Fourier Series and Filtering Periodic Signals

Lab 04

by Robert Throne (based on a Lab by Mark Yoder)

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. In this lab we will explore some analytical techniques for generating waveforms by summing complex exponentials.

The two main objectives of this lab are:

1. Improve your knowledge of programs and functions in MATLAB, and
2. Write a function that will sum several complex exponential terms.

Pre-Lab

Do the following before coming to lab. Record the results in your lab notebook and hand in a photocopy of your pre-lab.

a) Work the following matrix algebra problems by hand:

1. $\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} 6 & 4 & 2 \\ 5 & 3 & 1 \end{bmatrix} =$

2. $\begin{bmatrix} 6 & 4 & 2 \\ 5 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} =$

b) In MATLAB, a process called implicit type assignment can automatically create a matrix when it is called for in a computation. For example, in the computation below, MATLAB would understand that the result of the command cos(x) results in a matrix as shown. Find the result of the commands listed by hand.

$$x = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad ; \quad \cos(x) = \begin{bmatrix} \cos(a) & \cos(b) \\ \cos(c) & \cos(d) \end{bmatrix}$$

1. e^{jx}

2. xe^{jx}

c) Write a function that performs the same task as the following without using a `for` loop, and paste the code and verification that it works in your notebook.

```
function Z=replacez(A)
%REPLACEZ Function that replaces the positive elements of a matrix with 1
%  usage:
%    Z=replacez(A)
%    A=input matrix whose positive elements are to be replaced with 1
%
[M,N]=size(A);
for i=1:M
  for j=1:N
    if A(i,j)>0
      Z(i,j)=1;
    else
      Z(i,j)=A(i,j);
    end
  end
end
```

MATLAB hint: The MATLAB logical operators may be helpful in completing this exercise. Use `help` to understand what these commands do.

Procedure A: Efficient Synthesis of x(t)

The way the program **Fourier_Series.m** computes $x(t)$ is very inefficient, and we are going to try and speed this up and take advantage of Matlab's strengths. In what follows, remember we want to compute

$$x(t) \approx \sum_{k=-N}^{k=N} c_k e^{jk\omega_o t}$$

To do this we need to do the following:

a) Just after the period, T, is defined, write an expression for ω_0, the natural frequency.

b) Define an array $W = \left[-N\omega_0 \quad -(N-1)\omega_0 \quad \ldots \quad \omega_0 \quad 0 \quad \omega_0 \quad \ldots \quad (N-1)\omega_0 \quad N\omega_0\right]$
which we can write as $W = \left[-N \quad -(N-1) \quad \ldots \quad 1 \quad 0 \quad 1 \quad \ldots \quad (N-1) \quad N\right]\omega_o$

c) In order to use this sum, we need the array

$$C = [c_{-N} \quad c_{-(N-1)} \quad \ldots \quad c_{-1} \quad c_0 \quad c_1 \quad \ldots \quad c_{N-1} \quad c_N]$$

What we have is c_0 and the array $c = [c_1 \quad \ldots \quad c_{N-1} \quad c_N]$. Use these arrays and the properties of the coefficients to construct the array C. Your method of construction should work for any sized array. The Matlab commands **fliplr** and **conj** will be very helpful here. Constructing C should require one statement in Matlab, and, of course, must occur after c and c_0 are available.

d) Remove (delete) the following inefficient lines of code

```
 m = 1;
 est = T*c(1)*conj(feval(@periodic,t))+T*conj(c(1))*feval(@periodic,t);
%
 for i = 2:N
 m = i;
est = est + T*c(i)*conj(feval(@periodic,t));
est = est + T*conj(c(i))*feval(@periodic,t);
end;
est = est+c0;
```

e) Now write a routine **sum_exp** which takes as arguments W, C, and t, where t is an array of the times we want to find x at. The command **x = sum_exp(C,W,t)** should return the vector x evaluated at all times t. To understand how we are going to do this, let's consider the following situation: Assume we want to know $x(t)$ at three times and we are going to use up to the second harmonic in the Fourier series representation ($N = 2$), so we have $x(t_i) = \sum_{k=-2}^{k=2} c_k e^{jk\omega_0 t_i}$ for $i = 1,2,3$ Assume we have a row vector of the three times $t = \begin{bmatrix} t_1 & t_2 & t_3 \end{bmatrix}$, the row vector $W = \begin{bmatrix} -2\omega_0 & -\omega_0 & 0 & \omega_0 & 2\omega_0 \end{bmatrix}$ of the required frequencies, and the row vector $C = \begin{bmatrix} c_{-2} & c_{-1} & c_0 & c_1 & c_2 \end{bmatrix}$. First let's form

$$jW^T t = j\begin{bmatrix} -2\omega_0 \\ -1\omega_0 \\ 0 \\ \omega_0 \\ 2\omega_0 \end{bmatrix}\begin{bmatrix} t_1 & t_2 & t_3 \end{bmatrix} = \begin{bmatrix} -j2\omega_0 t_1 & -j2\omega_0 t_2 & -j2\omega_0 t_3 \\ -j1\omega_0 t_1 & -j1\omega_0 t_2 & -j1\omega_0 t_3 \\ 0 & 0 & 0 \\ j1\omega_0 t_1 & j1\omega_0 t_2 & j1\omega_0 t_3 \\ j2\omega_0 t_1 & j2\omega_0 t_2 & j2\omega_0 t_3 \end{bmatrix}$$

Then

$$e^{jW^T t} = \begin{bmatrix} e^{-j2\omega_0 t_1} & e^{-j2\omega_0 t_2} & e^{-j2\omega_0 t_3} \\ e^{-j1\omega_0 t_1} & e^{-j1\omega_0 t_2} & e^{-j1\omega_0 t_3} \\ 1 & 1 & 1 \\ e^{j1\omega_0 t_1} & e^{j1\omega_0 t_2} & e^{j1\omega_0 t_3} \\ e^{j2\omega_0 t_1} & e^{j2\omega_0 t_2} & e^{j2\omega_0 t_3} \end{bmatrix}$$

Finally, by premultiplying by C we will get the desired result. The function **sum_exp** should actually consist of one line of code.

f) Plot the original function and the Fourier Series representation of the function using 50 terms. The function is defined over the period -2 to 4. Include this figure in your lab notebook and have it checked off before you go on. The function you should be plotting is

$$f_3(t) = \begin{cases} 0 & -2 \le t < -1 \\ 1 & -1 \le t < 2 \\ 3 & 2 \le t < 3 \\ 0 & 3 \le t < 4 \end{cases}$$

Instructor Verification (see last page)

Procedure B: 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 $x(t)$. Recall from phasor analysis that if the input to a LTI system is $x(t) = A\cos(\omega_0 t + \phi)$ the steady state output will be

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

Let's rewrite the periodic input signal $x(t)$ using Euler's identity,

$$x(t) = \frac{A}{2}e^{j(\omega_0 t+\phi)} + \frac{A}{2}e^{-j(\omega_0 t+\phi)}$$

We can rewrite the output as

$$y(t) = \frac{A|H(j\omega_0)|}{2}e^{j(\omega_0 t+\phi+\angle H(j\omega_0))} + \frac{A|H(j\omega_0)|}{2}e^{-j(\omega_0 t+\phi+\angle H(j\omega_0))}$$

Using the properties that for a real system $|H(j\omega_0)| = |H(-j\omega_0)|$ (the magnitude is an even function) and $\angle H(j\omega_0) = -\angle H(-j\omega_0)$ (the phase is an odd function) , we get

$$y(t) = \frac{A|H(j\omega_0)|e^{j\angle H(j\omega_0)}}{2}e^{j(\omega_0 t+\phi)} + \frac{A|H(-j\omega_0)|e^{j\angle H(-j\omega_0)}}{2}e^{-j(\omega_0 t+\phi)}$$

Now if we recognize the above expression for $x(t)$ as its Fourier series representation, with coefficients

$$c_{-1} = \frac{Ae^{-j\phi}}{2} \quad c_0 = 0 \quad c_1 = \frac{Ae^{j\phi}}{2}$$

we can write

$$y(t) = c_{-1}H(-j\omega_0)e^{-j\omega_0 t} + c_1 H(j\omega_0)e^{j\omega_0 t} = b_{-1}e^{-j\omega_0 t} + b_1 e^{j\omega_0 t}$$

In general then, if $x(t)$ has the (finite) Fourier series representation

$$x(t) \approx \sum_{k=-N}^{k=N} c_k e^{jk\omega_o t}$$

then the steady state output will be given by

$$y(t) \approx \sum_{k=-N}^{k=N} b_k e^{jk\omega_o t} \quad \text{where} \quad b_k = c_k H(jk\omega_0)$$

We are now ready to modify **Fourier_Series.m** so we can filter signals.

a) We will start with a simple filter, $H(j\omega) = j\omega$. This filter computes the derivative of the input. For this filter, determine a new vector

$$H = [H(-jN\omega_0) \quad H(-j(N-1)\omega_0) \quad \ldots \quad H(j\omega_0) \quad \ldots \quad H(j(N-1)\omega_0) \quad H(jN\omega_0)]$$

b) Construct the new vector (this should be easy to do)

$$B = [c_{-N} H(-jN\omega_0) \quad c_{-(N-1)} H(-j(N-1)\omega_0) \quad \ldots \quad c_0 H(j\omega_0) \quad \ldots \quad c_{N-1} H(j(N-1)\omega_0) \quad c_N H(jN\omega_0)]$$

c) Plot the output vector $y(t)$ using its Fourier series representation. 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. Include a good plot in your lab notebook. 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)

d) Now we want look at different kinds of filters. 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 [Bu,Au] = butter(…..);

e)We now again need to find the vector

$$H = [H(-jN\omega_0) \quad H(-j(N-1)\omega_0) \quad \ldots \quad H(j\omega_0) \quad \ldots \quad H(j(N-1)\omega_0) \quad H(jN\omega_0)]$$

The command **freqs** will be helpful here.

e) Construct the B vector again, and plot both the input signal $x(t)$ and the output signal $y(t)$ on the same graph using 50 terms. (You do not need to plot the imaginary part of either signal, it gets too messy!) You should see two distinct phenomena. Include this graph and a brief discussion of what you see in your lab notebook. Have this part checked off before you go on.

Instructor Verification (see last page)

f) 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 Au and Bu vectors, using Matlab's **tf** command. The 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? Identify the cutoff frequency on the graph and put it in your lab notebook.

g) Estimate the slope of the phase of the filter (only near those frequencies we are allowing to pass), and the delay between the input and output signals. The phase and time delay should be approximately related though the relationship

$$-slope = t_d$$

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

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

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.

h) Change the Butterworth filter order to a 15^{th} and then a 25^{th} order filter. Plot the input and output signals on the same plot, and estimate the delay between the input and output signals. Plot the frequency response of the filter and estimate the slope of the filter. Make a table showing the slope of the phase and the time delay between the input and output (The table should have results for all three Butterworth filters.)

i) Modify the Butterworth filter to filter out all components of $x(t)$ except for the first harmonic. Plot both the input and output, as well as the frequency response of the filter you used, and put them in your lab notebook. Be sure to include the parameters you used in the filter.

j) Modify the Butterworth filter to filter out all components of $x(t)$ except for the second harmonic. Plot both the input and output, as well as the frequency response of the filter you used, and put them in your lab notebook. Be sure to include the parameters you used in the filter.

Have this last part checked off and you are done!

Instructor Verification (see last page)

Lab 04
Instructor Verification Sheet

Names _____

Date of Lab: _____

Part **A-f** Verified:_____

Part **B-c** Verified:_____

Part **B-e** Verified:_____

Part **B-j** Verified:_____