Assignment 4:
Fourier Approximations

Saurav Sachin Kale, EE19B141

March 10, 2021

1 Aim

- Study the fitting of functions using the fourier series, given by:

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

- Observe and compare the coefficients generated by direct integration, as well as by the method of least squares.
- Observe the variation of absolute error between the coefficients calculated using both of these methods.

2 Procedure

- We know that the fourier coefficients $a_n$and $b_n$are given by

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

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

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

- For the first method of direct integration, we implement the above formulae by integrating two functions $u(x,k) = f(x)cos(kx)$ and $v(x,k) = f(x)sin(kx)$ and integrating them for the various values of k using the Scipy function

```
scipy.integrate.quad(<function_to_be_integrated>, <lower_lim>, <upper_lim
    >, <extra_args>)
```

- Here, `<function_to_be_integrated>` is either $u(x,k)$ or $v(x,k)$. The limits of the integrals will be from $(0, 2\pi)$. In `<extra_args>`, we shall specify the value of k, since k is not the variable over which we are integrating.
- Using a loop, we can compute all the fourier coefficients $\{a_0, a_1, ..., a_{25}\}$ and $\{b_1, b_2, ..., b_n\}$.
- For the second menthod of least squares, we now define a vector x going from $(0, 2\pi)$ in 400 steps. Evaluating the function $f(x)$ at each x, we get another vector b. Now, for each $x_i \in x$:

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

- This expression can be written in matrix form as shown below:

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

$$Ac = b$$

- We can now use the method of least squares in order to compute the matrix c. We will use the following Scipy function for this:

  ```
  c = scipy.lstsq(A, b)
  ```

- Absolute error or deviation is obtained by subtracting the values of a_n and b_n calculated from direct integration from the values calculated by least squares and taking absolute value.

3 Results

1. Plotting the functions in interval $[-2\pi, 4\pi)$. The expected fourier series plots will be repeated segments of the function in the interval $x \in [0, 2\pi)$, since we have calculated the fourier series assuming the function is periodic with period 2π.

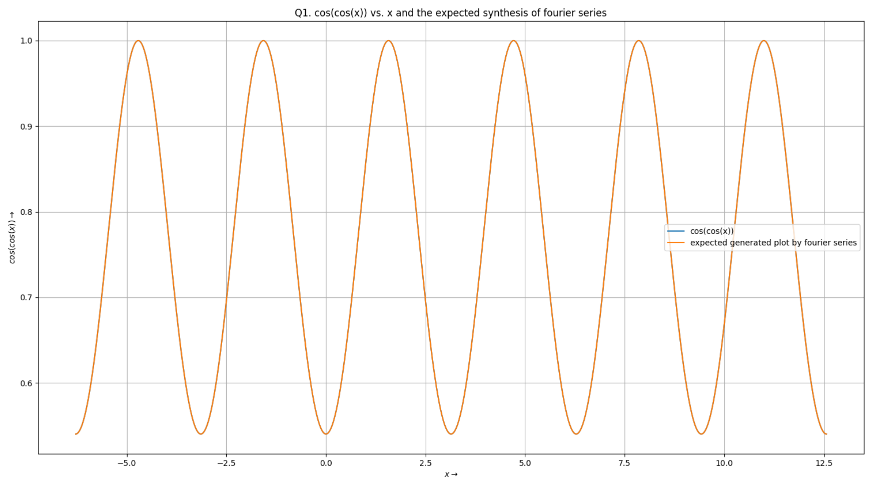

Figure 1: $cos(cos(x))$ vs. x and the expected fourier series plot

Note that the above function and the expected fourier plot are identical. This is because function itself is 2π periodic.

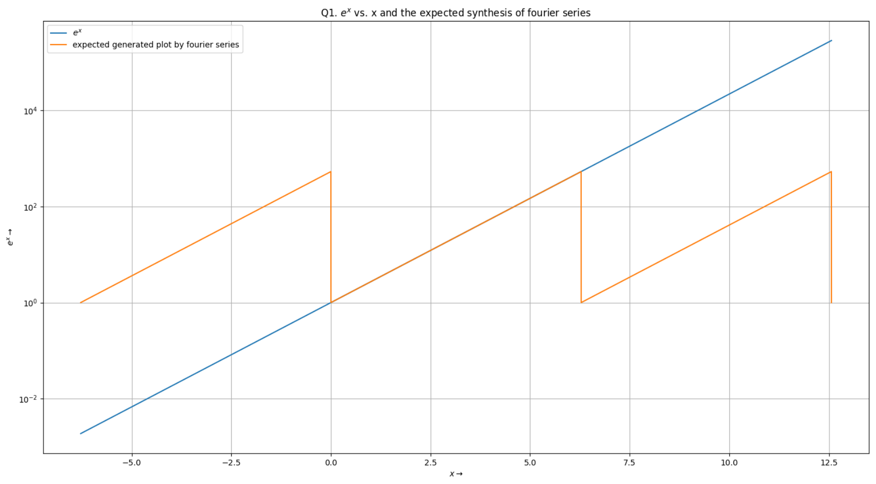

Figure 2: e^x vs. x and the expected fourier series plot in semilogy scale

This function is not periodic, therefore its expected fourier transform is not identical to the function.

2. In order to compute the fourier series we have created a class called Fourier, defined as shown below:

```
class Fourier:
        # class attributes
    f = None
    u = None
    v = None
    numCoeffs = 51
    cosineCoeffs = []
    sineCoeffs = []
    Fseries = None

        # constructor
    def __init__(self, _f, _numCoeffs = 51):
        self.f = _f
        self.numCoeffs = _numCoeffs
        self.u = lambda x, k : _f(x)*np.cos(x*k)
        self.v = lambda x, k : _f(x)*np.sin(x*k)
        self.cosineCoeffs.clear()
        self.sineCoeffs.clear()

        # calculating a_i
    def calcUk(self):
       print("Cosine")
        for k in range((self.numCoeffs + 1) // 2):
            fourierCoeff = (1/np.pi) * float(integrate.quad(self.u, 0, 2*
                np.pi, args=(k))[0])
            self.cosineCoeffs.append(fourierCoeff)
        print(self.cosineCoeffs)

        # calculating b_i
    def calcVk(self):
        print("Sine")
        for k in range(1, (self.numCoeffs + 1) // 2):
            fourierCoeff = (1/np.pi) * float(integrate.quad(self.v, 0, 2*
                np.pi, args=(k))[0])
            self.sineCoeffs.append(fourierCoeff)
        print(self.sineCoeffs)
```

The coefficients are stored in the two arrays, namely `cosineCoeffs` and `sineCoeffs`. The functions `calcUk` and `calcVk` calculate these values and populate these arrays with the values.

3. Plotting the magnitudes of the coefficients gives us the following plots. Note that the coefficients are plotted in this order:

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

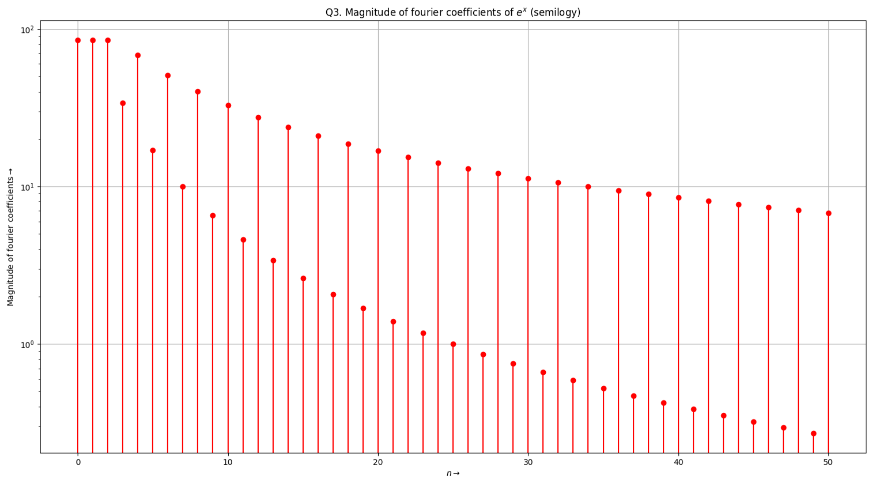

Figure 3: Magnitude of fourier coefficients of e^x in semilogy scale

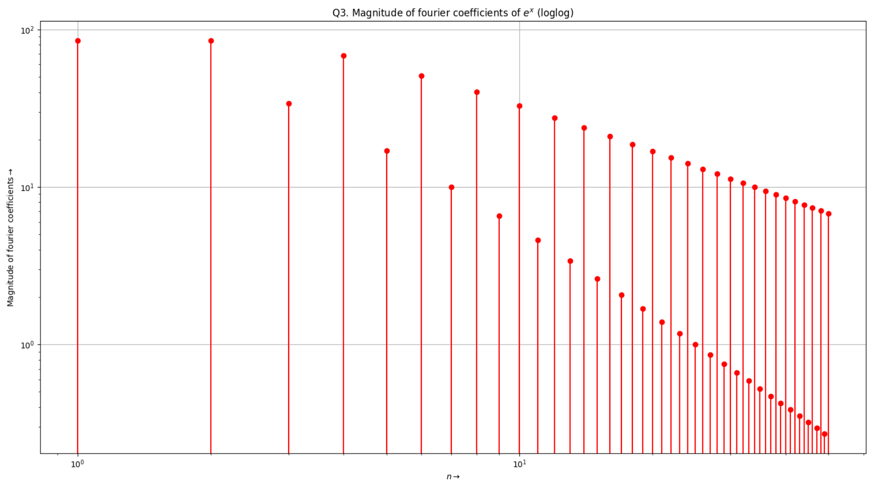

Figure 4: Magnitude of fourier coefficients of e^x in loglog scale

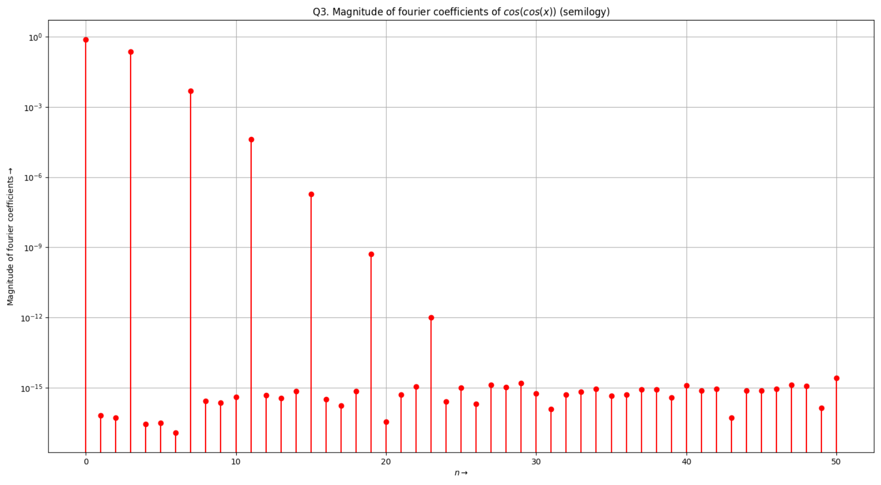

Figure 5: Magnitude of fourier coefficients of $cos(cos(x))$ in semilogy scale

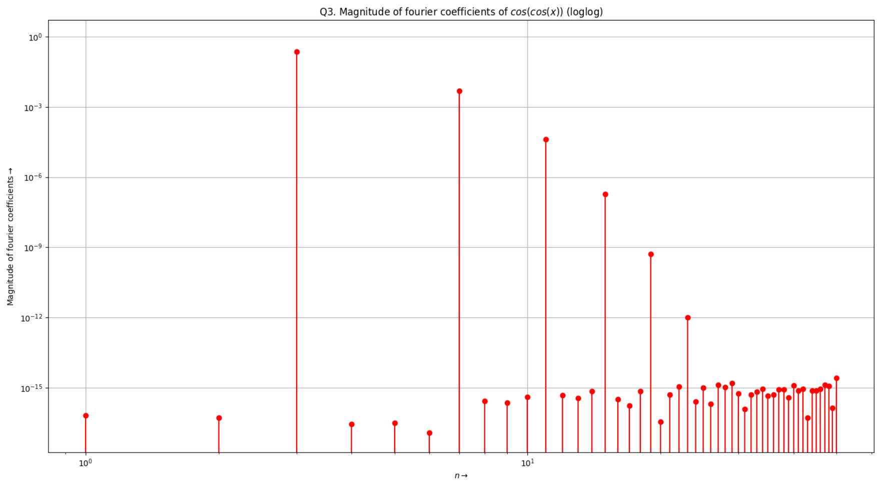

Figure 6: Magnitude of fourier coefficients of $cos(cos(x))$ in loglog scale

(a) Coefficients are zero for $cos(cos(x))$ since it is an even function.

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

We know that an even function has a cosine fourier series. Therefore all the sine coefficients are zero (here they are almost negligible but nonzero, most likely due to rounding errors)

(b) The closer a function is to a sinusoid, the lower the number of terms having appreciable contribution to the series. $cos(cos(x))$ is very close to a sinusoid, while e^x is not even periodic, let alone close to a sinusoid. Therefore the coefficients decay quickly for $cos(cos(x))$ and slowly for e^x

(c) If we attempt to calculate the fourier series coefficient for general case:

$$a_n = \frac{1}{\pi}\int\limits_0^{2\pi} e^x cos(nx)dx$$

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

taking log on both sides

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

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

We can perform similar analysis for b_n. This explains why the loglog plot of e^x coefficients is linear. Now let us look at $cos(cos(x))$:

$$a_n = \frac{1}{\pi}\int\limits_0^{2\pi} cos(cos(x))cos(nx)dx$$

$$\approx 2\pi cos(1) + \frac{1}{6}(2\pi)^3(sin(1) - n^2 cos(1)) + ...$$

taking log on both sides we get

$$log(a_n) \propto -n$$

Therefore semilogy plot is linear for $cos(cos(x))$.

4. Using `scipy.linalg.lstsq`, we determine the best fit coefficients satisfying $Ac = b$.

5. We now plot their magnitudes in green circles on top of figures (3), (4), (5), (6) and get the following plots:

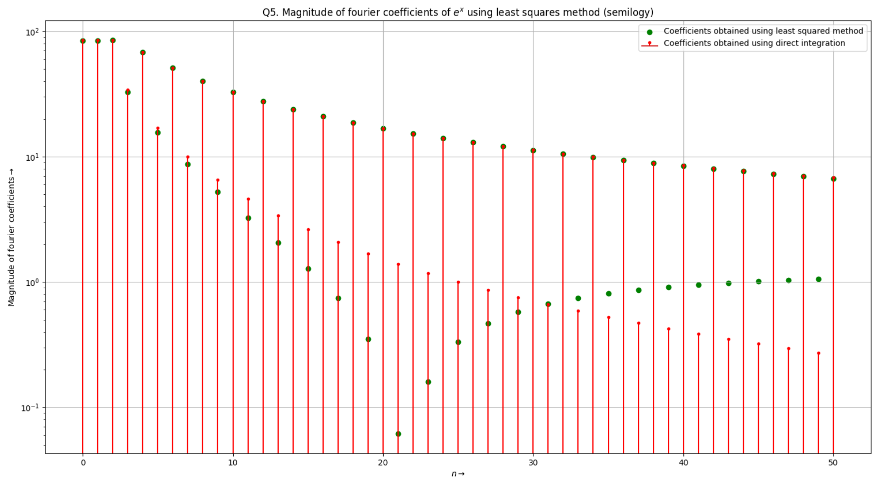

Figure 7: Fourier Coefficients of e^x calculated by direct integration and method of least squares on semilogy scale

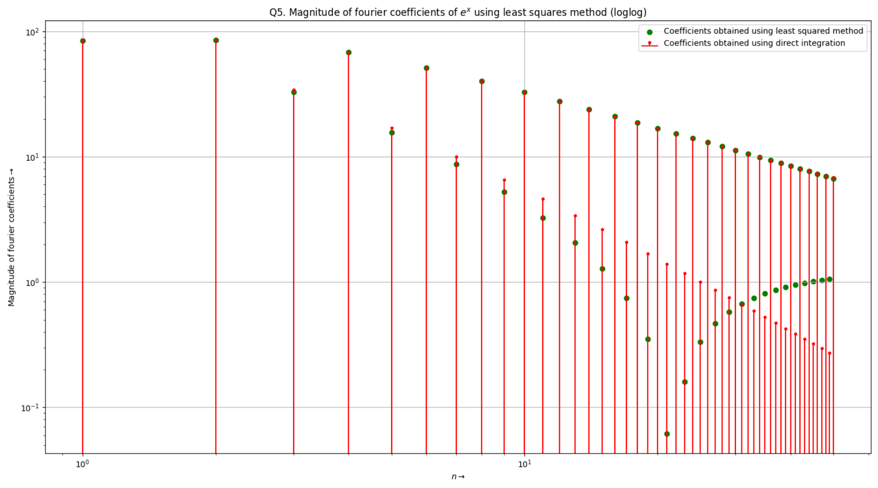

Figure 8: Fourier Coefficients of e^x calculated by direct integration and method of least squares on loglog scale

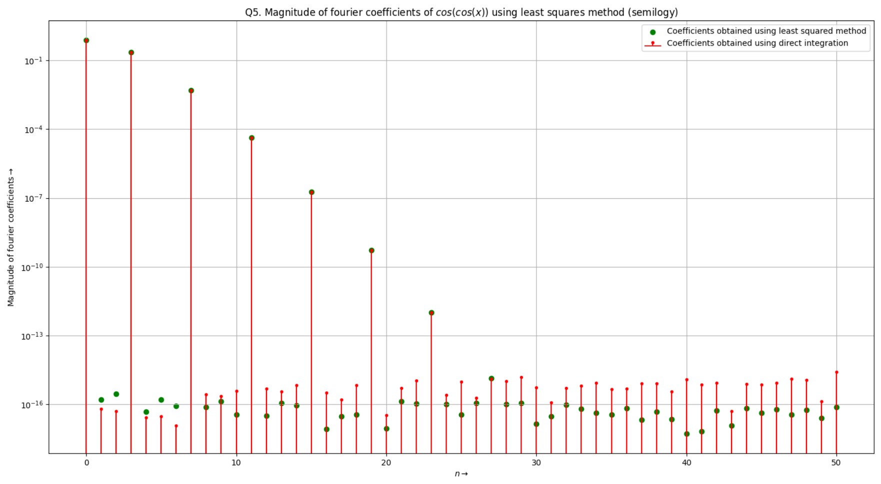

Figure 9: Fourier Coefficients of $cos(cos(x))$ calculated by direct integration and method of least squares on semilogy scale

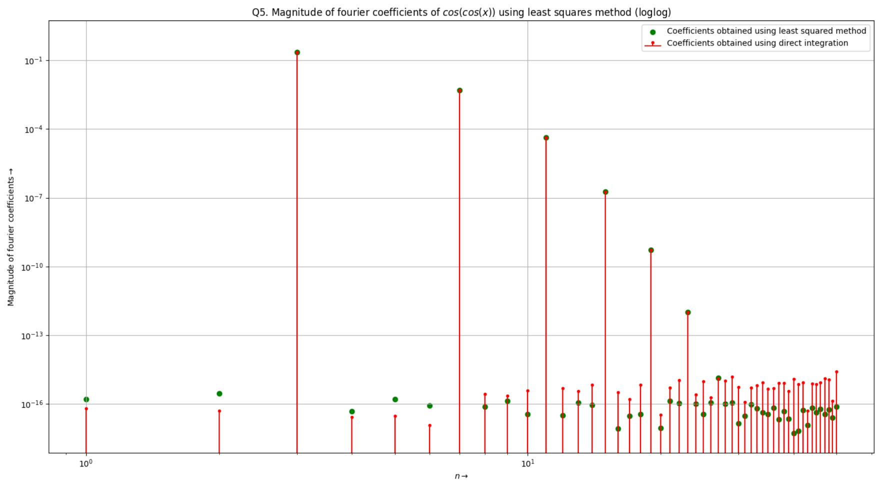

Figure 10: Fourier Coefficients of $cos(cos(x))$ calculated by direct integration and method of least squares on loglog scale

6. The answers by least squares matches quite closely for the $cos(cos(x))$ case, but has significant deviations for e^x. The appreciable deviation for e^x is expected, because the function, when made periodic by replicating the portion from $[0, 2\pi)$ contains discontinuities. Due to Gibbs Phenomenon, the values of the obtained Fourier Series synthesis vary wildly after encountering a discontinuity. Therefore, the coefficients obtained from the method of least squares should deviate significantly from the real values (those obtained from method of least squares), since our b matrix contained the values of $f(x) = e^x$ for all $x_i \in x$, and not of the fourier synthesis.

We now find and plot the deviation, obtained by the absolute value of the difference between coefficients calculated by both methods.

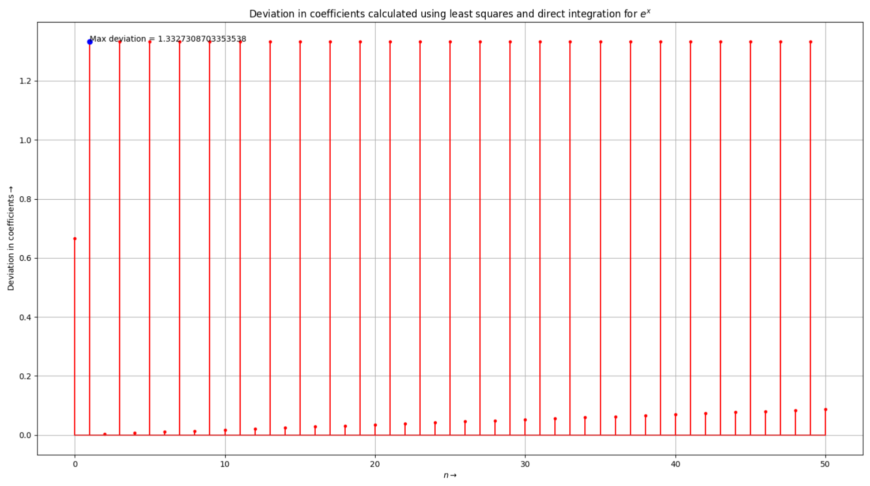

Figure 11: Absolute difference between fourier coefficients of e^x calculated using direct integration and method of least squares

From the graph, the largest deviation for coefficients of e^x is roughly 1.332, which is significant.

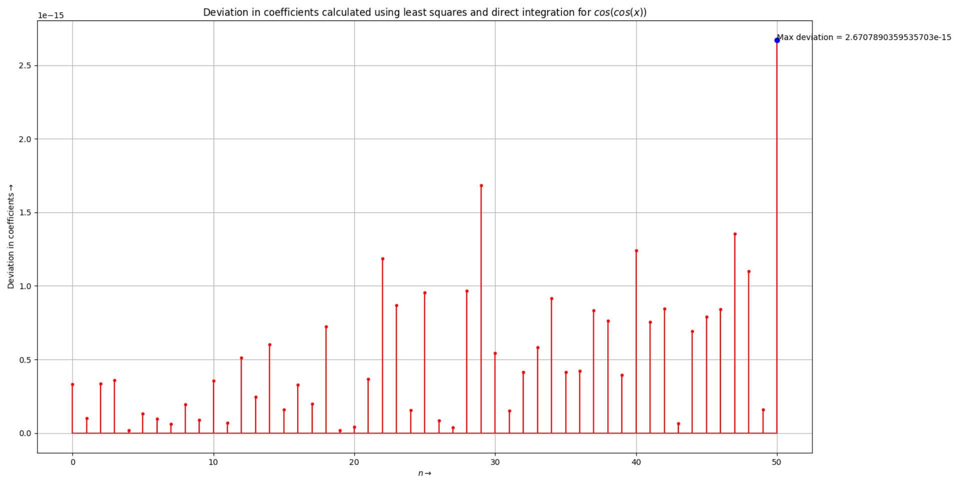

Figure 12: Absolute difference between fourier coefficients of $cos(cos(x))$ calculated using direct integration and method of least squares

From the graph, the maximum deviation for coefficients of $cos(cos(x))$ is 2.67×10^{-15}which is almost negligible.

This was done using vectors in the following manner:

```
deviation = [abs(i) for i in (f - c)]
```

f and c are vectors of size 51, containing the coefficients calculated by each of the two methods. A simple subtraction does this operation for all the 51 coefficients at once. Then we also take absolute value for each result.

7. We now perform the operation Ac to get b (the actual fourier series). We plot this vs. x with green circles on top of figure (1) and (2) to get the following plots:

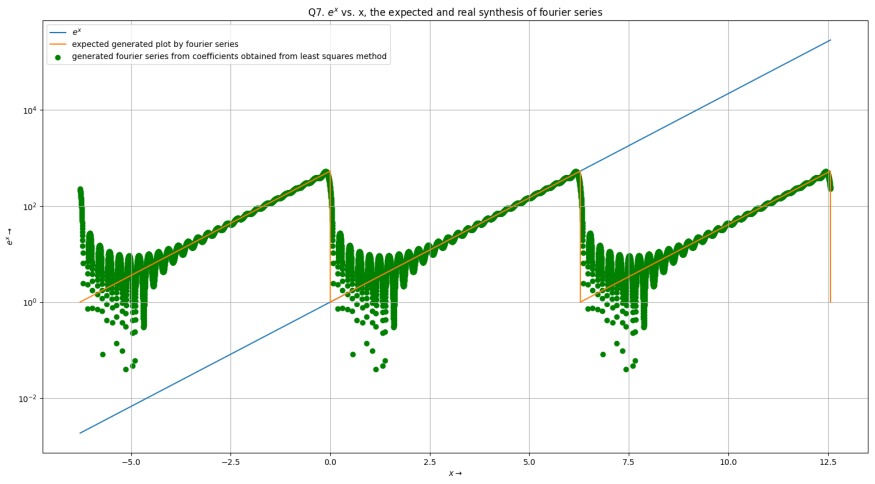

Figure 13: The actual fourier series along with the original function, and the expected fourier series for e^x

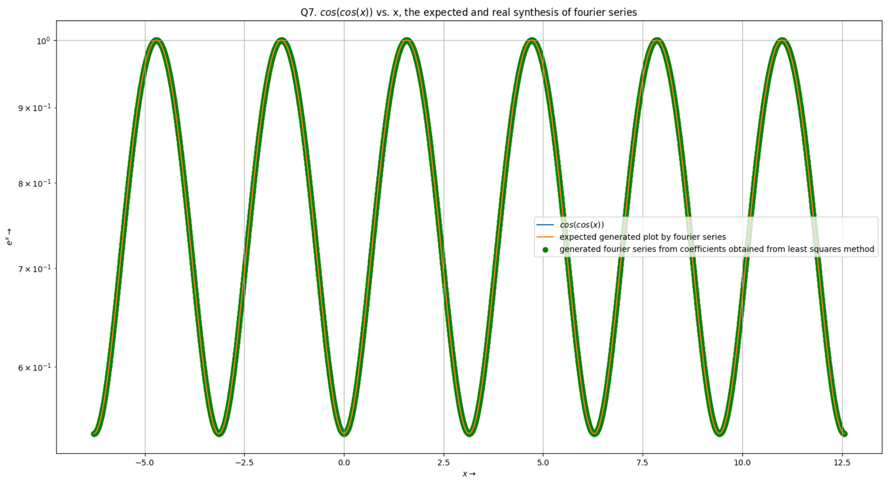

Figure 14: The actual fourier series along with the original function, and the expected fourier series for $cos(cos(x))$

The deviation in e^x fourier series is appreciable, because e^x, when considered as a periodic function with the segment from $[0, 2\pi)$ repeating itself, has discontinuities. These discontinuities result in the fourier series oscillating wildly after the discontinuity point, which causes these significant deviations. This is called the Gibbs Phenomenon. The Gibbs phenomenon is also related to the principle that the decay of the Fourier coefficients of a function is controlled by the smoothness of that function. Smooth functions will have very rapidly decaying Fourier coefficients, whereas discontinuous functions will have very slowly decaying Fourier coefficients.

4 Conclusions

- The fourier series coefficients can be calculated using two methods, from direct integration or by the method of least squares, and the results match quite well for continuous functions.
- The method of least squares gives significant deviations from the actual coefficients (those calculated by direct integration) in the case of discontinuous functions due to the fact that the fourier series representation deviates significantly from the function itself when close to a discontinuity, but the basis for calculation of least squares approach did not consider these deviations.
- The fourier series representation deviates significantly from the function itself when close to a discontinuity due to Gibbs Phenomenon.