EE2703 : Applied Programming Lab

Assignment 4

Harisankar K J
EE20B043

11 March 2022

Abstract

We will fit two functions, e^x and $cos(cos(x))$ over the interval $[0, 2\pi)$ using their computed Fourier Series Coefficients.

Introduction

The Fourier Series of a function $f(x)$ with period 2π is computed as follows:

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

where ,

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

Assignment

0.0.1 Initializing the functions

$cos(cos(x))$ is a periodic function with period 2π whereas e^x is not periodic. The functions that will be generated from the Fourier Series are $cos(cos(x))$ and $e^{x\%(2\pi)}$

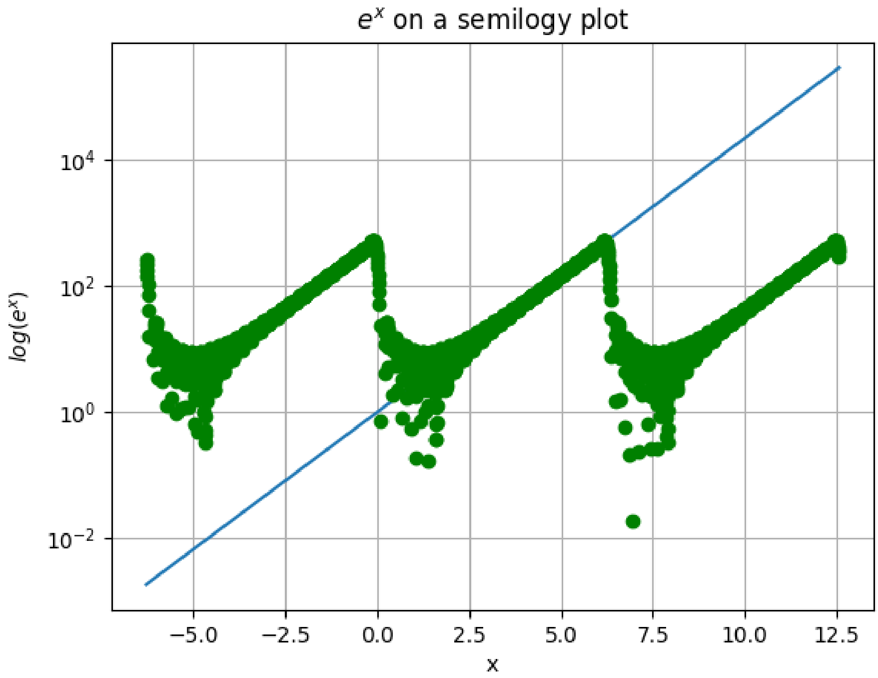
```
pi = np.pi


def exp(x):
    return np.exp(x)
def coscos(x):
    return np.cos(np.cos(x))
```

0.0.2 Generating Fourier Coefficients

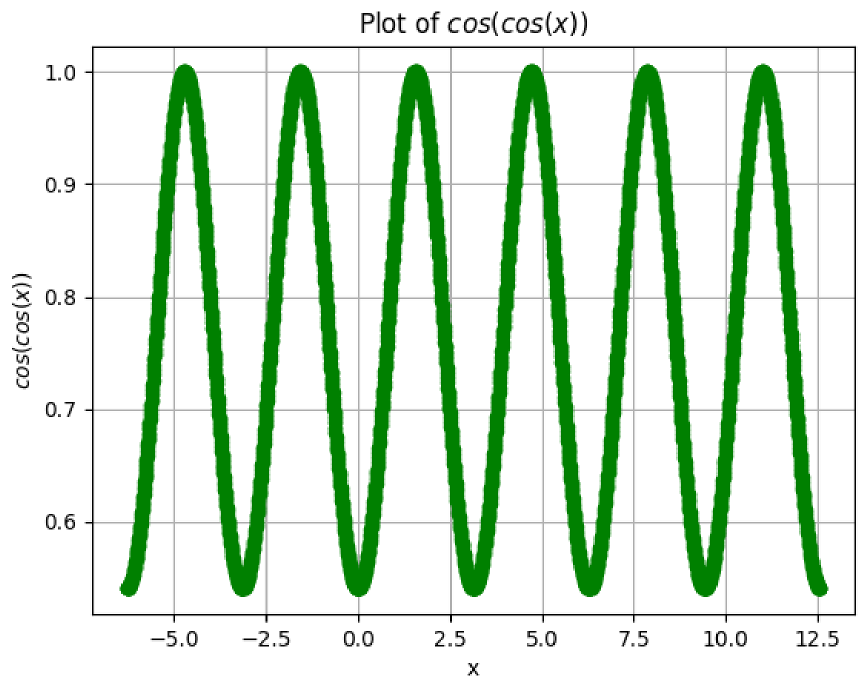
Obtaining the first 51 coefficients (excluding the Bn[0]=0) for the two functions above and the e^x and $cos(cos(x))$ is plotted.

```
x = np.arange(-2*pi, 4*pi, 0.01)
# x_fourier= np.arange(0,2*pi,0.01) #if we want to restrict
      the graphs
# print(x)
def fc_exp(x): return exp(x)*cos(i*x)  # i dummy index
def fs_exp(x): return exp(x)*sin(i*x)


n = 26  # max value of I, not taken infinity, better result
      with high value

Anexp = []  # defining array

Bnexp = []

sumexp = 0

for i in range(n):
    an = quad(fc_exp, 0, 2*pi)[0]*(1.0/np.pi)
    Anexp.append(an)

for i in range(n):
    bn = quad(fs_exp, 0, 2*pi)[0]*(1.0/np.pi)
    Bnexp.append(bn)  # putting value in array Bn

for i in range(n):
    if i == 0.0:
        sumexp = sumexp+Anexp[i]/2
    else:
        sumexp =     sumexp+(Anexp[i]*np.cos(i*x)+Bnexp[i]*
           np.sin(i*x))


def fc_coscos(x): return coscos(x)*cos(i*x)  # i dummy index
def fs_coscos(x): return coscos(x)*sin(i*x)


Ancoscos = []  # defining array

Bncoscos = []

sumcoscos = 0

for i in range(n):
```

```
    an1 = quad(fc_coscos, 0, 2*pi)[0]*(1.0/np.pi)
    Ancoscos.append(an1)

for i in range(n):
    bn1 = quad(fs_coscos, 0, 2*pi)[0]*(1.0/np.pi)
    Bncoscos.append(bn1)  # putting value in array Bn

or i in range(n):
    if i == 0.0:
        sumcoscos = sumcoscos+Ancoscos[i]/2
    else:
        sumcoscos =      sumcoscos+(Ancoscos[i]*np.cos(i*x)+
            Bncoscos[i]*np.sin(      i*x))

plt.figure(1)
plt.semilogy(x, exp(x))
plt.semilogy(x, sumexp, 'og')
plt.title(r'$e^x$ on a semilogy plot')
plt.xlabel('x')
plt.ylabel(r'$log(e^x)$')
plt.grid()
plt.show()

plt.figure(2)
plt.plot(x, coscos(x))
plt.plot(x, sumcoscos, 'og')
plt.title(r'Plot of $cos(cos(x))$')
plt.xlabel('x')
plt.ylabel(r'$cos(cos(x))$')
plt.grid()
plt.show()
```



e^x on a semilogy plot
$log(e^x)$
10^4
10^2
10^0
10^{-2}
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
x



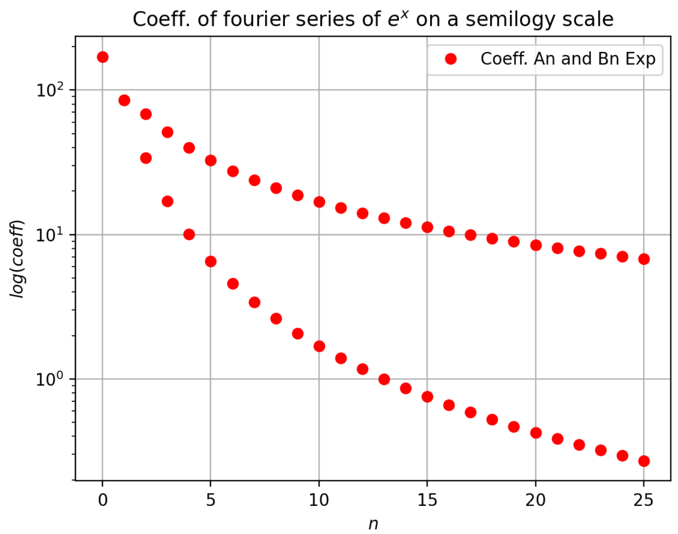
Plot of $cos(cos(x))$
$cos(cos(x))$
1.0
0.9
0.8
0.7
0.6
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
x

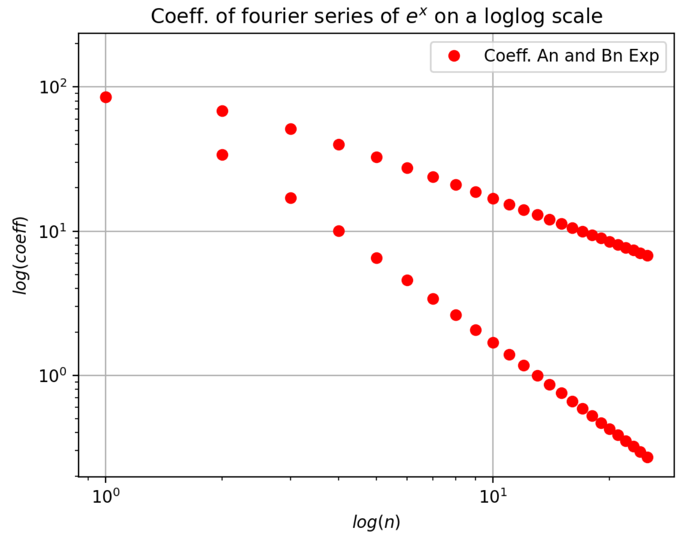
0.0.3 Plotting Semilog and Loglog of Fourier Coeff. for Exponentinal and Coscos Function

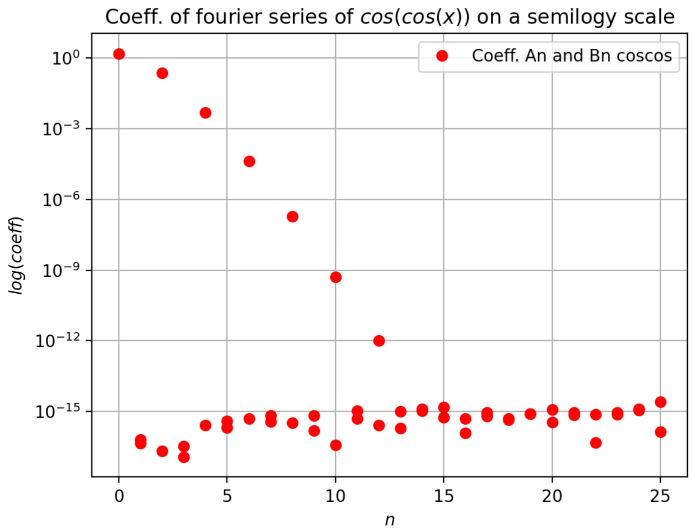
a)the Bn coeff. are nearly zero for $cos(cos(t))$ since it is an even function b)For the case of $exp(x)$, as it increases exponentially, it has many frequency components and hence the Fourier Coefficients do not die out easily for higher frequencies. Whereas, for the case of $cos(cos(x))$, it has a low frequency of $\frac{1}{\pi}$ and does not have higher frequency components. Hence, the coefficients decay quickly for the second case.
c)the loglog plot is linear for e^t since fourier coefficients decay asymptotically.The semilog plot is linear as shown in the plot.

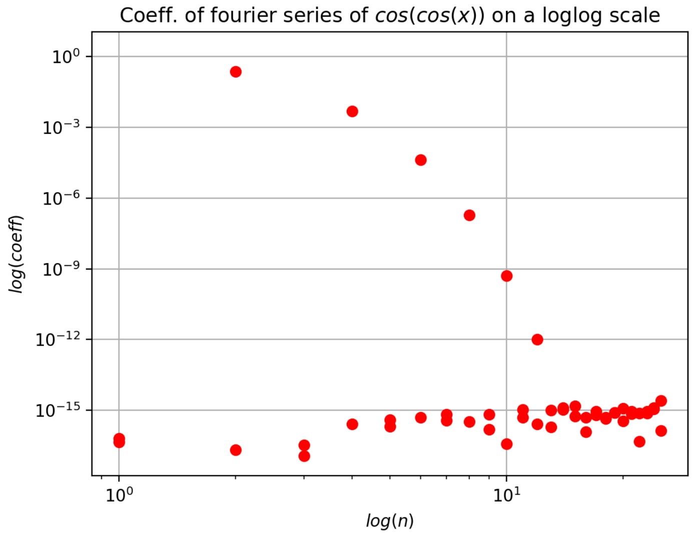
```
plt.figure(3)
plt.semilogy(np.arange(0, n, 1), np.abs(Anexp),
                "ro",label="Coeff. An Exp")
plt.semilogy(np.arange(0, n, 1), np.abs(Bnexp), "ro",
    abel="Coeff. Bn Exp")
plt.title(r"Coeff. of fourier series of $e^x$ on a semilogy 
        scale")
plt.xlabel(r'$n$')
plt.ylabel(r'$log(coeff)$')
plt.legend()
plt.grid()
plt.show()

plt.figure(4)
plt.loglog(np.arange(0, n, 1), np.abs(Anexp), "ro",
    label="Coeff. An Exp")
plt.loglog(np.arange(0, n, 1), np.abs(Bnexp), "og",
    label="Coeff. Bn Exp")
plt.title(r"Coeff. of fourier series of $e^x$ on a loglog   
      scale")
plt.xlabel(r'$log(n)$')
plt.ylabel(r'$log(coeff)$')
plt.legend()
plt.grid()
plt.show()

plt.figure(5)
plt.semilogy(np.arange(0, n, 1), np.abs(
     Ancoscos), "ro",
     label="Coeff. An coscos")
plt.semilogy(np.arange(0, n, 1), np.abs(
     Bncoscos), "og",
     label="Coeff. Bn coscos")
plt.title(r"Coeff. of fourier series of $cos(cos(x))$ on a  
       semilogy scale")
plt.xlabel(r'$n$')
```

```
plt.ylabel(r'$log(coeff)$')
plt.legend()
plt.grid()
plt.show()

plt.figure(6)
plt.loglog(np.arange(0, n, 1), np.abs(Ancoscos),
            "ro",
            label="Coeff. An Coscos")
plt.loglog(np.arange(0, n, 1), np.abs(Bncoscos),
            "og",
            label="Coeff. Bn Coscos")
plt.title(r"Coeff. of fourier series of $cos(cos(x))$ on a
    loglog scale")
plt.xlabel(r'$log(n)$')
plt.ylabel(r'$log(coeff)$')
plt.grid()
plt.show()
```




Coeff. of fourier series of e^x on a loglog scale
Coeff. An and Bn Exp
10^2
10^1
10^0
log(coeff)
10^0
10^1
log(n)


Coeff. of fourier series of cos(cos(x)) on a semilogy scale
Coeff. An and Bn coscos
10^0
10^{-3}
10^{-6}
10^{-9}
10^{-12}
10^{-15}
log(coeff)
0
5
10
15
20
25
n



0.0.4 Finding Coeff. Using Least square Method

The Coefficients are estimated using the least square method. The result we get will not be perfect.The true values of exponential and coscos are plotted along with the estimated coefficients using the least square method in semi-log and loglog.

$$Ac = b$$

where,

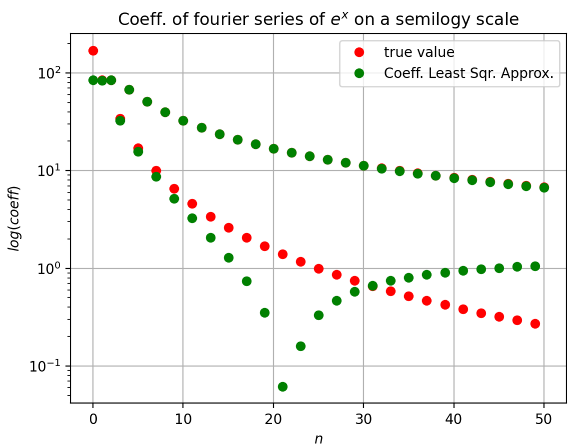
$$A = \begin{pmatrix} 1 & \cos(x_1) & \sin(x_1) & \dots & \cos(25x_1) & \sin(25x_1) \\ 1 & \cos(x_2) & \sin(x_2) & \dots & \cos(25x_2) & \sin(25x_2) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & \cos(x_{400}) & \sin(x_{400}) & \dots & \cos(25x_{400}) & \sin(25x_{400}) \end{pmatrix}$$

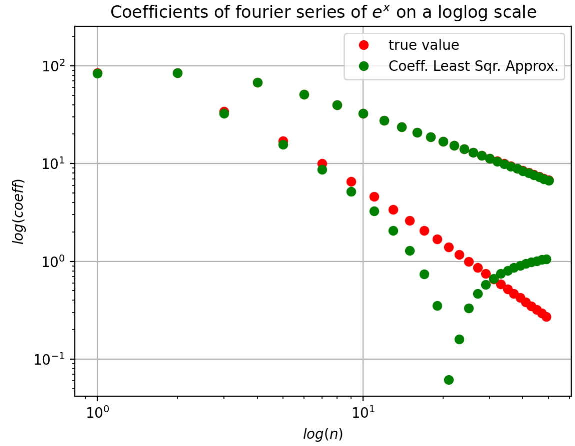
$$b = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \dots \\ f(x_{400}) \end{pmatrix}$$

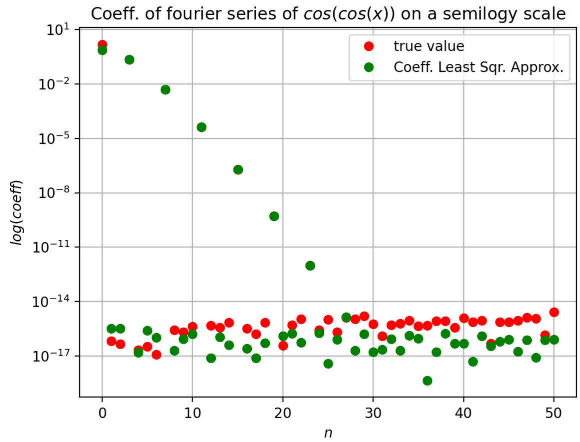
```
x1 = np.linspace(0, 2*pi, 401)
b = exp(x1)
x1 = x1[:-1]              # drop last term to have a proper
    periodic integral
```

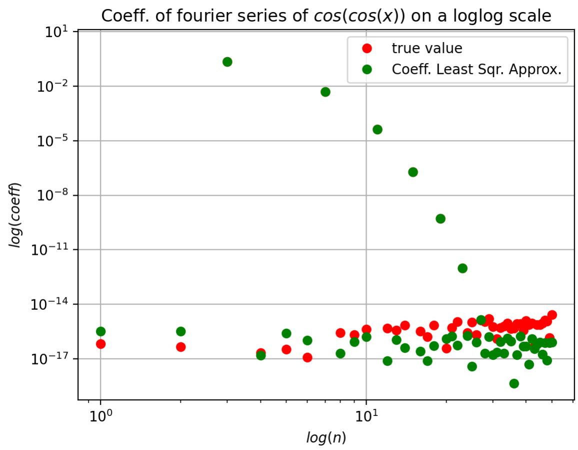
```
b = exp(x1)
A = np.zeros((400, 51))     # allocate space for A
A[:, 0] = 1                 # col 1 is all ones
for k in range(1, 26):
    A[:, 2*k-1] = np.cos(k*x1)    # cos(kx) column
    A[:, 2*k] = np.sin(k*x1)  # sin(kx) column
    # endfor
c1 = sp.linalg.lstsq(A, b)[0]       # the     [0]     is to pull out
     the
# best fit vector. lstsq returns a list.
# print(c1)

temp = []
temp.append(Anexp[0])
for i in range(1, 26):
    temp.append(Anexp[i])
    temp.append(Bnexp[i])
plt.figure(7)

plt.semilogy(np.arange(0, 51, 1), np.abs(temp), "ro")
plt.semilogy(np.arange(0, 51, 1), np.abs(c1), "og",
             label="Coeff. Least Sqr. Approx.")
plt.title(r"Coeff. of fourier series of $e^x$ on a semilogy
    scale")
plt.xlabel(r'$n$')
plt.ylabel(r'$log(coeff)$')
plt.legend()
plt.grid()
plt.show()

plt.figure(8)
plt.loglog(np.arange(0, 51, 1), np.abs(temp), "ro")
plt.loglog(np.arange(0, 51, 1), np.abs(c1), "og",
           label="Coeff. Least Sqr. Approx.")
plt.title(r"Coefficients of fourier series of $e^x$ on a loglog
    scale")
plt.xlabel(r'$log(n)$')
plt.ylabel(r'$log(coeff)$')
plt.legend()
plt.grid()
plt.show()

temp1 = []
temp1.append(Ancoscos[0])
for i in range(1, 26):
    temp1.append(Ancoscos[i])
    temp1.append(Bncoscos[i])

b1 = coscos(x1)
```

```
A1 = np.zeros((400, 51))       # allocate space for A
A1[:, 0] = 1                   # col 1 is all ones
for k in range(1, 26):
    A1[:, 2*k-1] = np.cos(k*x1)    # cos(kx) column
    A1[:, 2*k] = np.sin(k*x1)   # sin(kx) column
    # endfor
c2 = sp.linalg.lstsq(A1, b1)[0]

plt.figure(9)
plt.semilogy(np.arange(0, 51, 1), np.abs(temp1), "ro")
plt.semilogy(np.arange(0, 51, 1), np.abs(c2), "og",
             label="Coeff. Least Sqr. Approx.")
plt.title(r"Coeff. of fourier series of $cos(cos(x))$ on a 
    semilogy scale")
plt.xlabel(r'$n$')
plt.ylabel(r'$log(coeff)$')
plt.legend()
plt.grid()
plt.show()

plt.figure(10)
plt.loglog(np.arange(0, 51, 1), np.abs(temp1), "ro")
plt.loglog(np.arange(0, 51, 1), np.abs(c2), "og",
           label="Coeff. Least Sqr. Approx.")
plt.title(r"Coeff. of fourier series of $cos(cos(x))$ on a 
    loglog scale")
plt.xlabel(r'$log(n)$')
plt.ylabel(r'$log(coeff)$')
plt.legend()
plt.grid()
plt.show()
```









0.0.5 Comparing the both Methods

We can notice that there is some deviation of the Fourier Coefficients as estimated using Least Squares and by direct integration. We can treat the

coefficients estimated using direct integration as the true values because we use the exact formulae whereas the Least Squares coefficients are just estimates.
There is more deviation in the case of the exponential function as compared to $cos(cos(x))$ because the latter is a periodic function and we only take a periodic extension of the former.

0.0.6 Finding Maximum Deviation between both Functions and Printing Result

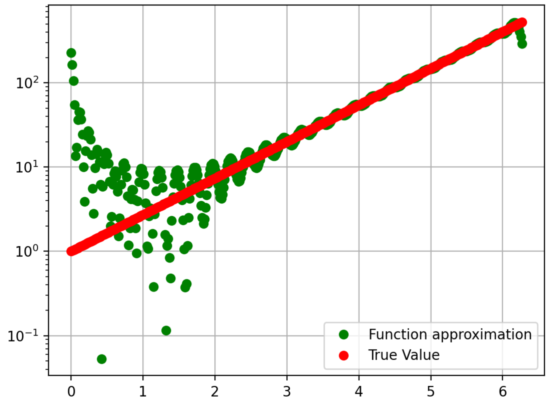
```
    exp_dev = []
    oscos_dev = []

for i in range(51):
    if i == 0:
        exp_dev.append(abs(c1[i]-temp[i]/2))
        coscos_dev.append(abs(c2[i]-temp1[i]/2))
    else:
        exp_dev.append(abs(c1[i]-temp[i]))
        coscos_dev.append(abs(c2[i]-temp1[i]))

print("Maximum deviation for exp(x): {}".format(max(exp_dev)))
print("Maximum deviation for coscos(x): {}".format(max(
    coscos_dev)))
```

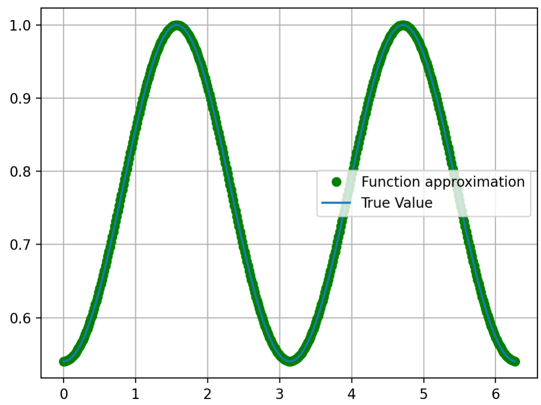
0.0.7 Plotting Results of Best Fit Matrix along with True

```
exp_approx = np.dot(A, c1)
coscos_approx = np.dot(A1, c2)

plt.figure(11)
plt.semilogy(x1, list(map(abs, exp_approx)),
             "go", label="Function approximation")
plt.semilogy(x1, exp(x1), "ro", label="True Value")
plt.legend()
plt.grid()
plt.show()


plt.figure(12)
plt.plot(x1, list(map(abs, coscos_approx)),
         "go", label="Function approximation")
plt.plot(x1, coscos(x1), label="True Value")
```

```
plt.legend()
plt.grid()
plt.show()
```

The matrix product Ac gives an estimate for the original functions using the coefficients estimated using Least Squares.
It can be noticed that there is considerable deviation for the case of $exp(x)$ whereas there is negligible deviation for the case of $cos(cos(x))$.
To approximate $exp(x)$ with greater accuracy, we need to consider components with higher frequencies.
This is not an issue for $cos(cos(x))$ because it is a periodic function.





Conclusion

In this assignment we approximated the functions $exp(x)$ and $cos(cos(x))$ using their respective Fourier Coefficients. We estimated the Fourier Coefficients using two different approaches, exact integration using the formulae, and using Least Squares. We understood that least square fitting can be helpful in simplifying the process but it is less accurate than using the integration method which is less efficient.