Assignment4 - EE2703

Vighnesh N, EE17B119

February 2019

1 Introduction - Fourier series

Periodic functions, in particular 2π-periodic functions can be represented as a Fourier Series, which is a sum of the Sinusoids of infinite harmonics of the fundamental frequency.

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

The Fourier coefficients a_n and b_n are given by

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

We wish to find the first twenty five Fourier Series coefficients for the functions $f(x) = e^x$ and $g(x) = cos(cos(x))$
We shall use two methods for this regard, first is getting each coefficient by integration, second is through least square matrix method.

2 Defining the functions

We perform the basic imports and define the basic functions to be integrated

```
#!/usr/bib/python3.5
from pylab import *
from scipy.integrate import quad
def ex(v):
    return exp(v) # Function returns e^x
def coscos(v):
    return cos(cos(v)) # Function returns cos(cos(x))
```

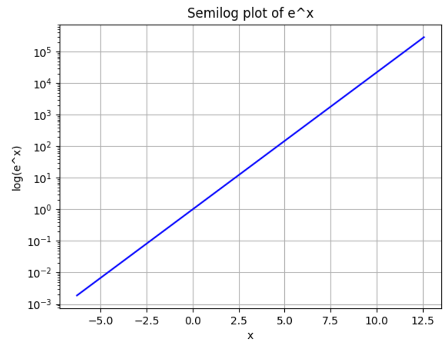

Figure 1: Plot of e^x from -2π to 4π

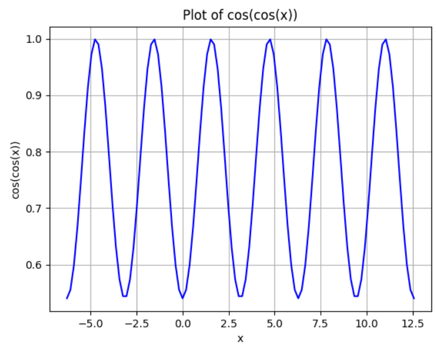

Figure 2: Plot of $cos(cos(x))$ from -2π to 4π

3 Visualizing plots

The code below gives the general template of how a function is plotted, taking into account the intricate details. This function will be called with appropriate parameters whenever a graph is to be plotted. Figure 1 and 2 are the plots of the actual functions

```
def plot_any(coeff, x, fig, _ylabel, _xlabel, _title, plot_type, color,
     l_dots='.'):
    '''
    This function plots any of the function/coefficient plots.
    It takes parameters for the x and y values, labels, titles,
        structure and color.
    '''
    figure(fig)
    title(_title)
    ylabel(_ylabel)
```

```
    xlabel(_xlabel)
    if plot_type == 1:
        plot(x,coeff,color+l_dots) #If we want regular plot
    elif plot_type == 2:
        semilogy(x,coeff,color+l_dots)
    else:
        loglog(x,coeff,color+l_dots)
```

4 Method 1 - Integration

The functions $u(x,k) = f(x)cos(kx)$ and $v(x) = f(x)sin(kx)$ are integrated to find the respective a_k and b_k. 51 coefficients are found, a_0 to a_25 and b_1 to b_25, approximating the Fourier expression to the sum of these first 50 Sinusoids.

The inbuilt scalar integral function 'quad' is used to determine the coefficients.

```
def u_x(x, k, func=ex):
    return func(x)*cos(k*x) # Passing the function to be multiplied by
        cos, will be integrated while finding coefficient
def v_x(x, k, func=ex):
    return func(x)*sin(k*x) # Passing the function to be multiplied by
        sin at that value, will be integrated.

def find_coeff(func=ex):
    a0 = quad(u_x,0,2*pi,args=(0,func))[0]/2/pi # code to compute the dc
        constant
    a = [quad(u_x,0,2*pi,args=(k,func))[0]/pi for k in range(1,26,1)] #
        code to compute the coefficients of cos
    b = [quad(v_x,0,2*pi,args=(k,func))[0]/pi for k in range(1,26,1)] #
        code to compute the coefficients of sin
    coeff = [a[i/2] if i%2 == 0 else b[(i-1)/2] for i in range(50)] #
        Code merging a and b
    coeff.insert(0,a0); a.insert(0,a0)
    return coeff, a, b
```

5 Method 2 - Least Squares Matrix Method

Following with the same approximation,

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

We convert this to a matrix problem

$$\begin{bmatrix} 1 & cos(x_1) & sin(x_1) & ... & cos(25x_1) & sin(25x_1) \\ 1 & cos(x_2) & sin(x_2) & ... & cos(25x_2) & sin(25x_2) \\ ... & ... & ... & ... & ... & ... \\ 1 & cos(x_400) & sin(x_400) & ... & cos(25x_400) & sin(25x_400) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ b_1 \\ ... \\ a_25 \\ b_25 \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ ... \\ f(x_400) \end{bmatrix}$$

Hence we have coverted this to the form

$$Ac = b$$

Where c are the required coefficients. By using the lstsq method of pylab, we can find the solutions.

```
def matrix(func=ex):
    x=linspace(0,2*pi,401)
    x=x[:-1] # drop last term to have a proper periodic integral
    b=func(x) # f has been written to take a vector
    A=zeros((400,51)) # allocate space for A
    A[:,0]=1 # col 1 is all ones
    for k in range(1,26):
        A[:,2*k-1]=cos(k*x) # cos(kx) column
        A[:,2*k]=sin(k*x) # sin(kx) column
    cl =lstsq(A,b)[0]    # the '[0]' is to pull out the best fit vector.
         lstsq returns a list.
    return cl, dot(A,cl), x
```

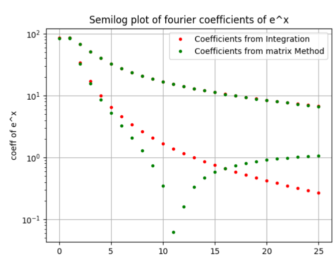

Figure 3: Plot of magnitude of first 51 Fourier coefficients of e^x on Semilog scale

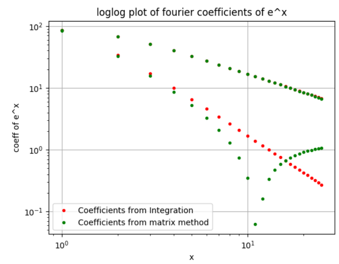

Figure 4: Plot of magnitude of first 51 Fourier coefficients of e^x on Loglog scale

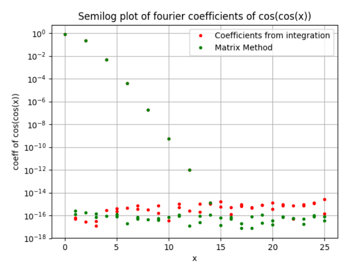

Figure 5: Plot of magnitude of first 51 Fourier coefficients of cos(cos(x)) on Semilog scale

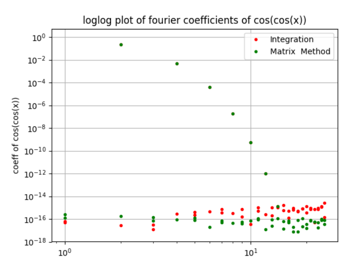

Figure 6: Plot of magnitude of first 51 Fourier coefficients of cos(cos(x)) on Loglog scale

This piece of code also finds the value of Ac for the just found c, hence evaluating the two functions at the range asked for in the question, 0 to 2π. The return statement contains the value of this. This is plotted along with the original curve as a comparison.

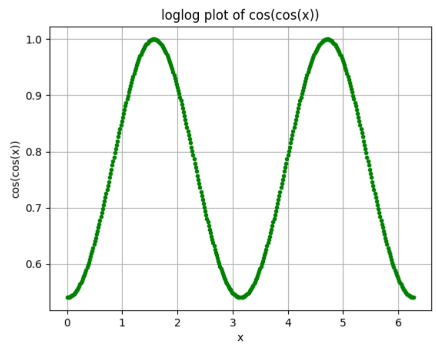

Figure 7: Plot of e^x using the found vector of Fourier coefficients

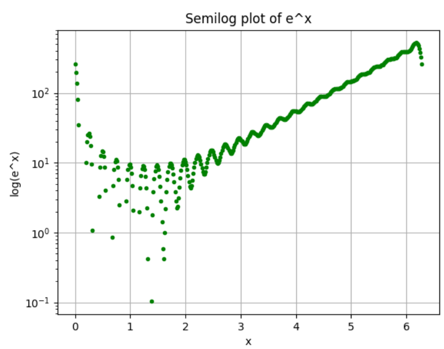

Figure 8: Plot of cos(cos(x)) using the found vector of Fourier coefficients

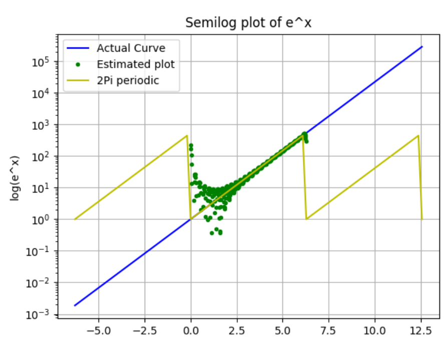

Figure 9: Plot of e^x using the found vector of Fourier coefficients from -2π to 4π

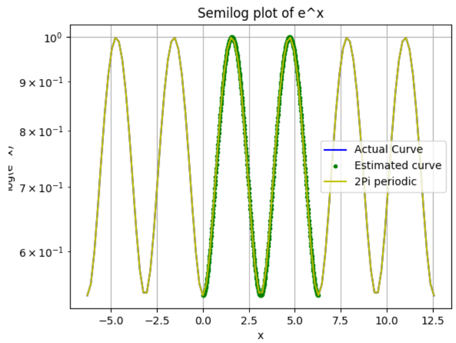

Figure 10: Plot of cos(cos(x)) using the found vector of Fourier coefficients from -2π to 4π

6 Comparing the two methods

The two methods, integration and matrix method are compared by taking the absolute difference between the two, and also finding the maximum difference between the two.

```
def compare_coeff(coeff_1, coeff_2):
    v = abs(coeff_1-coeff_2)
    return v, max(v)
```

The observations here are for e^x there exists a large variation between the two methods, where as for cos(cos(x)) the variation is much smaller.

Maximum Error for cos(cos(x)) is 2.6320616212618636e-15
Maximum Error for e^x is 0.08812169778765977
The Maximum error for e^x is 1.3327308703354248 if 2π is not part of the given data space.

7 Observation on the matrix method for e^x

If we change our data points we are fitting to to include 2π, as the current code omits it, the values of b_n visibly, and are more concurrent with the values of the integration method.

```
#In the previous case, the linspace command was till 401 elements,
    concurrently dropping the last element
x=linspace(0,2*pi,400)
```

The plots are evident of this fact.

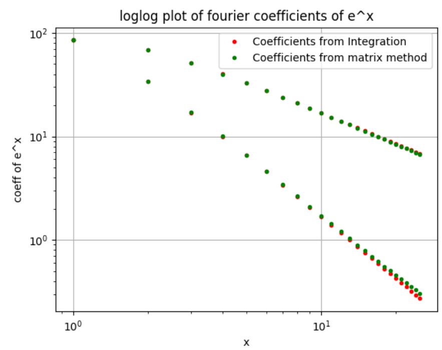

Figure 11: Plot of magnitude of first 51 Fourier coefficients of e^x on Semilog scale

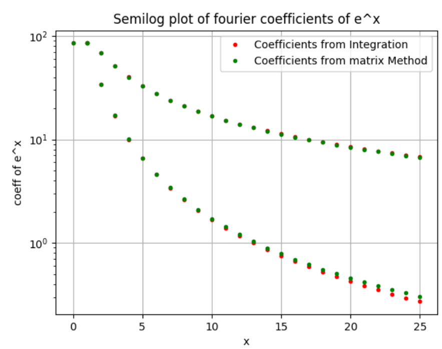

Figure 12: Plot of magnitude of first 51 Fourier coefficients of e^x on Loglog scale

8 Answers to the inline questions

1) From the Fourier series, I expect a 2π periodic function to be generated, with the value of the function from 0 to 2π comprising the values.
3)
a) cos(cos(x)) is clearly an even function, and for even functions, $b_n = 0$, as all the sin terms should vanish.
b) Since e^x is a non periodic function while cos(cos(x)) is a periodic function, to minimise the Gibbs effect, e^x requires higher frequencies/harmonics to get a better fit, hence, the coefficients die out more slowly for an inherently non periodic function as opposed to a periodic one. Also, it is due to the discontinuity of e^x at x = 2π
c) The loglog plot of figure 4 looks linear as the fourier coefficients $a_n \propto \dfrac{n}{n^2+1}$,

similarly for b_n also. Hence, at large n, $a_n \propto \frac{1}{n}$ as $n^2 + 1 \approx n$.

The semilog plot in figure 5 looks linear as it has a $\frac{1}{n}$ variation for small n and a $\frac{1}{n^2}$ variation for large n, due its dependence on zeroth and first order Bessel function.

6) There is a larger error in the case of e^x, due to the discontinuity

7) Due to the discontinuity when we fit e^x as a 2π periodic function, it requires many more Fourier terms to create a better fit, and hence we see a large variation.

9 Conclusion

We have approximated functions to their 2π periodic form upto a threshold on the number of coefficients. We perform this fit through two methods, integration and least square matrix method, for two functions, one is continuous and the other is discontinuous at the boundary. The methods are direct applications of the formulas to calculate Fourier coefficients, along with an L2 norm fit, We notice a large correlation for cos(cos(x)) while a larger error for e^x. The discontinuous case highlights the need for more coefficients to create a better fit.