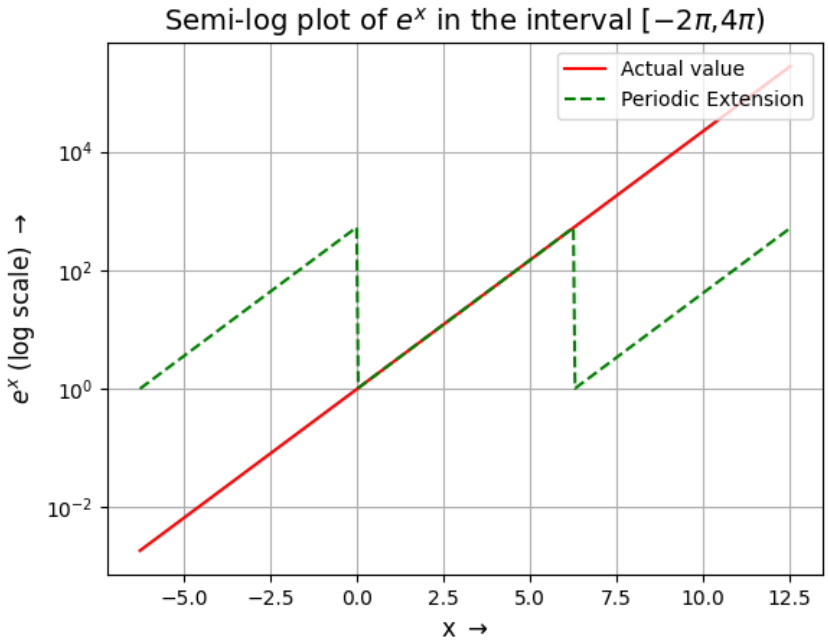

Figure 1: Plot of e^x in (log) scale and x in linear scale

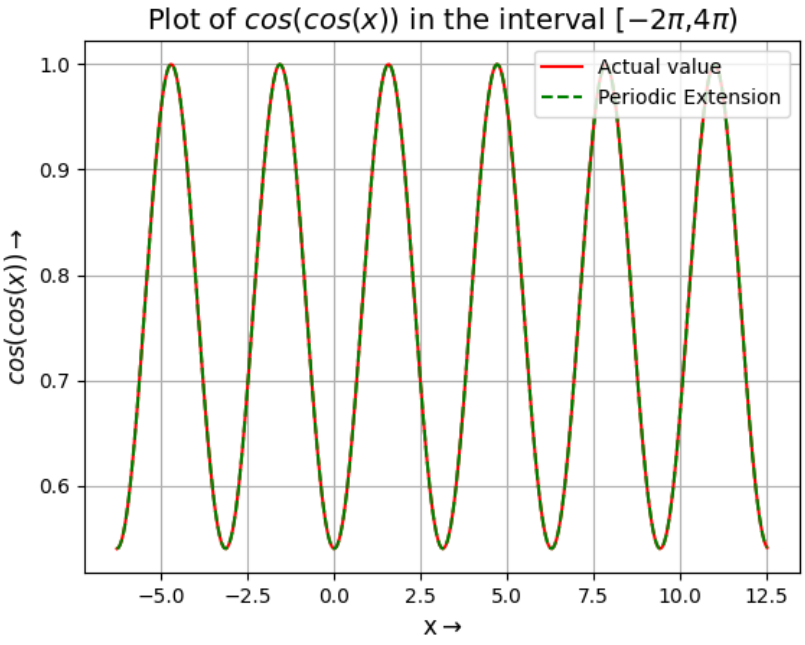

Figure 2: Plot of $cos(cos(x))$ in linear scale

Clearly it can be seen that $cos(cos(x))$ is periodic because of which its extension and the function overlap, while that's not the case of e^x. Note from the graph that $cos(cos(x))$ is 2π periodic while its fundamental period is π since $cos(x)$ is an even function of x.

Since Fourier series of periodic continuous functions converges pointwise, the function represented by Fourier series provides a very good approximation for $cos(cos(x))$, and this is true even if the series is truncated at lower number of terms, as the weight of the high frequency sinusoids is comparatively lower than that of e^x. On a similar notion the function approximated by Fourier series of e^x should converge to periodic extension of e^x, but since periodic extension is piece-wise continuous, the function converges to mid-point of left and right neighbourhood values at the discontinuity. Further since e^x grows rapidly and it is piece-wise continuous, the weight of higher frequency sinusoids is more when compared to $cos(cos(x))$ and a better approximation to periodic extension is obtained when relatively more number of terms are considered.

Coming to the code the first two python functions are used to generate the points for the functions e^x and $cos(cos(x))$. To generate the periodic extension, the last function has been defined. This function translates every element i in vector x to a periodic interval [lower,upper) such that $f(i) = f(i + ap)$, where a is an integer and p is the period. This works for any other interval also instead of hard coding [-2π,4π). The lines of code are as follows :

```
import numpy as np
def exp(x):
    return np.exp(x)
def cos_cos(x):
    return np.cos(np.cos(x))
def periodic_extension(x,lower,upper):
    if not(upper>lower):
        print('Please give proper range of limits')
        exit()
    p = upper - lower
    x_out = 2*pi*np.ceil(-x/p) + x
    return x_out
```

2 Fourier series coefficients using Integration

The following function is used to generate Fourier series coefficients, it takes in the name of the function, *exp* or *cos_cos* which are defined previously, and the highest index of Fourier coefficients to be generated. For say, coeff_intg(exp,25) returns the vector,

$$\begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ . \\ . \\ a_{25} \\ b_{25} \end{pmatrix}$$

conatining the the first 51 fourier coefficients of e^x. The lines of code are:

```
def coeff_intg(f,n):
    if n != int(n) or n < 0 :
        print('Please enter a positive integer value for n')
        exit()
    u = lambda x,k : f(x)*np.cos(k*x)
    v = lambda x,k : f(x)*np.sin(k*x)
    rtnval = np.zeros(2*n+1)
    rtnval[0] = (integrate.quad(f,0,2*pi)[0])/(2*pi)
    for k in range(1,n+1) :
        rtnval[2*k-1] = integrate.quad(u,0,2*pi,args=(k))[0]/pi
        rtnval[2*k]   = integrate.quad(v,0,2*pi,args=(k))[0]/pi
    return rtnval
```

The first 51 Fourier series coefficients are generated for e^x and $cos(cos(x))$ and are plotted in both semi-log, log-log scales as presented below.

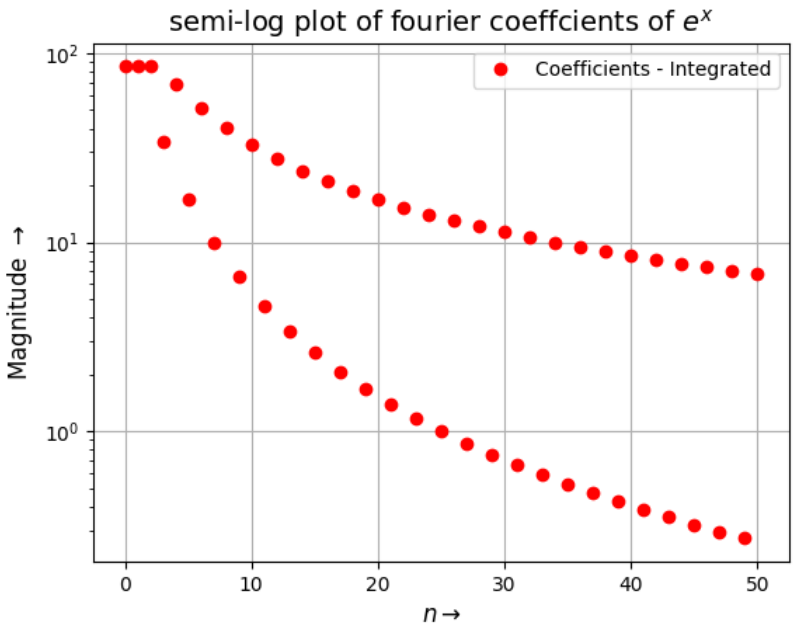

Figure 3: Fourier coefficients of e^x in (semi-log) scale

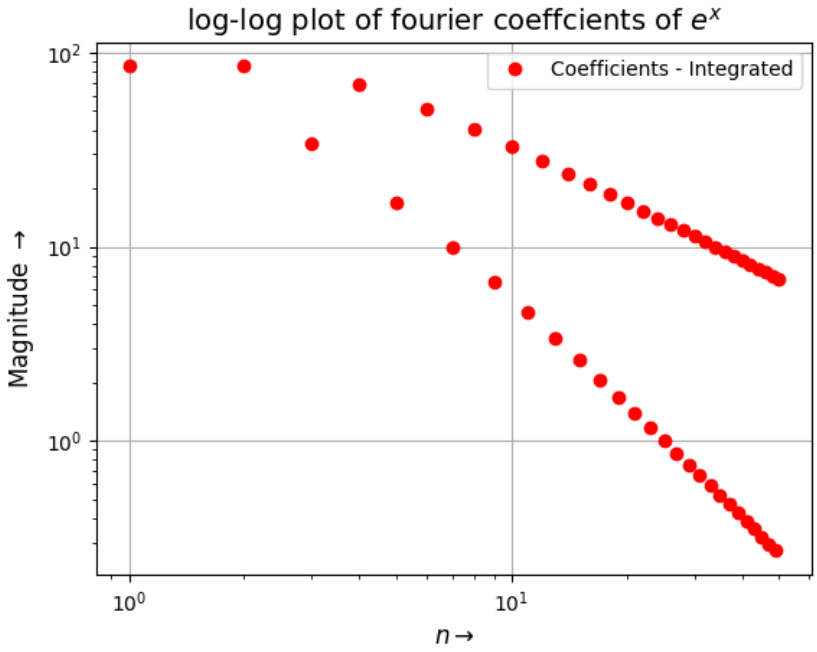

Figure 4: Fourier coefficients of e^x in (log-log) scale

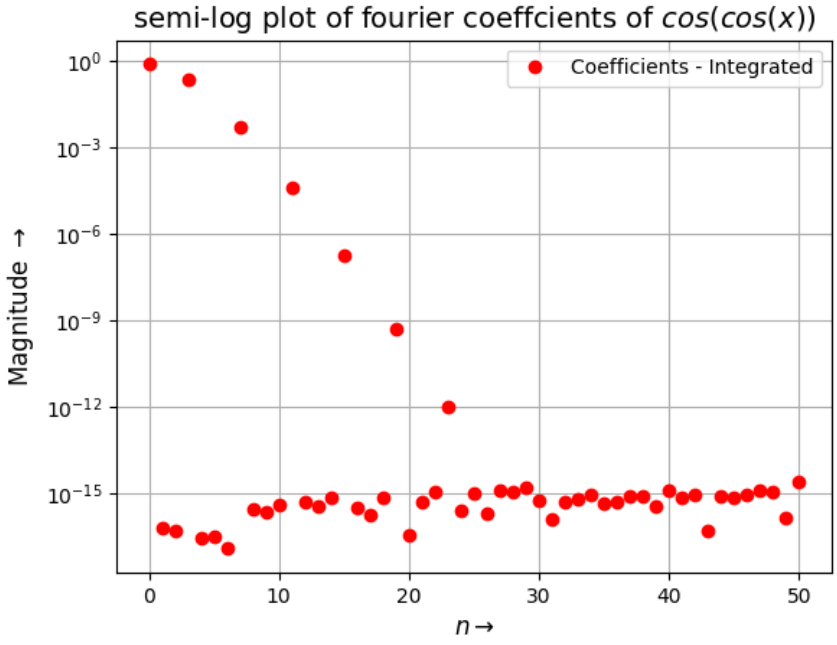

Figure 5: Fourier coefficients of $cos(cos(x))$ in (semi-log) scale

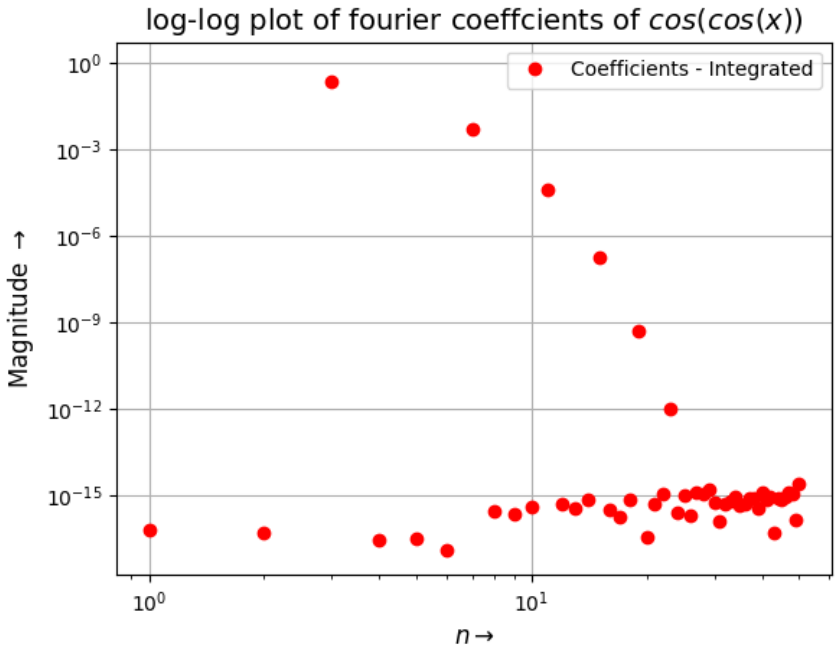

Figure 6: Fourier coefficients of $cos(cos(x))$ in (log-log) scale

2.1 Analysis of coefficients

- The b_n coefficients in case of $cos(cos(x))$ should be nearly zero. Why does this happen ?
  Reason : The b_n coefficients in case of $cos(cos(x))$ are nearly zero follows from the fact that $cos(cos(x))$ is an even function of x. The coefficient b_n is given by,

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

  Since $sin(nx)$ is an odd function, while $cos(cos(x))$ is an even function, their product is also a odd function, because of which b_n are nearly zero, as inferred from both Figure 5,6. While analytical approach gives b_n as zero, because of numerical approximation, they converge to a small value.

- In the case of e^x, the coefficients do not decay as quickly as the coefficients of $cos(cos(x))$. Why not?
  Reason : The fundamental period of $cos(cos(x))$ is π, so the fundamental frequency is $\frac{1}{\pi}$, and since the function is continuous, the coefficients of higher frequency sinusoids do not contribute significantly to the Fourier series. While at the same time e^x is discontinuous at

every even multiple of π, and further it grows rapidly when compared to that of $cos(cos(x))$, as a result of which higher frequency sinusoids do contribute greater unlike the previous case. Because of these factors, the coefficients of e^x do not decay as quickly as the coefficients of $cos(cos(x))$.

- Why does log-log plot in Figure 4 look linear, whereas the semi-log plot in Figure 5 looks linear?
  Reason : Upon evaluation the coefficients in the case of e^x are given by,

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

$$b_n = \frac{n(1 - e^{2\pi})}{(n^2 + 1)\pi} \tag{3}$$

  For large n,

$$|a_n| \sim \frac{1}{n^2} \tag{4}$$

$$|b_n| \sim \frac{1}{n} \tag{5}$$

  Taking logarithm gives :

$$log(\frac{1}{n^2}) = -2log(n) \tag{6}$$

$$log(\frac{1}{n}) = -log(n) \tag{7}$$

  Because of the above variation, the log-log plot in Figure 4 looks linear. While the semi-log plot of $cos(cos(x))$ is linear as the magnitude of coefficients decays exponentially with n.

3 Least Squares Approach

The Fourier coefficients can also be computed using a least squares approach, where the scipy operation 'lstsq' returns the least square solution to the following linear matrix equation. Here in this case the equation Ac = b is :

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

The following lines of code gives the function which is used to generate A,b. The functions takes the name of the function, exp or cos_cos which are defined previously, and the highest index of Fourier coefficients to be generated, data-points x and returns the matrices A,b.

```
def vec_lstsq(f,n,x):
    if n != int(n) or n < 0 :
        print('Please enter a integer value for n')
        exit()
    pts = x.shape[0]
    b = f(x)
    A = np.zeros((pts,2*n+1))
    A[:,0] = 1
    for k in range(1,n+1) :
        A[:,2*k-1] = np.cos(k*x)
        A[:,2*k]   = np.sin(k*x)
    return A,b
```

The returned values of A,b when passed as inputs to 'lstsq' return the estimate for the coefficient vector c, the same is plotted in the previous plots in green as shown, while the true value of coefficients are shown in red. Here in this case the vector x is uniform sample of 400 points in [0,2π).

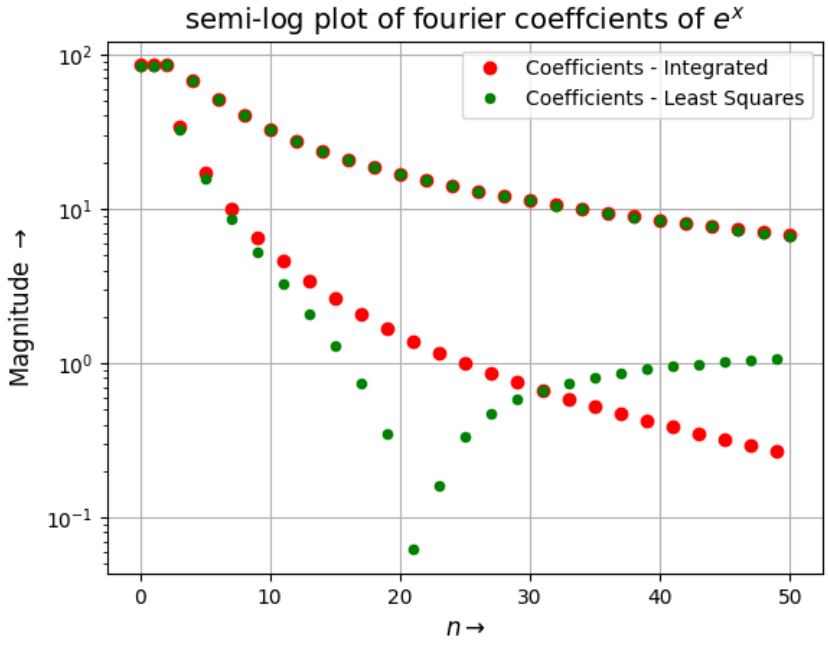

Figure 7: Fourier coefficients of e^x (400 x data-points)

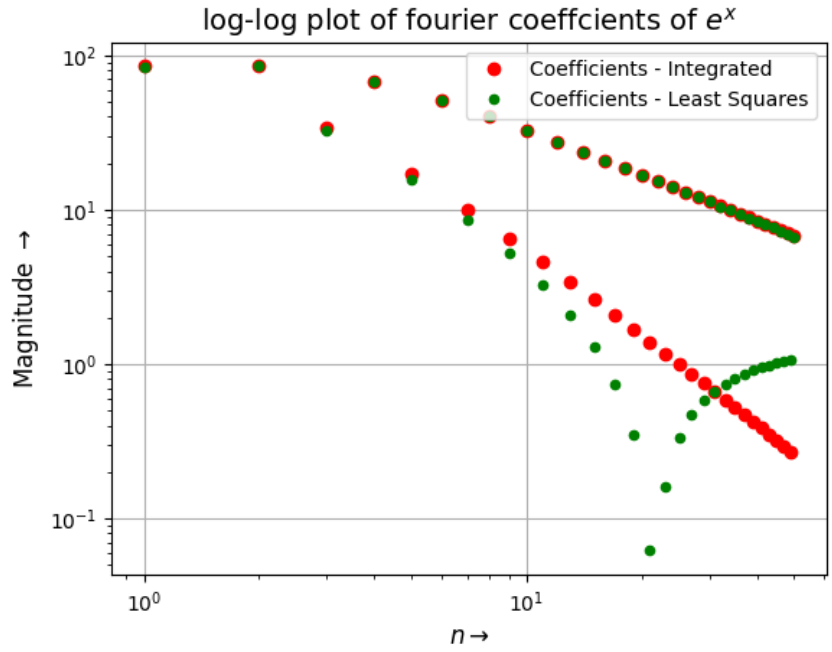

Figure 8: Fourier coefficients of e^x (400 x data-points)

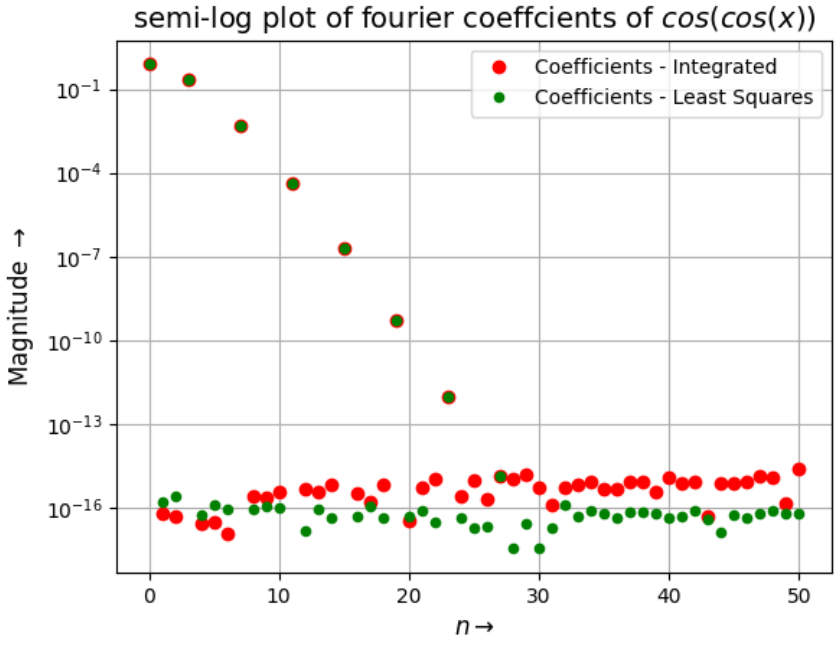

Figure 9: Fourier coefficients of $cos(cos(x))$ (400 x data-points)

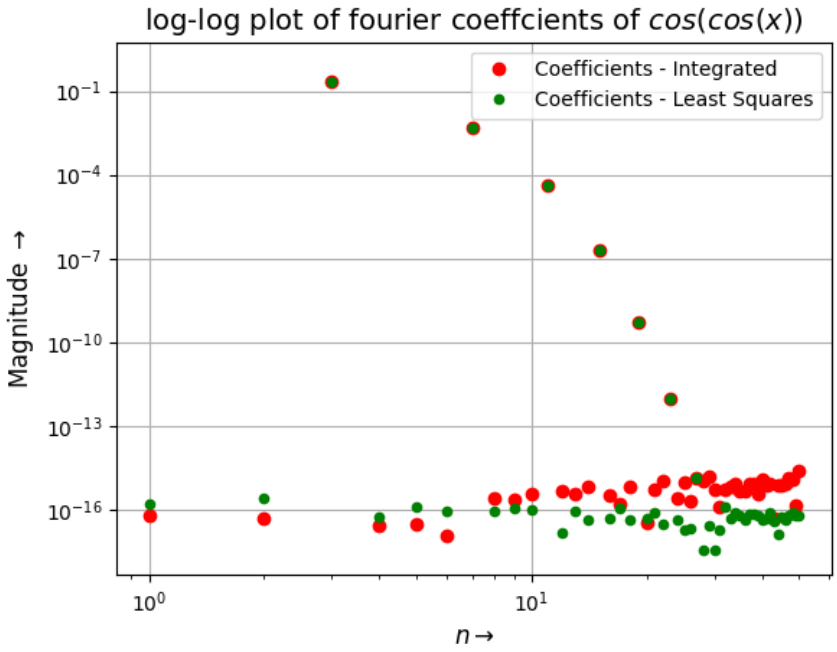

Figure 10: Fourier coefficients of $cos(cos(x))$ (400 x data-points)

3.1 Analysis of coefficients

From the above plots, clearly the degree of deviation in answers obtained using least squares is significant in case of e^x than that of $cos(cos(x))$, while the answers relatively agree in the case of $cos(cos(x))$. Further as e^x has greater contribution from higher frequency sinusoids, it requires more data-points for a better estimate to be produced using the least squares approach unlike the numerical integration. It is evident from the following plots (Fig. 11,12) that the deviation that occurred previously was because of the number of data-points and they do agree when more data-points are considered. Here in this case x is uniform sample of 10000 points in $[0,2\pi)$. The deviation computed are as follows :

- For 10000 data-points :
  The deviation in case of exp(x) is 5.339319e-01
  The deviation in case of cos(cos(x)) is 2.662603e-15
- For 400 data-points :
  The deviation in case of exp(x) is 1.332731e+00
  The deviation in case of cos(cos(x)) is 2.656647e-15

Clearly there isn't much variation in the latter while the former changes by a factor of 10.

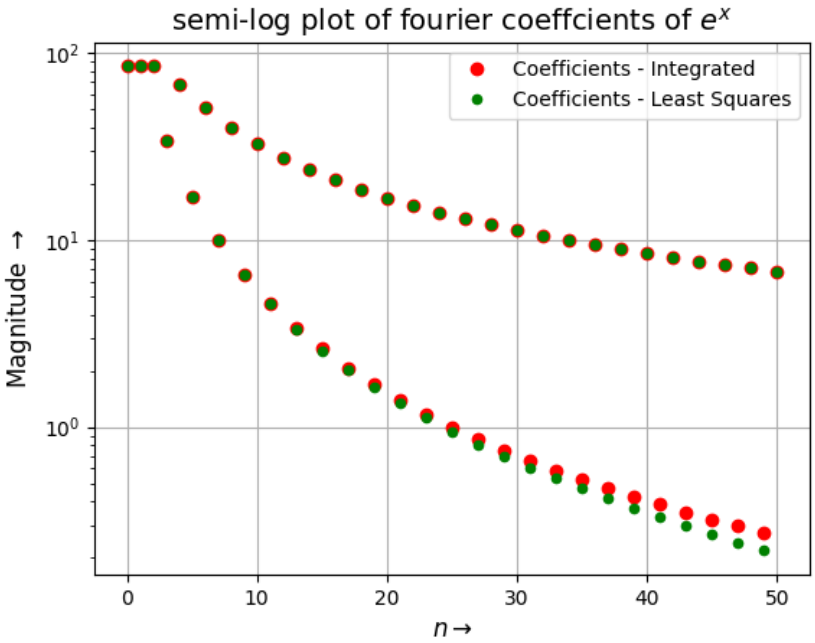

Figure 11: Fourier coefficients of e^x (10000 x data-points)

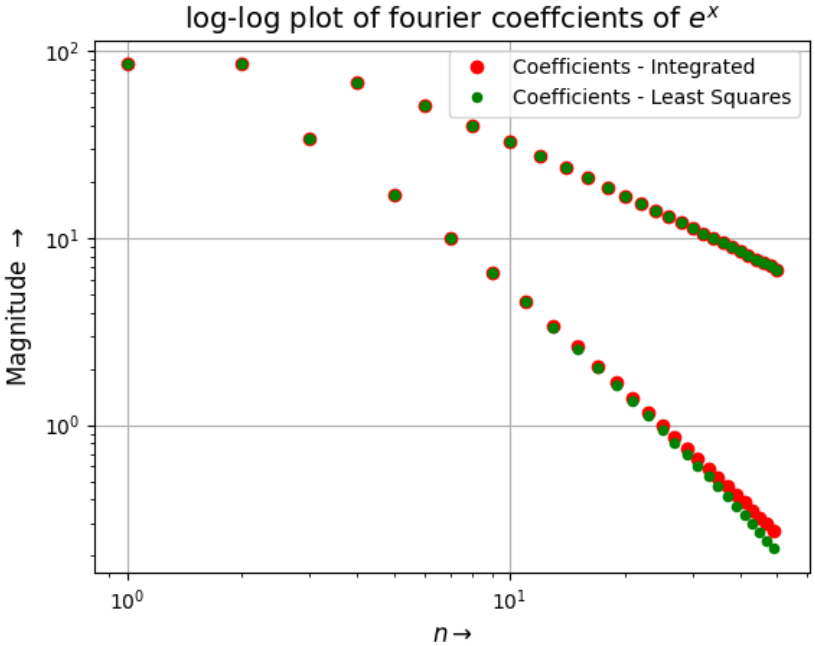

Figure 12: Fourier coefficients of e^x (10000 x data-points)

The following line of code is used to compute the deviation, from vectors coeff_intg (generated using numerical integration) and coeff_lstq (generated using least squares method).

```
delta = np.max(abs(coeff_intg - coeff_lstq))
```

4 Estimated function

Using the values of c estimated using least squares, the product of Ac is taken which gives a vector of length 400, and which is a function of x, when these are plotted along with the original function, the following plots are obtained.

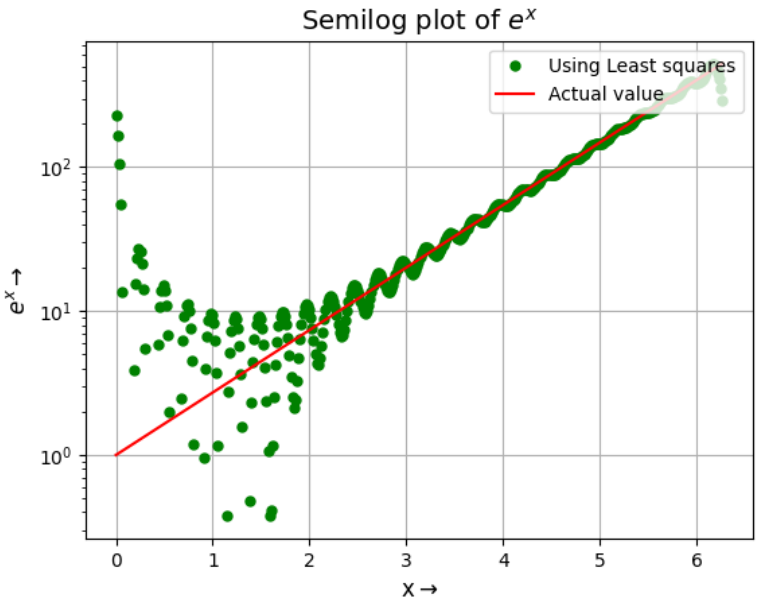

Figure 13: Function approximation of e^x

Clearly there is a deviation in case of e^x while there is nearly perfect agreement in case of $cos(cos(x))$, this is because only first 51 coefficients are considered, and since $cos(cos(x))$ is perfectly continuous and has its most of the contribution from lower frequency sinusoids, so it is perfectly approximated , while the periodic extension of e^x is discontinuous and has relatively more contribution from higher frequency sinusoids also because of which the truncated series shows large deviation. However, from Fig. 15 it can be seen that, although it shows some deviation,it provides a good approximation for e^x, further it can be seen that the Fourier series converges to midpoint of the neighbourhood values at discontinuities.

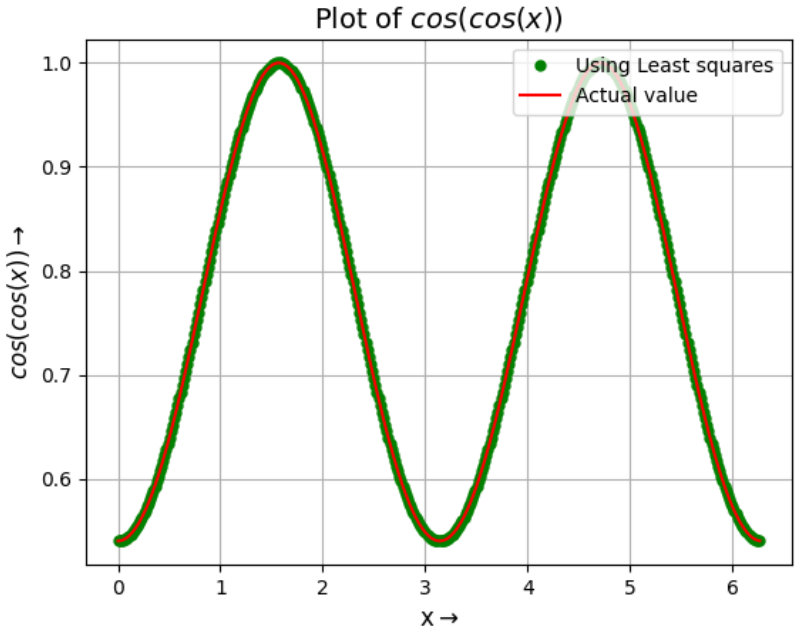

Figure 14: Function approximation of $cos(cos(x))$

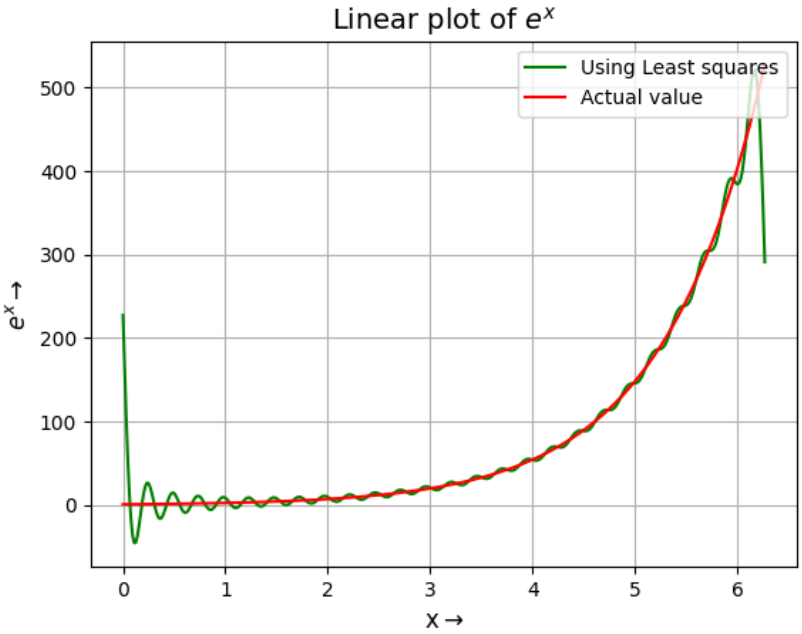

Figure 15: Function approximation of e^x

5 Conclusion

Fourier series provides a convenient way of approximating periodic functions, through this assignment two approaches have been used to find the Fourier series approximations of functions e^x and $cos(cos(x))$ in the interval $[0,2\pi)$, since the former is a piece-wise discontinuous function and increases exponentially in a period, it has greater contributions from higher frequency sinusoids, while the latter is periodic with period π, and has relatively lower contribution from higher frequency sinusoids, because of which coefficients of $cos(cos(x))$ fall rapidly. Furthermore least squares method gives less deviation in case of $cos(cos(x))$ thus providing a way which is more analytic than numerical integration, while at the same time least squares approach produces relatively more deviation in case of e^x. The approximated function shows relatively greater deviation in case of e^x, but the approximation is fairly good, and Fourier series is converging to mid-point of neighbourhood values in case of discontinuities.