Systems Modelling and Control: Tutorial 3

Due on April 25, 2016 at 3:10pm

Damien Hill

S.Reynolds (262538)

Problem 1

Frequency Response

QUESTION A

A linear time-invariant continuous-time system has some frequency function $H(\omega)$. Some input $x(t)$, which is composed of a DC component and 2 sinusoids with distinct multiples of some fundamental frequency, w_0, is given as:

$$\begin{aligned} x(t) &= 1 + 4\cos(2\pi t) + 8\sin(3\pi t - \pi/2) \\ &= 1 + 4\cos(2\pi t) + 8\cos(3\pi t - \pi) \end{aligned}$$

The fundamental frequency of the sinusoids, w_0, is the lowest common divisor of 2π and 3π. Hence, the fundamental frequency is given by:

$$w_0 = \pi$$

The output to the system, $y(t)$, is given by:

$$\begin{aligned} y(t) &= 2 - 2\sin(2\pi t) \\ &= 2 + 2\sin(-2\pi t) \\ &= 2 + 2\cos(-2\pi t - \pi/2) \\ &= 2 + 2\cos\big(-(2\pi + \pi/2)\big) \\ &= 2 + 2\cos(2\pi t + \pi/2) \end{aligned}$$

Now, the above input signal, $x(t)$, is periodic and as such has the complex Fourier series representation:

$$x(t) = \sum_{k=-\infty}^{\infty} c_k^x e^{jk\omega_0 t}$$

A single instance, k, of this infinite series is given by:

$$x_k(t) = c_k^x e^{jk\omega_0 t}$$

Because the system is linear, we can apply the transfer function to each instance of complex Fourier series representation of $x(t)$, and rely on superposition to obtain the output, $y(t)$. This is represented as:

$$y(t) = \sum_{k=-\infty}^{\infty} H(\omega_0 k) c_k^x e^{jk\omega_0 t}$$

A single instance, k, of the infinite Fourier series representing $y(t)$ can be written as:

$$y_k(t) = c_k^y e^{jk\omega_0 t} = H(\omega_0 k) c_k^x e^{jk\omega_0 t}$$

Problem 1 continued on next page...

Hence, we can see that:

$$c_k^y = H(\omega_0 k) c_k^x$$

Which allows us to write an expression for the transfer function, $H(\omega)$, in terms of c_k^x and c_k^y:

$$H(\omega) = H(k\omega_0) = \frac{c_k^y}{c_k^x}, \quad \omega = k\omega_0$$

Hence, $\exists\ H(\omega)\ \forall\ c_k^x \neq 0$. That is to say, we can find $H(\omega)$ for $k = 0, 2, 3$.

Question 2

$H(\omega) = H(k\omega_0)$ can be found for $k = 0, 2, 3$. We first note that:

$$\begin{aligned} c_0^x &= 1 \\ c_2^x &= \frac{1}{2} \cdot 4 = 2 \\ c_3^x &= \frac{1}{2} \cdot 8 \cdot e^{-j\pi} = 4 \cdot e^{-j\pi} \end{aligned}$$

Further, we can see that:

$$\begin{aligned} c_0^y &= 2 \\ c_2^y &= \frac{1}{2} \cdot 2 \cdot e^{j \cdot \pi/2} = e^{j \cdot \pi/2} \\ c_3^y &= 0 \end{aligned}$$

Hence, we get that:

$$\begin{aligned} H(0 \cdot \pi) &= \frac{c_0^y}{c_0^x} = \frac{2}{1} = 2 \\ H(2 \cdot \pi) &= \frac{c_2^y}{c_2^x} = \frac{e^{j \cdot \pi/2}}{2} = \frac{1}{2} \cdot e^{j \cdot \pi/2} \\ H(3 \cdot \pi) &= \frac{c_3^y}{c_3^x} = \frac{0}{4 \cdot e^{-j\pi}} = 0 \end{aligned}$$

Problem 2

Filtering of sound

Suppose a continuous-time signal, $x(t)$, is given by the following:

$$x(t) = \cos(440 \cdot 2\pi \cdot t) + \cos(554 \cdot 2\pi \cdot t) + \cos(659 \cdot 2\pi \cdot t)$$

To play the sound of this signal for 2 seconds, the following Matlab code was implemented:

```
% Script plays audio of signal x(t)
dur = 2.0;
fs = 44100;
t = 0:(1/fs):dur;

% Complex representation of the signal
xe1 = 0.5*(exp(1i*440*2*pi*t) + exp(-1i*440*2*pi*t));
xe2 = 0.5*(exp(1i*554*2*pi*t) + exp(-1i*554*2*pi*t));
xe3 = 0.5*(exp(1i*659*2*pi*t) + exp(-1i*659*2*pi*t));

x = xe1 + xe2 + xe3;

sound(x,fs)
```

Problem 3

Filtering of sound

Suppose a continuous-time signal, $x(t)$, is given by the following:

$$x(t) = \cos(440 \cdot 2\pi \cdot t) + \cos(554 \cdot 2\pi \cdot t) + \cos(659 \cdot 2\pi \cdot t)$$

The representation of a periodic signal as a complex Fourier series can be written as follows:

$$x(t) = \sum_{k=-\infty}^{\infty} c_k^x e^{jk\omega_0 t}$$

The complex coefficients, c_k^x, can be written as:

$$c_k^x = \frac{1}{2} \cdot A_k \cdot e^{j\theta_k}$$

It must be noted that $|c_k^x| = |c_{-k}^x| = \frac{A_k}{2}\ \forall\ k \in \mathbb{Z}^+$, however, the same is not true for $\angle c_k^x$. In fact, $\angle c_k^x = -\angle c_{-k}^x\ \forall\ k \in \mathbb{Z}^+$.

For the signal in this problem we note that $w_0 = 2\pi$, which greatly simplifies the complex Fourier series representation since there are only three terms in the sequence whose amplitudes are non-zero: $k = 440$, $k = 554$,and $k = 659$. Further, we see that $A_{440} = A_{554} = A_{659} = 1$, and that $\theta_{440} = \theta_{554} = \theta_{659} = 0$. Hence, the complex exponential Fourier series representation of $x(t)$ is as follows:

$$x(t) = \frac{1}{2} \cdot \left[(e^{j \cdot 440 \cdot 2\pi \cdot t} + e^{-j \cdot 440 \cdot 2\pi \cdot t}) + (e^{j \cdot 554 \cdot 2\pi \cdot t} + e^{-j \cdot 554 \cdot 2\pi \cdot t}) + (e^{j \cdot 659 \cdot 2\pi \cdot t} + e^{-j \cdot 659 \cdot 2\pi \cdot t}) \right]$$

Problem 3 continued on next page. . .

The following Matlab code was run to plot the magnitudes, $|c_k^x|$, versus k:

```
% Script simply plots k vs ck for a signal composed of a finite number
% of cosine functions

% Determine size of plot
n = 1000;

% Create values for k
k = -n:n;

% Create storage vector for |ck|
pos_ck = zeros(1,n);

% Insert ck amplitude into storage vector
A = [440 554 659];
pos_ck(A) = 0.5;

% Compose |ck|
ck = [fliplr(pos_ck) 0 pos_ck];

% Print and label graph of k vs |ck|
stem(k,ck,'marker','.')
title('Magnitude of Complex Exponetials')
xlabel('Frequency')
ylabel('Amplitude |ck|')
axis([-1000 1000 0 1]);
```

The plot can be seen in Figure 1.

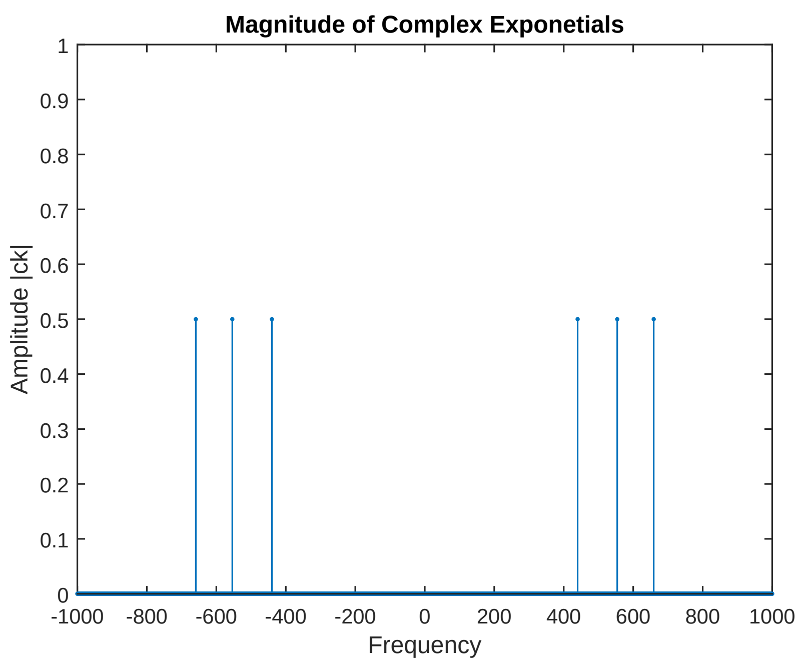

Figure 1: Magnitude of Complex Exponetials

Problem 4

A Matlab function was written to apply an ideal low pass filter to the signal. The Matlab code can be seen below:

```
function ck_out = low_pass_filter(k,ck,w0,B)

    % Function has one output ck_out which is the output
    % of the input signal with the filter applied to it.
    % The inputs for the function are the fundamental
    % frequency (w0), the input signal complex coefficients (ck),
    % the bandwidth of the filter (B)

    filter = abs(-k:k) < (B/w0);
    ck_out = ck.*filter;

end
```

A Matlab script, implementing the above function, was written to plot the signal output for filters of bandwidth varying from 0 to 5000 rad /s. The script can be seen below:

```
% Script simply plots k vs ck for a signal composed of a finite number
% of cosine functions

% Determine size of plot
n = 1000;
w0 = 2*pi;

% Create values for k
k = -n:n;

% Create storage vector for |ck|
pos_ck = zeros(1,n);

% Insert ck amplitude into storage vector
A = [440 554 659];
pos_ck(A) = 0.5;

% Compose |ck|
ck = [fliplr(pos_ck) 0 pos_ck];
ck_out = zeros(1,6);

for i = 0:5

    % Run filter over input signal
    ck_out = low_pass_filter(n,ck,w0,1000*i);

    % Create subplot template
    subplot(3,2,i+1)

    % Plot individual stem plots for each B
    stem(k,ck_out,'marker','.')
    title(sprintf('B: %d', 1000*i))
    xlabel('Frequency')
    ylabel('Amplitude |ck|')
    axis([-1000 1000 0 1]);

end
```

The plot can be seen in Figure 2.

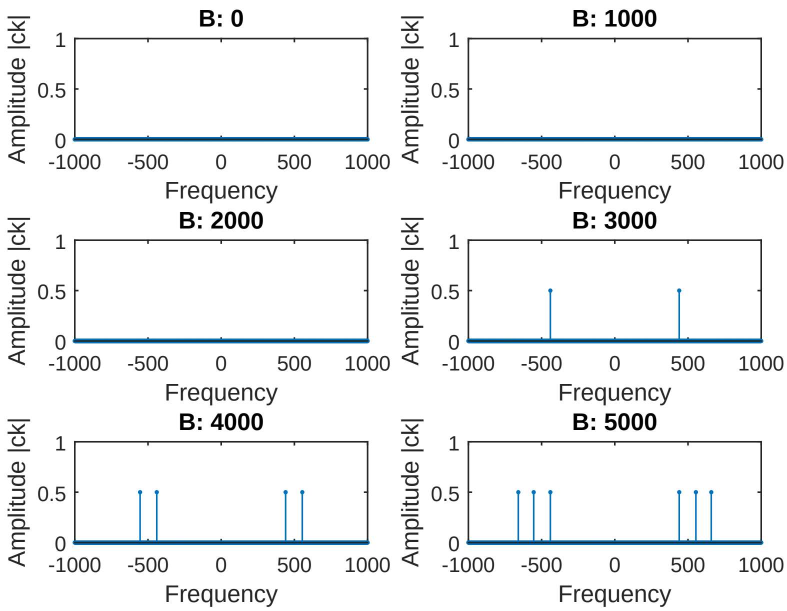

Figure 2: Magnitude of Complex Exponetials

Problem 5

Finally, the script from problem 4 was modified so that the filtered signal would play after each iteration of filtering. The code can be seen below:

```
% Script simply plots k vs ck for a signal composed of a finite number
% of cosine functions

% Determine size of plot
n = 1000;
w0 = 2*pi;

% Create values for k
k_pos = 1:n;
k = -n:n;

% Create storage vector for |ck|
pos_ck = zeros(1,n);

% Insert ck amplitude into storage vector
A = [440 554 659];
pos_ck(A) = 0.5;

% Compose |ck|
ck = [fliplr(pos_ck) 0 pos_ck];
ck_out = zeros(1,6);

% Set up sampling frequency and time duration to play back filtered signal
dur = 2.0;
fs = 44100;
t = 0:(1/fs):dur;

for i = 0:5

    % Plot individual stem plots for each B
    ck_out = low_pass_filter(n,ck,w0,1000*i);
    subplot(3,2,i+1)
    stem(k,ck_out,'marker','.')
    title(sprintf('B: %d', 1000*i))
    xlabel('Frequency')
    ylabel('Amplitude |ck|')
    axis([-1000 1000 0 1]);

    for j = 1:length(k_pos)

        % Check for magnitudes at frequency
        if ck_out(j) ~= 0

            % Convert back to time series
            x = ck_out(j)*( exp(1i*k(j)*w0*t) + exp(-1i*k(j)*w0*t) );

            % Exclude any complex exponentials
            real_x = real(x);
            sound(x,fs)
            pause(3)
        end
    end
end
```