EE2703 : Applied Programming Lab
Assignment 4

Potta Muni Asheesh
EE19B048

March 23, 2021

1 Aim

We will be fitting two functions, namely, $exp(x)$ and $cos(cos(x))$ over an interval of [0, 2π) using Fourier series.

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

The coefficients a_0, a_n, b_n are given by

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

2 Assignment questions

2.1 Plotting the two functions

We can define functions which use functions from `numpy` to compute the required function for a given vector(or scalar) and returns a vector(or scalar). The functions are named as `e` and `coscos` in the code.

```
def e(x):
    return np.exp(x)
```

```python
def coscos(x):
    return np.cos(np.cos(x))
```

The plot of these two functions over the interval $[-2\pi, 4\pi)$ is given below. It is seen that $cos(cos(x))$ is periodic and $exp(x)$ is not periodic. When we compute the Fourier series of these two functions, the generated function is expected to be over the interval $[0, 2\pi)$, as shown below

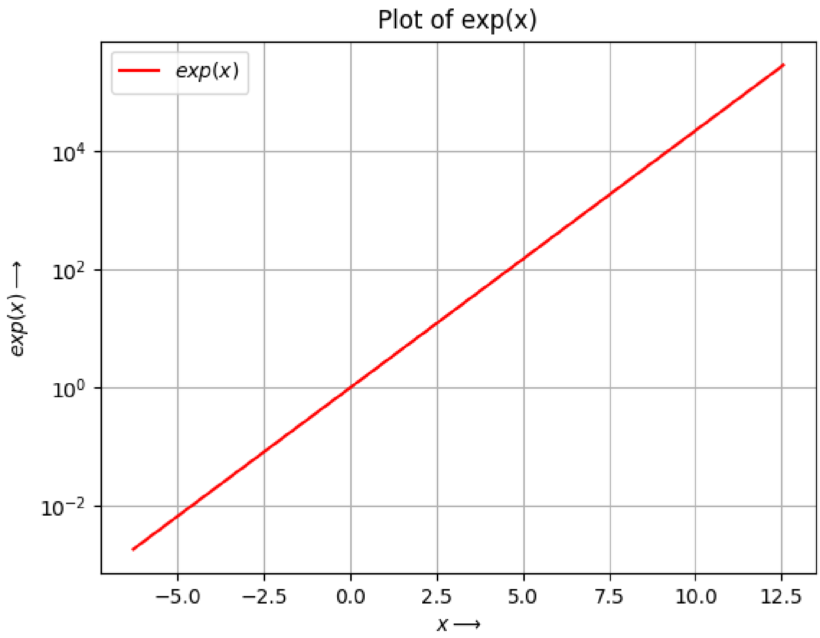

Figure 1: Plot of $exp(x)$ (semi-log scale)

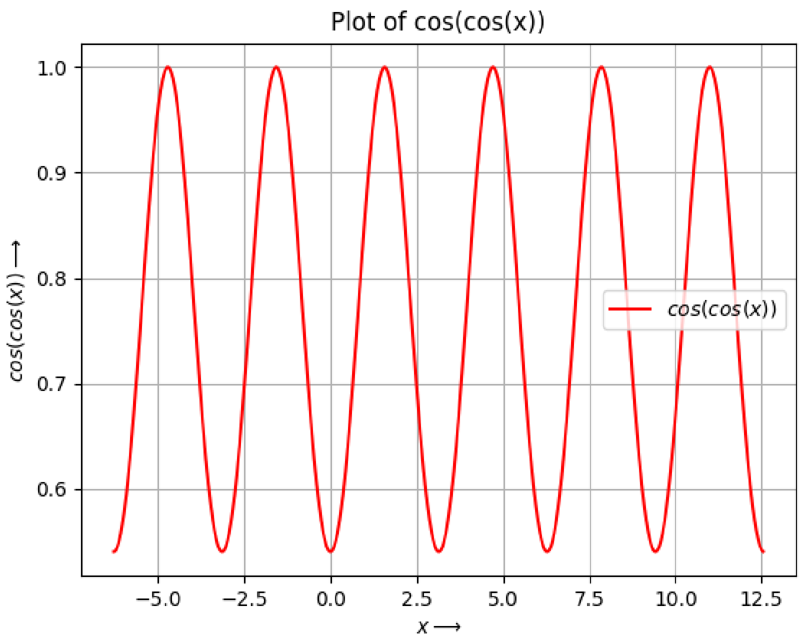

Figure 2: Plot of $cos(cos(x))$

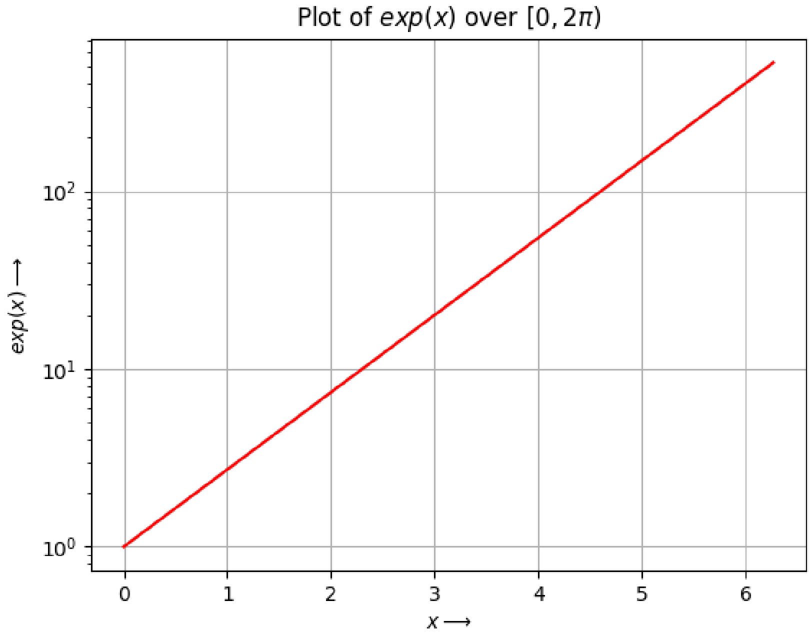

Figure 3: Plot of $exp(x)$

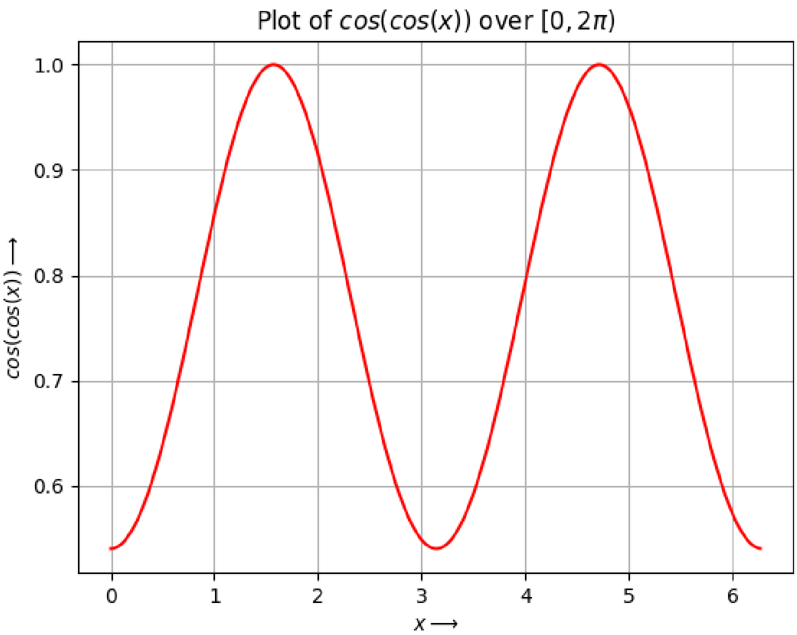

Figure 4: Plot of $cos(cos(x))$

2.2 Obtaining Fourier coefficients using direct integration

As discussed above, we can use integration to find the coefficients of the Fourier series. In Python, we can do definite integration using `quad` function from `scipy.integrate`. Two Python functions `u(x, k, f) = f(x)cos(kx)` and `v(x, k, f) = f(x)sin(kx)` are defined, to pass to `quad` function, as shown below

```
def u(x, k, f):
    return f(x)*np.cos(k*x)

def v(x, k, f):
    return f(x)*np.sin(k*x)
```

Here, the parameter `f` should be a function and it should return a scalar or vector. The functions defined previously are used here. `for` loop is used to find the coefficients iteratively as shown below. The coefficients for $exp(x)$ and $cos(cos(x))$ are stored in arrays named `coeffs_f1` and `coeffs_f2` respectively.

```
coeffs_f1 = np.zeros(51)
coeffs_f2 = np.zeros(51)
coeffs_f1[0] = quad(u, 0, 2*pi, args=(0,e))[0]/(2*pi)
coeffs_f2[0] = quad(u, 0, 2*pi, args=(0,coscos))[0]/(2*pi)

for i in range(1, 26):
    coeffs_f1[2*i-1] = quad(u, 0, 2*pi, args=(i,e))[0]/pi
    coeffs_f1[2*i] = quad(v, 0, 2*pi, args=(i,e))[0]/pi
    coeffs_f2[2*i-1] = quad(u, 0, 2*pi, args=(i,coscos))[0]/pi
    coeffs_f2[2*i] = quad(v, 0, 2*pi, args=(i,coscos))[0]/pi
```

2.3 Plotting the coefficients

To plot the 51 coefficients, they are first arranged in a vector as shown below.

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

Now, this vector is plotted against n as n varies from 1 to 51. For each set of the coefficients, two plots are made, one with *log* scale on y-axis only and the other with *log* scale on both x and y axes. The plots of absolute values of the coefficients are shown below.
It is found that b_n for the function $cos(cos(x))$ are of the order $10^{-15} - 10^{-17}$ which for practical purposes can be approximated to 0. This happens because $cos(cos(x))$ is an even function and coefficients of sin terms in Fourier series are 0 for even functions.

The coefficients for $exp(x)$ are given as

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

So, they decay as $\sim \frac{1}{n^2}$ and $\sim \frac{1}{n}$ which is not as quick as the coefficients of $cos(cos(x))$. This is because, as $exp(x)$ is not periodic, higher frequency sinusoids are also required for good enough convergence. They look linear in the *loglog* plot because they decay as shown above. The coefficients of $cos(cos(x))$ are related to Bessel functions as they can be evaluated from the integral form of Bessels functions, so they look linear in semi-log-y plot.

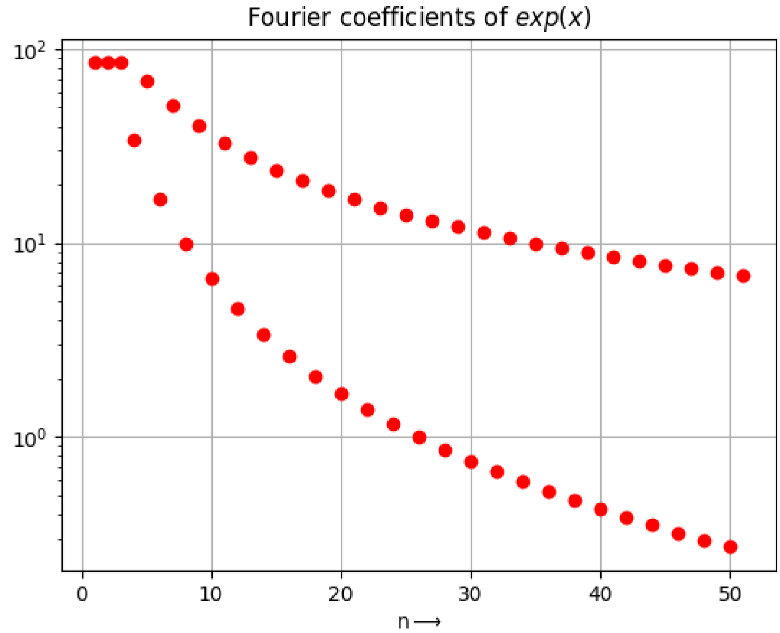

Figure 5: Coefficients for $exp(x)$ (semilogy)

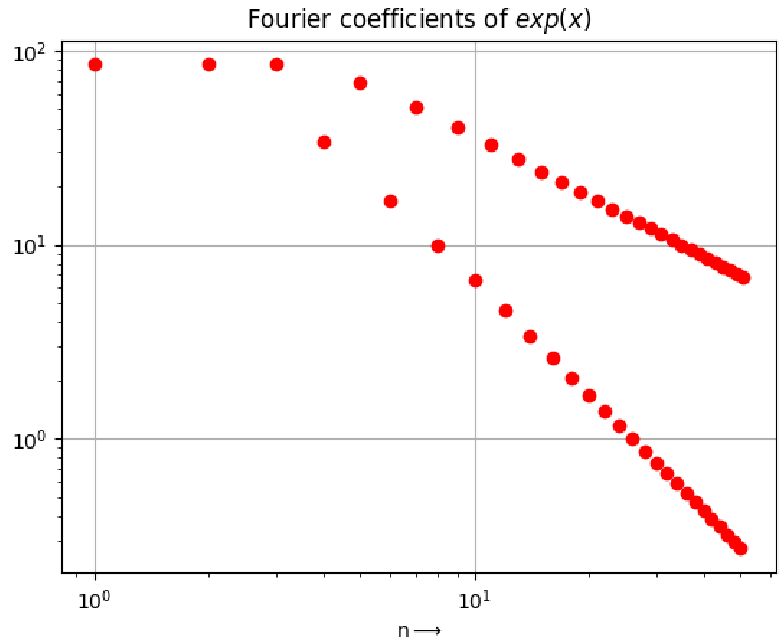

Figure 6: Coefficients for $exp(x)$ (loglog)

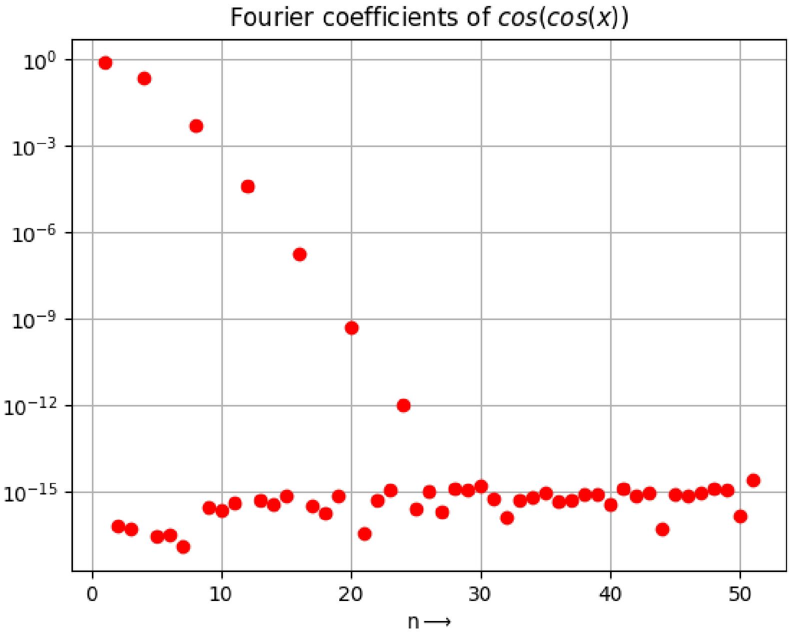

Figure 7: Coefficients for $cos(cos(x))$ (semilogy)

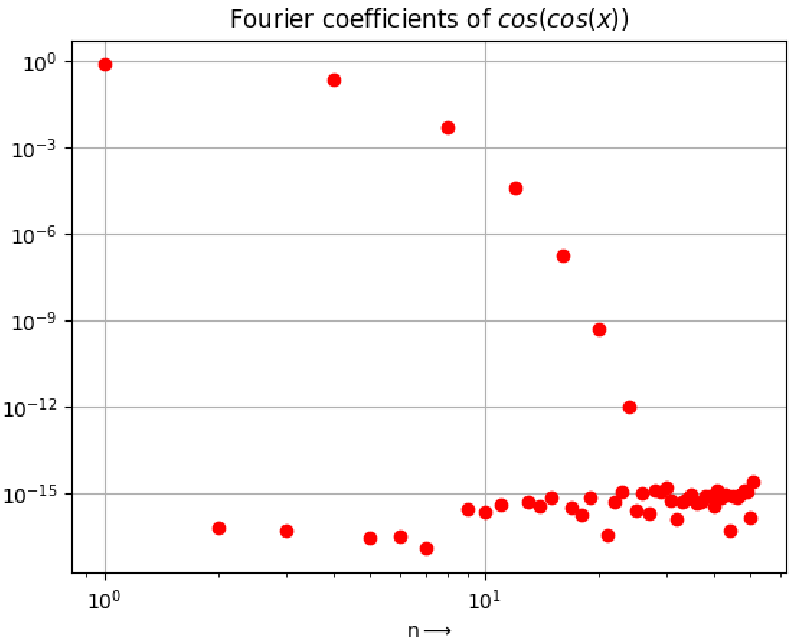

Figure 8: Coefficients for $cos(cos(x))$ (loglog)

2.4 Least squares approach

A finite set of Fourier coeffcients can be found using the **Least squares approximation** by looking at this as a matrix problem. 400 distinct x values over the interval $[0, 2\pi)$ are considered in this problem. The number of x values can be increased to get better approximations. The resultant matrix equation is

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

Consider the matrix to be A, the coefficients vector be c and the function values vector be b. Then the matrix equation is

$$Ac = b$$

We can use `lstsq` function from `scipy.linalg` to approximate the vector c. The coefficients for $exp(x)$ and $cos(cos(x))$ can be approximated as shown below

```
x = np.linspace(0, 2*pi, 401)
x = x[:-1]
b1 = e(x)
b2 = coscos(x)
A = np.zeros((400, 51))
A[:,0] = 1
for k in range(1, 26):
    A[:,2*k-1] = np.cos(k*x)
    A[:,2*k] = np.sin(k*x)
c1 = lstsq(A, b1)[0]
c2 = lstsq(A, b2)[0]
```

2.5 Plotting the approximated coefficients

The *best fit* coefficients for both the functions $exp(x)$ and $cos(cos(s))$ are computed above. They can be plotted along the coefficients compute by direct integration. To differentiate them, green circles are used.

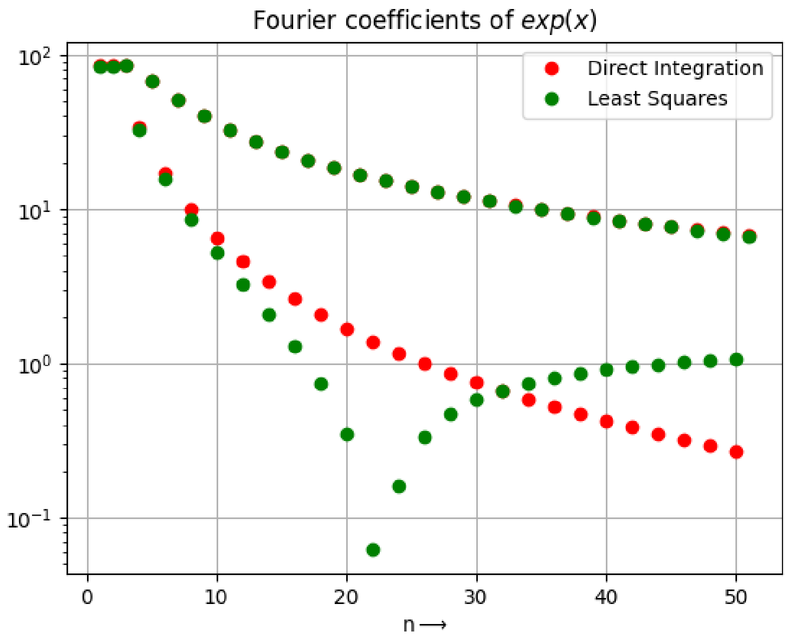

Figure 9: Coefficients for $exp(x)$ (semilogy)

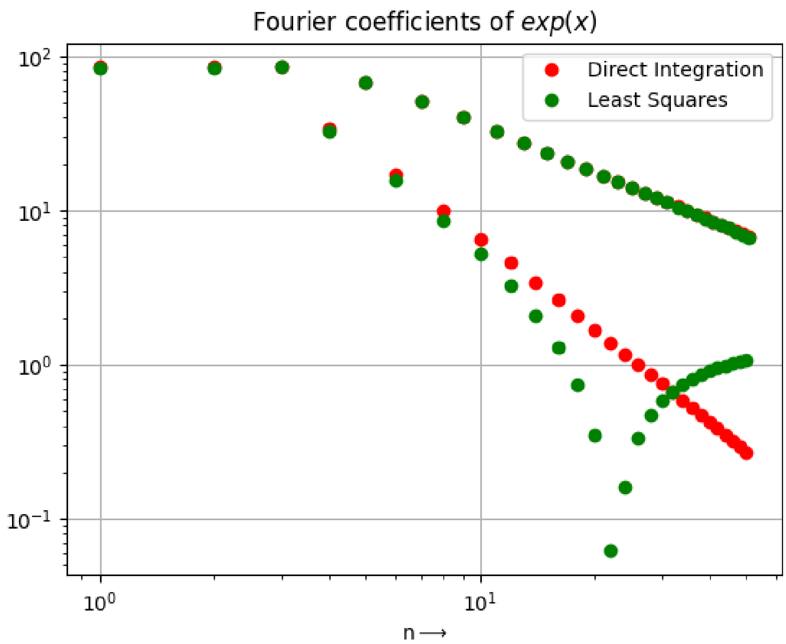

Figure 10: Coefficients for $exp(x)$ (loglog)

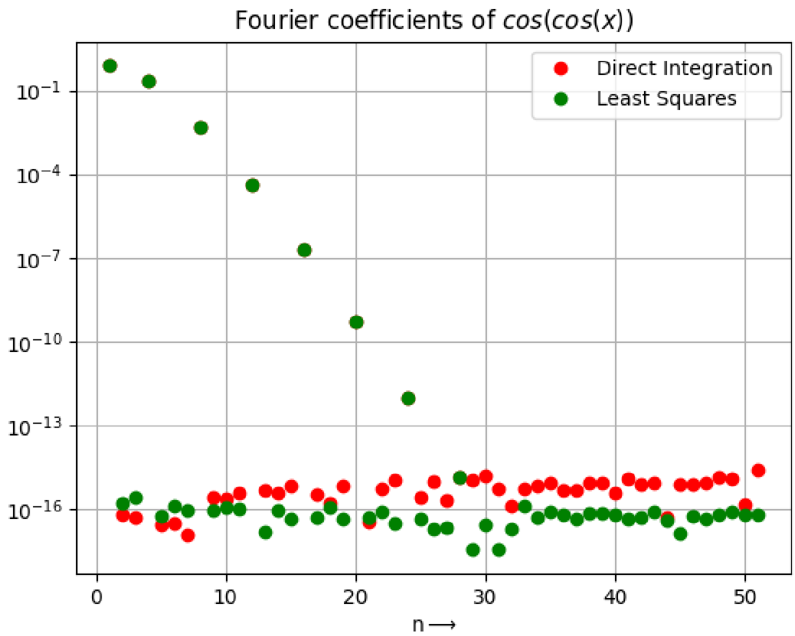

Figure 11: Coefficients for $cos(cos(x))$ (semilogy)

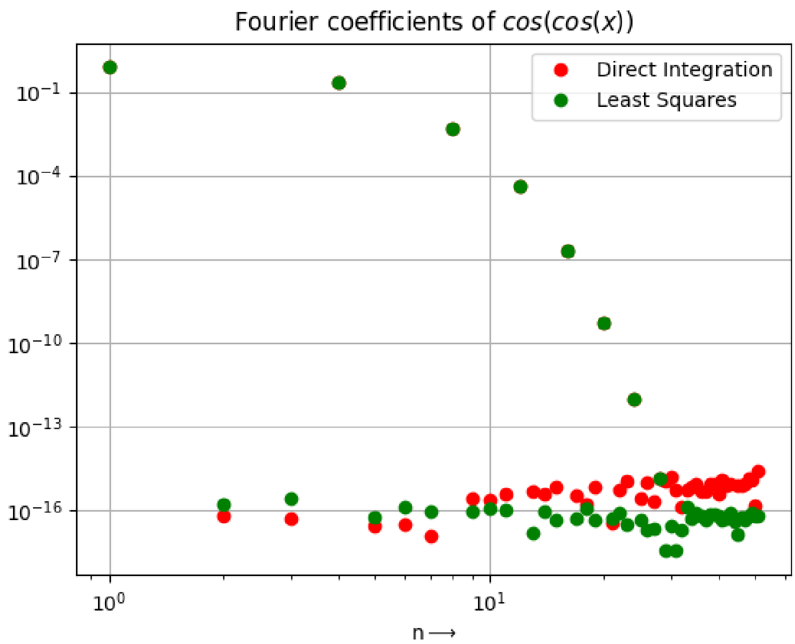

Figure 12: Coefficients for $cos(cos(x))$ (loglog)

2.6 Comparing the coefficients

The results from direct integration and least squares approximation agree for $cos(cos(x))$, but there is significant deviation in coefficients of cos terms for $exp(x)$. To find the maximum deviation from actual values, `np.max` can be used.

```
dev_f1 = np.max(np.abs(c1 - coeffs_f1))
dev_f2 = np.max(np.abs(c2 - coeffs_f2))
```

Here, `c1, c2` are the approximated coefficients and `coeffs_f1, coeffs_f2` are the coefficients from direct integration.

- The maximum absolute deviation for $exp(x)$ is 1.3327308703353964
- The maximum absolute deviation for $cos(cos(x))$ is 2.6566469738662125e-15

2.7 Plotting the approximated functions

The function values for the approxiamted coefficients can be found from the multiplication of matrix A and vector c. This will the function values for x over the interval $[0, 2\pi)$. For matrix multiplication, `np.dot` function can be used. The plots of the functions are given below.
High deviation is seen in case of $exp(x)$, but almost no deviation is seen for $cos(cos(x))$. This happens because 2π-periodic extension of $exp(x)$ has a finite discontinuity and the Fourier series deviates from the actual function at the point of discontinuity. This is called *Gibbs phenomenon*. This deviation seen in the plot has very little to do with error in least squares approximation as similar deviation can be seen for direct integration coefficients as well. This does not happen for $cos(cos(s))$ as there is no discontinuity.

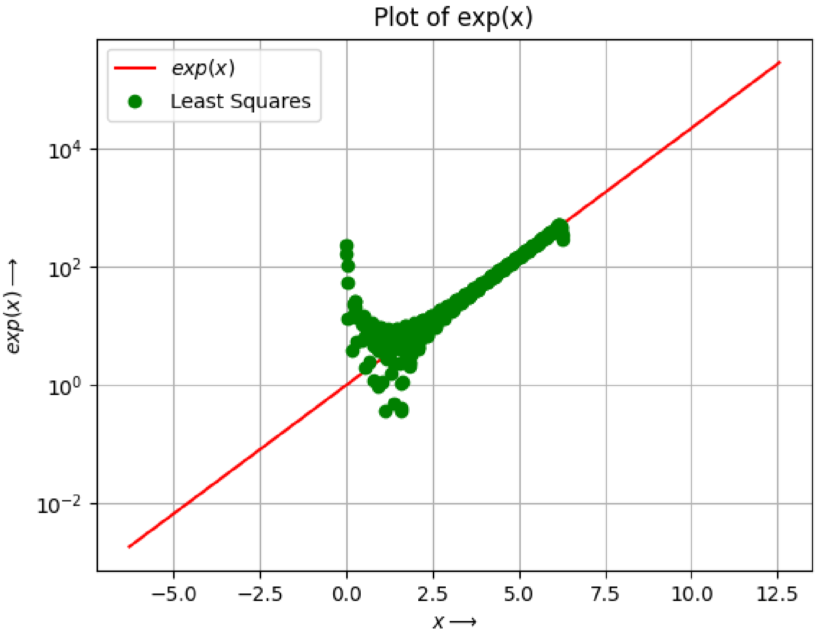

Figure 13: Approximated $exp(x)$

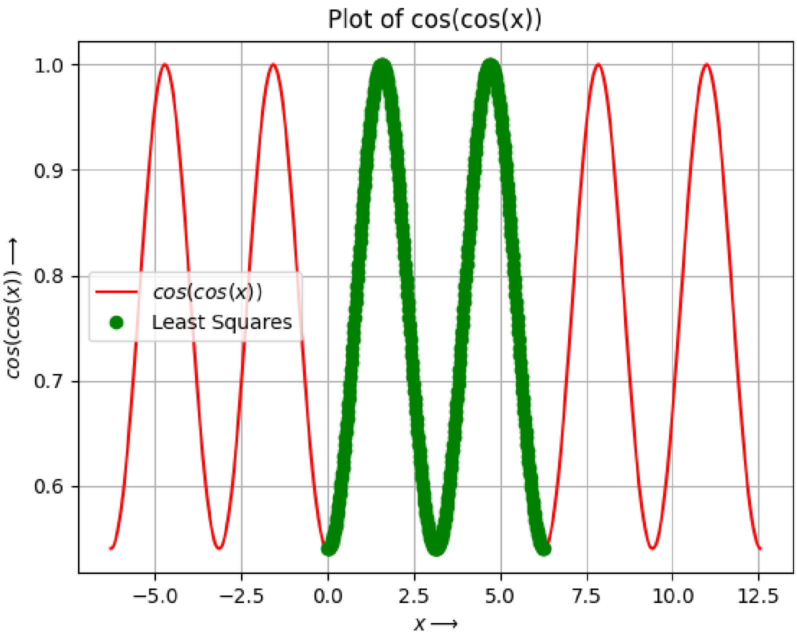

Figure 14: Approximated $cos(cos(x))$

3 Conclusion

51 Fourier coefficients of $exp(x)$ and $cos(cos(x))$ are computed by integration and their variation with n is observed. It was seen that even functions have sine-coefficients as 0 in Fourier series. The coeffcients are also approximated using *least squares method* and the deviation from integration coefficients was observed. The peculiar deviation of Fourier series from actual function after jump discontinuity, called *Gibbs phenomenon*, was observed.