

## Fourier Series and Filtering Periodic Signals

Lab 06

by Robert Throne

### Objectives

A variety of interesting waveforms can be expressed as sums of complex exponentials of different frequencies. The pulse trains used in communication systems, speech waveforms, and the waveforms produced by musical instruments can be modeled in this way. It is also important to determine how these periodic signals are modified when they are the input to a linear time invariant system.

The four main objectives of this lab are:

1. Improve your knowledge of programs in MATLAB, and
2. Understand how Fourier series coefficients are changed when a periodic signal goes through a system.
3. Review filtering of signals and develop an understanding of the relationship between the phase of a system and the time delay in the output signal.

### Procedure 1: Plotting a Periodic Signal using the Complex Fourier Series

a) Copy your code **Complex\_Fourier\_Series.m** from the previous homework to a new file **lab6.m**, and then use it to determine and plot the Complex Fourier series representation of the following periodic function (defined over one period) using 20 terms of the series.

$$x(t) = \begin{cases} 0 & -2 \leq t < -1 \\ 1 & -1 \leq t < 2 \\ 3 & 2 \leq t < 3 \\ 0 & 3 \leq t < 4 \end{cases}$$

### Procedure 2: Filtering Periodic Signals

One of the reasons for using a Fourier Series representation of a periodic signal instead of a different type of representation is that we get a frequency domain representation of the original signal  $x(t)$ . If  $x(t)$  is a periodic signal with period  $T$ , then it has the Fourier series

representation  $x(t) = \sum_{k=-\infty}^{k=\infty} c_k e^{jk\omega_0 t}$  where  $\omega_0 = \frac{2\pi}{T}$  and  $c_k = \frac{1}{T} \int_T x(t) e^{-jk\omega_0 t} dt$ . Using the fact that

the magnitude of the  $c_k$  is even and the phase of the  $c_k$  is odd, we can rewrite the complex Fourier series as a Fourier cosine series

$$x(t) = \sum_{k=-\infty}^{k=\infty} c_k e^{jk\omega_0 t} = c_0 + \sum_{k=1}^{k=\infty} 2|c_k| \cos(k\omega_0 t + \angle c_k)$$

Assume next we have a periodic signal  $x(t) = A\cos(\omega_0 t + \phi)$ . We can represent this as a phasor  $\mathbf{X} = A\angle\phi$ . Assume this signal is the input to an LTI system with transfer function  $H(j\omega)$ . We know we are only interested in the response of the transfer function at the same frequency as the input, so we want  $H(j\omega_0)$ . Again we can represent the transfer function as the phasor  $\mathbf{H} = |H(j\omega_0)|\angle H(j\omega_0)$ . The output of the system *in steady state* is then

$$\mathbf{Y} = \mathbf{H}\mathbf{X} = A|H(j\omega_0)|\angle[\phi + \angle H(j\omega_0)]$$

Going back from phasor notation to the time domain, we have the steady state output is

$$y(t) = A|H(j\omega_0)|\cos(\omega_0 t + \phi + \angle H(j\omega_0))$$

Now assume we have two inputs to the system

$$\begin{aligned} x_1(t) &= A_1 \cos(\omega_1 t + \phi_1) \\ x_2(t) &= A_2 \cos(\omega_2 t + \phi_2) \end{aligned}$$

We can represent these two inputs as phasors

$$\begin{aligned} \mathbf{X}_1 &= A_1\angle\phi_1 \\ \mathbf{X}_2 &= A_2\angle\phi_2 \end{aligned}$$

For each input, the corresponding output will be

$$\begin{aligned} \mathbf{Y}_1 &= \mathbf{H}_1\mathbf{X}_1 \text{ where } \mathbf{H}_1 = |H(j\omega_1)|\angle H(j\omega_1) \\ \mathbf{Y}_2 &= \mathbf{H}_2\mathbf{X}_2 \text{ where } \mathbf{H}_2 = |H(j\omega_2)|\angle H(j\omega_2) \end{aligned}$$

The steady state output due to each input is then

$$\begin{aligned} y_1(t) &= A_1 |H_1(j\omega_1)| \cos(\omega_1 t + \phi_1 + \angle H(j\omega_1)) \\ y_2(t) &= A_2 |H_2(j\omega_2)| \cos(\omega_2 t + \phi_2 + \angle H(j\omega_2)) \end{aligned}$$

If the input is  $x(t) = x_1(t) + x_2(t)$ , then by linearity our output will be

$$y(t) = y_1(t) + y_2(t) = A_1 |H_1(j\omega_1)| \cos(\omega_1 t + \phi_1 + \angle H(j\omega_1)) + A_2 |H_2(j\omega_2)| \cos(\omega_2 t + \phi_2 + \angle H(j\omega_2))$$

Now assume we have a periodic signal  $x(t)$  as the input to an LTI system with transfer function  $H(j\omega)$ . Since this signal is periodic it has a Fourier series representation:

$$x(t) = \sum_{k=-\infty}^{k=\infty} c_k e^{jk\omega_0 t} = c_0 + \sum_{k=1}^{k=\infty} 2|c_k| \cos(k\omega_0 t + \angle c_k)$$

Extending our analysis above, the steady state output of the system will be

$$y(t) = c_0 H(0) + \sum_{k=1}^{k=\infty} 2|c_k| |H(jk\omega_0)| \cos(k\omega_0 t + \angle c_k + \angle H(jk\omega_0))$$

a) Modify your code in **lab6.m** to produce the output signal for a filtered signal. We will actually only be looking at the first N terms in the Fourier series, as we have been. We will start with a simple filter,  $H(j\omega) = j\omega$ . This filter computes the derivative of the input. In order to do this so we can reuse the code, we will break this up into two pieces. Let's define the variable H0 to be the value of the transfer function at  $\omega = 0$ , and the variable (array) H to be the value of the transfer function at the discrete frequencies  $\omega_0, 2\omega_0, 3\omega_0, \dots, N\omega_0$ . It is probably easiest to assign these frequencies to a variable (array)  $W = [1:N]*\omega_0$ ;

b) For the periodic signal you determined the complex Fourier series of in part a, plot the Fourier series representation of both the input signal and the output signal *using different line types and a legend*. Start with  $N = 5$  and gradually increase N until you see what is happening (and you get a good graph). If N becomes too large you will not be able to see much. *Print out a good plot and attach it to the worksheet at the end of this lab*. Describe what you see (think about the derivative of the step function...). Have this part checked off before you go on.

Instructor Verification (see last page)

### **Procedure 3: Slope of Phase and Delay in the Time Response**

a) Now we want look at using Matlab to generate filter coefficients. For the majority of the filters it uses, Matlab assumes filter has the form

$$H(s) = \frac{b_N s^N + b_{N-1} s^{N-1} + \dots + b_2 s^2 + b_1 s + b_0}{a_N s^N + a_{N-1} s^{N-1} + \dots + a_2 s^2 + a_1 s + a_0}$$

Hence, in order to represent any filter, Matlab just uses an array for the  $b$  coefficients and an array for the  $a$  coefficients. For example, we might have two variables (arrays) B and A to store the coefficients. These variables would be

$$B = [b_N \quad \dots \quad b_1 \quad b_0]$$

$$A = [a_N \quad \dots \quad a_1 \quad a_0]$$

Let's assume we want to use a 10<sup>th</sup> order Butterworth lowpass filter with a frequency cutoff of  $20\omega_0$ . Use the help command to look up the Matlab function **butter**. You should return the

coefficients in two arrays, i.e. your command should be `[B,A] = butter(...)`; Note that we are constructing an **analog** filter here, so read **all** of the description for the **butter** command.

b) We now again need to find the variable  $H_0 = H(0)$  the variable (array)

$H = [H(j\omega_0) \ H(j2\omega_0) \ \dots \ H(jN\omega_0)]$ . To find  $H(0)$  we use the fact that

$$H(0) = \frac{b_0}{a_0}$$

If the  $b$  and  $a$  coefficients are stored in arrays  $B$  and  $A$ , we can write

`H0 = B(end)/A(end);`

Where **end** tells Matlab to get the last element of the array. The command **freqs** will be helpful for finding the variable (array)  $H$ .

c) Plot the Fourier series representation for both the input signal  $x(t)$  and the output signal  $y(t)$  on the same graph for  $N=25$  terms, *using different line types and a legend*. The title of your graph should indicate that you are using a Butterworth filter. If you have done everything correctly your graph should look like that in Figure 1. *Print out this graph and attach it to the worksheet at the end of this lab.*

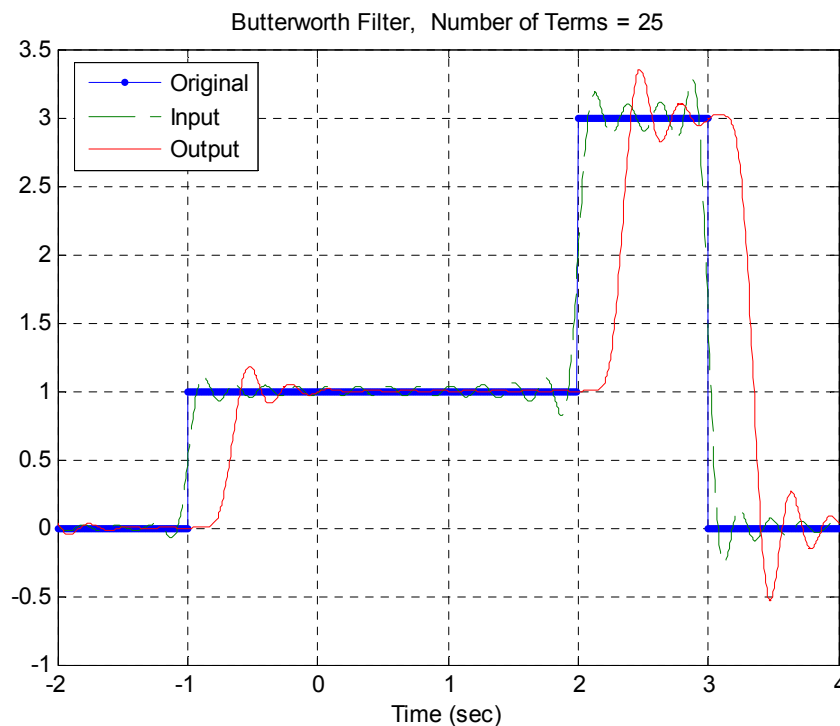


Figure 1. Input and Fourier series representation of the input and output using a 10<sup>th</sup> order Butterworth filter.

d) It will be useful to look at the frequency response (Bode) plot of the filter you are using. To do this, you need to first construct a transfer function using the A and B vectors, using Matlab's **tf** command. Then use Matlab's **bode** command to plot the frequency response of the filter. Use the **grid** command to put a grid on the graph. Is the cutoff frequency where you think it should be? *Print out the bode plot, label the cutoff frequency on the printed graph, and then attach it to the end of the sheet at the end of the lab.*

e) Estimate the delay between the input and output signals. The easiest way to do this is to determine the delay between a point halfway up the left side of  $f(t)$  (the point (-1,0.5)) and location of the same point for the filtered signal (i.e., what value of  $t$  has amplitude of 0.5 on the filtered signal?) Next estimate the slope of the phase of the filter (only near those frequencies we are allowing to pass.) The phase and time delay should be approximately related through the relationship

$$-\text{slope of filter (in sec)} = \text{time delay}$$

We only need to estimate the slope over those frequencies we are interested in, so we'll use

$$\text{slope} = \frac{[\angle H(j\omega_0) - \angle H(j20\omega_0)] \times \left[ \frac{\pi}{180} \right]}{\omega_0 - 20\omega_0}$$

Note that this formula for the slope assumes that the phase is measured in degrees and the phase has been **unwrapped**. Both of these are true if you use the Bode plot. You may not be able to get the phase at  $\omega_0$  or  $20\omega_0$  exactly, so you may have to modify the formula. You can click on the bode plot and slide the cursor to get accurate readings. Note that the phase is not really linear, but we'll just approximate it this way for now. Record your values for the measured and approximate time delays.

f) Change the Butterworth filter order to a 15<sup>th</sup> and then a 25<sup>th</sup> order filter. Plot the input and output signals on the same plot, and measure the time delays for each. Also plot the frequency response of each filter and estimate the time delay from the slope of the phase response. Fill in the remaining entries in the table on the worksheet showing measured and estimated values of the time delay. (The table should have results for all three Butterworth filters) **Note: You do not need to print out these figures!**

Instructor Verification (see last page)

**Procedure 4: Filtering to Isolate Harmonics**

a) Modify the 10<sup>th</sup> order Butterworth filter to filter out all components of  $x(t)$  except for the **first harmonic**. The only output should be of the form  $D_1 \cos(\omega_o t + \phi_1)$ . Plot both the input and output signals, as well as the frequency response of the filter you used, *and attach them to the worksheet at the end of the lab*. Be sure to include the parameters you used in the filter.

b) Modify the Butterworth filter to filter out all components of  $x(t)$  except for the **second harmonic**. The only output should be of the form  $D_2 \cos(2\omega_o t + \phi_2)$ . Plot both the input and output signals, as well as the frequency response of the filter you used, *and attach them to the worksheet at the end of the lab*.

[Instructor Verification \(see last page\)](#)

**Lab 06**  
**Instructor Verification Sheet**

*You should have graphs from Parts 2-b, 3-c, 3-d, 4-a, and 4-b.*

**Attach your code (Lab6.m) to this sheet**

Name \_\_\_\_\_ Date: \_\_\_\_\_

Part **2-b** Verified: \_\_\_\_\_

Describe the relevant information in the plot of the *system's* output:

Filter Type	Predicted Delay (based on slope of phase)	Measured Delay (measured in time domain)
10 <sup>th</sup> order Butterworth		
15 <sup>th</sup> order Butterworth		
25 <sup>th</sup> order Butterworth		

Part **3-f** Verified: \_\_\_\_\_

Part **4-b** Verified: \_\_\_\_\_

In part **3-c** we used  $N=25$  terms in the Fourier series representation of  $x(t)$ . What is the smallest number of terms we can use in the Fourier series representation of  $x(t)$  that will produce the same output of the filter  $y(t)$  that we obtained with  $N=25$  terms? Why?