ROSE-HULMAN INSTITUTE OF TECHNOLOGY
Department of Electrical and Computer Engineering

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. Understand how to get a DC value from a pulse width modulated (pwm) signal.

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

Copy your file **Complex_Fourier_Series.m** to a new file **lab6.m**. Be sure to delete anything that is not necessary in this code, since part of your grade will be based on your code.

Plot the Complex Fourier series representation of the following periodic function (defined over one period) using 20 terms of the series.

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

Note that the function does not start at zero, be sure your **quadl** function does not assume this!

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

$$x(t) = \sum_{k=-\infty}^{k=\infty} c_k e^{jk\omega_o t} = c_o + \sum_{k=1}^{k=\infty} 2\,|c_k|\cos(k\omega_o t + \measuredangle c_k)$$

Assume next we have a periodic signal $x(t) = A\cos(\omega_0 t + \phi)$. We can represent this as a phasor $\mathbf{X} = A\measuredangle\phi$. 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 $\mathbf{H} = |H(j\omega_0)|\measuredangle H(j\omega_0)$. The output of the system <u>*in steady state*</u> is then

$$\mathbf{Y} = \mathbf{HX} = A\,|H(j\omega_0)|\measuredangle\left[\phi + \measuredangle H(j\omega_0)\right]$$

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

$$x_1(t) = A_1\cos(\omega_1 t + \phi_1)$$
$$x2(t) = A_2\cos(\omega_2 t + \phi_2)$$

We can represent these two inputs as phasors

$$\mathbf{X}_1 = A_1\measuredangle\phi_1$$
$$\mathbf{X}_2 = A_2\measuredangle\phi_2$$

For each input, the corresponding output will be

$$\mathbf{Y}_1 = \mathbf{H}_1\mathbf{X}_1 \text{ where } \mathbf{H}_1 = |H(j\omega_1)|\measuredangle H(j\omega_1)$$
$$\mathbf{Y}_2 = \mathbf{H}_2\mathbf{X}_2 \text{where } \mathbf{H}_2 = |H(j\omega_2)|\measuredangle H(j\omega_2)$$

The steady state output due to each input is then

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

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:

$$x(t) = \sum_{k=-\infty}^{k=\infty} c_k e^{jk\omega_o t} = c_o + \sum_{k=1}^{k=\infty} 2 \,|\, c_k \,|\, \cos(k\omega_o t + \measuredangle c_k)$$

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

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

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 $\omega_0, 2\omega_0, 3\omega_0, ..., N\omega_0$. It is probably easiest to assign these frequencies to a variable (array) W = [1:N]* ω_o;

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

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

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

$$B = \begin{bmatrix} b_N & \ldots & b_1 & b_0 \end{bmatrix}$$
$$A = \begin{bmatrix} a_N & \ldots & a_1 & a_0 \end{bmatrix}$$

Let's assume we want to use a 10^{th} 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 H0 = $H(0)$ the variable (array) $H = \begin{bmatrix} H(j\omega_0) & H(j2\omega_0) & \ldots & H(jN\omega_0) \end{bmatrix}$. 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.*

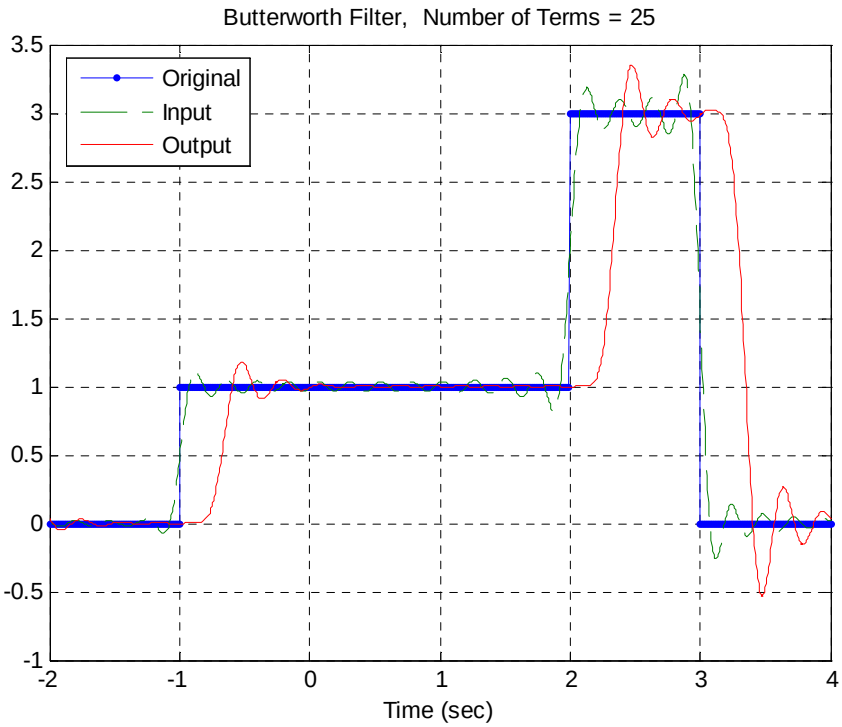

Figure 1. Input and Fourier series representation of the input and output using a 10^{th} order Butterworth filter.

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 though the relationship

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

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

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

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 ω_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 on the worksheet at the end of this lab.*

Instructor Verification (see last page)

Procedure 4: Analog Voltages from Pulse Width Modulated (PWM) Signals

Microcontrollers often need to output an analog signal in order to control sensors or motors. Although it is possible to use analog to digital converters, they take up a lot of space on a chip. An alternative is to use a form of pulse width modulation (PWM) and a lowpass filter. For our purposes, the pulse width modulated signal is a periodic square wave which is either at 5 volts or 0 volts. We only change *period* of the pwm signal and the *duty cycle* of the pwm signal since this can easily be done with timers. We can now use our knowledge of periodic signals, Fourier series representation of periodic signals, and filtering to determine how we get an analog DC signal out of a pwm signal.

a) Construct a square wave signal $x(t)$ with an amplitude of 5 volts (it goes from a minimum of 0 volts to a maximum of 5 volt), a period of 1 microsecond (the signal should start at zero), and a duty cycle of 80%. Leave the duty cycle a variable in your program.

b) Plot the original signal and the Fourier series representation of the signal (using 10 terms) over one period to verify everything is working before going on.

c) Assume the pwm signal is the input to a 2^{nd} order lowpass Butterworth filter with cutoff frequency of $\frac{\omega_0}{2}$. Since we want to get a DC signal, we want the lowpass filter to only retain the DC term, and filter out the higher harmonics. Plot the true input, the Fourier series representation of the input, and the Fourier series representation of the output for 4 periods, all on one graph with *appropriate legends and line types.* You may find the *mod* function helpful here, but you only need it in the plotting function when evaluating the true signal $x(t)$.

d) Modify the order of your Butterworth filter until you have the smallest filter order that produces a reasonably flat output. *Print out your plot and attach it to the worksheet at the end of this lab. Your graph should look a lot like that in Figure 2.*

e) Modify the duty cycle of your input signal so the output of the lowpass filter will be 1 volt. *Print out your plot and attach it to the worksheet at the end of this lab.*

Instructor Verification (see last page)

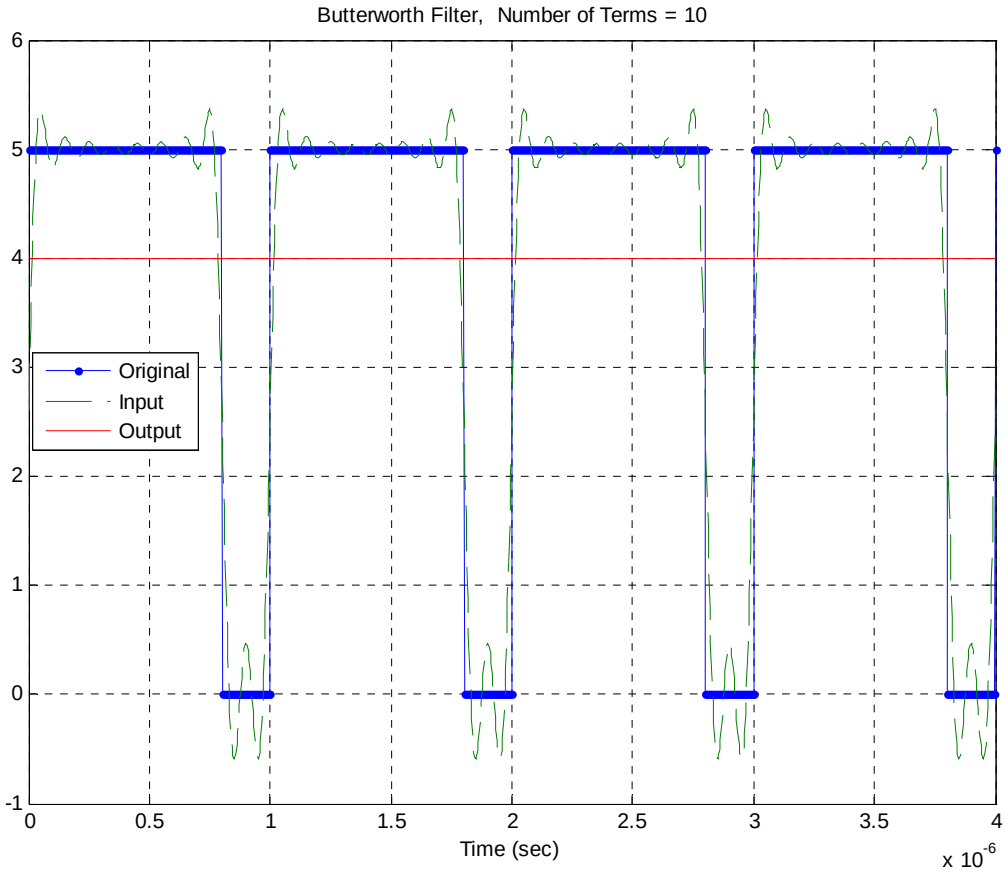

Figure 2. Results for part 4d, obtaining an analog voltage from a pwm signal.

Procedure 5: Generating Sinusoids from PWM signals

By applying an appropriate bandpass filter, we can also generate sine waves from pwm signals. For this problem use the same signal as in part 4, but with a 50% duty cycle.

a) Use a 5th 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 the true input, the Fourier series representation of the input, and the Fourier series representation of the output for 5 periods, all on one graph with *appropriate legends and line types.* You may find the *mod* function helpful here, but you only need it in the plotting function when evaluating the true signal $x(t)$. *Print out your plot and attach it to the worksheet at the end of this lab.*

b) Use a 5th order 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 the true input, the

Fourier series representation of the input, and the Fourier series representation of the output for 5 periods, all on one graph with *appropriate legends and line types.* You may find the *mod* function helpful here, but you only need it in the plotting function when evaluating the true signal $x(t)$. *Print out your plot and attach it to the worksheet at the end of this lab.*

Instructor Verification (see last page)

Lab 06
Instructor Verification Sheet

You should have graphs from Parts 2-b, 3-c, 3-d, 4-d, 4-e, 5-a and 5-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:

Predicted delay (base on slope of phase):

Measured delay (measured in time domain):

Part **3-e** Verified:_____

Part **4-e** Verified:_____

Part **5-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?