EE2703 : Applied Programming Lab
Assignment 4
Fourier Approximation

Arun Krishna A M S
EE19B001

March 10, 2021

Aim

A Fourier series is a way of representing a periodic function as a (possibly infinite) sum of sine and cosine functions. In this assignment, we mainly focus on:

- Approximating functions e^x and $cos(cos(x))$ in the range $[0, 2\pi)$ by calculating fourier coefficients through
  - Analysis Equation
  - Least Squares approach i.e., Curve fitting approach
- Calculate the fourier series through the synthesis equation with coefficients obtained from both the methods
- Study the error between the two approaches by plotting the coefficients obtained from the two methods

Introduction

Consider a real valued periodic function that is integrable on an interval of length 2π, which will be the period of the Fourier series. The **synthesis process** i.e., the actual Fourier series is given by:

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

The weights: $a_0, a_1, a_2..., a_n...b_1, b_2...b_n..$ for the harmonics is determined by the **analysis equation**:

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

The existence of the fourier series alone doesn't determine the convergence or equality of $F(x)$ at all points. At discontinuities

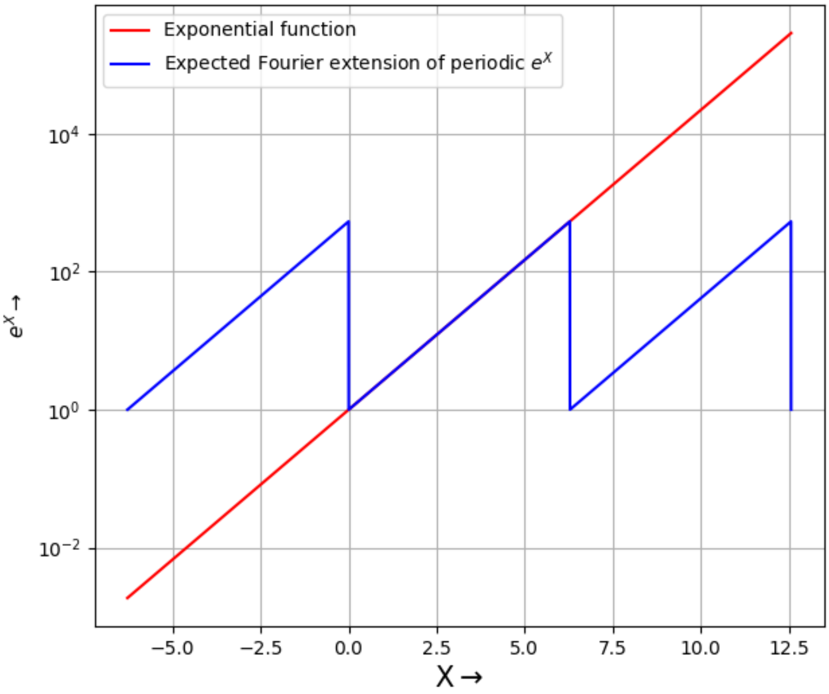
$$F(c) = \frac{1}{2}(f(c^+) + f(c^-))$$

Fourier series doesn't converge at discontinuities, and at these discontinuities, Gibbs phenomenon takes place.

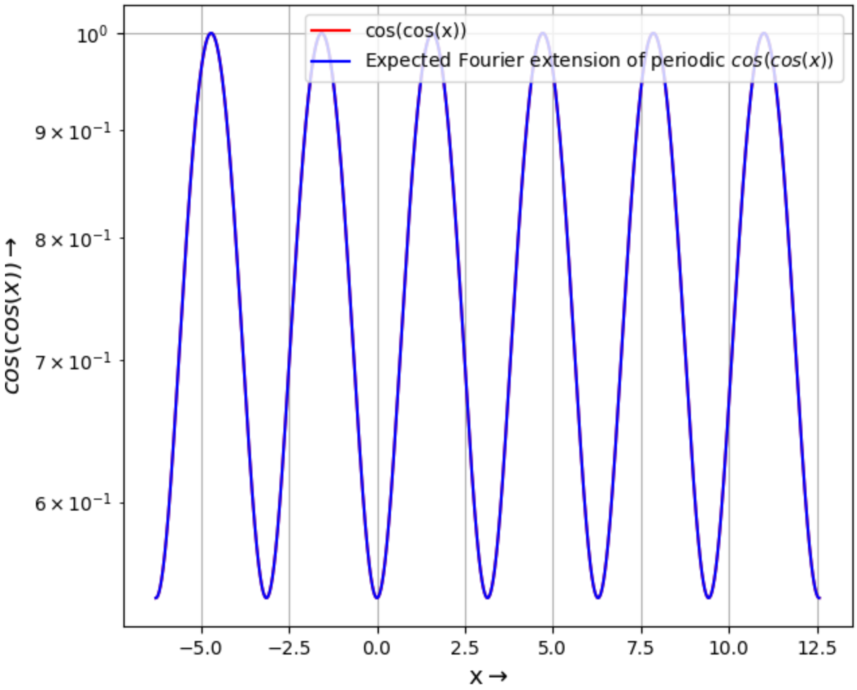
The assignment starts with defining the `exponential(x)` and `cosCos(x)` as

```
def exponential(x):
    return exp(x);
def cosCos(x):
    return cos(cos(x));
```

The expected fourier series is periodic with interval 2π when computed and plotted should look like



semilogY plot of e^x



plot of $cos(cos(x))$

It can be observed that the function e^x **is non-periodic while** $cos(cos(x))$ **is periodic** (Question 1) with periodicity π. Function $cos(cos(x))$ and its fourier series, overlaps with each other. It can be observed that fourier transform of e^x repeats every 2π with discontinuities existing at the multiples of 2π.

Generation of Fourier Coefficients

The fourier coefficients is calculated by the analysis equation. In general, N tends to ∞, but for all practical purposes, N can be observed to be a large finite number. Here we consider only $N = 25$ which is sufficiently large. This may not be correct if the function is significantly made up of higher frequencies.

The functions $u(x, k) = f(x)\cos(kx)$ and $v(x, k) = f(x)\sin(kx)$ is generated by

```
func_dict = {’exponential(x)’:exponential,’cosCos(x)’: cosCos}
def u_Func(x,argument):
    k = argument[0];
    label = argument[1];
    func = func_dict[label];
    return func(x)*cos(k*x);

def v_Func(x,argument):
    k = argument[0];
    label = argument[1];
    func = func_dict[label];
    return func(x)*sin(k*x);
```

Functions $u(x,k)$ and $v(x,k)$ is then used in the analysis equation to determine the fourier coefficients.

$$a_k = \frac{1}{\pi}\int_0^{2\pi} u(x,k)dx$$
$$b_k = \frac{1}{\pi}\int_0^{2\pi} v(x,k)dx$$

The integration is completed by the `scipy.integrate.quad`, with arguments for $u(x,k)$ and $v(x,k)$ is passed by creating the list `argument = [i,label]`

```
def functionCoefficient(label, numcoefficient):
    function = func_dict[label];
    cosCoefficient = np.zeros((numcoefficient+1)//2);
    sinCoefficient = np.zeros(numcoefficient//2);
    Coefficient = np.zeros(numcoefficient);

    cosCoefficient[0] = (SP.quad(function, 0, 2*pi)[0]) / (2*pi);
    Coefficient[0] = cosCoefficient[0];

    for i in range(1,cosCoefficient.size,1):
        argument = [i,label];
        cosCoefficient[i] = (SP.quad(u_Func,0,2*pi,args = argument)[0])/pi;
        Coefficient[2*i-1] = cosCoefficient[i];

    for i in range(1,sinCoefficient.size+1,1):
        argument = [i,label];
        sinCoefficient[i-1] = (SP.quad(v_Func,0,2*pi,args = argument)[0])/pi;
```

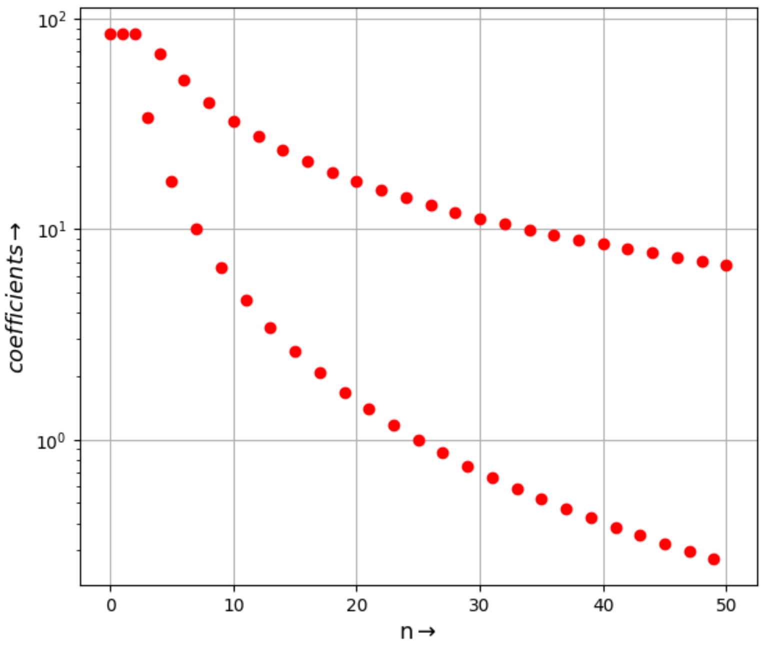
```
          Coefficient [2*i] = sinCoefficient[i-1];

     return Coefficient,cosCoefficient,sinCoefficient;
```

The fourier series coefficients is given by

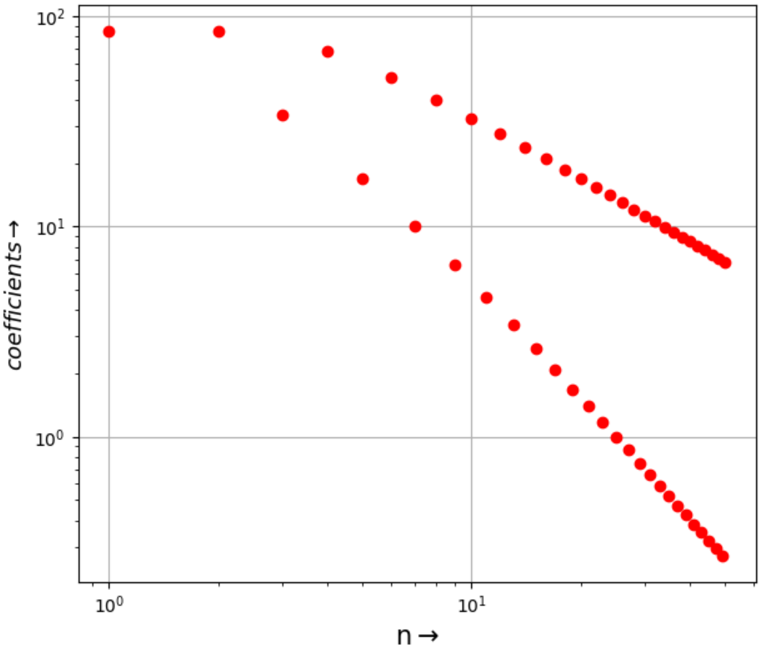
$$\texttt{Coefficient} = \begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ . \\ . \\ a_{25} \\ b_{25} \end{pmatrix} \quad \texttt{cosCoefficient} = \begin{pmatrix} a_0 \\ a_1 \\ . \\ . \\ a_{25} \end{pmatrix} \quad \texttt{sinCoefficient} = \begin{pmatrix} b_1 \\ b_2 \\ . \\ . \\ b_{25} \end{pmatrix}$$

Visualization of Fourier Coefficients

Fourier coefficients of e^x are plotted below:



Fourier coefficients of e^x (semilogY plot)



Fourier coefficients of e^x (loglog plot)

The coefficients of e^x were found to be

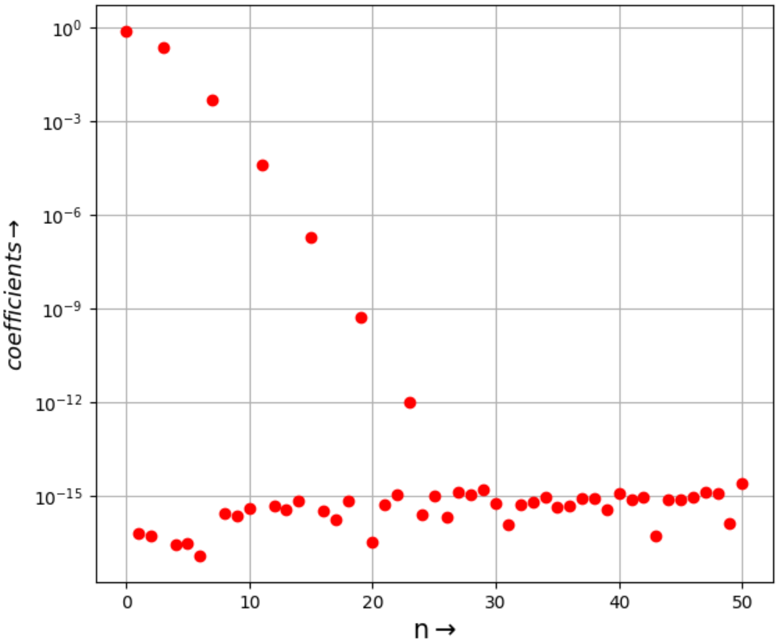
$$
\begin{aligned}
a_k &= \frac{1}{\pi}\int_0^{2\pi} e^x dx = \frac{e^{2\pi}-1}{\pi(1+k^2)} &&\implies a_k \propto \frac{1}{k^2} &&\implies \log a_k \propto -2\log k \\
b_k &= \frac{1}{\pi}\int_0^{2\pi} e^x dx = \frac{k(1-e^{2\pi})}{\pi(1+k^2)} &&\implies b_k \propto \frac{1}{k} &&\implies \log b_k \propto -\log k
\end{aligned}
$$

Thus the `loglog` **plot for e^x is found to be nearly linear** (Question 3c)

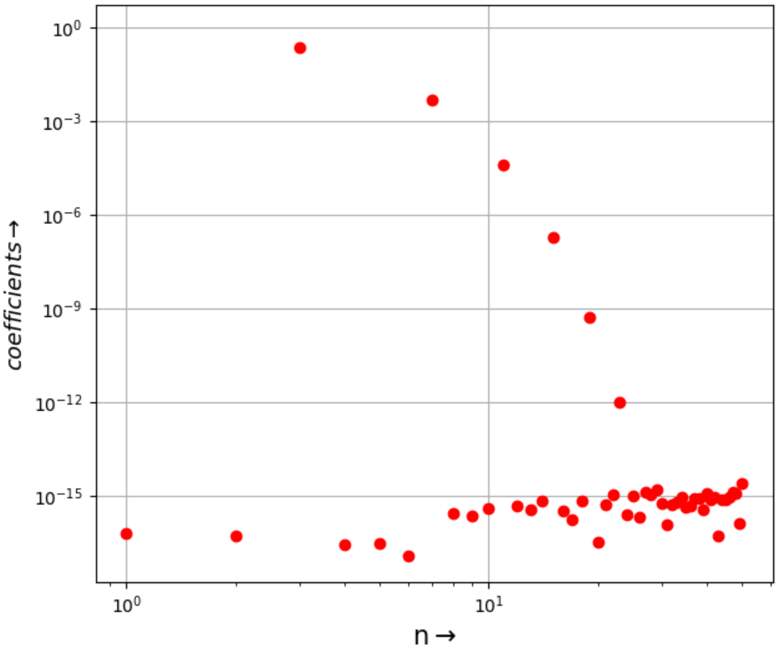
It can be observed that $u(x) = cos(cos(x))cos(nx)$ and $v(x) = cos(cos(x))sin(nx)$ doesn't have closed form antiderivative.

From `Figure 2`, we can intuitively say that the function $cos(cos(x))$ is periodic (with time period π), thus it is made up of low frequencies $\implies$ the **coefficients for higher frequencies for $cos(cos(x))$ decreases exponentially**. Whereas we observed earlier that the weight **for higher harmonics in e^x is indirectly proportional to k and k^2**. (Question 3b)

Fourier coefficients of $cos(cos(x))$ are plotted:



Fourier coefficients of $cos(cos(x))$ (semilogY plot)



Fourier coefficients of $cos(cos(x))$ (loglog plot)

As expected, the coefficients for the second case decreases exponentially. This is shown in the `semilogY plot - figure 6` (Question 3b).

Note that $sin(x)$ function is odd, while $cos(x)$ function is even. Since the function $cos(cos(x))$ is even (i.e., $cos(cos(-x)) = cos(cos(x))$). This means that there are no odd sine components. Thus $b_n = 0$. Since the `quad` function computes integration numerically, values of b_n may not be accurately be 0. (Question 3a)

Fourier Coefficients by Linear Square Approach

The first 51 coefficients can also be approximately figured out by **Linear Square Approach**. Define a vector `X` going from 0 to 2π in 400 steps using `linspace`.

```
numData = 400;
X = np.linspace(0,2*pi,numData+1)[:-1];
```

Evaluate the function $f(x)$ at those x values and call it b. Now this function is to be approximated by the synthesis equation. So for each x_i we want,

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

Turning this into a matrix problem

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

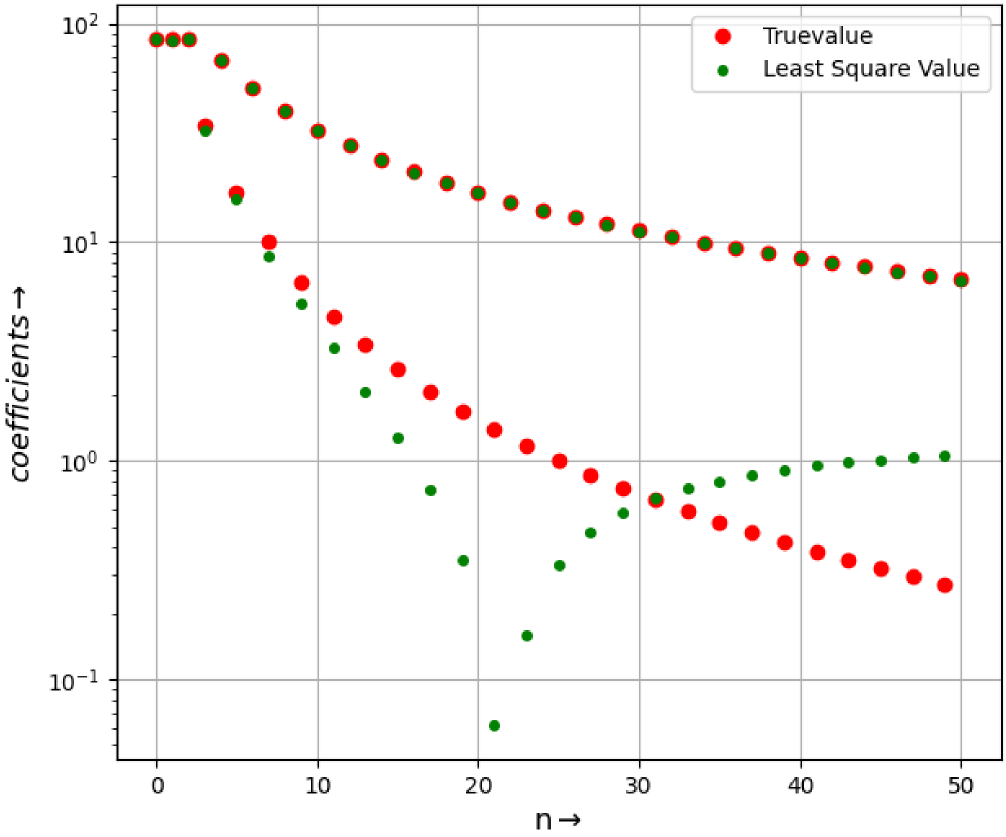
Let the matrix on the left side be A. We want to solve $Ac = b$ where c are the fourier coefficients.

```
A = np.zeros((numData,numcoefficient));
A[:,0] = 1;
for i in range(1,(numcoefficient+1)//2):
      A[:,2*i-1] = cos(i*X);
      A[:,2*i] = sin(i*X);
```

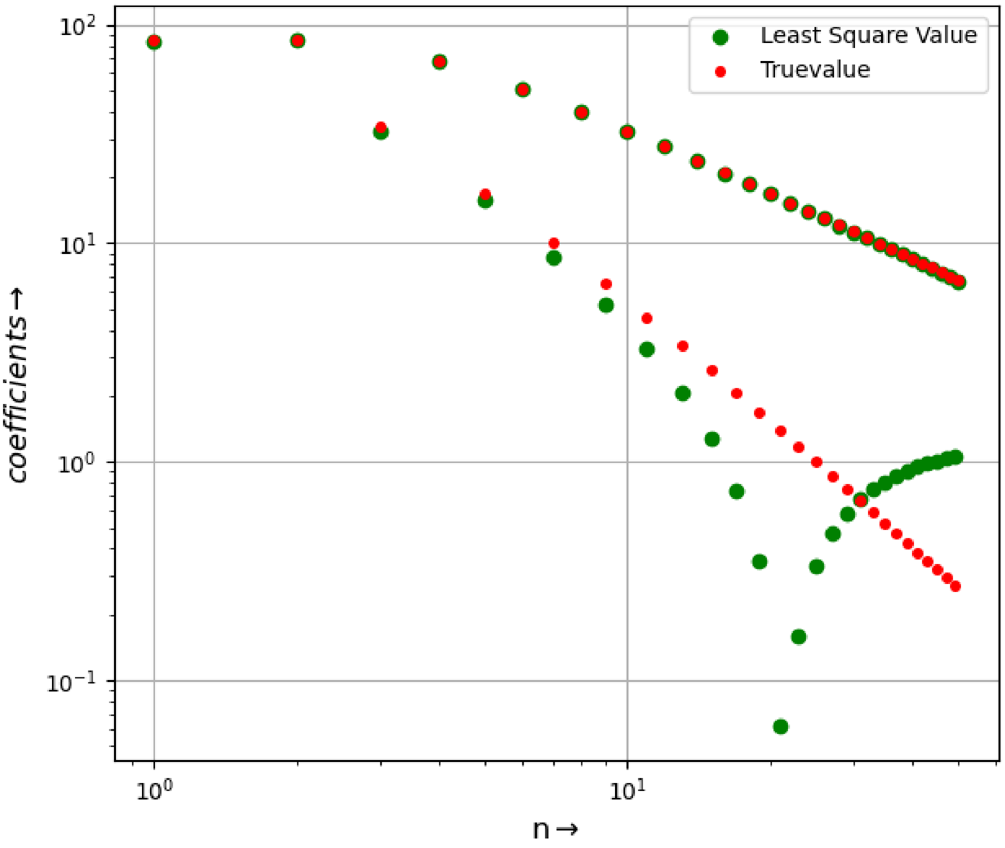
Using `scipy.linalg.lstsq` we solve this problem by executing

```
b = exponential(X);
C_Expo = scipy.linalg.lstsq(A,b)[0];

b = cosCos(X);
C_CosCos = scipy.linalg.lstsq(A,b)[0];
```

This finds the "best fit" numbers that will satisfy the matrix equation at exactly the points we have evaluated $f(x_i)$. Plotting the obtained coefficients for the given functions.



Fourier coefficients of e^x (semilogY plot)



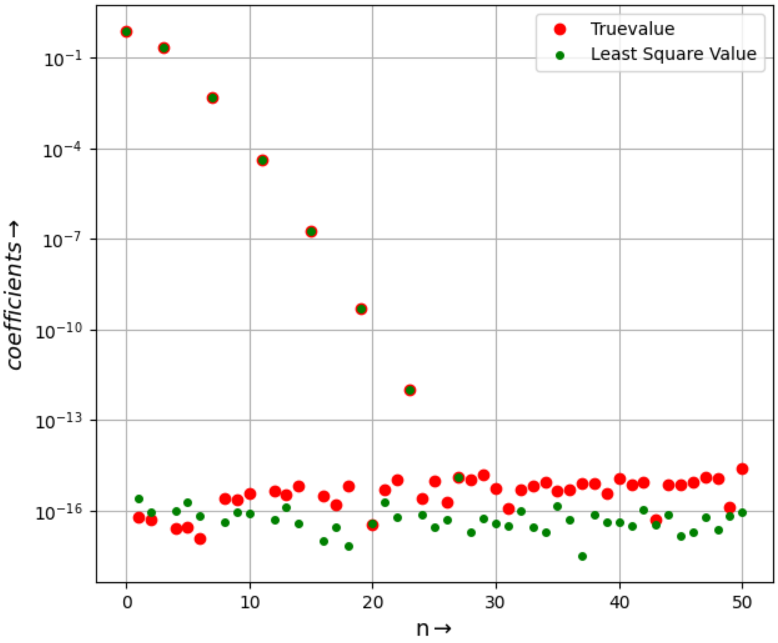
Fourier coefficients of e^x (loglog plot)

```
errorMax_Exp = np.amax(np.abs(CoefficientExpo - C_Expo))
errorMax_CosCos = np.amax(np.abs(CoefficientCosCos - C_CosCos))
```

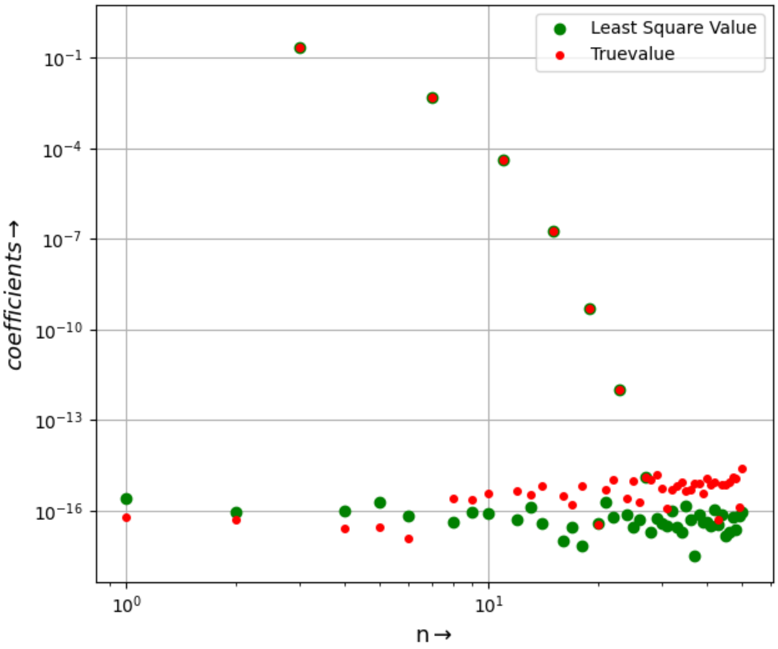
The maximum absolute error between the coefficients obtained from analysis equation and the coefficients obtained from least square method for the **function** e^x **is** 1.3327308703354248**.** (Question 6)

The maximum absolute error between the coefficients obtained from analysis equation and the coefficients obtained from least square method for the **function** $cos(cos(x))$ **is** $2.682986310394032 \times 10^{-15}$**.** (Question 6)

Our predictions for e^x **is worse compared to** $cos(cos(x))$. The error deviation for $cos(cos(x))$ is plotted below.



Fourier coefficients of $cos(cos(x))$ (semilogY plot)



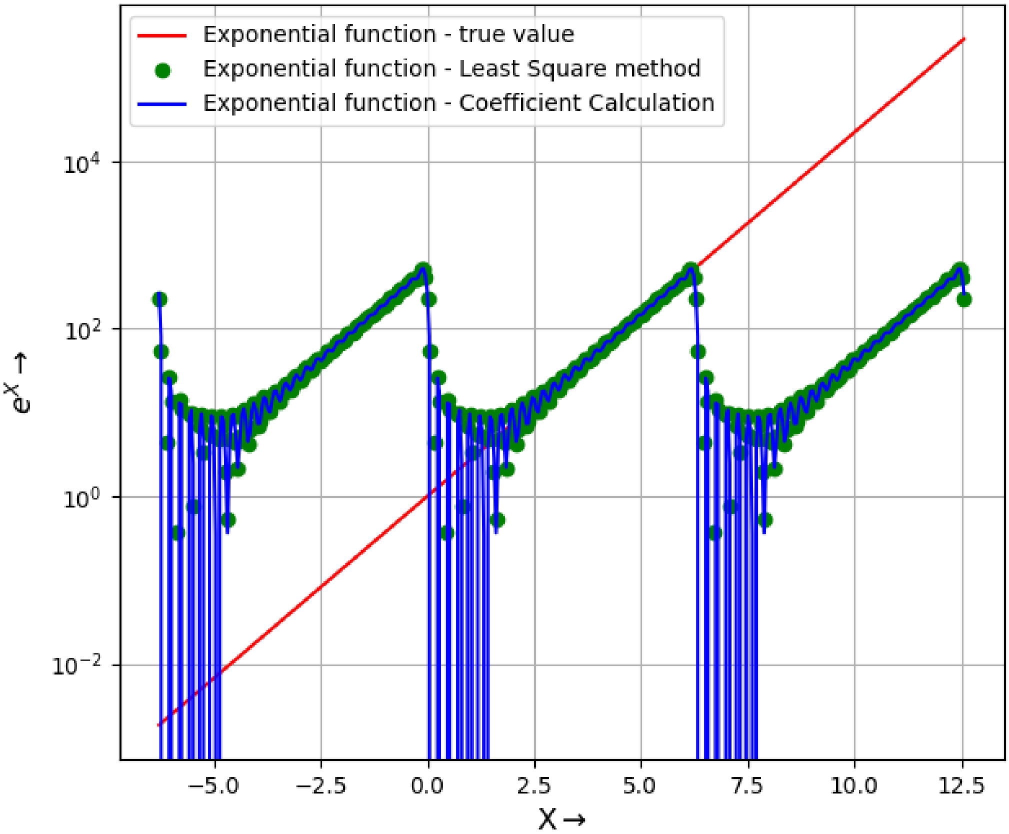
Fourier coefficients of $cos(cos(x))$ (loglog plot)

Function Approximation

The matrix $V = Ac$ gives the approximate values at $(x_1, x_2, x_3 ... x_{400})$ where

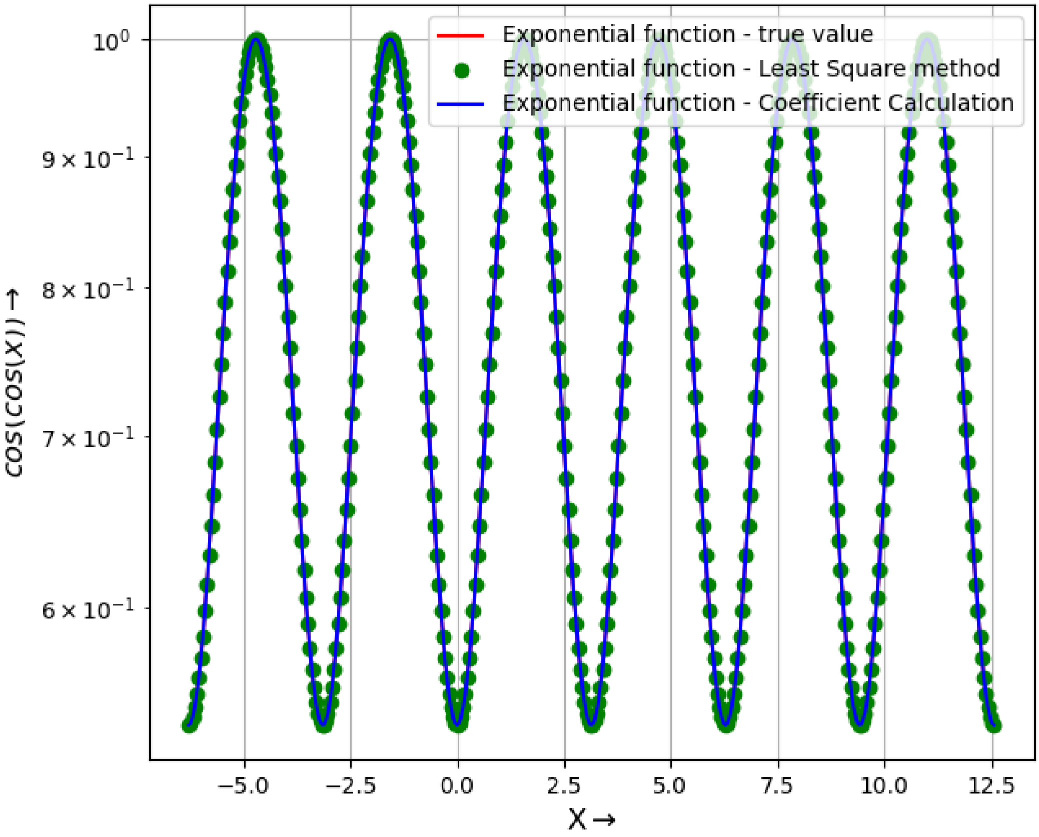
- c column vector is calculated by the least square process
- c column vector is calculated by the analysis equation

and then plotted.



e^x (semilogY plot)

The graph obtained by the analysis equation show rapid rise and fall at the discontinuities in the e^x graph - **Gibbs Phenomenon**. The function e^x has significant weight for higher frequencies, which is ignored in this calculation, resulting in great deviation.(Question 7)



$cos(cos(x))$

The function $cos(cos(x))$ is predominantly made up of lower frequencies, thus the fourier series is very close to the actual function. Since the function doesn't have discontinuities, Gibbs Phenomenon is not observed.

Thus we can say that **fourier series of $cos(cos(x))$ is a much better approximate to the $cos(cos(x))$ compared to e^x** (Question 7)

Conclusion

Fourier Coefficients of $cos(cos(x))$ and e^x is calculated by both Analysis and Least Square method. We also observed that the presence of discontinuities and higher frequencies made the fourier series approximations for e^x significantly diverge from its function compared to the function $cos(cos(x))$