CANDIDATE CODE : KYBS6

BARC0130 : Advanced Mathematical Modelling & Analysis

2021-2022

Topic 1 : Series & Approximations

COURSEWORK 1

Appendices referenced as [number]

Part A: The Source Data

The curve chosen to study is the Walt Disney Castle. The original inspiration for the curve was taken from *Image 1*. This curve was then modified using Adobe Illustrator for it to obey Dirichlet's conditions i.e make it piece-wise continuous and the final curve to be used was the one shown in *Image 2.*

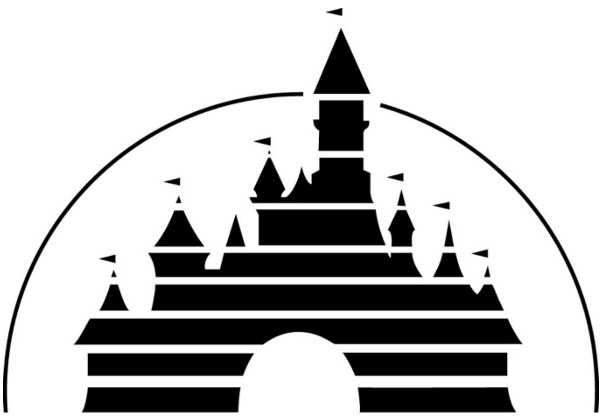


Image 1: Walt Disney Castle

Image 2: Modified Walt Disney Castle

The reason this curve was chosen was because on simplification (*Image 3*) it can be observed that it tends to form a very interesting asymmetrical step path which can be used to create aesthetical tessellations to be used in a design (*Image 4*).

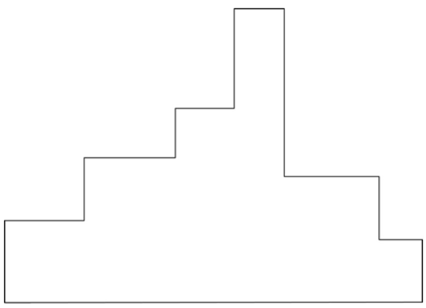


Image 3: Simplified Castle Outline

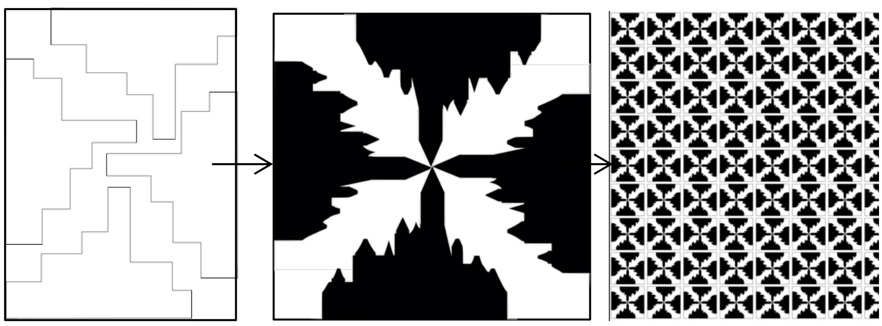


Image 4: Walt Disney Castle tessellated tiles[1]

Part B: Digitization

STEP 1-TRACE CURVE ON RHINO
To obtain Data Points of the selected curve, *Image 2* was imported to Rhino and the curve was traced using the polyline tool to reveal an outline of the curve.

STEP 2-SCALE THE CURVE TO FIT INDEPENDENT VARIABLES FROM 0 TO 2π
Using Grasshopper, a line of 2π units is created and baked. The polyline curve is then scaled down to this line.

STEP 3-PRODUCE COORDINATES OF THE CURVE
The line is then divided into different parts (controlled by a slider) and vertical lines are generated from these points. The points of intersection of these vertical lines with the polyline curve are recorded in a panel with separated x and y coordinates.

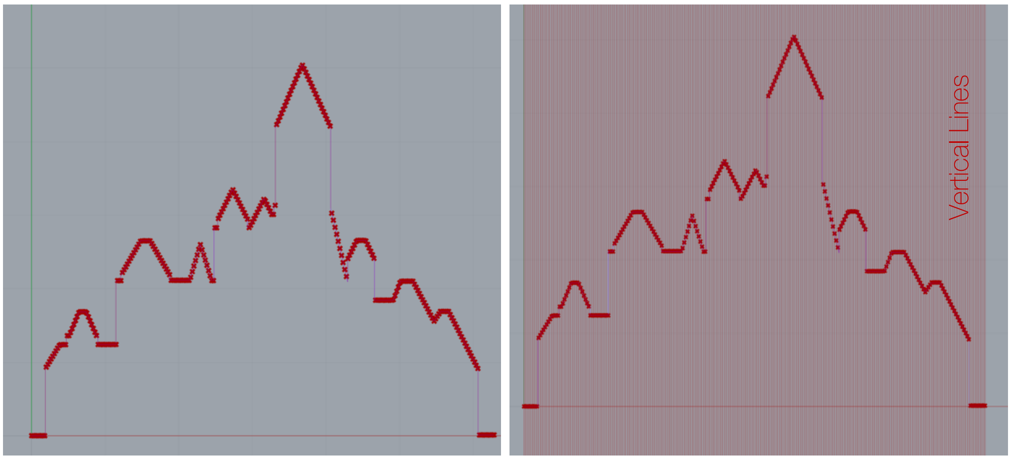


Image 5: Digitization of points on Rhino using Grasshopper [1]

STEP 4-EXPORTING TO EXCEL
The values recorded in the panel are copied and pasted onto an excel sheet which is later uploaded onto MATLAB to be used in a code. These values have been recorded in an excel sheet attached titled *'Fourier_Approximation_Values.xlsx'*.

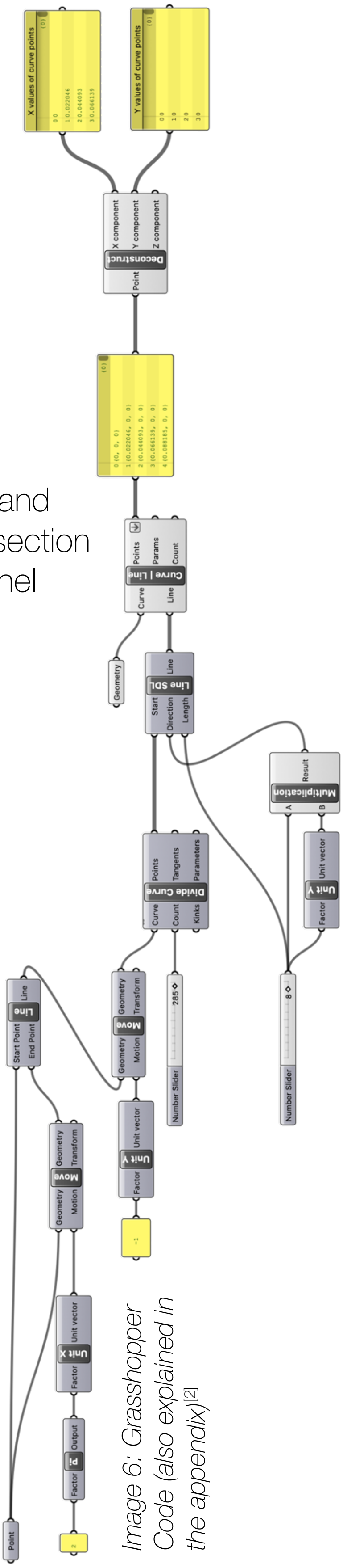


Image 6: Grasshopper Code (also explained in the appendix) [2]

Part C: Modelling

The Fourier Approximation for the selected curve is carried out using the 'Numerical Harmonic Analysis' Method and computed using MATLAB.

At first the number of Data Points collected (m) from the chosen curve is recorded.

m = 286

The data points are represented by x_i and y_i wherein x_i represents the array of x coordinates and y_i represents the array of the y coordinates.

Then, the total number of iterations of the Fourier Approximation (n) is decided.

In order to calculate the Fourier Approximation, the Fourier coefficients a_0 , a_n and b_n are evaluated using the following equations:

$a_0 = 2/m\ (\sum_{im=1}^{m} y_i)$

$a_n = 2/m\ ((\sum_{im=1}^{m} y_i\ \cos(nx_i))$

$b_n = 2/m\ ((\sum_{im=1}^{m} y_i\ \sin(nx_i))$

Here a_n and b_n are evaluated using 'for loops' in MATLAB.

The Fourier Approximation ($f_n(x_i)$) is calculated then using the following equation:

$f_n(x_i) \simeq a_0/2. + \sum_{jn=1}^{n}(a_n\cos(nx_i) + b_n\sin(nx_i))$

Then the quality of the model is assessed on the basis of the difference between the Fourier Approximations generated and the individual Data Points and are represented by V_i.

$V_i = y_i - f_n(x_i)$

We also calculate the RMS of fit:

$$\text{RMS of fit} = \sqrt{\frac{\sum_{im=1}^{m}(V_i)^2}{m}}$$

Initially, the number of iterations taken was 20, i.e., n = 20

The following results were obtained[3]:

a_0 = 4.5452 (remains constant for all values of n)
a_n = an array of 20 values
b_n = an array of 20 values
$f_n(x_i)$ = an array of 286 values

We plot the values of the Fourier Approximation alongside the original curve (*Image 7*).

The V_i values obtained are plotted to show the deviation of the Fourier Approximated values from the original Data Points (*Image 8*).

These values go upto 0.5 indicating that the Fourier Approximation is not yet extremely accurate.

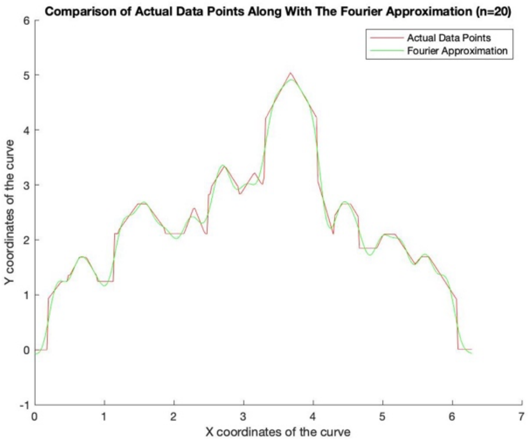


Image 7

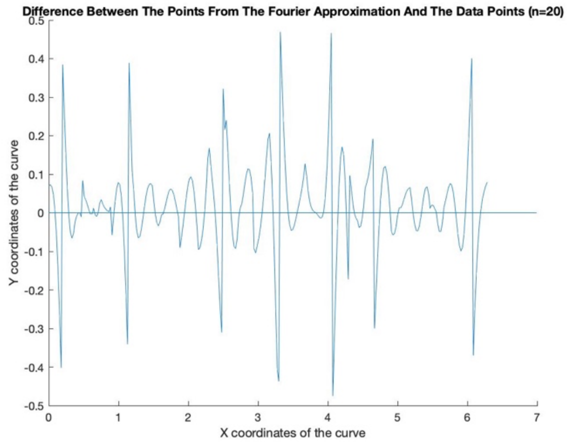


Image 8

The RMS of fit = 0.1259

For the number of iterations taken as 100, i.e., n = 100 we obtained the following results[4]:

We plot the values of the Fourier Approximation alongside the original curve (*Image 9*).

The V_i values obtained are plotted to show the deviation of the Fourier Approximated values from the original Data Points (*Image 10*).

These values are significantly less that the previous case as the go upto a highest of about 0.2. So we can observe as we increase the value of ‘n’, the quality of approximation improves.

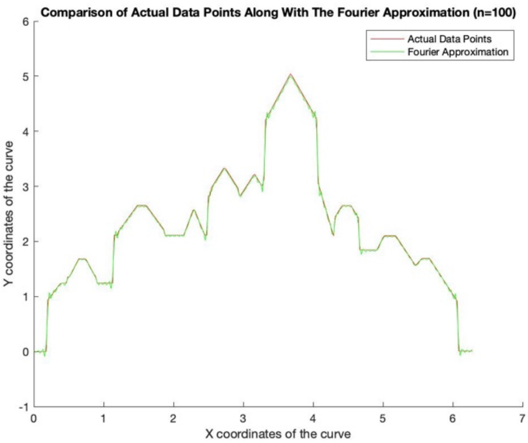


Image 9

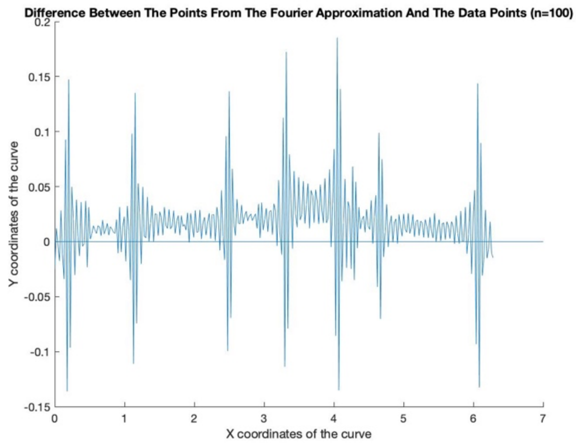


Image 10

The RMS of fit = 0.0432

The number of iterations (n) were adjusted and it was observed that the lowest value for RMS of fit was recorded at n = 143

For the number of iterations taken as 143, i.e., n = 143 we obtained the following results[5]:

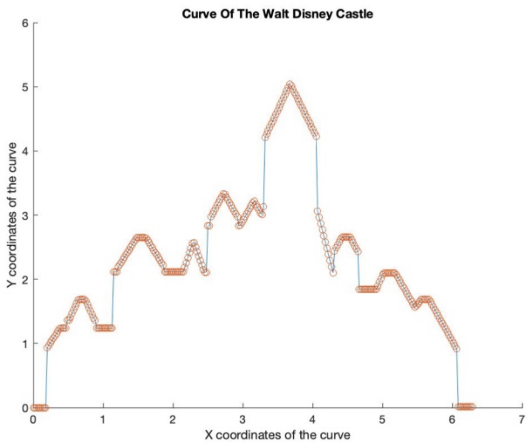


Image 11: Original Curve From Data Points

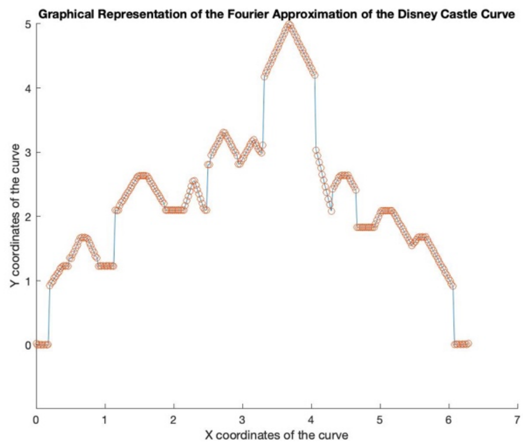


Image 12: Curve From Fourier Approximation

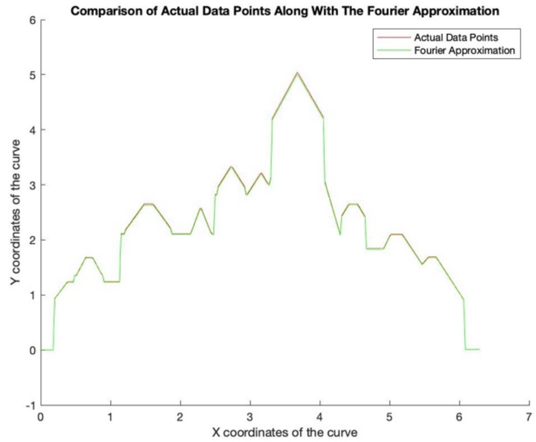


Image 13: Comparison of Original Curve and Fourier Curve

The V_i values obtained are plotted to show the deviation of the Fourier Approximated values from the original Data Points (*Image 14*).

These are significantly lower than the ones observed for n=20 and n=100.

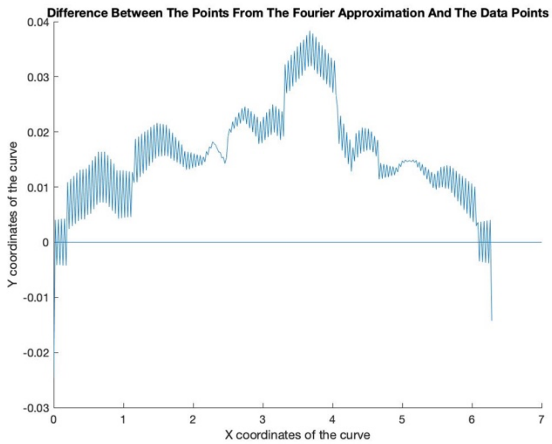


Image 14

The RMS of fit = 0.0182.

This low value of RMS of fit suggests that the difference between the actual data point values and the Fourier Approximation values is low (with a maximum of 0.04), and hence we can access the quality of the curve to be good.

The values of a_n, b_n, $f_n(x_i)$ and V_i have been recorded in an excel sheet attached by the name of *'Fourier_Approximation_Values.xlsx'*.

It was noted that while adjusting the number of iterations (n), the lowest value for the RMS of fit, i.e., the most accurate Fourier Approximation occurred when the number of iterations was exactly half of the total number of Data Points, that is, when n = m/2.

Part D: Curve Generation

The Fourier curve obtained in Part C is then transposed and plotted from x values 0 to 8π [6]. This is demonstrated in *Image15.*

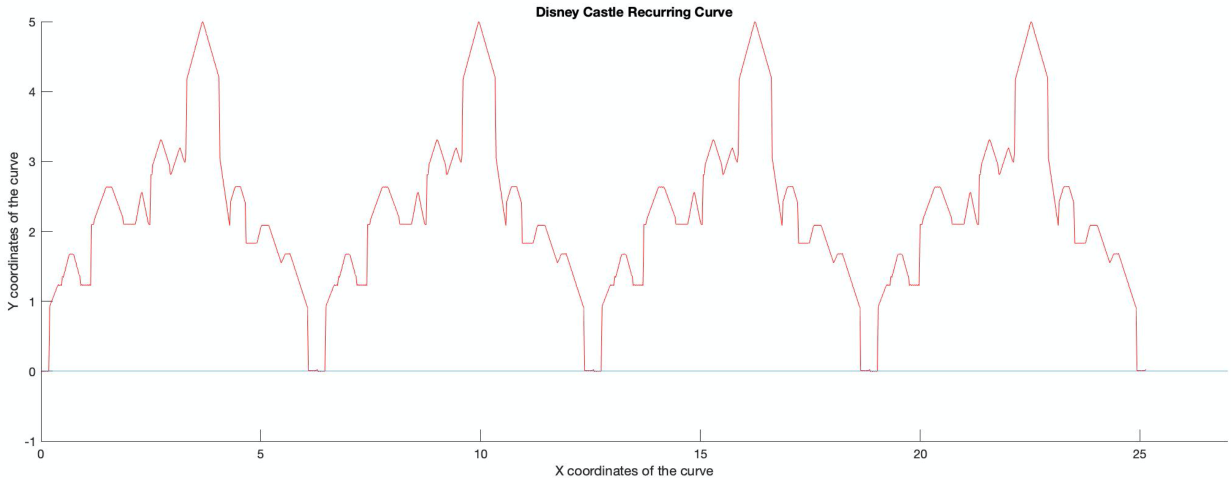
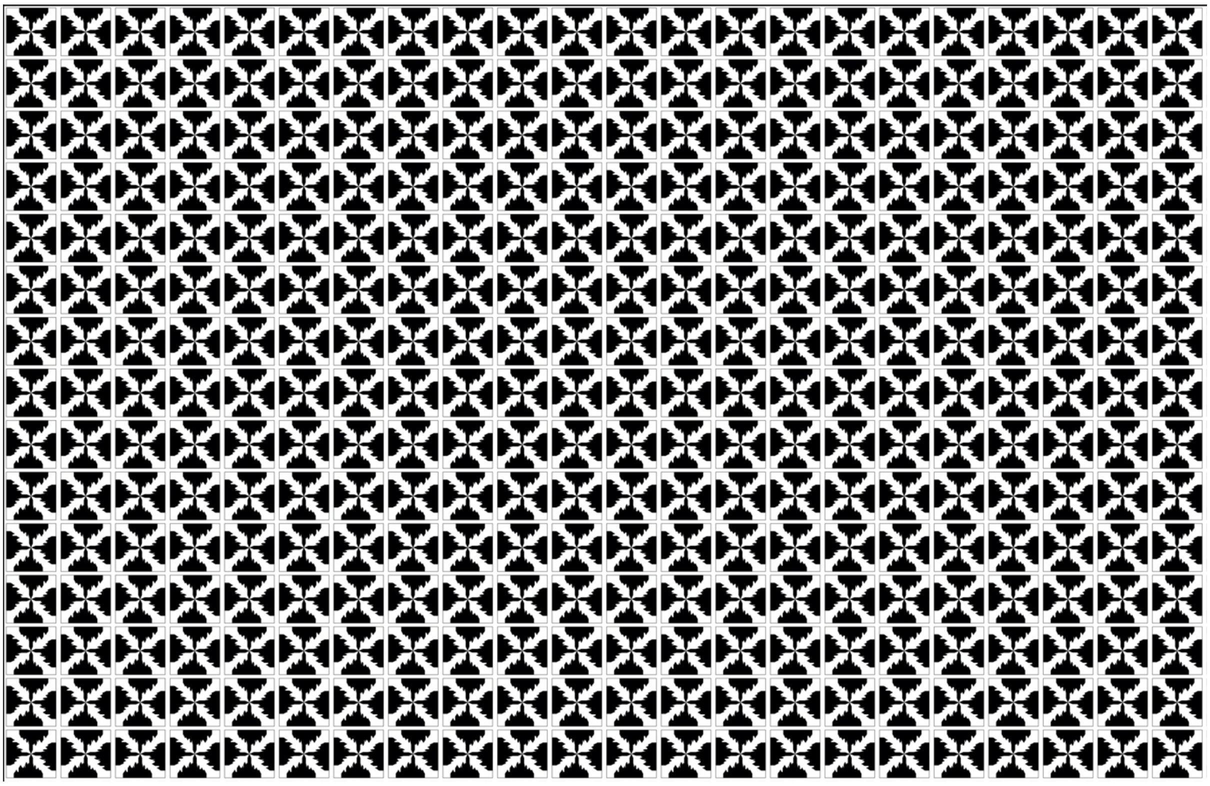


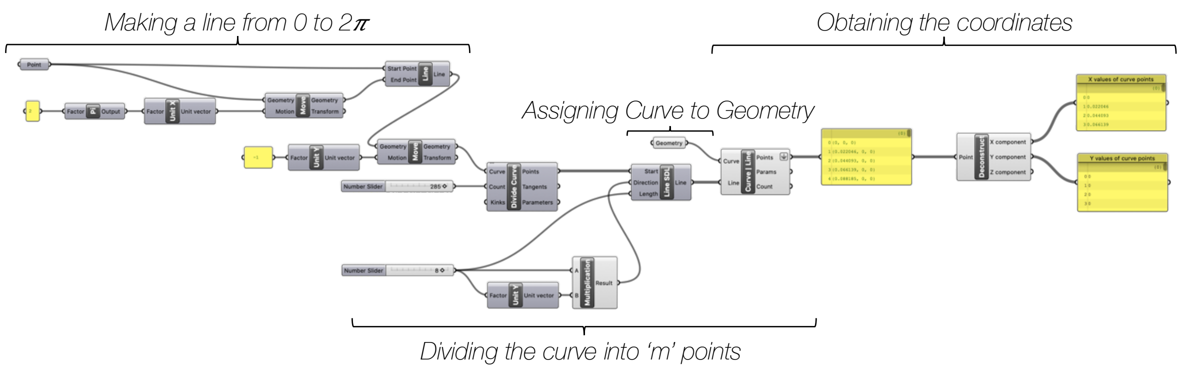
Image 15

Appendix

1. Tessellated Tile : Example of an application of using the selected curve in a design:



2. Grasshopper Code For Generating Data Points:



3. Matlab Code for n = 20

```
% Upload data of the coordinates of the castle from Excel onto MATLAB
% Name it 'castle_coordinates'

xl_filename = fullfile(pwd,'DATASETS','Disney_castle_coordinates.xlsx');
castle_coordinates = xlsread(xl_filename);
```

```
% Separate x and y coordinates and

x = (castle_coordinates(:,1))';
y = (castle_coordinates(:,2))';

% Define 'm' as the total number of data points collected

m = length(castle_coordinates);

% Define 'n' as the number of iterations

n = 144;

% Define Fourier Coefficient a_0

a_0 = (2/m)*sum(y);

% Defining a loop to obtain values of Fourier Coefficients a_n and b_n

for in = 1:n

    s_an = 0;
    s_bn = 0;

    for im = 1:m

    s_an = s_an + ((y(im))*cos(in*(x(im))));
    s_bn = s_bn + ((y(im))*sin(in*(x(im))));

    end
   an(in) = (2/m)*(s_an);
   bn(in) = (2/m)*(s_bn);

end

% Carry our fourier harmonic analysis by defining the fourier expansion

fn(im) = 0;

 for jm = 1:m

    s_fn = a_0/2 ;

    for jn = 1:n

    s_fn = s_fn + (an(jn)*(cos(jn*x(jm)))+(bn(jn)*sin(jn*x(jm)))) ;

    end
    fn(jm) = s_fn;
 end

% Plot the castle_coordinates and fourier approximation on the same graph

f1 = figure;
hold on

plot(x,y,'r')
```

```
plot (x,fn,'g')

xlabel ('X coordinates of the curve')
ylabel ('Y coordinates of the curve')
title ('Comparison of Actual Data Points Along With The Fourier
Approximation (n=20)')
legend ('Actual Data Points','Fourier Approximation')

hold off

% Assessing the quality of the Fourier Model using the difference between
% the series to the nth term and the actual data points

for im = 1:m
    V_i(im) = y(im) - fn(im);
end

% Plotting the diff

f2 = figure;
hold on

plot (x, V_i)
line(xlim(), [0,0])

xlabel ('X coordinates of the curve')
ylabel ('Y coordinates of the curve')
title ('Difference Between The Points From The Fourier Approximation And
The Data Points (n=20)')

hold off

% Calculating the RMS of fit

rms_fit = sqrt((sum ((V_i).^2)/m));
```

4. *Matlab Code for n = 100*

```
xl_filename = fullfile(pwd,'DATASETS','Disney_castle_coordinates.xlsx');
castle_coordinates = xlsread(xl_filename);

% Separate x and y coordinates

x = (castle_coordinates(:,1))';
y = (castle_coordinates(:,2))';

% Define 'm' as the total number of data points collected

m = length(castle_coordinates);

% Define 'n' as the number of iterations

n = 143;

% Define Fourier Coefficient a_0
```

```
a_0 = (2/m)*sum(y);

% Defining a loop to obtain values of Fourier Coefficients a_n and b_n

for in = 1:n

    s_an = 0;
    s_bn = 0;

    for im = 1:m

    s_an = s_an + ((y(im))*cos(in*(x(im))));
    s_bn = s_bn + ((y(im))*sin(in*(x(im))));

    end
   an(in) = (2/m)*(s_an);
   bn(in) = (2/m)*(s_bn);

end

% Carry our fourier harmonic analysis by defining the fourier expansion

fn(im) = 0;

 for jm = 1:m

    s_fn = a_0/2 ;

    for jn = 1:n

    s_fn = s_fn + (an(jn)*(cos(jn*x(jm)))+(bn(jn)*sin(jn*x(jm)))) ;

    end
    fn(jm) = s_fn;
 end

% Plot the castle_coordinates and fourier approximation on the same graph

f1 = figure;
hold on

plot(x,y,'r')
plot (x,fn,'g')

xlabel ('X coordinates of the curve')
ylabel ('Y coordinates of the curve')
title ('Comparison of Actual Data Points Along With The Fourier
Approximation (n=100)')
legend ('Actual Data Points','Fourier Approximation')

hold off

% Assessing the quality of the Fourier Model using the difference between
% the series to the nth term and the actual data points

for im = 1:m
    V_i(im) = y(im) - fn(im);
end
```

```
% Plotting the diff

f2 = figure;
hold on

plot (x, V_i)
line(xlim(), [0,0])

xlabel ('X coordinates of the curve')
ylabel ('Y coordinates of the curve')
title ('Difference Between The Points From The Fourier Approximation And
The Data Points (n=100)')

hold off

% Calculating the RMS of fit

rms_fit = sqrt((sum ((V_i).^2)/m));
```

5. Matlab Code for n = 143

```
% Sara Motwani
% Coursework 1 Series and Approximations

% Upload data of the coordinates of the castle from Excel onto MATLAB
% Name it 'castle_coordinates'

xl_filename = fullfile(pwd,'DATASETS','Disney_castle_coordinates.xlsx');
castle_coordinates = xlsread(xl_filename);

% Separate x and y coordinates and

x = (castle_coordinates(:,1))';
y = (castle_coordinates(:,2))';

% Plot the castle_coordinates to reveal the castle shaped graph
f1 = figure;
hold on

plot(x,y)
scatter(x,y)

xlabel ('X coordinates of the curve')
ylabel ('Y coordinates of the curve')
title ('Curve Of The Walt Disney Castle')

hold off

% Define 'm' as the total number of data points collected

m = length(castle_coordinates);

% Define 'n' as the number of iterations

n = 143;
```

```
% Define Fourier Coefficient a_0

a_0 = (2/m)*sum(y);

% Defining a loop to obtain values of Fourier Coefficients a_n and b_n

for in = 1:n

    s_an = 0;
    s_bn = 0;

    for im = 1:m

    s_an = s_an + ((y(im))*cos(in*(x(im))));
    s_bn = s_bn + ((y(im))*sin(in*(x(im))));

    end
   an(in) = (2/m)*(s_an);
   bn(in) = (2/m)*(s_bn);

end

% Carry our fourier harmonic analysis by defining the fourier expansion

fn(im) = 0;

 for jm = 1:m

    s_fn = a_0/2 ;

    for jn = 1:n

    s_fn = s_fn + (an(jn)*(cos(jn*x(jm)))+(bn(jn)*sin(jn*x(jm)))) ;

    end
    fn(jm) = s_fn;
 end

% Plotting the fourier approximation

f2 = figure;
hold on

plot (x,fn)
scatter (x,fn)

xlabel ('X coordinates of the curve')
ylabel ('Y coordinates of the curve')
title ('Graphical Representation of the Fourier Approximation of the Disney
Castle Curve')

hold off

% Plot the castle_coordinates and fourier approximation on the same graph

f3 = figure;
hold on
```

```
plot(x,y,'r')
plot (x,fn,'g')

xlabel ('X coordinates of the curve')
ylabel ('Y coordinates of the curve')
title ('Comparison of Actual Data Points Along With The Fourier
Approximation')
legend ('Actual Data Points','Fourier Approximation')

hold off

% Assessing the quality of the Fourier Model using the difference between
% the series to the nth term and the actual data points

for im = 1:m
    V_i(im) = y(im) - fn(im);
end

% Plotting the diff

f4 = figure;
hold on

plot (x, V_i)
line(xlim(), [0,0])

xlabel ('X coordinates of the curve')
ylabel ('Y coordinates of the curve')
title ('Difference Between The Points From The Fourier Approximation And
The Data Points')

hold off

% Calculating the RMS of fit

rms_fit = sqrt((sum ((V_i).^2)/m));
```

6. Matlab Code For Part D

```
% Plotting the curve from 0 to 8pi

f5 = figure;
hold on

plot (x,fn, 'r')
plot (x+(2*pi),fn,'r')
plot (x+(4*pi),fn,'r')
plot (x+(6*pi),fn, 'r')

line(xlim(), [0,0])

xlabel ('X coordinates of the curve')
ylabel ('Y coordinates of the curve')
title ('Disney Castle Recurring Curve')

hold off
```