MA441Hon Team Project 1

Fourier Series Representation for Rocket Density Measurements with Altitude

**October 11, 2016**

**Group F**

**Group Members**

Stirling Brandt

Jason Koch

Brennan McCann

Embry-Riddle Aeronautical University

Daytona Beach, FL 32114

**Introduction**

A Fourier series is a commonly used series, which represents the expansion of a periodic function. While it is an approximation containing several terms of an infinite series, its usefulness extends into several applications of data analysis. For this analysis, density readings through the atmosphere are recorded as electrical signal measurements taken aboard a rocket. During the data recording time of 850 to 900 seconds into the takeoff, density measurements vary significantly. Particularly due to noise and extraneous data, the readings do not represent an observable function of density with altitude. Using a Fourier series approximation, however, it is possible to produce a function that represents the data points as a curve. Due to the nature of the series, any amount of terms can be included in the summation. The resulting calculation of the curve will change as more terms in the Fourier series are added. For this data set, 5, 7, and 10 terms will be used to approximate the representative function. In addition, both a cosine and full Fourier series is calculated and compared. The usefulness of such analysis can be extended to any number of included terms, or in essence, any set of data points.

**Methodology**

In the determination of both the Fourier cosine series and full Fourier series the values for a0, an, and bn are typically determined by an integral featuring a function f(x).However, given that there is no function given for the rocket data set, the integrals were evaluated as a series. The transition can be found below.

For the values from a to b, the integral of the function is equal to the summation of all f(x) values multiplied by a very small change in x. For the cosine series, it was only necessary to calculate a0 and an. The integrals for these values are given as

In series notation, these would be given by

In this case, L would vary over a 50s time frame from 850 to 900 seconds in the flight for the given data set. The full Fourier series would be given by

Substituting the values for a0, an, and bn, the Fourier series would be given by

Using a similar fashion, the cosine series would be given by

For the full Fourier series, the value of bn would have to be calculated slightly differently. As the sine expansion is not symmetric about the y axis, the value for bn would have to be calculated from -50s to 0s and from 0s to 50s and then added together. The values for the density would stay the same over both intervals.

This calculation can be completed in MATLAB through a pair of nested for loops, calculating the values of the coefficients over the range of x values from -50s to 50s and determining those values for each of the n number of terms (5, 7, and 10 in this case).

**Results**

**Discussion**

**Conclusion**

**Appendix**

% Rocket Fourier

% Creates and plots a fourier sine series with variable number of terms for

% MA 441 rocket data

% Load data

Data = xlsread('Rocket\_Data');

% Calculate first 100 coefficients

Coeffs = zeros(3,101);

for n = 0:100

% Treat term 0 differently

if n == 0

% Take the average of the data as a0

a = sum(Data(:,1)) / length(Data);

ac = a;

% Set b0 as 0

b = 0;

else

a = 0;

b = 0;

time = 850;

for t = 1:length(Data)

% Take the integral of the data's periodic extension multiplied

% by cos(n\*pi\*x/L) from 800 to 900

a = a + Data(t,1) \* cos(n\*pi()\*Data(t,2)/50) \* (Data(t,2) - time);

a = a + Data(t,1) \* cos(n\*pi()\*(Data(t,2)-50)/50) \* (Data(t,2) - time);

% Take the integral of the data's periodic extension multiplied

% by sin(n\*pi\*x/L) from 800 to 900

b = b + Data(t,1) \* sin(n\*pi()\*Data(t,2)/50) \* (Data(t,2) - time);

b = b + Data(t,1) \* sin(n\*pi()\*(Data(t,2)-50)/50) \* (Data(t,2) - time);

time = Data(t,2);

end

time = 850;

for t = 1:length(Data)

% Take the integral of the data multiplied by cos(n\*pi\*x/L)

ac = ac + Data(t,1) \* cos(n\*pi()\*Data(t,2)/50) \* (Data(t,2) - time);

time = Data(t,2);

end

% Multiply by 2/L for the cosine series coefficients and 1/L for

% the Fourier series

ac = ac\*2/50;

a = a/50;

b = b/50;

end

% Store coefficients in a matrix

Coeffs(1, n+1) = a;

Coeffs(2, n+1) = b;

Coeffs(3, n+1) = ac;

end

% Call simple GUI to select number of terms to plot

termselector(Data, Coeffs)

% Rocket\_Fourier\_GUI

function termselector(Data, Coeffs)

% Create frame

frame = figure('Resize', 'Off');

set(frame,'MenuBar','none');

set(frame,'Name','Term Selector');

set(frame,'NumberTitle','off');

set(frame,'Position', [100,100,200,260])

% Create edit box to show number of terms

terms = uicontrol('Style', 'Edit', 'Units', 'Normalized', 'Position', [.4,.5,.2,.1], 'String', '10');

% Create button to plot fourier approximation

button = uicontrol('Style', 'Pushbutton', 'Units', 'Normalized', 'Position', [.3,.3,.4,.1], 'String', 'Plot Fourier');

set(button, 'Callback', {@forplotter, Data, Coeffs, terms})

% Create button to plot cosine approximation

button = uicontrol('Style', 'Pushbutton', 'Units', 'Normalized', 'Position', [.3,.1,.4,.1], 'String', 'Plot Cosine');

set(button, 'Callback', {@cosplotter, Data, Coeffs, terms})

end

% Callback for plot button

function forplotter(hObject, eventdata, Data, Coeffs, terms)

% Extract number of terms to plot

num = str2double(get(terms, 'String'));

% Calculate points to fit fourier approximation

ys = [];

for t = 850:.001:900

y = Coeffs(1,1);

for n = 2:num + 1;

y = y + Coeffs(1,n) \* cos(n\*pi()\*t/50) + Coeffs(2,n) \* sin(n\*pi()\*t/50);

end

ys = [ys,y];

end

t = 850:.001:900;

% Create a new axes in a new frame

frame2 = figure('Resize', 'Off');

set(frame2,'MenuBar','none');

set(frame2,'Name','Plot');

set(frame2,'NumberTitle','off');

set(frame2,'Position', [300,300,400,300]);

% Plot fourier approximation and data

plot(Data(:,2),Data(:,1),t,ys)

plottitle = sprintf('Fourier Approximation with %d Terms', num);

title(plottitle)

xlabel('Time (Seconds)')

ylabel('Density of Air (Micro-amps)')

legend('Data', 'Approximation')

end

% Callback for cosine plot button

function cosplotter(hObject, eventdata, Data, Coeffs, terms)

% Extract number of terms to plot

num = str2double(get(terms, 'String'));

% Calculate points to fit cosine approximation

ys = [];

for t = 850:.001:900

y = Coeffs(3,1);

for n = 2:num + 1;

y = y + Coeffs(3,n) \* cos(n\*pi()\*t/50);

end

ys = [ys,y];

end

t = 850:.001:900;

% Create a new axes in a new frame

frame2 = figure('Resize', 'Off');

set(frame2,'MenuBar','none');

set(frame2,'Name','Plot');

set(frame2,'NumberTitle','off');

set(frame2,'Position', [300,300,400,300]);

% Plot Cosine approximation and data

plot(Data(:,2),Data(:,1),t,ys)

plottitle = sprintf('Cosine Approximation with %d Terms', num);

title(plottitle)

xlabel('Time (Seconds)')

ylabel('Density of Air (Micro-amps)')

legend('Data', 'Approximation')

end