Week 3: Fourier Approximations

Ganga Meghanath
EE15B025
Electrical Engineering

January 30, 2017

Introduction

This report features the analysis of coefficients of the Fourier series expansion of e^x and cos(cos(x)) over the interval [0,2π). Two different approaches have been followed to compute the coefficients - Integration and Least Squares. For each of the above mentioned methods, graphs in semilog and log scales are plotted against frequency to get more insights into the two approaches and analyse the deviation in the Least Square approach in comparison to Direct Integration. The absolute difference between the two sets of coefficients obtained would serve as a means to quantify the deviation and also to measure the maximum deviation.

The Fourier series is given by :

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

For simplicity of computation, we will not compute the coefficients till infinity , but till 25, as our main purpose is to compare the two methods mentioned above. Hence,

$$f(x) = a_0 + \sum_{n=1}^{25} \{a_n cos(nx) + b_n sin(nx)\}$$

Integration method, the coefficients $a_n$and $b_n$are given by :

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

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

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

Least square method , it turns into a matrix problem :

Defining a vector x going from 0 to 2π in 400 steps and evaluating the function f (x) at those x values.

$$\begin{pmatrix} 1 & cosx_1 & sinx_1 & ... & cos25x_1 & sin25x_1 \\ 1 & cosx_2 & sinx_2 & ... & cos25x_2 & sin25x_2 \\ ... & ... & ... & ... & ... & ... \\ 1 & cosx_{400} & sinx_{400} & ... & cos25x_{400} & sin25x_{400} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ ... \\ a_{25} \\ b_{25} \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ ... \\ f(x_{400}) \end{pmatrix}$$

Hence, we will solve Ac=b where c contains the fourier coefficients.

Python and associated libraries have been used for computing the coefficients and for plotting the graphs. Python provides an efficient and easy way for the computations by providing libraries having in-build functions for integration and least squares. This makes our task much simple as seperate codes for the implementation needn't be written. Python also provides matplotlib for plotting functions and even includes latex for displaying mathematical formulae and expressions.

1. Define Python functions for $e^x$and cos(cosx), that take a vector (or scalar) input, and return a vector (or scalar) value.

```
def f1(x):
        return exp(x)

def f2(x):
        return cos(cos(x))

x = raw_input("Enter the vector x for Q1 : ") #accept vector x
x=re.sub('[^0-9 ]+', ' ',x)#accept digits & spaces from string
x=x.split()                                  #split the string
x=[float(a) for a in x] #convert each element of list to float
x=np.array(x)                        #making it an array

y1=f1(x)                            #finding exp(x)
y2=f2(x)                            #finding cos(cosx)

#printing out exp(x) and cos(cosx)
print("\nexp(x)given as : {}".format(y1))
print("\n\ncos(cos(x))given as : {}".format(y2))
print "\n\nwhere x is taken in radians\n\n"
```

The output comes out as :

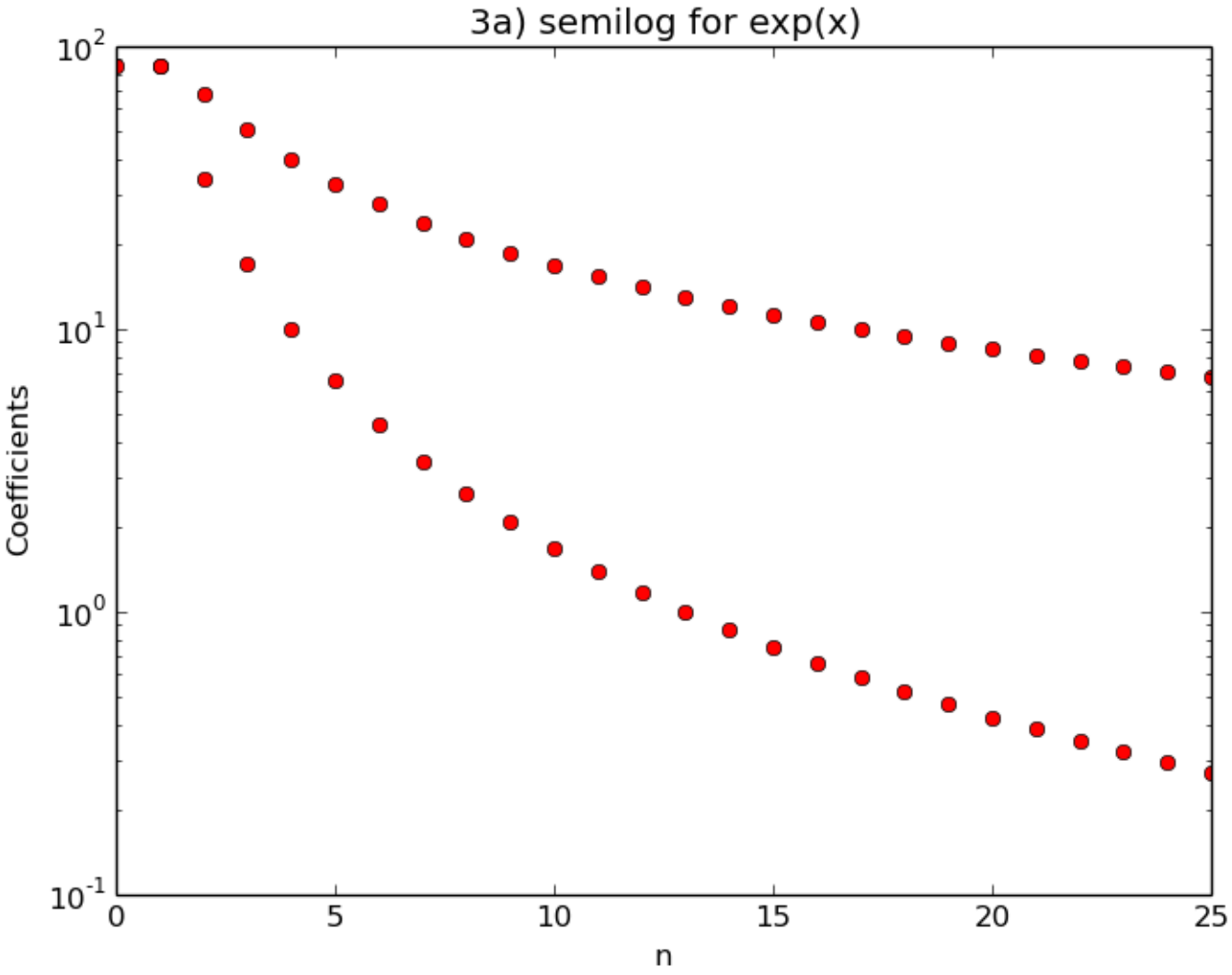
```
Enter the vector x for Q1 : 1,,,2...3   4//5

exp(x)given as:[2.71828183 7.3890561 20.08553692 54.59815003 148.4131591

cos(cos(x))given as:[0.85755322 0.91465333 0.54869613 0.79387345 0.96003

where x is taken in radians
```

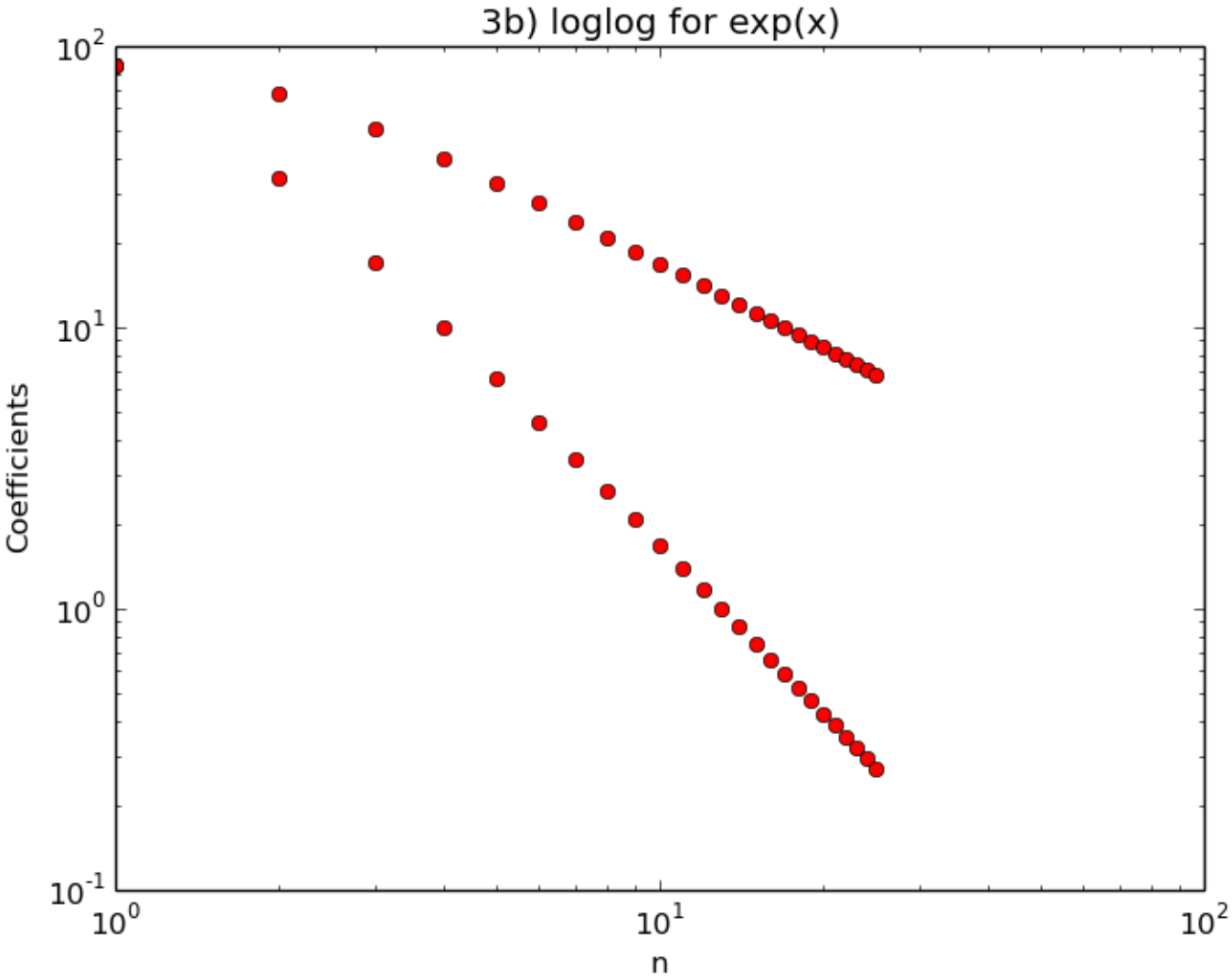
2. Obtain the first 51 coefficients for the two functions above using integration.

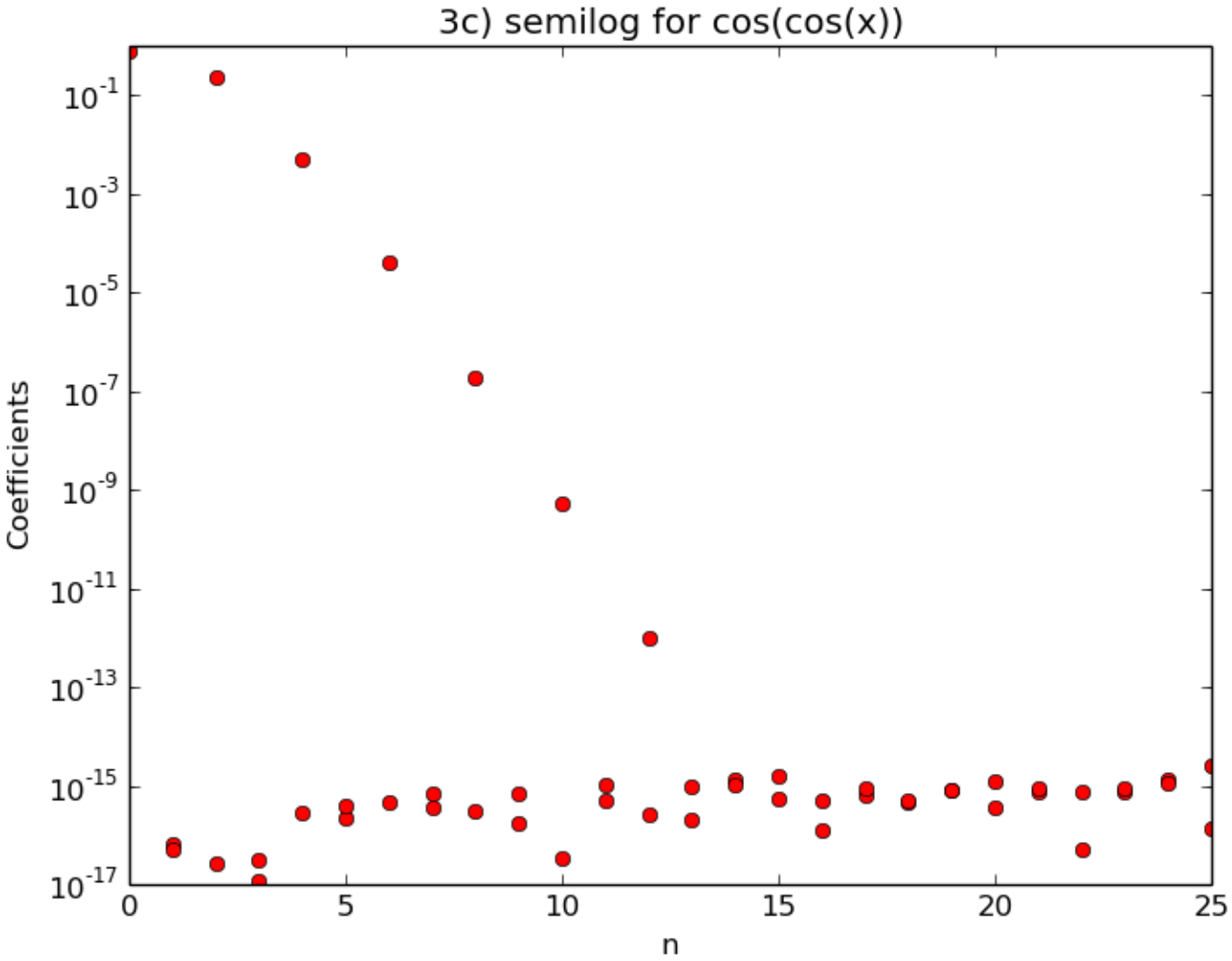
```
def f1_sin(n):
        ans= lambda x,a : exp(x)*sin(n*x)
        return ans

def f1_cos(n):
        ans= lambda x,a : exp(x)*cos(n*x)
        return ans

def f2_sin(n):
        ans= lambda x,a : cos(cos(x))*sin(n*x)
        return ans

def f2_cos(n):
        ans= lambda x,a : cos(cos(x))*cos(n*x)
        return ans

expx=[] #coefficient vector for exp(x)
coscosx=[]#coefficient vector for cos(cosx)

temp=quad(f1_cos(0),0,2*pi,args=(0))
expx.append(temp[0]/(2*pi)) #compute a0
temp=quad(f2_cos(0),0,2*pi,args=(0))
coscosx.append(temp[0]/(2*pi)) #compute a0

for i in range(1,26):   #loop for computing values of the coefficients
        temp=quad(f1_cos(i),0,2*pi,args=(i)) #computes a0 for exp(x)
        expx.append(temp[0]/pi)
        temp=quad(f2_cos(i),0,2*pi,args=(i)) #computes a0 for cos(cosx)
        coscosx.append(temp[0]/pi)

        temp=quad(f1_sin(i),0,2*pi,args=(i)) #computes b0 for exp(X)
        expx.append(temp[0]/pi)
        temp=quad(f2_sin(i),0,2*pi,args=(i))#computes b0 for cos(cosx)
        coscosx.append(temp[0]/pi)

with open('Output.txt','a') as file_obj:
        file_obj.write("\nThe 51 coefficients for exp(x): {}".format(exp
        file_obj.write("\n\nThe 51 coefficients for cos(cos(x)): {}".for
```

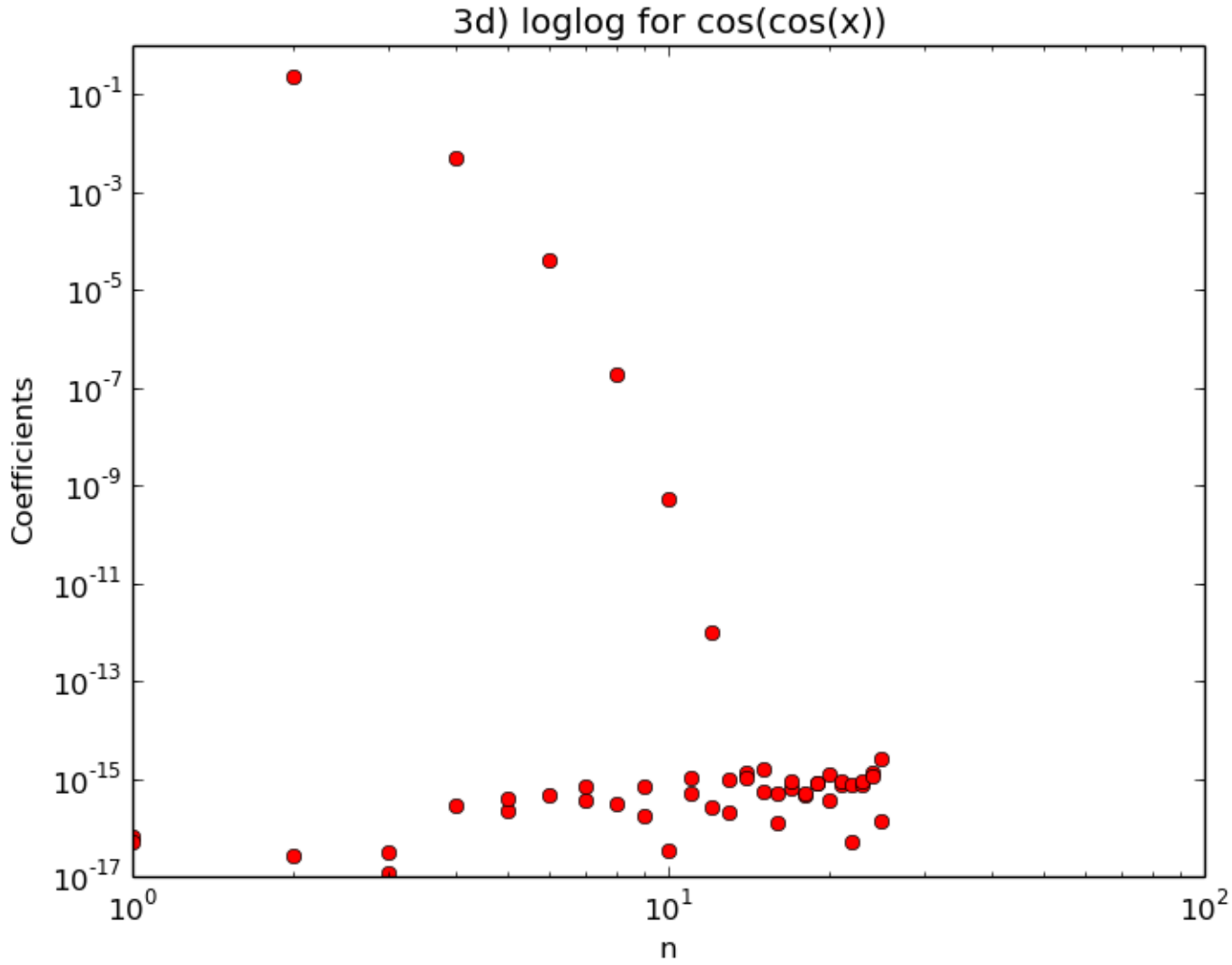
3. For each of the two functions, make two different plots using "semilogy" and "loglog" and plot the magnitude of the coefficients vs n . The values should be plotted with red circles. Note that the answer should be a vector of the form :

$$\begin{pmatrix} a_0 \\ a_1 \\ b_2 \\ \dots \\ a_{25} \\ b_{25} \end{pmatrix}$$


3a) semilog for exp(x)
Coefficients
n
10^2
10^1
10^0
10^{-1}
0
5
10
15
20
25


3b) loglog for exp(x)
Coefficients
n
10^{2}
10^{1}
10^{0}
10^{-1}
10^{0}
10^{1}
10^{2}


3c) semilog for cos(cos(x))
Coefficients
n
10^{-1}
10^{-3}
10^{-5}
10^{-7}
10^{-9}
10^{-11}
10^{-13}
10^{-15}
10^{-17}
0
5
10
15
20
25



4. In integration method, the b_n coefficients for $cos(cosx)$ is nearly zero. Why does this happen?

Tailor expansion of $cosx = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - ....$

Hence, cos(cosx)=$1 - \frac{(cosx)^2}{2!} + \frac{(cosx)^4}{4!} - \frac{(cosx)^6}{6!} + \frac{(cosx)^8}{8!} - ...$

We know that cos(2x)=1-2cos^2x

Hence, each of the terms in the tailor expansion can be replaced by cosine functions of frequencies which are multiples of x. This can be made to look like a fourier series entirely in cosine function. Hence, cos(cosx) can be expressed as a sum of cosines.

cos(cosx) is an even and positive(ranges from nearly 0.54 to 1) function.

According to theorem, if f(x) is a function defined and integrable on interval $[-\pi,\pi]$ and even, then

$$b_n = 0, \text{ pour tout } n \geq 1;$$

and

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

The fourier series is then called the Fourier Cosine series and is given by,

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

if f(x) is odd, then

$$a_n = 0, \text{ pour tout n} \geq 0$$

and

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

The fourier series is then called the Fourier Sine series and is given by,

$$f(x) \sim \sum_{n=1}^{\infty} a_n sin(nx)$$

5. In the e^x case in integration method, the coefficients do not decay quickly in the semilogy plot. Why not? Why does the plot look linear in the loglog plot?

$e^x$is not a periodic function. We have defined an interval for it and repeated it for infinite times to make a periodic function of its value in the interval [0,2π) , having time period 2π. Fourier series is defined for a periodic signal.

e^xfourier series in the given interval is as follows :

$$e^x = \frac{e^{2\pi}-1}{\pi}(\frac{1}{2} + \sum_{n=1}^{\infty} \frac{cos(nx)-nsin(nx)}{n^2-1})$$

Here,

$$a_n \ \alpha \ \frac{1}{n^2-1}$$

and

$$b_n \ \alpha \ \frac{n}{n^2-1}$$

This clearly explains why the loglog plot is linear. The y α $\frac{1}{x}$and $\frac{1}{x^2}$.

This also explains why the coefficients don't decay quickly in the semilogy plot since y-axis has $\frac{1}{n^2}$and $\frac{1}{n}$relations. Hence the decay will be gradual when Y-axis is plotted in log scale.

6. Why are the b_n coefficients larger than the a_n coefficients for e^xcase in integration method?

This is because, the fourier series is as follows :

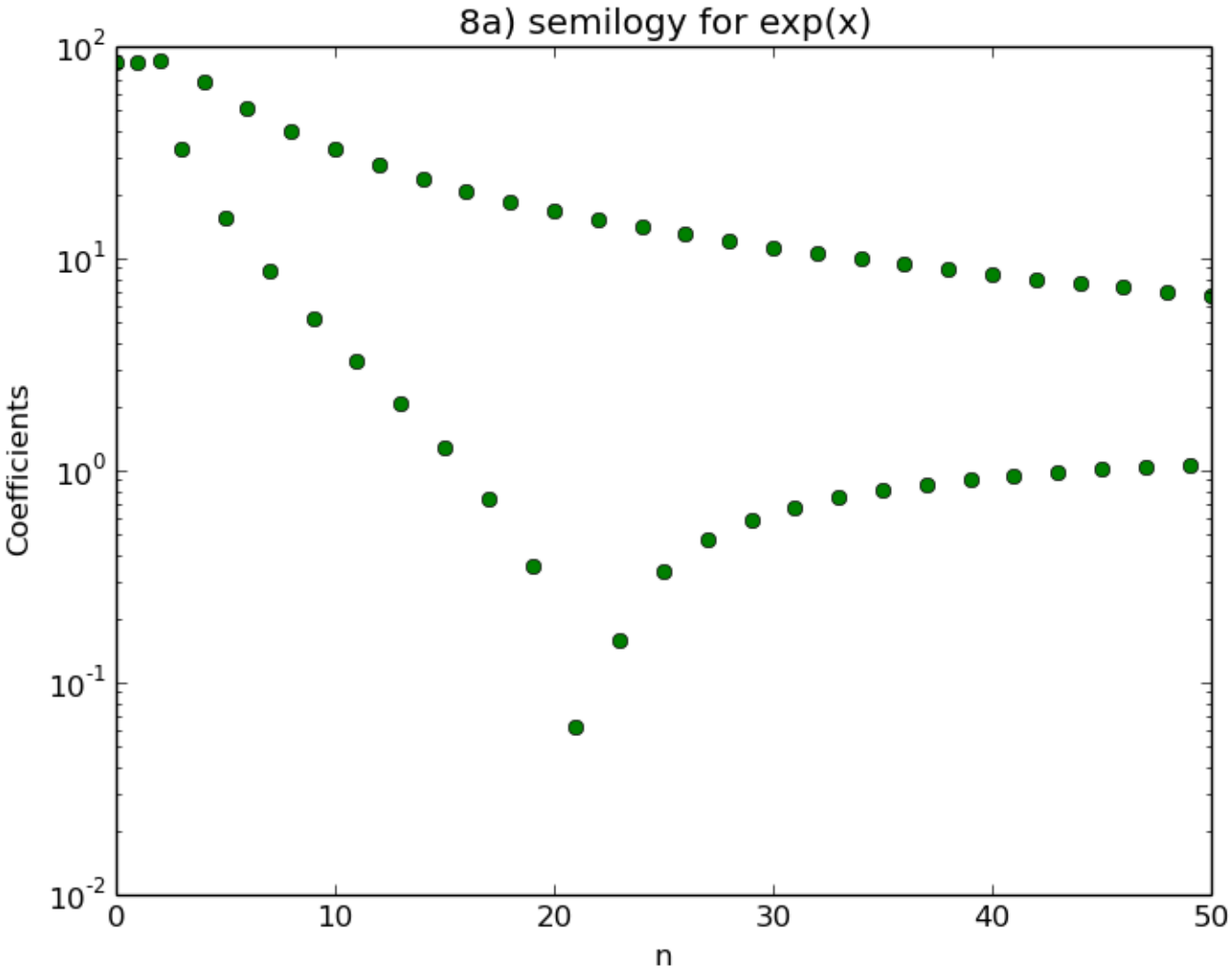
$$e^x = \frac{e^{2\pi}-1}{\pi}(\frac{1}{2} + \sum_{n=1}^{\infty} \frac{cos(nx)-nsin(nx)}{n^2-1})$$

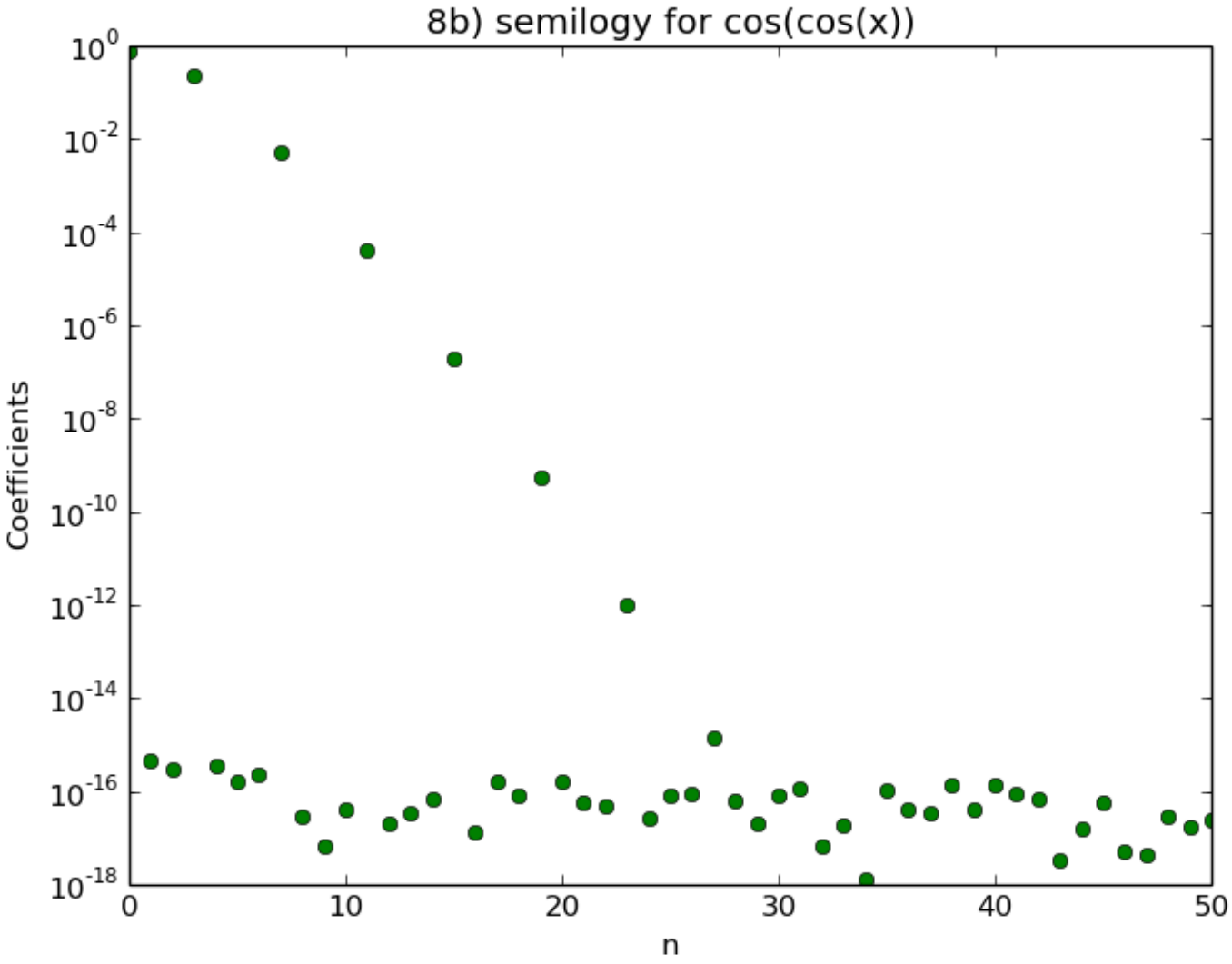
Hence, $a_n = abs\left[\frac{e^{2\pi}-1}{\pi(n^2-1)}\right] < abs\left[-\frac{(e^{2\pi}-1)n}{\pi(n^2-1)}\right] = b_n$

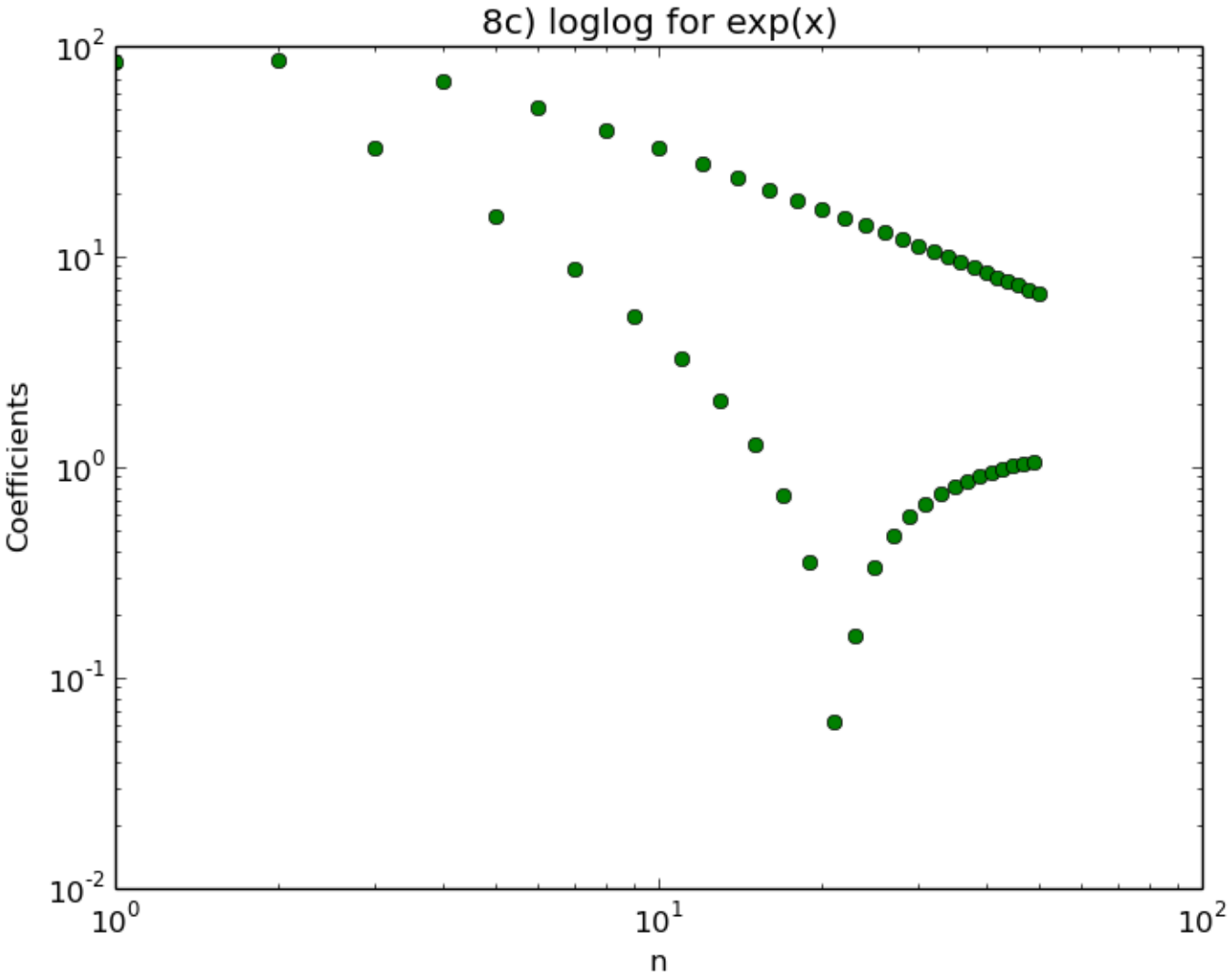
7. For Least square method, build up x,b and A and use lstsq() to solve the problem.

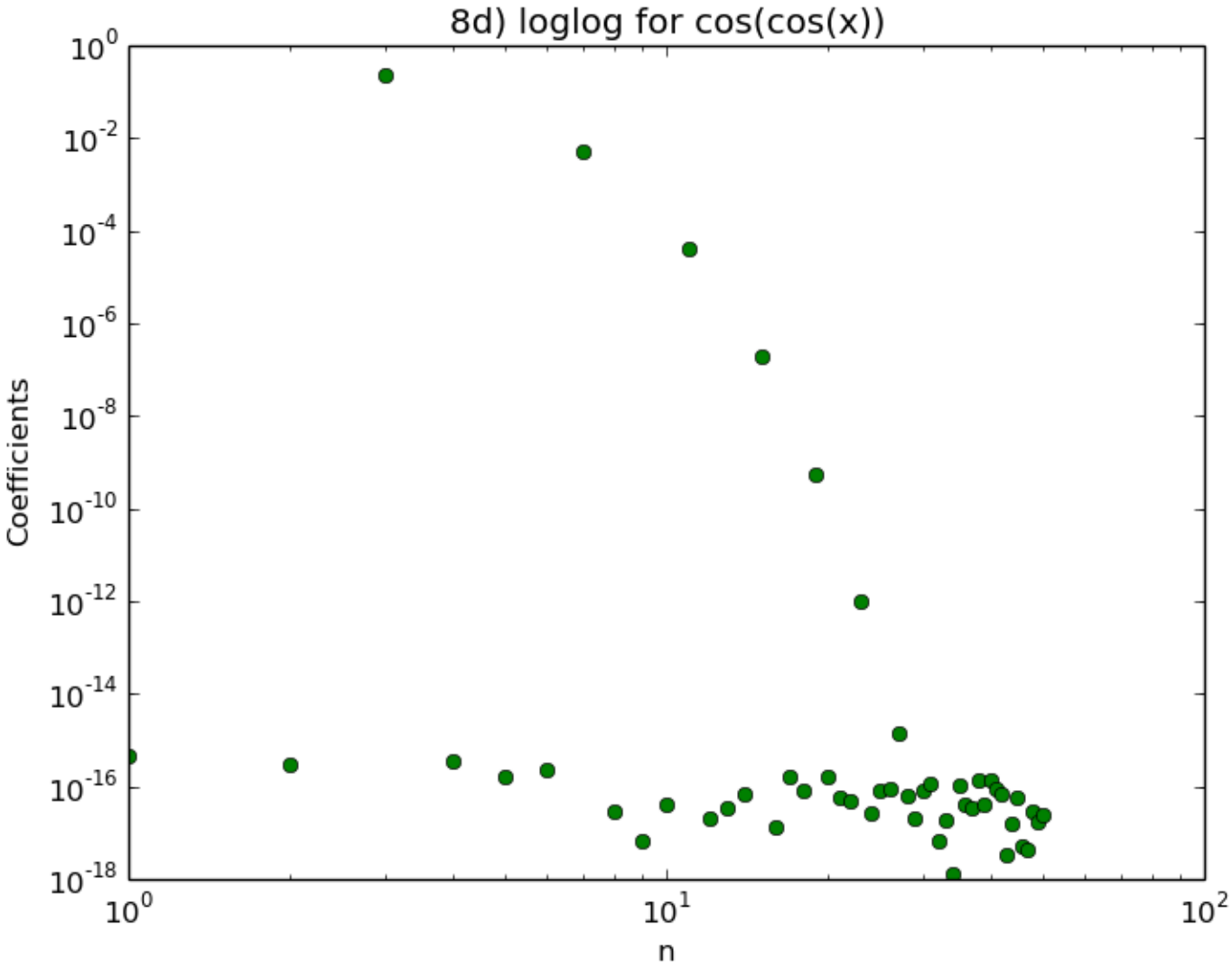
```
x = linspace(0,2*pi,num=401)#creating x for least square approx
x=x[:-1] # drop last term to have proper periodic integral
b1=f1(x) # f1->exp() has been written to take a vector
b2=f2(x) # f2->cos(cosx) has been written to take a vector

with open('Output.txt','a') as file_obj:
        file_obj.write(("\nexp(x) given as : {}".format(b1,b2)))
        file_obj.write("\n\ncos(cos(x)) is given as : {}".format(b2))
        file_obj.write("\n\nwhere x is taken in radians\n\n")

A1=np.zeros((400,51))    # allocate space for A1->exp(x)
A2=np.zeros((400,51))    # allocate space for A2->cos(cosx)

A1[:,0]=1         # col 1 is all one
A2[:,0]=1         # col 1 is all one

for k in range(1,26):
        A1[:,2*k-1]=cos(k*x)    # cos(kx) column
        A2[:,2*k-1]=cos(k*x)    # sin(kx) column

        A1[:,2*k]=sin(k*x)              # cos(kx) column
        A2[:,2*k]=sin(k*x)              # sin(kx) column

c11=lstsq(A1,b1)[0]
c12=lstsq(A2,b2)[0]
```

8. Plot the coefficients obtained using lstsq() with green circles in the corresponding plots.


8a) semilogy for exp(x)
Coefficients
n
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
0
10
20
30
40
50

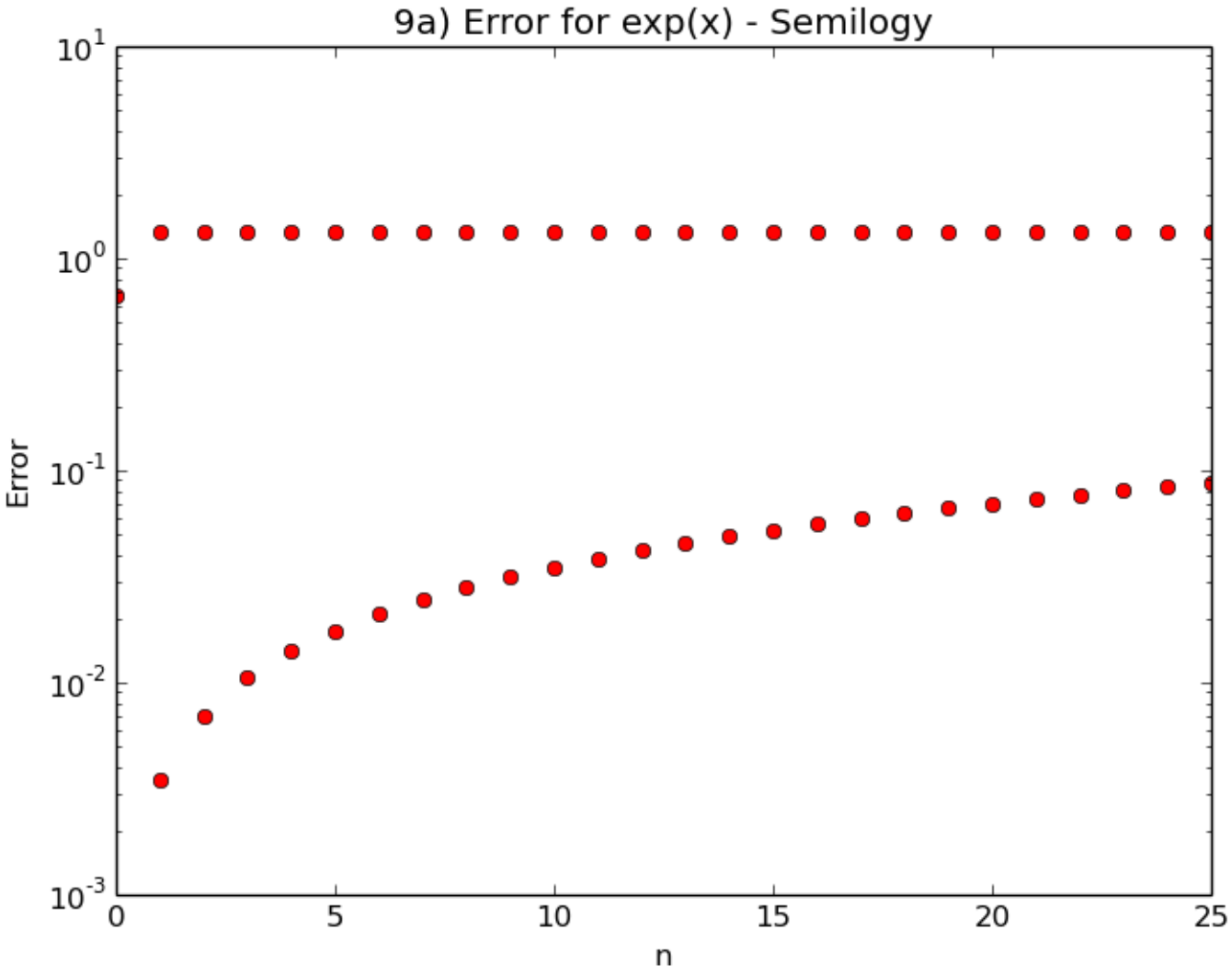

8b) semilogy for cos(cos(x))
Coefficients
n
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
10^{-16}
10^{-18}
0
10
20
30
40
50

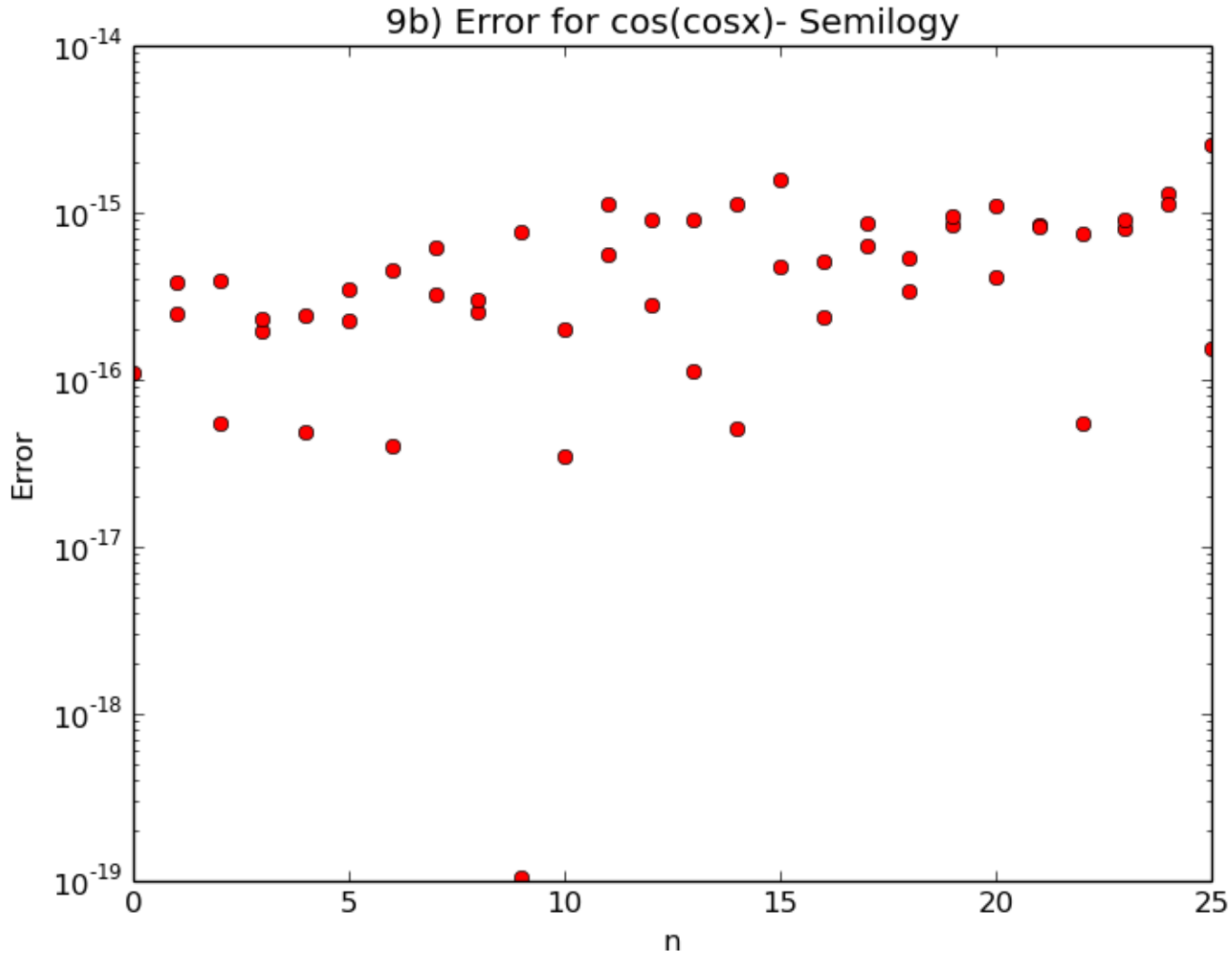

8c) loglog for exp(x)
Coefficients
n
10^2
10^1
10^0
10^{-1}
10^{-2}
10^0
10^1
10^2



9. Compare the answers got by least squares and by the direct integration. Do they agree? Should they? How much deviation is there (find the absolute difference between the two sets of coefficients and find the largest deviation. How will you do this using vectors?)

```
error_exp=abs(c11-expx)          #deviation for exp(x)
error_cos=abs(c12-coscosx)       #deviation for cos(cosx)
```


9a) Error for exp(x) - Semilogy
Error
n
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
0
5
10
15
20
25



As we can see from the graphs, the deviation is quite small. For exp(x) , the maximum deviation is about 1.33 and for cos(cosx), the maximum deviation is of the order of 10^{-15}. This would tell us that the approximation for the fourier coefficients of cos(cosx) is quite close to the real value. Hence it can be used for further computations effectively. As in the case of e^x,the values obtained are decent enough, but the error value obtained is accountable. Hence, integration method would be more preferred for e^x though least square method is a fair enough approximation.

The whole code :

Note : Some parts of the code might not be visile in the pdf, though the entire code is available in the .lyx file

```
from pylab import *
import re
import numpy as np
from scipy.integrate import quad

def f1(x):
        return exp(x)

def f2(x):
        return cos(cos(x))
```

```
def f1_sin(n):
        ans= lambda x,a : exp(x)*sin(n*x)
        return ans

def f1_cos(n):
        ans= lambda x,a : exp(x)*cos(n*x)
        return ans

def f2_sin(n):
        ans= lambda x,a : cos(cos(x))*sin(n*x)
        return ans

def f2_cos(n):
        ans= lambda x,a : cos(cos(x))*cos(n*x)
        return ans

################################################ Question 1 ###########

x = raw_input("Enter the vector x for Q1 : ")   #accept a vector x from
x=re.sub('[^0-9 ]+', ' ',x)
x=x.split()
x=[float(a) for a in x]
x=np.array(x)

y1=f1(x)                          #finding exp(x)
y2=f2(x)                          #finding cos(cosx)

#printing out exp(x) and cos(cosx)
print("\nexp(x) given as : {} \n\ncos(cos(x)) is given as : {}\n\nwhere

################################################ Question 2 ###########

expx=[] #coefficient vector for exp(x)
coscosx=[]#coefficient vector for cos(cosx)

temp=quad(f1_cos(0),0,2*pi,args=(0))
expx.append(temp[0]/(2*pi)) #compute a0
temp=quad(f2_cos(0),0,2*pi,args=(0))
coscosx.append(temp[0]/(2*pi)) #compute a0

for i in range(1,26):   #loop for computing values of the coefficients
        temp=quad(f1_cos(i),0,2*pi,args=(i)) #computes a0 for exp(x)
        expx.append(temp[0]/pi)
        temp=quad(f2_cos(i),0,2*pi,args=(i)) #computes a0 for cos(cosx)
        coscosx.append(temp[0]/pi)

        temp=quad(f1_sin(i),0,2*pi,args=(i)) #computes b0 for exp(X)
        expx.append(temp[0]/pi)
        temp=quad(f2_sin(i),0,2*pi,args=(i))#computes b0 for cos(cosx)
        coscosx.append(temp[0]/pi)

with open('Output.txt','a') as file_obj:
```

```
        file_obj.write("\nThe 51 coefficients for exp(x): {}".format(exp
        file_obj.write("\n\nThe 51 coefficients for cos(cos(x)): {}".for

################################################ Question 3 ############
expx=np.array(expx)
coscosx=np.array(coscosx)

temp=linspace(1,25,num=25,endpoint=True)#defining a list ranging from 1

n=np.zeros(51)  #n denotes x axis ->different frequencies

n[1::2]=temp #n will have one 0 and then doublets of the
n[2::2]=temp #rest of the numbers till 25

line1, = semilogy(n,abs(expx),'ro',linewidth=10)
xlabel('n')
ylabel('Coefficients')
title("3a) semilog for exp(x)")
savefig("3a.png")
show()

line2, = loglog(n,abs(expx),'ro',linewidth=10)
xlabel('n')
ylabel('Coefficients')
title("3b) loglog for exp(x)")
savefig("3b.png")
show()

line3, = semilogy(n,abs(coscosx),'ro',linewidth=10)
xlabel('n')
ylabel('Coefficients')
title("3c) semilog for cos(cos(x))")
savefig("3c.png")
show()

line4, = loglog(n,abs(coscosx),'ro',linewidth=10)
xlabel('n')
ylabel('Coefficients')
title("3d) loglog for cos(cos(x))")
savefig("3d.png")
show()

################################################ Question 3 ############

x = linspace(0,2*pi,num=401)    #creating x for least square approximati
x=x[:-1]                # drop last term to have a proper periodic inte
b1=f1(x)                # f1->exp() has been written to take a vector
b2=f2(x)                # f2->cos(cosx) has been written to take a vect

with open('Output.txt','a') as file_obj:
        file_obj.write(("\nexp(x) given as : {} \n\ncos(cos(x)) is given

A1=np.zeros((400,51))    # allocate space for A1->exp(x)
```

```
A2=np.zeros((400,51))    # allocate space for A2->cos(cosx)

A1[:,0]=1         # col 1 is all one
A2[:,0]=1         # col 1 is all one

for k in range(1,26):
        A1[:,2*k-1]=cos(k*x)    # cos(kx) column
        A2[:,2*k-1]=cos(k*x)    # sin(kx) column

        A1[:,2*k]=sin(k*x)              # cos(kx) column
        A2[:,2*k]=sin(k*x)              # sin(kx) column

c11=lstsq(A1,b1)[0]
c12=lstsq(A2,b2)[0]

line5, = semilogy(list(range(0,51)),abs(c11),'go',linewidth=10)
xlabel('n')
ylabel('Coefficients')
title("8a) semilogy for exp(x)")
savefig("8a.png")
show()

line6, = semilogy(list(range(0,51)),abs(c12),'go',linewidth=10)
xlabel('n')
ylabel('Coefficients')
title("8b) semilogy for cos(cos(x))")
savefig("8b.png")
show()

line7, = loglog(list(range(0,51)),abs(c11),'go',linewidth=10)
xlabel('n')
ylabel('Coefficients')
title("8c) loglog for exp(x)")
savefig("8c.png")
show()

line8, = loglog(list(range(0,51)),abs(c12),'go',linewidth=10)
xlabel('n')
ylabel('Coefficients')
title("8d) loglog for cos(cos(x))")
savefig("8d.png")
show()

################################### Question 6 ##########################

error_exp=abs(c11-expx)          #deviation for exp(x)
error_cos=abs(c12-coscosx)       #deviation for cos(cosx)

line9, = semilogy(n,error_exp,'ro',linewidth=10)
xlabel('n')
ylabel('Error')
title("9a) Error for exp(x) - Semilogy")
savefig("9a.png")
```

```
show()

line10, = semilogy(n,error_cos,'ro',linewidth=10)
xlabel('n')
ylabel('Error')
title("9b) Error for cos(cosx)- Semilogy")
savefig("9b.png")
show()

with open('Output.txt','a') as file_obj:
        file_obj.write("\n\nMaximum deviation for exp(x) is {} and cos(c
```