Assignment No 4

Subhankar Chakraborty

February 26, 2019

1 Fourier Series

A Fourier series is an expansion of a periodic function $f(x)$ in terms of an infinite sum of sines and cosines. Fourier series make use of the orthogonality relationships of the sine and cosine functions. The computation and study of Fourier series is known as **harmonic analysis** and is extremely useful as a way to break up an arbitrary periodic function into a set of simple terms that can be plugged in, solved individually, and then recombined to obtain the solution to the original problem or an approximation to it to whatever accuracy is desired or practical.

1.1 The fourier series representation

The fourier series representation of a function f(x) is given by the equation

$$f(x) = a_0 + \sum_{n=0}^{\infty} a_n cos(nx) + b_n sin(nx) \tag{1}$$

1.2 Calculating the fourier coefficients

The fourier coefficients can be calculated as below.

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

$$a_n = \frac{1}{\pi} \int_0^{2\pi} f(x)cos(nx)dx \tag{3}$$

$$b_n = \frac{1}{\pi} \int_0^{2\pi} f(x)sin(nx)dx \tag{4}$$

2 The given functions

The functions to be analysed in this assignment are e^x and $Cos(Cos(x))$.Figures 1 and 2 show their plots in the interval $[-2\pi, 4\pi)$.

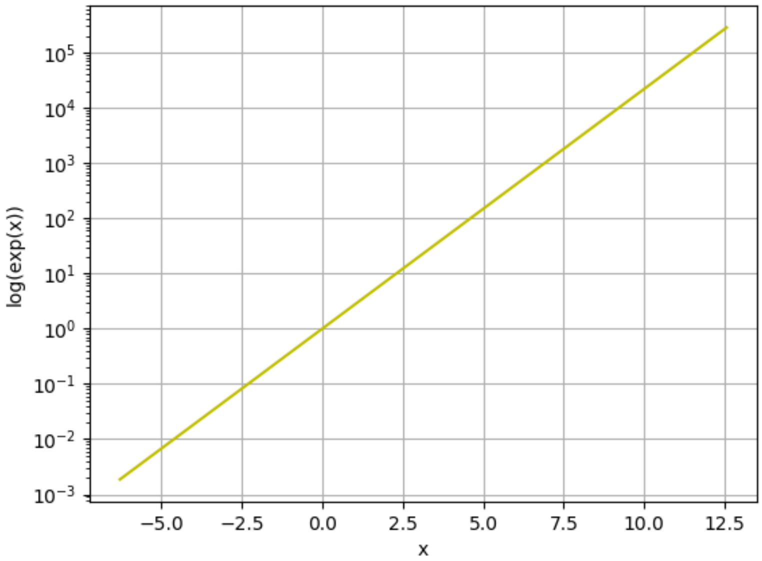

Figure 1: e^x

The following section of code calculates the values of e^x and $cos(cos(x))$ for a given x.

```
exp = lambda x:np.exp(x)
cos_cos = lambda x:np.cos(np.cos(x))
```

3 Integration over $[0, 2\pi)$

3.1 Calculating the coefficients

We obtain the first 51 fourier coefficients for the two functions mentioned.We first write functions which calculate $f(x)Sin(kx)$ and $f(x)Cos(kx)$.

```
a_vals_calc_exp = lambda x,k:exp(x)*np.cos(k*x)
b_vals_calc_exp = lambda x,k:exp(x)*np.sin(k*x)
```

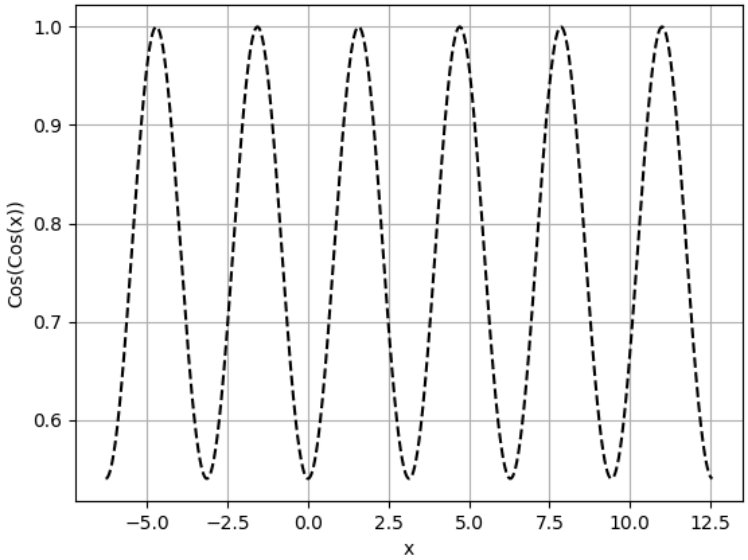

Figure 2: $cos(cos(x))$

The exact same logic is used for $Cos(Cos(x))$.To perform the integration, we use the **quad** function from the **scipy.integrate** module and run it within a *for loop* to get the values of a_k and b_k for different k.

```
a_values_exp =
np.asanyarray([(sp.integrate.quad(a_vals_calc_exp,0,2*pi,
args = k))[0]/pi for k in range(N)]).reshape(-1,1)
```

The code for calculating b_k values for e^x and the fourier coefficients of $Cos(Cos(x))$ is almost exactly similar and is hence, not shown.Note the the values of a_0 have to be further divided by 2 after this step. From the values obtained above, the vector of the form as desired in the PDF is created using simple *append* functionality of python lists.

```
for k in range(int(N/2)-1):
    cos_cos_to_plot.append(a_values_cos_cos[k+1])
    cos_cos_to_plot.append(b_values_cos_cos[k])
```

3.1.1 Plotting the values

After creating the vector above , we plot it with respect to n. First we plot a *loglog* plot and then a *semilogy* plot.

```
plt.loglog(np.arange(N-1).reshape(-1,1), exp_to_plot, 'ro')
plt.semilogy(np.arange(N-1).reshape(-1,1), exp_to_plot, 'ro')
```

The same procedure is repeated for the $cos(cos(x))$ case.Figures 3,4,5 and 6 show the plots.

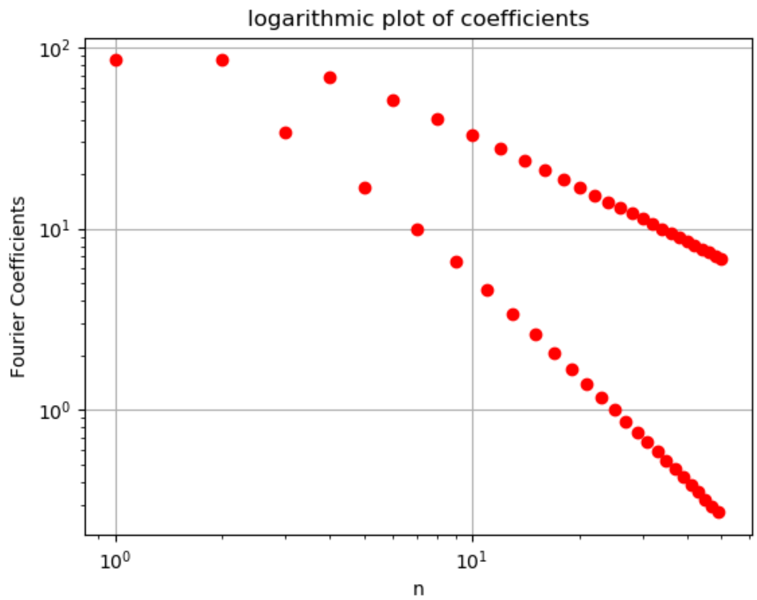

Figure 3: Coefficients of e^x

4 Least square method

Instead of using integration, we use a least square approximation as used in the previous assignment.The equation we want is

$$a_0 + \sum_{n=0}^{25} a_n cos(nx_i) + b_n sin(nx_i) \approx f(x_i) \tag{5}$$

4.1 Creating the matrix

We create a matrix A and use the *lstsq* function from scipy to approximate the coefficients for a given function.In equation form it is

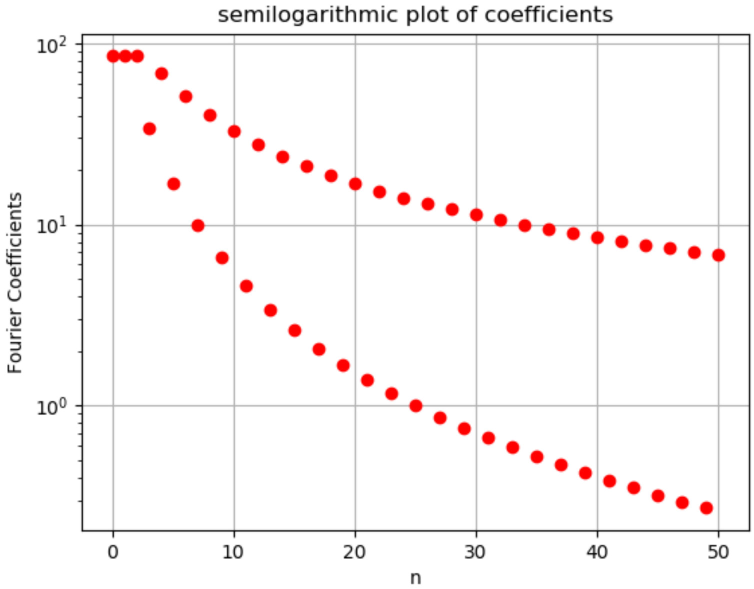

Figure 4: Coefficients of e^x

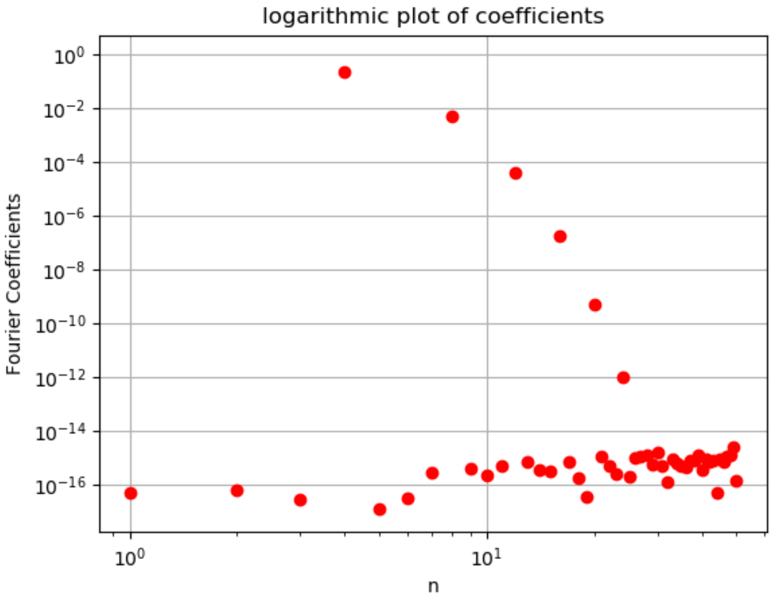

Figure 5: Coefficients of $Cos(Cos(x))$

$$Ac = b \tag{6}$$

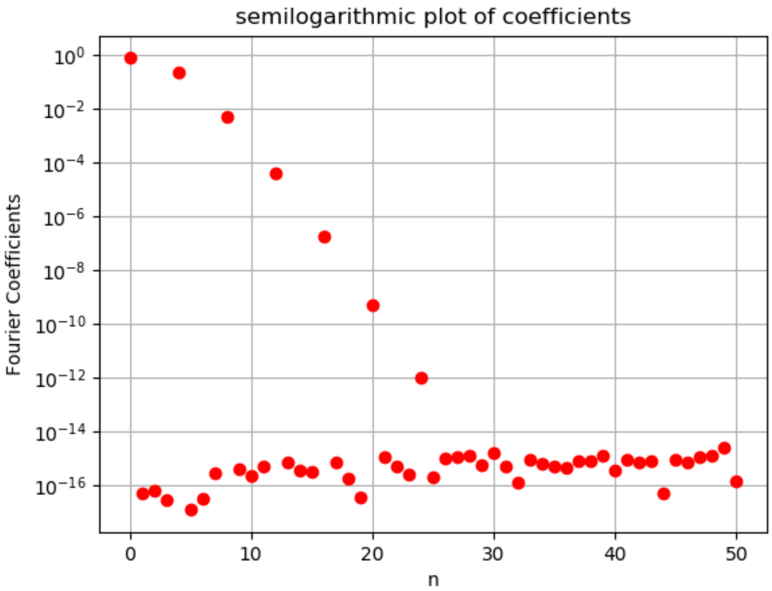

Figure 6: Coefficients of $Cos(Cos(x))$

where c is the set of coefficients we want and b is a vector containing values of $f(x)$ for different values of x.The following piece of code will make it clear.

```
#creating the matrix A
x = np.linspace(0,2*pi, 400)
A = np.zeros(rows*columns).reshape(rows, columns)
A[:,0] = 1
for k in range(1,int(N/2)):
    A[:, 2*k-1] = np.cos(k*x)
    A[:,2*k] = np.sin(k*x)

#lstsq approximation for exp(x)
b_exp = exp(x)
c_exp = np.asanyarray(sp.linalg.lstsq(A,b_exp)[0]).reshape(-1,1)
```

4.2 Plotting the coefficients

The coefficients generated from this method are plotted in figures 7,8,9 and 10.**Note that all coefficients are plotted for their absolute values only.**

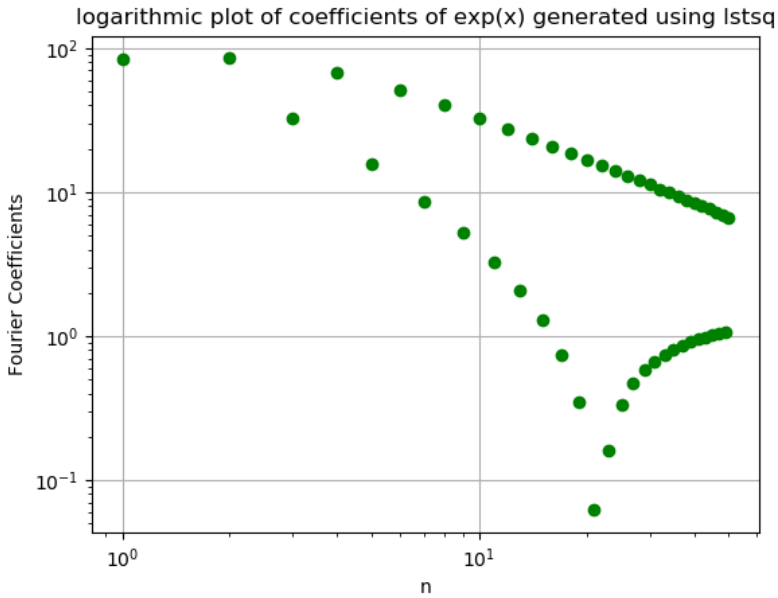

Figure 7: Coefficients of e^x

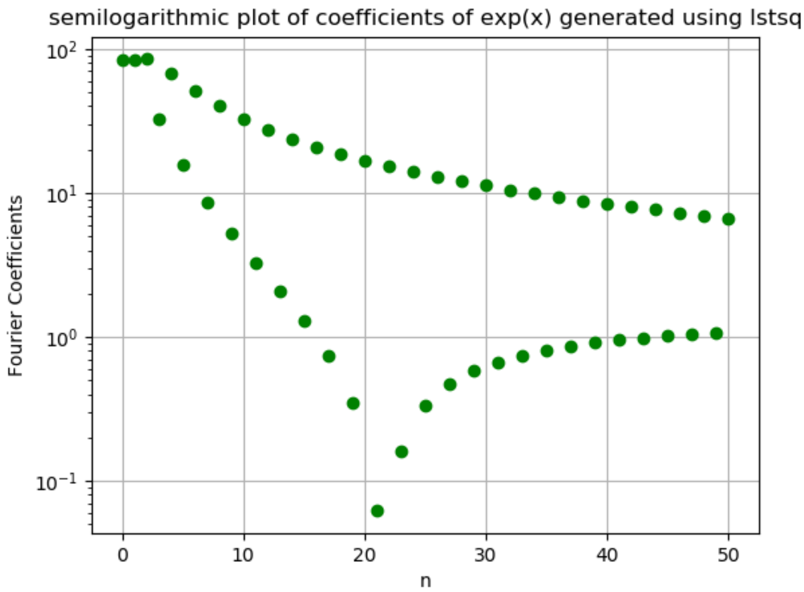

Figure 8: Coefficients of e^x

5 Generating the functions from the coefficients

We use the coefficients to generate values of e^x and $Cos(Cos(x))$ and plot it together with the original values to get an estimate of how close our ap-

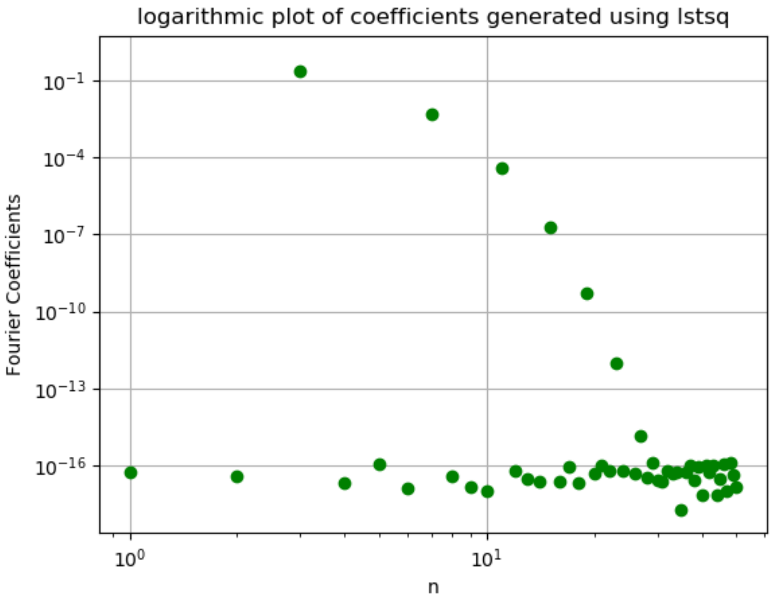

Figure 9: Coefficients of $Cos(Cos(x))$

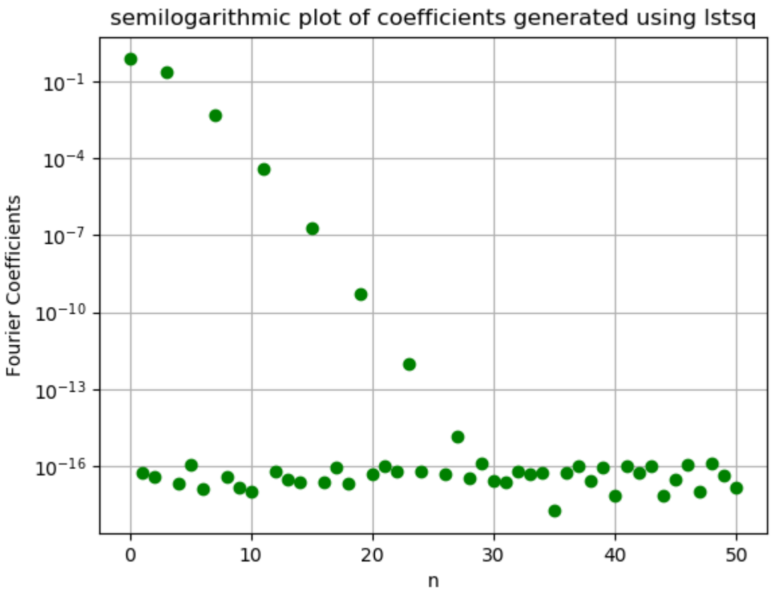

Figure 10: Coefficients of $Cos(Cos(x))$

proximation is.The code for one of the cases is given here.Figures 11 and 12 show the generated values versus the true values.

```
plt.plot(x[::10], cos_cos(x[::10]), 'k-o')
plt.stem(x[::10],np.dot(A,c_cos_cos)[::10], 'r' )
```

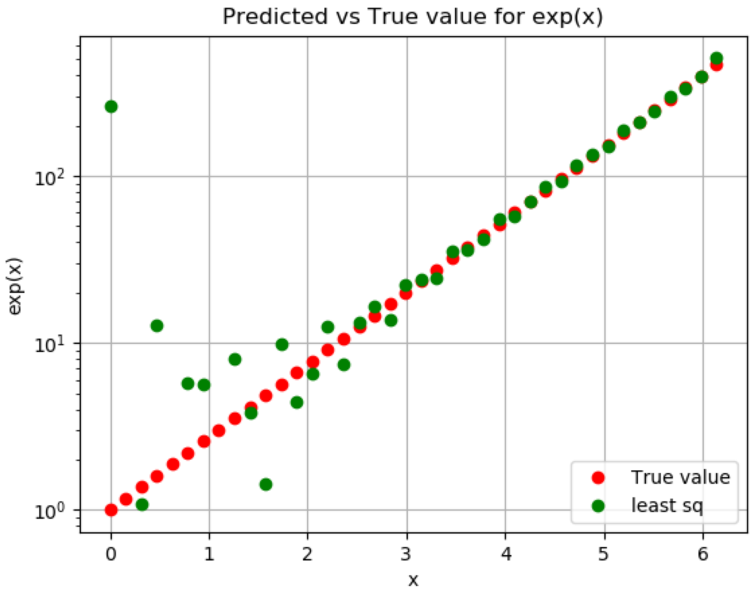

Figure 11: Predicted vs true values of e^x

6 Inferences

The following inferences can be drawn

- The b_n coefficients for $Cos(Cos(x))$ are almost zero.This is primarily due to two reasons.

  1. Since both $Cos(Cos(x))$ and $Sin(kx)$ are 2π periodic, we can choose any interval for the calculation of fourier coefficients.This means the interval $[-\pi, \pi)$ should also give exact same results provided no discontinuities at the boundaries.

  2. In this interval $Sin(kx)$ is an odd function of x and $Cos(Cos(x))$ is an even function. This means the integral value will be 0.We get values of the order 10^-15 because of limited machine precision.

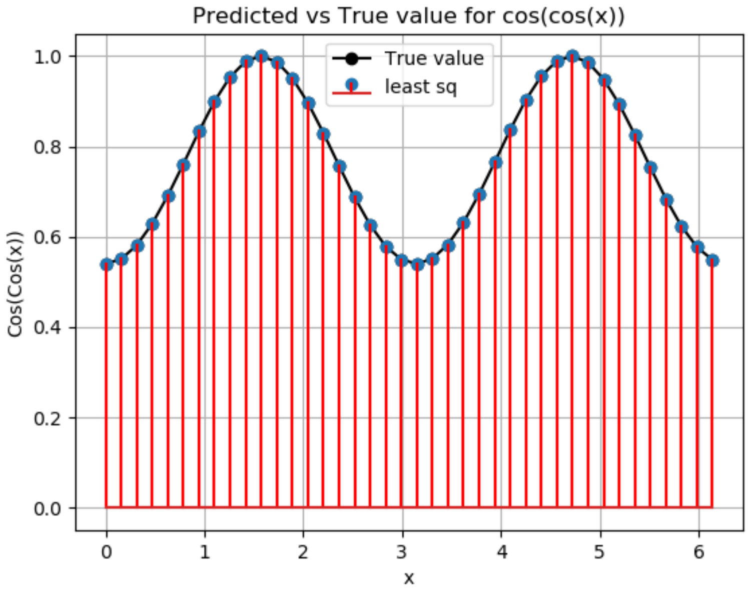

Figure 12: Predicted vs true values of $Cos(Cos(x))$

- The coefficients for e^x do not decay as quickly as in the case of $Cos(Cos(x))$ as the second function here is very similar in structure to a sinusoid with period π and will hence have contributions from a limited number of frequencies only. e^x on the other hand is very different from a sinusoid and is a rapidly increasing function. So you do need contribution from the higher frequency coefficients also ,especially near the boundaries where there is a discontinuity.

- The coefficients of $Cos(Cos(x)$ are exponential with respect to n.So the semilog gives a linera plot. Whereas the coefficients for e^x,

$$|\, a_n \,|, |\, b_n \,| \quad \propto \quad \frac{1}{n^2+1}$$

Thus, for large n ,

$$log(|\, a_n \,|), log(|\, b_n \,|) \quad \propto \quad \log(\frac{1}{n^2+1}) \approx -2\log n$$

So we get a linear plot in log scale.

- The deviation in the coefficients obtained from the two methods is much lesser in the case of $Cos(Cos(x))$(maximum is of the order 10^-15) whereas the deviation in coefficients of e^x is significantly larger. This should have been expected.

  1. The fourier representation using finite number of coefficients is the projection of a vector from an infinite dimensional space with all frequencies to a finite dimensional space. This tends to the initial vector as the number of terms tend to infinity

  2. The least square is used in the case where we have more equations than variables and to get the curve that gives the least mean squared error. This means that there will be deviations from the fourier series when there are discontinuities at boundaries.
  3. In the $Cos(Cos(x))$ case, we do not have any discontinuity at the boundaries. So, the coefficients match very nicely. On the other hand for the e^x case, there is a discontinuity of $e^{2\pi} - 1$ which means we see quite a bit of deviation.

- There are primarily two reasons for almost perfect agreement in one case and wide deviations in the other.

  1. $Cos(Cos(x))$ is very close to a 2π periodic sinusoid and hence, a few number of frequencies give an almost exact approximation. Plus, the fouries series converges to the the average of the LHS and RHS near a discontinuity,and since there is no discontinuity in this case, we get an almost perfect match.

  2. e^x increases from 1 to over 500 in an interval of just 2π. So we need much more high frequency terms for an accurate representation. Plus, the discontinuity at end points mean it should theoretically converge to the average value of 1 and $e^{2\pi}$ near 0, and this means we get a big deviation there.