Assignment 4 - Fourier Approximations

Aravint Annamalai, EE18B125

February 25, 2020

1 Abstract

In this assignment, I am going to deal with Fourier approximations of a periodic signal. According to this, any periodic signal can be represented as a linear combination of sines and cosines. I am going to analyse the Fourier representation of two signals: exp(x), which is not periodic, and cos(cos(x)), which is periodic. The analysis is done over a period from 0 to 2π. The computed Fourier coefficients is compared with the values obtained by the least square error approximation. The approximated functions are then compared with the actual function.

2 Introduction

Every periodic function can be written with the help of Fourier approximation in this form:

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

where the coefficients can be computed by

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

There is a built-in function `scipy.integrate.quad()` which computes the integral of any function within the specified limits, all passed as arguments. If the function to be integrated also requires some parameters to be passed, that can also be done as we will see here.

3 Task 1

Here, we define the functions `exp(x)` and `cos(cos(x))` for values ranging from -2π to 4π, make them 2π periodic and plot all the graphs.

3.1 Defining the functions

Firstly the libraries `numpy` and `matplotlib.pyplot` are imported for performing array related operations and graphing respectively.
Here, we will define 2 functions `exponential(x)` and `coscos(x)` which returns exp() and cos(cos()) values of the value/array passed to it as arguments. We use the built-in functions `np.exp()` and `np.cos()` to perform our required task.

3.2 Plotting graphs from -2π to 4π

The constant pi is declared and is given the approximate value of 3.1416. A list x is declared which stores 300 samples between -2π to 4π. `np.linspace()` is used for the purpose. Using `np.vectorize()` method, the above two functions defined are made to return lists/arrays when the argument passed is having that data type. The new functions are `exp1(x)` and `coscos1(x)`.

Then the graphs of these functions are plotted. Since `exp(x)` grows really fast, without any bounds, it is plotted as a semilog graph whereas the graph of `cos(cos(x))` is plotted using regular scale. Legends are given to the graphs appropriately.

3.3 Plotting 2π periodic functions

Using `np.linspace()` method, a list containing 400 samples between 0 and 2π is declared. The above functions `exp1(x)` and `coscos1(x)` are used to assign the respective function values to separate lists. These functions are then plotted from -2π to 4π by appropriately shifting the values on the x-axis. The resultat graphs are such that the functions are 2π periodic .

4 Task 2

Here, we are going to compute the first 51 Fourier series coefficients of the two functions `exp(x)` and `cos(cos(x))`.

4.1 Defining the functions to be integrated

We define two functions `u()` and `v()` as follows:

```
def u(x, func, k):
    u1 = func(x) * np.cos(k*x)
    return u1

def v(x, func, k):
    v1 = func(x) * np.sin(k*x)
    return v1
```

As we can see in (2) above, these are the functions to be integrated to compute the Fourier series coefficients.

4.2 Computing Fourier series coefficients

In order to perform the integrals, the function `scipy.integrate.quad()` is imported. Lists to store the Fourier series coefficients are declared and initialised with null values. Then, the integrals in (2) are computed and stored in the respective arrays. The technique for computation is depicted as an example below:

```
a_exp[i] = (1/pi) * quad(u, 0, (2*pi), args = (exp1,i))[0]
```

Like the above example, the lists declared in order to store the coefficients are populated.

5 Task 3

Here, I am combining the Fourier series coefficients commputed in the previous task into a single list for each function and I am plotting them in various scales.

5.1 Storing the Fourier series coefficients

First I am going to store the values of the coefficients of `exp(x)` into a single list. For that, I have declared a list `pred.exp` of size 51, initially storing null values. The first element of this list is assigned as the first value of the `a.exp` list computed in the previous task. Then, the subsequent elements are asiigned to subsequent elements of `a.exp` and `b.exp` values in that order. All the values in this list are made positive in order to plot the graphs when the scale on the y-axis will be log scale. The same process is repeated for the coefficients for `cos(cos(x))` as well.

5.2 Plotting the Fourier series coefficients

I am plotting a scatter plot of the coefficients vs the number assigned to the coefficient, as was assigned to them in the previous subsection. First, the

graph is plotted in the loglog scale and then the graph between the same quantities are plotted in the semilogy scale as well. These graphs are given appropriate titles and are displayed. This is done for both the functions.

5.3 Answers to the questions

(a) If you did Q1 correctly, the b_n coefficients in the second case should be nearly zero. Why does this happen?
Since the function in the second case is `cos(cos(x))`, there is no sine harmonic in there, so the contribution from the sine part of Fourier coefficients, represented by b_n, are close to zero.

(b) In the first case, the coefficients do not decay as quickly as the coefficients for the second case. Why not?
The function in the second case is a cosine, and since Fouier series coefficients are computed in terms of cosines, there exists only a very few harmonics in the second case and the value of coefficients decay faster.

6 Tasks 4 and 5

Here, I am going to construct matrices so that Fourier coefficients could be computed using the least square error method `numpy.linalg.lstsq()`. The computed values are then compared with the values of the coefiicients obtained using the integrals done in Task 2.

6.1 Populating the matrices

A list x is declared which stores 400 samples between 0 and 2π. A list `b.exp` is initialised with the list returned after x is passed to the function `exp1(x)`. A matrix `A` of order 400*51 is declared and is initialised with zeros. The first column of A is assigned identically 1, and the remining columns are assigned as follows:

```
for k in range(1,26):
    A[:, (2*k -1)] = np.cos(k*x)
    A[:, (2*k)] = np.sin(k*x)
```

6.2 Estimating the coefficients and plotting them

The coefficients are estimated using the function `numpy.linalg.lstsq()`, by passing the matrices `A` and `b.exp` as parameters. The elements of the returned list `lstsq.exp` is then made positive for graphing, which may involve log scaling. Both the list of coefficients, one obtained by doing direct integration (`pred.exp`) and the other, obtained by the least square error

minimisation `lstsq.exp`). These are plotted as scatter plots and are plotted using different colours for ease of identification. The same process is repeated for the other function as well.

7 Task 6

In this task, the maximum error between `lstsq.exp` and `pred.exp` is computed using list comprehension and `max()` method. This process is repeated for `cos(cos(x))` as well.

```
err_exp = [abs(lstsq_exp[i] - pred_exp[i]) for i in range(51)]
print(max(err_exp))
```

8 Task 7

The actual value of the function can be obtained by multiplying the A matrix declared in Task 4 with the solution matrix. The multiplication is done using the in-built matrix multiplication method. The value obtained after using the solution matrix from the least square error method and the actual function valus computed in Task 1. This process is done for both the functions.

9 Observation

Here, it can be seen that for periodic functions(like `cos(cos(x))` which was done in this assignment), the Fourier representation gives an exact value for each point within the domain which was used for performing the Fourier analysis. Whereas, for non-periodic functions like `exp(x)`, the Fourier analysis does not provide an exact graph as the function itself, but provides a good enough approximation within the domain which was used to perform the Fourier analysis.