APPLIED PROGRAMMING LAB (Week - 3) Fourier Coefficients

Shubhangi Ghosh
EE15B129
Department of Electrical Engineering

February 10, 2017

1 INTRODUCTION:

This report presents two methods of computing Fourier series coefficients. A function $f(x)$ can be represented as a Fouriers series, *i.e.* a summation of cosine and sine functions as:

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

1.1 Functions:

In this code, computation of Fourier coefficients are illustrated for the functions:

1. e^x
2. $cos(cosx))$

1.2 Methods Illustrated for computing Fourier Coefficients:

The two methods presented in this report are:

1. **Computation of coefficients by Integration:**

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

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

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

2. **Least Squares Approximation:**

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

We want to solve,

$$Ac = b,$$

where,

$$A - the\, matrix\, on\, the\, left\, hand\, side$$

$$c - the\, matrix\, of\, Fourier\, coefficients$$

$$b - the\, matrix\, of\, function\, values$$

This can be solved by *Least Squares Approximation* by solving the equation:

$$A^*Ac = A^*b$$

Python provides the function lstsq in its *numpy.linalg* library.

2 BRIEF CODE EXPLANATION:

The following steps have been followed in the code:

1. First, we define functions which return e^x and $cos(cosx)$ on passing x as an argument.
2. Then, we define a function, f_coeff which computes Fourier coefficients by method of integration.
3. Then, we write code for computing Fourier coefficients by Least Square Approximation.
4. The fourier coefficients as computed by integration are stored in ans_exp for periodic exponential function $e^x over the interval[0, 2\pi)$ and ans_cos for $cos(cosx) over the interval[0, 2\pi)$.
5. Then, we plot Fourier coefficients computed by integration and that by Least square approximation on a semilog plot $w.r.t$ n and compare.
6. We do the same in a log-log plot.
7. Then, we plot the error in the coefficients as computed by integration and least square approximations for both functions.

2.1 CODE:

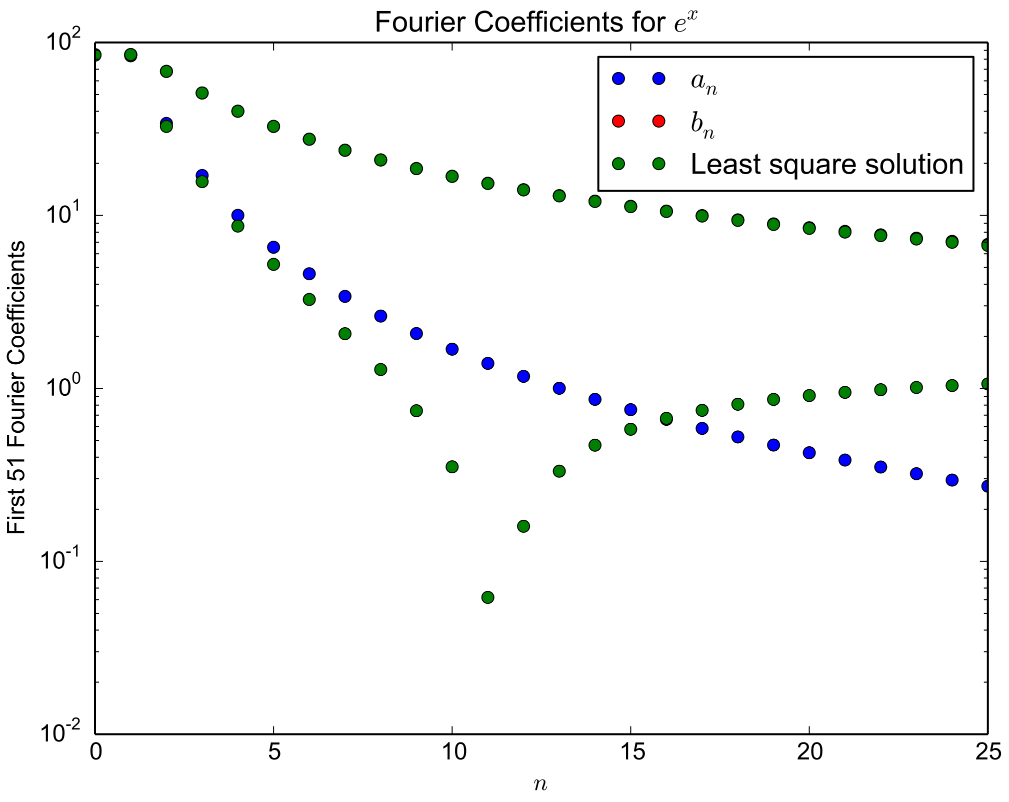
```
#Importing header files
import numpy as np
from scipy import *
from matplotlib.pyplot import *
from scipy.integrate import quad
from math import *
#Exponential Function
def expo(x):
return e**x
#cos(cos(x)) Funxtion
```

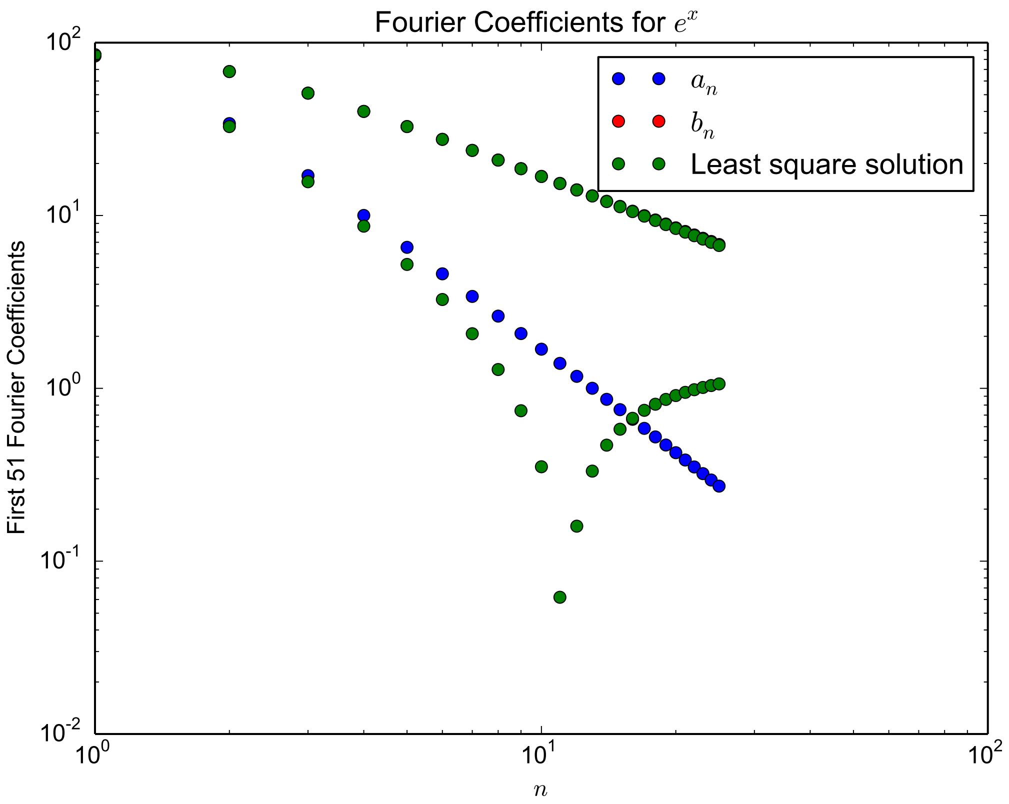
```
def cosine(x):
return np.cos(np.cos(x))
#Returns f(x)cos(x)
def u(x,func,k):
return func(x)*np.cos(k*x)
#Return f(x)sin(x)
def v(x,func,k):
return func(x)*np.sin(k*x)
#Finding 2*n+1 fourier coefficients for func(x) i.e. a0,a1,..an and b1,b2,..bn
def f_coeff(func,n):
a = []
a.append((1/(2*pi))*quad(func,0,2*pi)[0]) #a0
for k in range(1,n+1):
a.append((1/pi)*quad(u,0,2*pi,args=(func,k))[0]) #ak
a.append((1/pi)*quad(v,0,2*pi,args=(func,k))[0]) #bk
return a
#Fourier coeffcients for e^x
ans_exp = f_coeff(expo,25)
n= range(0,26) #Indices of fourier coefficients
#Fourier coeffcients for cos(cos(x))
ans_cos = f_coeff(cosine,25)
#Fourier coefficients by Least Square Approximation
#Creating a linearly spaced array with 400 points between 0 and 2*pi
x = linspace(0,2*pi,400)
b = expo(x) #exponential func for vector
A = zeros((400,51)) # allocate a matrix A with zeroes
A[:,0] = 1 #first column is 1s
for k in range(1,26):
A[:,2*k-1] = np.cos(k*x) #cos(kx) column
A[:,2*k] = np.sin(k*x) #sin(kx) column
c = np.linalg.lstsq(A,b)[0] #[0] gives first col of lstsq which is the best fit vector
a_coeff = np.fabs(insert(c[1::2],0,c[0])) #extracting a0,a1,...an in an array
#plotting first 51 fourier coefficients for e^x in semilog scale
title('Fourier Coefficients for $e^x$')
xlabel('$n$')
ylabel('First 51 Fourier Coefficients')
semilogy(n,np.fabs([ans_exp[0]]+ans_exp[1::2]),'bo') #an
semilogy(n[1:],np.fabs(ans_exp[2::2]),'ro') #bn
semilogy(n,a_coeff,'go') #least square an
semilogy(n[1:],np.fabs(c[2::2]),'go') #least square bn
legend(('$a_n$','$b_n$','Least square solution'),loc='best')
savefig('fig1_exp.pdf',format='pdf') #save figure as pdf
show()
#plotting first 51 fourier coefficients for e^x in loglog scale
title('Fourier Coefficients for $e^x$')
xlabel('$n$')
ylabel('First 51 Fourier Coefficients')
loglog(n,np.fabs([ans_exp[0]]+ans_exp[1::2]),'bo') #an
loglog(n[1:],np.fabs(ans_exp[2::2]),'ro') #bn
loglog(n,a_coeff,'go') #least square an
loglog(n[1:],np.fabs(c[2::2]),'go') #least square bn
legend(('$a_n$','$b_n$','Least square solution'),loc='best')
savefig('fig2_exp.pdf',format='pdf') #save figure as pdf
```

```
show()
#error plot for first 51 fourier coefficients for e^x in semilog scale
title('Absolute value of error in Fourier Coefficients for $e^x$')
xlabel('$n$')
ylabel('Error')
error = np.fabs(ans_exp-c)
semilogy(n,insert(error[1::2],0,error[0]),'bo') #an
semilogy(n[1:],error[2::2],'ro') #bn
legend(('Error in $a_n$','Error in $b_n$'),loc='best')
savefig('error_exp.pdf',format='pdf') #save figure as pdf
show()
#printing maximum error for e^x
print "Maximum error between Coefficients found by Integration and Least Squares Metho
print 'e^x:'
print np.amax(error)
b = cosine(x) #cos(cos(x)) func for vector
c = np.linalg.lstsq(A,b)[0] #[0] gives first col of lstsq which is the best fit vector
a_coeff = np.fabs(insert(c[1::2],0,c[0])) #extracting a0,a1,...an in an array
title('Fourier Coefficients for $cos(cos(x))$')
xlabel('$n$')
ylabel('First 51 Fourier Coefficients')
#plotting first 51 fourier coefficients for cos(cos(x)) in semilog scale
semilogy(n,np.fabs([ans_cos[0]]+ans_cos[1::2]),'bo') #an
semilogy(n[1:],np.fabs(ans_cos[2::2]),'ro') #bn
semilogy(n,a_coeff,'go') #least square an
semilogy(n[1:],np.fabs(c[2::2]),'go') #least square bn
legend(('$a_n$','$b_n$','Least square solution'),loc='best')
savefig('fig1_cos(cos).pdf',format='pdf') #save figure as pdf
show()
title('Fourier Coefficients for $cos(cos(x))$')
xlabel('$n$')
ylabel('First 51 Fourier Coefficients')
loglog(n,np.fabs([ans_cos[0]]+ans_cos[1::2]),'bo') #an
loglog(n[1:],np.fabs(ans_cos[2::2]),'ro') #bn
loglog(n,a_coeff,'go') #least square an
loglog(n[1:],np.fabs(c[2::2]),'go') #least square bn
legend(('$a_n$','$b_n$','Least square solution'),loc='best')
savefig('fig2_cos(cos).pdf',format='pdf') #save figure as pdf
show()
#error plot for first 51 fourier coefficients for cos(cos(x)) in semilog scale
title('Absolute value of error in Fourier Coefficients for $cos(cos(x))$')
xlabel('$n$')
ylabel('Error')
error = np.fabs(ans_cos-c)
#printing maximum error for e^x
print 'cos(cos(x)):'
print np.amax(error)
semilogy(n,insert(error[1::2],0,error[0]),'bo')
semilogy(n[1:],error[2::2],'ro')
legend(('Error in $a_n$','Error in $b_n$'),loc='best')
savefig('error_cos(cos)).pdf',format='pdf') #save figure as pdf
show()
```

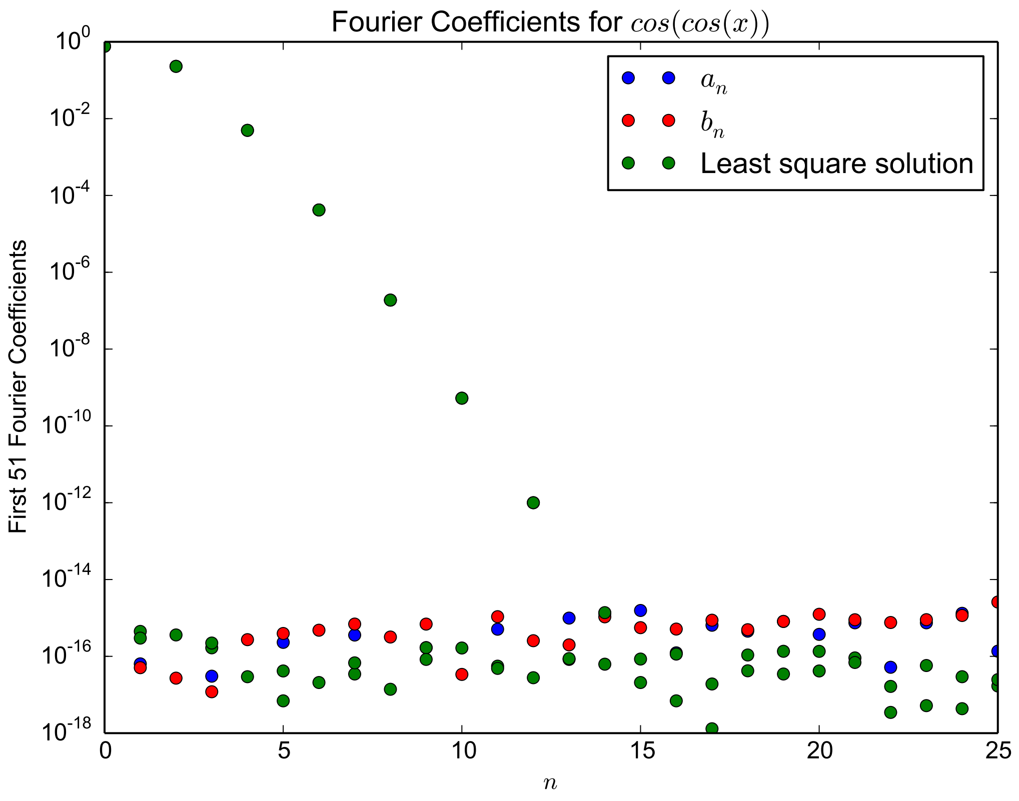
2.2 PLOTS:

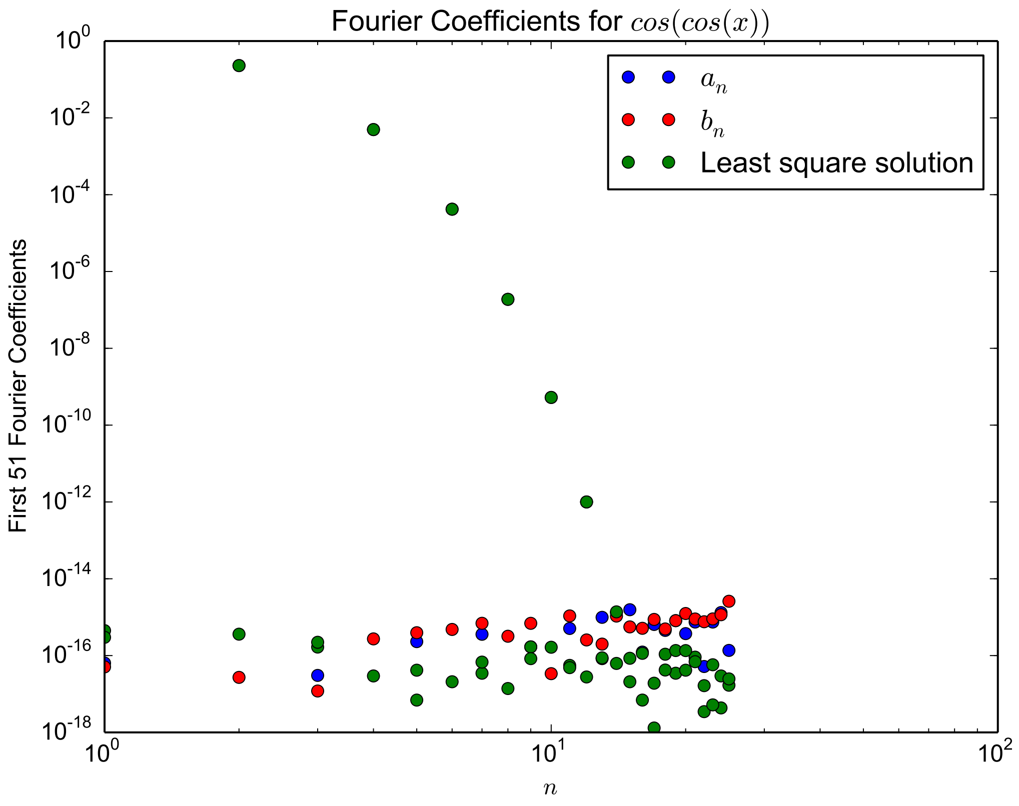
2.2.1 Plot of Fourier Coefficients of e^x as computed by Integration and Least Squares Method



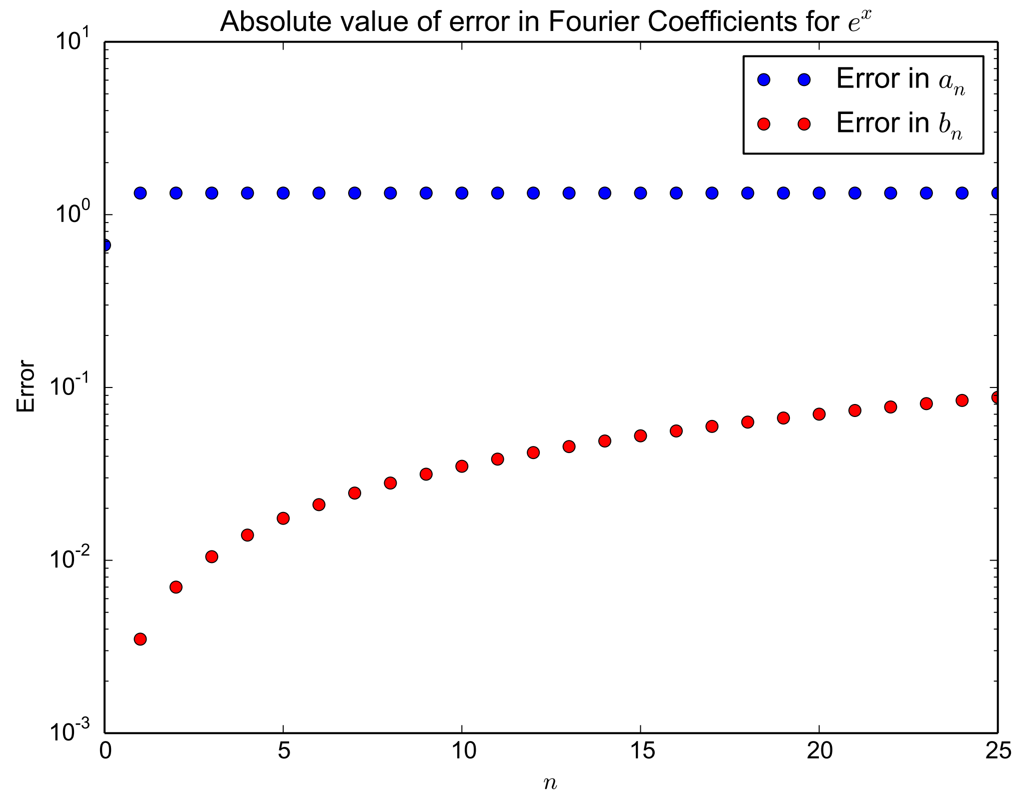


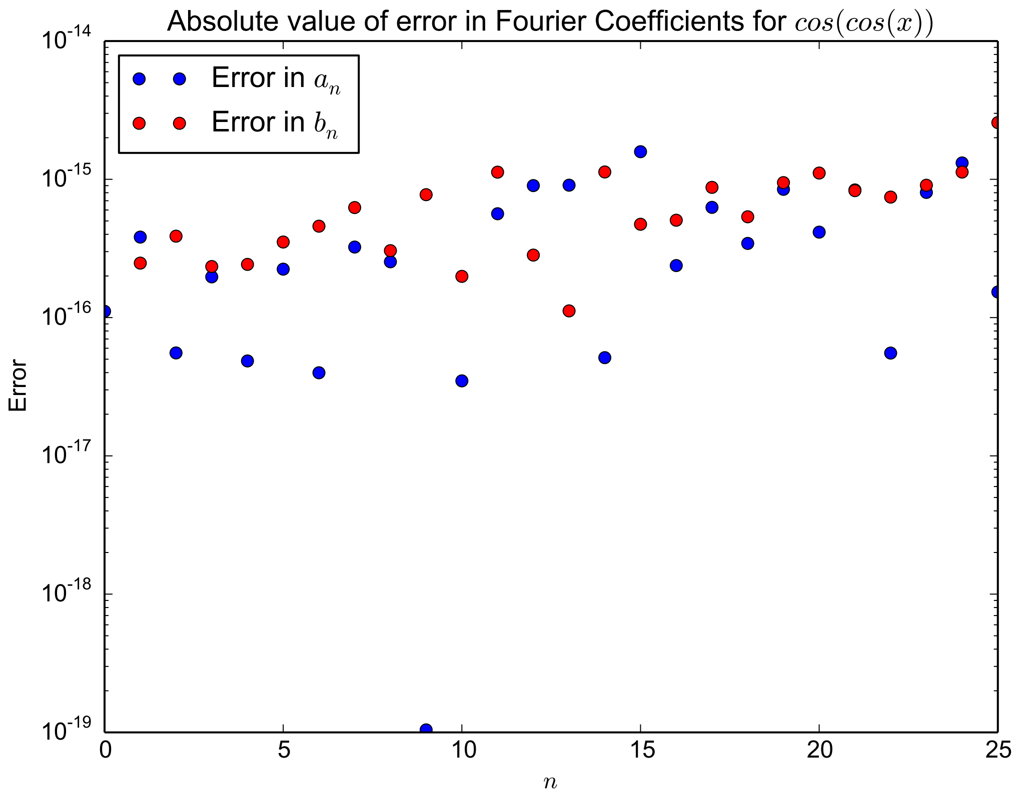
2.2.2 Plot of Fourier Coefficients of $cos(cosx)$ as computed by Integration and Least Squares Method





2.2.3 Plot of error in Fourier Coefficients of $e^x$and $cos(cosx)$ as computed by Integration and Least Squares Method





2.3 ERROR ANALYSIS:

1. The value of maximum error in $e^x$is 1.33273087034.
2. The value of maximum error in $cos(cos\, x)$ is $2.57022928446\, x\, 10^{-15}$.

3 RESULTS AND DISCUSSION:

3.1 Question 1.

Why are the b_n coefficients nearly zero for $cos(cos(x))$?
Ans. $cos(cos(x))$ is an even function. Hence, the b_n terms, being the coefficients of sine terms, which are odd functions are essentially.
But, since integration by quad function is not exact, we obtain values very close to zero.

3.2 Question 2.

Why are the don't the coefficients of e^x not decay quickly in semilog plot?
Ans. Substantial high frequency components exist for e^x, hence coefficients don't decay quickly wrt n.

3.3 Question 3.

Why is the log-log plot of e^x linear?
Ans. The coefficients of e^x vary as $\frac{1}{1+n^2}$. This can be approximated as $\frac{1}{n^2}$ for large n.

$$a_n = \frac{1}{n^2}$$

$$log\, a_n = -2log\, n$$

Thus, the $a_n vs. n$ plot is an approximately linear plot of slope -2.

3.4 Question 4.

Why are b_n coefficients larger than a_n coefficients?
Ans. The b_n coefficients being larger $a_n$coefficients imply that the odd part of the function e^x periodic in the interval $[0, 2\pi)$ has a larger magnitude than the even part of the function.

3.5 Question 5.

Should the coefficients agree as computed by Least Squares Approximation Method and by
Ans. The coefficients need not agree.

1. The value of maximum error in $e^x$is 1.33273087034.
2. The value of maximum error in $cos(cos\, x)$ is $2.57022928446\, x\, 10^{-15}$.

But Least Square is a very good approximation, when actual integration is not feasible