A Discrete Fourier Series Application

Matthew Liepke, Tanner Rosolino and Derek Stahl

Embry-Riddle Aeronautical University, Fall 2020

Introduction

A Fourier series can be used for multiple things. It can be used for approximating a function over an interval and it can also be used to approximate data over an interval. In this project, we used a cosine Fourier series and a full Fourier series to achieve the second goal. Fourier series can be used on a specific set of data, rather than a piecewise function. This is called a Discrete Fourier Series. We can do this for multiple reasons, including eliminating noise and providing a trendline for the measured data. In this project this principle will be used to model noisy data as a smooth function, which can be more beneficial than the original data when analyzing it.

Methodology

Using the MATLAB code included in appendix A, it is possible to approximate the collected data with a Fourier series. Both a cosine Fourier series and a full Fourier series were used to approximate the data using 5,7 and 10 terms for each. The Y values of our data are the current collected by a probe on a rocket. This current is closely related to the density of the atmosphere. The X values are the time values ranging from 850 to 900 seconds.

For the cosine series the existence of an even expansion in the negative X direction was assumed. This made the calculations easier as it forces the coefficient of the sine term (b_n) to be zero. This means we only have to calculate the cosine coefficient (a_n), and the initial term (a_0). In a cosine series, $a_0 = \frac{2}{L}\int_0^L f(x)dx$ where L is the half period of the function, or in this case, the total time elapsed. This integral is equivalent to the integral of f(x) over any half period, so we used the data provided, which is actually from 850 seconds to 900 seconds as the limits of integration. The integral was approximated using a trapezoidal riemann sum. a_n was then calculated using the equation $a_n = \frac{2}{L}\int_0^L f(x)cos(\frac{n\pi x}{L})dx$. Again a trapezoidal Riemann sum was used to approximate the integral. N is the series variable that changes for each term in the Fourier series. This means that a_n has a different value for each term, so it was calculated for each n value.

Using the values calculated for each coefficient we the calculated the value of the full series, using the equation $f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n cos(\frac{n\pi x}{L})$. This approximation, however, only used a

few terms, so n ranges from 1 to the number of terms used in that approximation. The approximations were graphed over the graph of the original data.

The full series is slightly more complicated, as it involves an extra term. It was assumed that the data was part of a periodic function that repeats every 50 seconds, the length of the data. This gave a half period of L=25. For the full series, $a_0 = \frac{1}{L}\int_{-L}^{L} f(x)dx$, but again this is equivalent to any integral of f(x) over a full period, so 850 and 900 were used for the limits of integration. The integral was approximated with another trapezoidal Riemann sum.

For a full series, both the sine coefficient, b_n, and the cosine coefficient, a_n, need to be calculated. b_n is calculated using the equation $b_n = \frac{1}{L}\int_{-L}^{L} f(x)sin(\frac{n\pi x}{L})dx$, and a_n is calculated using the equation $a_n = \frac{1}{L}\int_{-L}^{L} f(x)cos(\frac{n\pi x}{L})dx$. Again the limits of integration were changed, and the integral was approximated using a trapezoidal Riemann sum.

Using the calculated coefficients, the function can be modeled using the equation $f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n cos(\frac{n\pi x}{L}) + b_n sin(\frac{n\pi x}{L})$, again with n only reaching the number of terms used in that approximation. These approximations were then graphed with the original data to compare.

Results

After running the MATLAB script, the values for the coefficients of the Fourier series populated according to their respective algorithms, outputting the following values:

	n = 0	n =1	n =2	n =3	n =4	n =5	n =6	n =7	n =8	n =9	n =10
A_n	28025.8	262.3	-387.3	183.2	96.7	92.8	-35.1	30.7	110.2	-35.2	-9.6

Figure 1: Cosine Fourier Coefficients

	n = 0	n =1	n =2	n =3	n =4	n =5	n =6	n =7	n =8	n =9	n =10
A_n	28025.8	258.6	262.3	-238.2	-387.3	10.9	183.2	352.0	96.7	134.4	92.8
B_n	-	-315.0	156.9	233.9	-156.6	-168.1	-123.7	-10.8	-72.5	-165.7	-21.1

Figure 2: Full Fourier Coefficients

The values tabulated in Figure 1 and Figure 2 were then used to construct a Fourier series, which could be visually compared to the original data. This comparison is a valuable step in determining the accuracy of the series as calculated with the MATLAB code, and determining the value of a Fourier series in analyzing data.

Starting with the cosine series, the even expansion using discrete data provided a good trendline to the original data. It is visually clear that the code applies the discrete Fourier transform to the data in order to create a function that aligns with collected data. As expected, the higher order series follows the data more closely, which indicates again that the implementation of the discrete Fourier transform is correct.

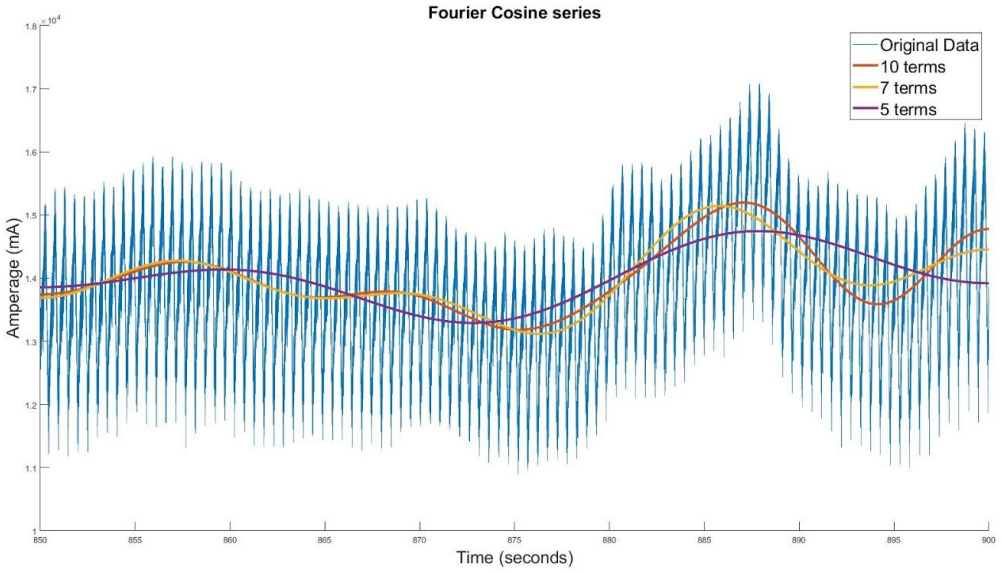

Figure 3: Fourier Cosine Series

The full Fourier discrete transform, as expected, follows the trend of the data with a higher degree of precision. Smaller oscillations are visible, unlike with the cosine series using the same number of terms. If more terms were added, it is clear that the full series would approach the original data before the cosine series would, which is another indication that the MATLAB code was successful.

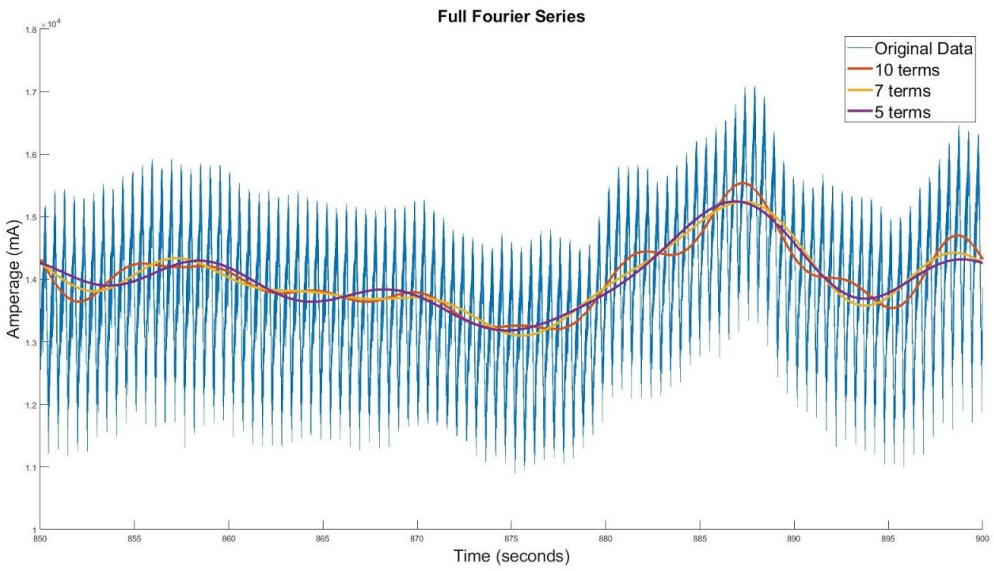

Figure 4: Full Fourier Series

Discussion

This project showed that you can use a Fourier series to help “clean up data” and create a trendline for the data. This can help with analyzing data as it allows you to see a trend and analyze how the data is influenced. As stated before, using a full Fourier series creates a trendline that is closer to the original data. Adding more terms to the series creates a trendline with more precision with respect to the original data. Using a Fourier series to clean up data has some common implications in computing, mainly in compression. The lower the number of terms there are, the compression ratio for the data will be higher. The more terms there are in the series, the compression ratio will be lower. That means that the trendline is much closer to the actual data.

Conclusion

Complex data recorded from a rocket probe was effectively transformed into a Fourier series, requiring minimal code and computation for a low number of terms. This validates the Fourier series’ ability to model any output data as a function. Although only plotted with a low amount of terms, as the number of terms was increased for the cosine and full Fourier series, a

function for the data was almost exactly created, with little to no visual variance between the outputted data and the Fourier series representation. In a practical sense, this means that any data can be recreated using the methods that were used to recreate the Fourier series for this project. This has far reaching implications - it means that any measured data can be mathematically represented, even if a theory or law does not exist to model the expected data. It also means that any mathematical operation can be performed (with help from a computer) on measured data if it is transformed into a Fourier series.

Appendix A

```
%Tanner Rosolino
%MA 441 Project 1
%Simulating data using a Fourier series

clear;
clc;
close all;

%Import data
RocketData=dlmread('rocketProbe-1.txt');

%set up figure for graphs
figure('units','normalized','outerposition',[0.05,0.1,0.9,0.85]);

%define important data
nData=length(RocketData);
nTerms=[1:10];

%% %cosine series

%define half period
HalfPeriodCos=max(RocketData(:,2))-min(RocketData(:,2));

%find A0 using riemann sum
A0Cos=2/HalfPeriodCos*trapz(RocketData(:,2),RocketData(:,1));

%find an for cosine using riemann sum
AnCos=2/HalfPeriodCos*trapz(RocketData(:,2),RocketData(:,1).*cos(nTerms*pi.
*RocketData(:,2)/HalfPeriodCos));

%calculate series
ysCosSum=zeros(nData,10);
for x=1:nData
    for n=nTerms

ysCosSum(x,n)=sum(AnCos(n)*cos(n*RocketData(x,2)*pi/HalfPeriodCos));
    end
end
ysCos10=A0Cos/2+sum(ysCosSum,2);
```

```
ysCos7=A0Cos/2+sum(ysCosSum(:,1:7),2);
ysCos5=A0Cos/2+sum(ysCosSum(:,1:5),2);

%plot data (subplot 2)
subplot(2,1,1);
hold on
plot(RocketData(:,2),RocketData(:,1));
plot(RocketData(:,2),ysCos10,'linewidth',3);
plot(RocketData(:,2),ysCos7,'linewidth',3);
plot(RocketData(:,2),ysCos5,'linewidth',3);
legend('Original data','10 terms','7 terms','5 terms');
title('Approximation using a cosine series')
axis([min(RocketData(:,2)),max(RocketData(:,2)),min(RocketData(:,1))*.95,ma
x(RocketData(:,1))*1.1]);
xlabel('Time (Sec)');
ylabel('Current (nAmps)');

%% %full series

%define half period
HalfPeriodFull=(max(RocketData(:,2))-min(RocketData(:,2)))/2;

%find A0 using riemann sum
A0Full=1/HalfPeriodFull*trapz(RocketData(:,2),RocketData(:,1));

%find an for cosine using riemann sum
AnFull=1/HalfPeriodFull*trapz(RocketData(:,2),RocketData(:,1).*cos(nTerms*p
i.*RocketData(:,2)/HalfPeriodFull));

%find bn for sine using riemann sum
BnFull=1/HalfPeriodFull*trapz(RocketData(:,2),RocketData(:,1).*sin(nTerms*p
i.*RocketData(:,2)/HalfPeriodFull));

%calculate series
ysFullSum=zeros(nData,10);
for x=1:nData
    for n=nTerms

ysFullSum(x,n)=sum(AnFull(n)*cos(n*RocketData(x,2)*pi/HalfPeriodFull)+BnFul
l(n)*sin(n*RocketData(x,2)*pi/HalfPeriodFull));
    end
end
ysFull10=A0Full/2+sum(ysFullSum,2);
```

```
ysFull7=A0Full/2+sum(ysFullSum(:,1:7),2);
ysFull5=A0Full/2+sum(ysFullSum(:,1:5),2);

%plot data (subplot 3)
subplot(2,1,2);
hold on
plot(RocketData(:,2),RocketData(:,1));
plot(RocketData(:,2),ysFull10,'linewidth',3);
plot(RocketData(:,2),ysFull7,'linewidth',3);
plot(RocketData(:,2),ysFull5,'linewidth',3);
legend('Original Data','10 terms','7 terms','5 terms');
title('Approximation using a full series')
axis([min(RocketData(:,2)),max(RocketData(:,2)),min(RocketData(:,1))*.95,ma
x(RocketData(:,1))*1.1]);
xlabel('Time (Sec)');
ylabel('Current (nAmps)');
```

References

Collins, Richard, et al. “Using Lidar and Rockets to Explore Turbulence in the Atmosphere.” SPIE Newsroom, 2015, doi:10.1117/2.1201505.005922.