# Assignment No. 4: Report

-Akash Reddy A, EE17B001

February 27, 2019

## 1 Overview

Assignment 4 attempts to fit, over the interval $[0, 2\pi)$, two given functions

$$f_1(x) = exp(x)$$

and

$$f_2(x) = cos(cos(x))$$

using the Fourier Series:

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

where:

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

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

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

## 2 The Assignment

This report presents the assignment in a Part-by-Part approach.

### 2.1 Part 1

Two functions named `eexp` and `coscos` are defined using `numpy` in order to accept both scalars and vectors as inputs, and correspondingly return scalar and vector outputs.

`eexp` returns *exp(x)* and `coscos` returns *cos(cos(x))*, where $x$ is the argument to the respective function.

**CAVEAT:** If `math` library is used instead of `numpy`, the exponential and cosine functions cannot accept vector arguments or return vector outputs.

We create an array of $x$ values, which are uniformly spaced in the interval [-2$\pi$, 4$\pi$). The number of values we divide the interval into is arbitrary. (However, it is necessary that a large enough number of $x$ values is taken, so thay we may obtain a smooth curve to fit these points.) We choose a value of **1200 steps**.

Now, the values of *exp(x* and *cos(cos(x))* at the 1200 values of $x$ are calculated using the functions that have already been defined. The x values are given as a vector input, and we obtain the corresponding values of *exp(x)* and *cos(cos(x))* as a vector output each. The graphs are then plotted using `plot`. The data points themselves are not displayed in order to show a smooth curve.

Additionally, the plots of the functions that are expected from the Fourier series are also plotted. The values of these functions for the 1200 values of $x$ are calculated.

**A Fourier Series expansion is defined for a periodic function over an interval [0, T) and the function obtained in the interval is expected to repeat itself indefinitely (periodic extension) on either side of the interval.**

When we compare our Fourier Series expansion equations with the general form of the Fourier Series for a function defined in [0, T) and periodically extended on either side, the function *f(x)* can be represented as:

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

where:

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

it is understood that $T = 2\pi$ and $\omega = 1$.

Since the integrals for the Fourier Coefficients are calculated by integrating over [0, 2$\pi$), what we are indeed calculating using our Fourier Series expansion is the ***periodic extension* of the approximate function over the interval [0,2$\pi$)**.

Therefore, when we generate functions using the Fourier Series, what we expect for each function is the function evaluated approximately in [0, $2\pi$), **and a periodic extension of this segment on either side of this interval**. Hence, for:

- *exp(x)*, since we are plotting the function in the interval [-$2\pi$, $4\pi$), we expect three iterations of the function that we obtained in [0, $2\pi$) over [-$2\pi$,0), [0, $2\pi$) and [$2\pi$, $4\pi$) respectively. Hence we expect a **sawtooth like wave-form** due to the discontinuities at 0, $2\pi$ and $4\pi$, as the semilog plot of the original function is linear.
- *cos(cos(x))*, the original function itself is perfectly periodic over a period of $2\pi$. As a result, we expect the Fourier Series or the periodic extension over [-$2\pi$, $4\pi$) to perfectly align with our original function.

### 2.1.1 Relevant Code:

```
def eexp(i):
    return np.exp(i)
def coscos(i):
    return np.cos(np.cos(i))
```

`eexp` and `coscos` are the functions for *exp(x)* and *cos(cos(x))* that accept both scalar and vector inputs and return scalar and vector outputs respectively.

```
x=np.linspace(-2*pi,4*pi, 1200)
epoints=eexp(x)
coscospoints=coscos(x)
fourierexp=eexp(x%(2*pi))
fouriercoscos=coscos(x%(2*pi))
```

`x` contains the 1200 x values in the interval [-$2\pi$, $4\pi$) for which we calculate function values.

`epoints` is an array holding the values of *exp(x)*, `coscospoints` the values of *(cos(cos(x))*, `fourierexp` the values of the expected Fourier Series for *exp(x)* and `fouriercoscos` the values of the expected Fourier Series for *cos(cos(x))* respectively, calculated at the 1200 points.

### 2.1.2 Plots Generated:

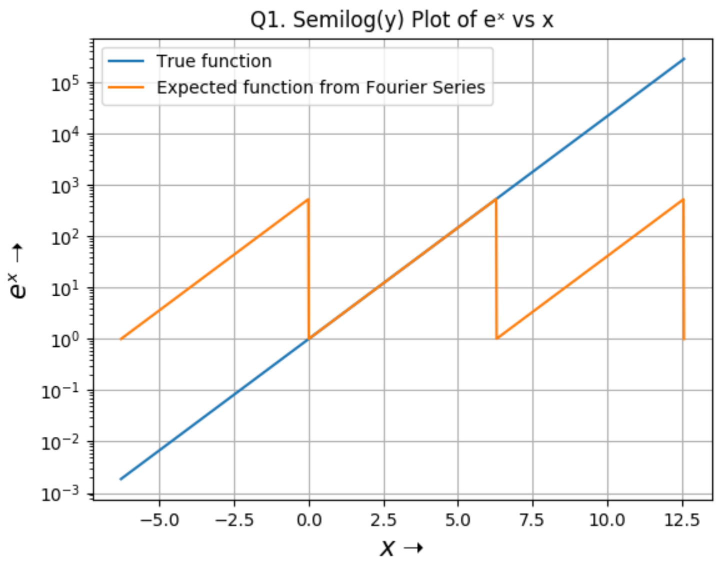

Figure 1: Semilog(y) plot of exp(x) vs x

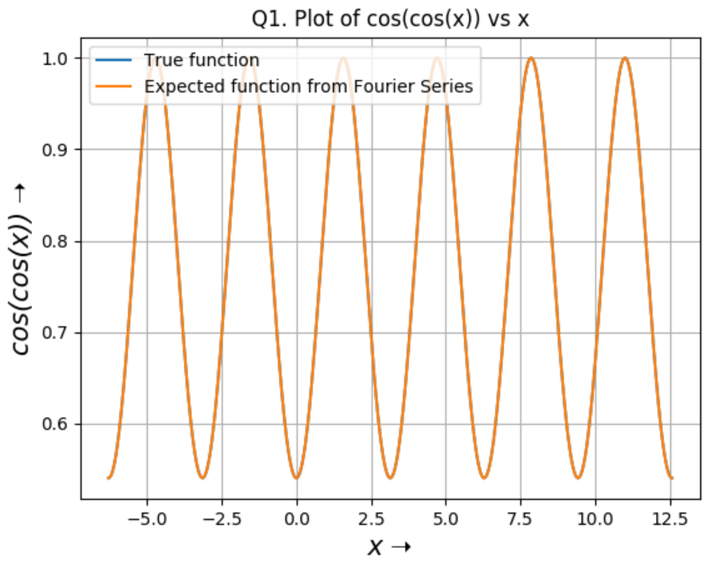

Figure 2: Plot of cos(cos(x)) vs x

### 2.2 Part 2

The first 51 Fourier coefficients for the two functions are generated as expected using `scipy.integrate.quad`.

We define 4 new functions, 2 *u(x,k)* and *v(x,k)* functions each for *exp(x)* and *cos(cos(x))*, which are `eexpu` and `eexpv`, and `coscosu` and `coscosv` respectively. This is done because these are the functions that have to be integrated over the interval $[0, 2\pi)$ in order to obtain our Fourier coefficients.

The Fourier coefficients, given according to:

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

are calculated.

### 2.2.1 Relevant Code:

```
def a_list(fna):
    a=[list(scint.quad(fna,0,2*pi,args=(n))) for n in range(26)]
    a=1/pi*array([a[i][0] for i in range(len(a))])
    a[0]=a[0]/2
    return(a)
def b_list(fnb):
    b=[list(scint.quad(fnb,0,2*pi,args=(n))) for n in range(26)]
    b.remove(b[0])
    b=1/pi*array([b[i][0] for i in range(len(b))])
    return(b)
```

The above two functions are defined for ease of calculation of Fourier coefficients. Each of these functions accepts a function as a parameter and calculates the $a_n$ and $b_n$ coefficients respectively for the function.

```
expa=a_list(eexpu)
expb=b_list(eexpv)
coscosa=a_list(coscosu)
coscosb=b_list(coscosv)
```

The above code computes the values of the $a_n$ and $b_n$ coefficients for *exp(x)* and *cos(cos(x))*.

## 2.3 Part 3

### 2.3.1 Relevant Code:

```
expco = [None]*(len(expa)+len(expb))
coscosco = [None]*(len(coscosa)+len(coscosb))
```

The above code is used to generate two vectors of `NoneType` so that they can later be filled up in the required order of $(a_0, a_1, b_1, a_2, b_2, ..., a_{25}, b_{25})$.

```
expco[0] = expa[0]
expco[1::2] = expa[1:]
expco[2::2] = expb

coscosco[0] = coscosa[0]
coscosco[1::2] = coscosa[1:]
coscosco[2::2] = coscosb

coscoscoabs=[abs(m) for m in coscosco]
expcoabs=[abs(m) for m in expco]
```

This code is used to fill up the `NoneType` vectors in the required order, and then take the magnitudes of the coefficients.

```
xaxis_n=[int((m+1)/2) for m in range(51)]
```

The above line of code is used to generate the list [0, 1, 1, 2, 2, 3, 3, ... , 24, 24, 25, 25], so that when `expco` and `coscosco` (which are of the form $(a_0, a_1, b_1, a_2, b_2, ..., a_{25}, b_{25})$) are plotted against this `xaxis_n`, both $a_n$ and $b_n$ are plotted against the correct value of n. The 51 coefficients map correctly to the 51 indices/x-axis n values of the list [0, 1, 1, 2, 2, ... , 24, 24, 25, 25].

**In the following plots, the red circles are relevant here. The green circles are the Fourier coefficients obtained through `lstsq` estimation, which will be relevant in Part 5 of the Assignment.**

#### 2.3.2 Plots Generated:

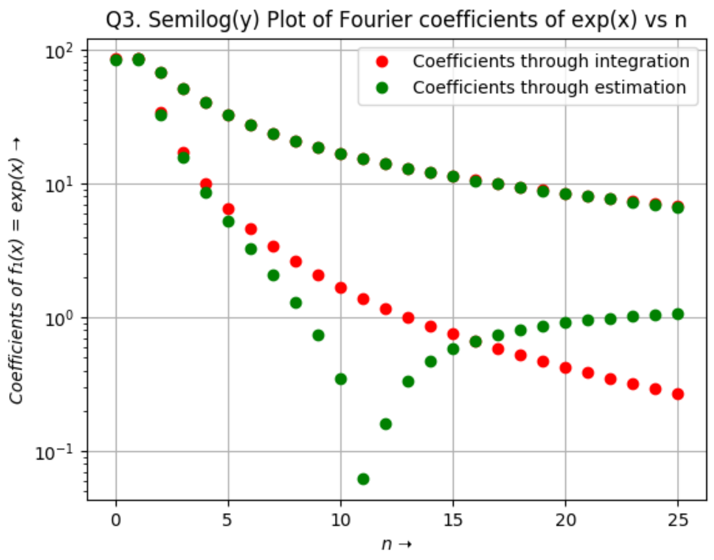

Figure 3: Semilog(y) plot of Fourier coefficients of exp(x) vs n

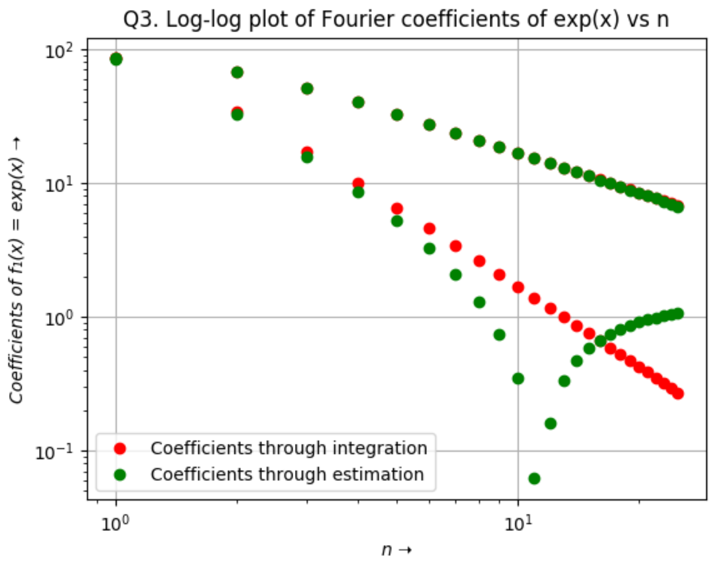

Figure 4: Log-log plot of Fourier coefficients of exp(x) vs n

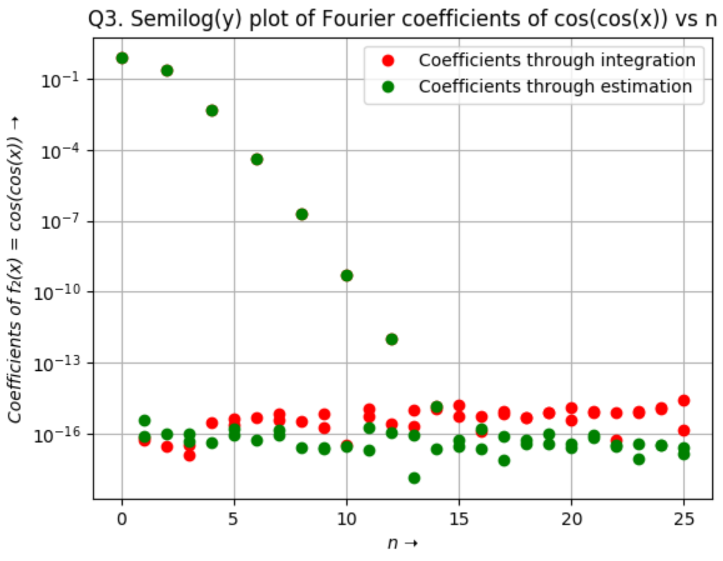

Figure 5: Semilog(y) plot of Fourier coefficients of cos(cos(x)) vs n

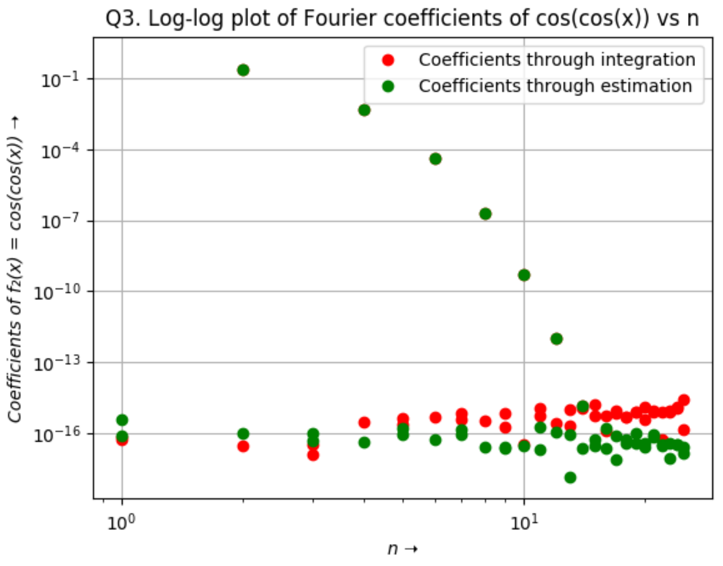

Figure 6: Log-log plot of Fourier coefficients of cos(cos(x)) vs n

### 2.3.3 Answers to the Questions:

a. The "second case" refers to the function $cos(cos(x))$. The $b_n$ coefficients are nearly zero in this case. **This is because the function is an even function.**
The function is also even in nature about x=$\pi$, which is the midpoint of the interval [0, $2\pi$) over which it is integrated in product with $sin(nx)$ in order to obtain $b_n$ coefficients.
However, $sin(nx)$ is odd in the same interval about x=$\pi$, therefore:

$$\begin{aligned} b_n &= \frac{1}{\pi}\int_0^{2\pi} f(x)sin(nx)dx \\ &= \frac{1}{\pi}\int_0^{\pi} f(x)sin(nx)dx + \frac{1}{\pi}\int_{\pi}^{2\pi} f(x)sin(nx)dx \\ &= \frac{1}{\pi}\int_0^{\pi} f(x)sin(nx)dx + (-\frac{1}{\pi}\int_0^{\pi} f(x)sin(nx)dx) \\ &= 0 \end{aligned}$$

This is why the $b_n$ coefficients are nearly zero.

b. In the case of $exp(x)$, we obtain a linear graph in the semilog(y) plot. In order to fit our Fourier Series to this function, **we require a larger number of Fourier sinusoid components** (in theory, we need infinite sinusoids to fit the line perfectly). This is why the Fourier coefficients decay very slowly, as we need nearly every frequency.

However, in the case of $cos(cos(x))$, the true function already oscillates nearly sinusoidally. As a result, **we do not need too many Fourier sinusoid components** in order to reach this true function through the Fourier Series. Only the first few frequencies are sufficient to accurately represent $cos(cos(x))$, therefore the coefficients decay rapidly as the Fourier Series does not need the later frequencies.

c. In Figure 4 (loglog plot for $exp(x)$), we plot $log(a_n)$ vs $log(n)$. This graph would be nearly linear, as we can see below:

$$
\begin{aligned}
a_n &= \frac{1}{\pi}\int_0^{2\pi} f(x)cos(nx)dx \\
&= \frac{1}{\pi}\int_0^{2\pi} exp(x)cos(nx)dx \\
&= \frac{1}{\pi}\int_0^{2\pi} exp(x)\frac{exp(jnx)+exp(-jnx)}{2}dx \\
&= \frac{e^{2\pi}-1}{\pi(1+n^2)} \\
log(a_n) &= log(\frac{e^{2\pi}-1}{\pi}) - log(1+n^2) \\
\implies log(a_n) &\approx log(\frac{e^{2\pi}-1}{\pi}) - 2log(n)
\end{aligned}
$$

*(similar analysis for $b_n$ coefficients)*

This is the equation of a straight line with negative slope, for the **loglog** plot in Figure 4.

In the case of Figure 5 (semilog(y) plot for $cos(cos(x))$), we see that the coefficients decay exponentially and quickly, because many Fourier frequencies are not needed in order to fit the near-sinusoidal $cos(cos(x))$. This would mean that:

$$
\begin{aligned}
a_n &\propto \frac{k}{e^n} \\
\implies log(a_n) &\approx logk' - n
\end{aligned}
$$

This is the equation of a straight line with negative slope, for the **semilog** plot in Figure 5.

## 2.4 Parts 4 and 5

### 2.4.1 Relevant Code:

```
x=linspace(0,2*pi,401)
x=x[:-1]
b1=eexp(x)
b2=coscos(x)
A=zeros((400,51))
A[:,0]=1
for k in range(1,26):
    A[:,2*k-1]=np.cos(k*x)
    A[:,2*k]=np.sin(k*x)
```

```
c1=lstsq(A,b1,rcond=None)[0]
c2=lstsq(A,b2,rcond=None)[0]
c1abs=[abs(m) for m in c1]
c2abs=[abs(m) for m in c2]
```

Using the above code, we first define an x vector in the interval [0, $2\pi$) in 400 steps, after which we calculate the functions' values at these points. We then generate the matrix equation $Ac = b$ and solve it using `lstsq`. Through this, we get the best-fit Fourier coefficients that generate the best-fit Fourier Series function passing through the function values we have just calculated.

`b1` and `b2` are the function values for $exp(x)$ and $cos(cos(x))$, and `c1` and `c2` are the best-fit Fourier coefficients for the respective functions.

Finally, c1abs and `c2abs` are used to take the *magnitudes* of the best-fit Fourier coefficients for the sake of the plots, as required.

#### 2.4.2 Plots Generated:

**The plots that are required in Part 5 of the Assignment are already shown under the part of the report dealing with Part 3 of the Assignment. The green circles are now relevant: they are the best-fit Fourier coefficients obtained here using `lstsq`.**

### 2.5 Part 6

We use the plots in Part 3 of this report to compare the Fourier coefficients obtained through integration (red circles), and those obtained through `lstsq` (green circles). We find that the $a_n$ coefficients agree almost perfectly, and the $b_n$ coefficients, while still pretty close, tend to deviate more from each other as $n$ increases. Deviation in general increases as $n$ increases.

The coefficients *should* nearly agree, as we are attempting to find the same values, albeit using two different processes. The coefficients obtained through integration are the perfect coefficients (except the small errors due to the errors of the `quad` function used for integration), and the coefficients obtained through `lstsq` are the best-fit coefficients, and deviate from the true values. Therefore, they are not expected to *exactly* agree.

We find that the largest absolute deviation between the coefficients for:

- $exp(x)$=1.3327308703353822
- $cos(cos(x))$=2.5816134072666375 x $10^{-15}$

#### 2.5.1 Relevant Code:

```
expmaxdev=max([abs(expco[m]-c1[m]) for m in range(51)])
```

```
coscosmaxdev=max([abs(coscosco[m]-c2[m]) for m in range(51)])

print('Largest absolute deviation in exp(x) coefficients='
,expmaxdev)
print('Largest absolute deviation in cos(cos(x)) coefficients='
,coscosmaxdev)
```

This code calculates the **maximum absolute deviation** between the coefficients obtained through integration and the best-fit coefficients. The plots of Figure 1 and Figure 2 obtained in Part 1 of the Assignment are essentially re-plotted with the values of the best-fit function marked in green circles.

## 2.6 Part 7

This part involves the calculation of the values at $x_i$ of the functions generated using the **estimated** Fourier Series.

### 2.6.1 Relevant Code:

```
expest=matmul(A,c1)
coscosest=matmul(A,c2)
epoints=eexp(x)
coscospoints=coscos(x)
fourierexp=eexp(x%(2*pi))
fouriercoscos=coscos(x%(2*pi))
```

The code now uses `numpy.matmul` in order to solve the matrix equation $Ac = b$ to obtain the function values $b$ using the best-fit estimated Fourier Series.

`epoints` and `coscospoints` are re-calculated because our array of x-axis values has changed (we are now working in the interval $[0, 2\pi)$ with 400 steps). The same goes for `fourierexp` and `fouriercoscos`.

### 2.6.2 Plots Generated:

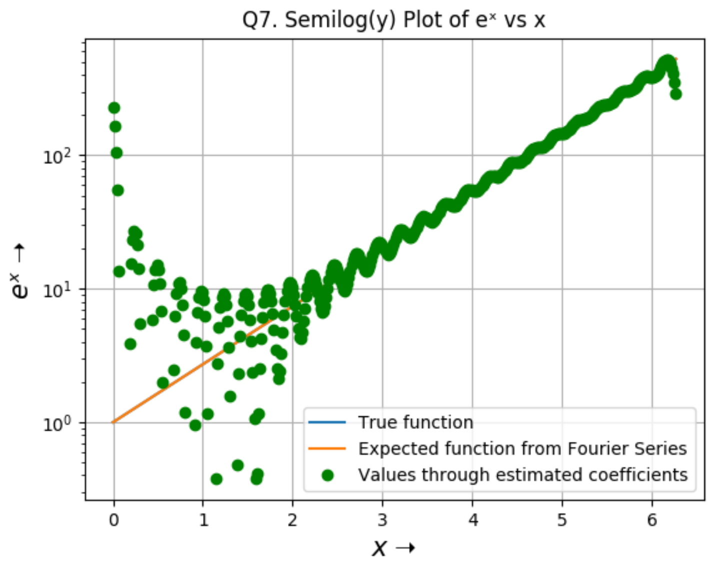

Figure 7: Semilog(y) plot of exp(x) vs x (Figure 1 re-plotted)

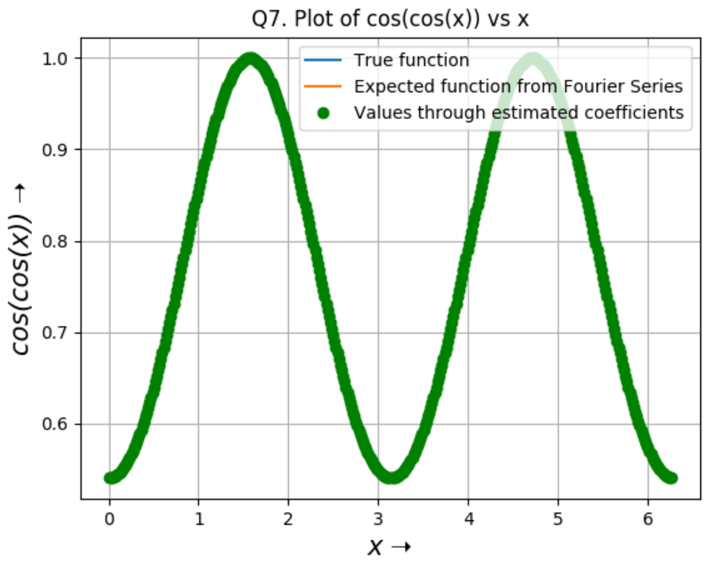

Figure 8: Plot of cos(cos(x)) vs x (Figure 2 re-plotted)

### 2.6.3 Answers to the Questions:

In Figure 1, there is significant deviation between the true function and the values through estimated Fourier Series. In Figure 2, they agree nearly perfectly.

- In Figure 1 for $exp(x)$, there is a huge deviation near $x = 0$, almost like a jump discontinuity. This is because our green points are values of the estimated **Fourier Series**.
  When we use Fourier Series we observe that, even in our expected function from Fourier Series (plotted using the red line in Figure 1), there is a discontinuity at $x = 0$.
  This discontinuity is a result of **periodic extension beyond the interval [0, 2$\pi$) to the left and right, and is not perfectly accounted for by our best-fit curve**, and it spills a little past x=0. This is why we get the initial disparity.

  However, in the case of $cos(cos(x))$ in Figure 2, the value of the function at $x = 0$ coincides with the value at $x = 2\pi$. Therefore there is **no jump discontinuity even due to periodic extension.** This is why we do not see such a disparity here.

- In Figure 1 for $exp(x)$, the green points oscillate in a somewhat sinusoidal fashion as they fit the linear curve. This is because we have taken only 51 frequencies of the Fourier Series to represent it, therefore the sinusoidal nature is still visible. It would take all the infinite component frequencies of the Fourier series in order for this to disappear. **In other words, it is more apparent as our true function is linear.**

  However, in the case of $cos(cos(x))$ in Figure 2, the true function itself is nearly sinusoidal. Therefore, it does not take the consideration of too many component frequencies for this error to disappear, as it becomes easier for sinusoids to align with the true function. **This is why this error is much less apparent in Figure 2, and there is near perfect agreement.**

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *