EE3TR4 – Lab 1 Report

Yiming Chen, 400230266

Ruiyi Deng, 400240387

cheny466@mcmaster.ca

dengr6@mcmaster.ca

Feb 4th, 2022

Design Objective:

The output of a function generator is a square wave with 50% duty cycle and the period of the wave is 0.1 ms. You may assume that the amplitude of the square wave is 1 Volt. Design a second order Butterworth filter so that any harmonics in the output is at least 23 dB below the fundamental sinusoid at 10 kHz, and the loss of the fundamental sinusoid due to filtering should be less than 2 dB. The purpose of this experiment is to generate a high amplitude sinusoidal signal out of a square wave (to have a high-quality sinusoid, the amplitudes of harmonics should be sufficiently low).

Analysis of frequency response of the second order Butterworth Filter:

In this lab we are required to design the second order Butterworth filter. To simulate it, the transfer function of the filter should be obtained. From the textbook, we know that

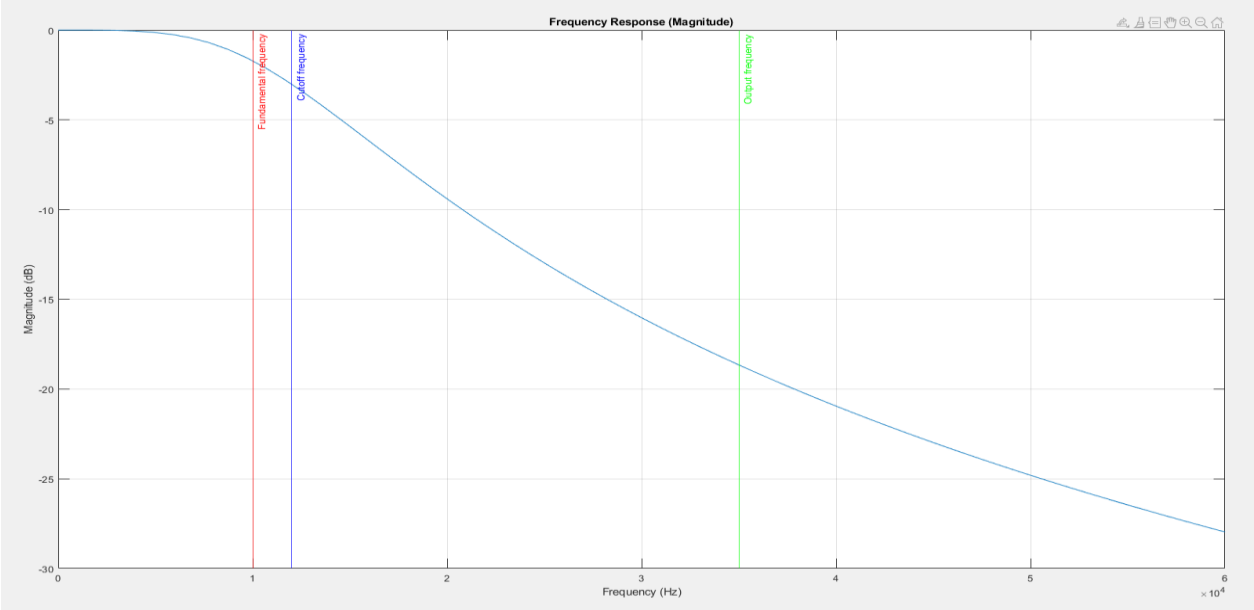
$$H(s) = \frac{1}{s^2 + 1.414s + 1}$$

The file used in this section is frequencyResponse.m. The purpose of this script is to simulate the frequency response of the second order Butterworth filter. Attached is the code used to build the filter.

```
%% set parameter of ButterWorth Filter
% n = 2; % order of the filter
fc = 1.2*10e3; % cutoff frequency
f0 = 10e3; % fundamental frequency
f_o = 3.5*10e3; % output frequency, 23dB below

% List of the parameter for ButterWorth Filter from Wikipedia
% n = 1: Bn = (s+1)
% n = 2: Bn = s^{2}+1.414s+1
% n = 3: Bn = (s+1)(s^{2}+s+1)
% n = 4: Bn = (s^{2}+0.7654s+1)(s^{2}+1.8478s+1)
% n = 5: Bn = (s+1)(s^{2}+0.6180s+1)(s^{2}+1.6180s+1)

% for denominator
a = [1 1.414 1];

% for nominator
b = 1;
```

```
%% Build the plot of ButterworthFilter
step = linspace(0,8);
h = freqs(b,a,step); % a for poles, b for zeros, step for normalized freq range from 0 to 8 rad/s
mag = 20*log10(abs(h)); % convert magnitude to dB

% plot frequency response
figure(1)

% frequency response

plt = plot(fc*step,mag);
xlim([0 60000]);

grid on
title('Frequency Response (Magnitude)');
xlabel('Frequency (Hz)')
ylabel('Magnitude (dB)')
xline(fc,'b',{'Cutoff frequency'}); % 3dB frequency (at cutoff frequency)
xline(f0,'r',{'Fundamental frequency'}); % fundamental frequency
xline(f_o,'g',{'Output frequency'}); % 23dB below
```

The first step is to set some basic parameters. We first set three different frequencies - cutoff frequency, fundamental frequency, and output frequency. The fundamental frequency can be directly determined at 10 kHz as required from the lab document which is the frequency of the sinusoid. The reason for the decision of the cutoff frequency and the output frequency will be illustrated later. The command xline is used to plot the reference line at three pre-set frequencies. After executing the script, we will have the figure below:



From this figure, we can clearly observe the frequency response of the filter. This is a low pass filter and with the rise of frequency, the magnitude will go through a steep falling edge and then becomes smooth.

We also implemented the verification part and it helps us to check whether our design met the requirements.

```
%% Verification of the constraints
% obtain the interpolation points
cq = interpl(fc*step, mag, [fc f0 f_o]);

message = ['Attenuation at cutoff frequency: ', num2str(cq(1)), ' dB'];
disp(message);

message = ['Attenuation at fundamental frequency: ', num2str(cq(2)), ' dB'];
disp(message);

message = ['Attenuation at output harmonic: ', num2str(cq(3)), ' dB'];
disp(message);

if (abs(cq(2)) < 2)
    flag = 'Yes';
else
    flag = 'No';
end


message = ['The loss of the fundamental frequency due to filtering is less than 2 dB: ', flag];
disp(message);

message = ['Output of harmonics is at least ', num2str(-(cq(3)-cq(2))), ' dB below fundamental frequency'];
disp(message);

if (abs(cq(3)-cq(2)) > 13.46)
    flag = 'Yes';
else
    flag = 'No';
end

message = ['Does the design meet the required attentuation from filtering (at least 13.46 (23-9.54)dB): ', flag];
disp(message);
```

The code shown above will detect the amplitude at specific frequencies and do some comparisons to check the correctness. And when

Because the constraint requires us to build the loss of the filtered signal with less than 2 dB gain, we chose the cutoff frequency at 12 kHz. Moreover, it is asked that any harmonics in the output should be at least 23 dB below the fundamental frequency. With the analysis of Fourier transform on the square wave function, it naturally has an attenuation at approximately 9.5 dB. As a result, the filter will only need to provide an attenuation larger than 13.5 dB. We chose the output frequency at 35 kHz and the numerical comparison result from our implemented checking application is shown below:

```
Attenuation at cutoff frequency: -3.0155 dB
Attenuation at fundamental frequency: -1.7175 dB
Attenuation at output harmonic: -18.6542 dB
The loss of the fundamental frequency due to filtering is less than 2 dB: Yes
Output of harmonics is at least 16.9367 dB below fundamental frequency
Does the design meet the required attentuation from filtering (at least 13.5 (23-9.5)dB): Yes
```

This result indicates that our design has met all the requirements.

Impact of the Cutoff Frequency:

Apart from the analysis of the frequency response, some other points are also worth mentioning.

We think the key point of finding acceptable range of the cutoff frequency to look for the maximum cutoff frequency allowing the loss of magnitude less than 2 dB. The lower bound should be larger than 11.5 kHz according to the previous tests. After several tests, we find the maximum cutoff frequency should be lower than 15.6 kHz to meet the output frequency at 35 kHz. Below is the verification part:

```
Attenuation at cutoff frequency: -3.0155 dB
Attenuation at fundamental frequency: -0.67962 dB
Attenuation at output harmonic: -14.2037 dB
The loss of the fundamental frequency due to filtering is less than 2 dB: Yes
Output of harmonics is 13.5241 dB below fundamental frequency
Does the design meet the required attentuation from filtering (at least 13.5 (23-9.5)dB): Yes
```

To avoid posing negative effects on the quality of the output signal, the frequency of the output signal should be as low as possible. We prefer 35 kHz as a reasonable choice and under this choice, the acceptable cutoff frequency is from 11.5 kHz to 15.6 kHz.

Because the rate of attenuation becomes slower at high frequency, even if to choose a comparatively high cutoff frequency can meet the 2 dB requirement, it will be harder to determine the output frequency with the attenuation of the 13.5 dB. The resulted frequency will be at very high level thus ruining the quality of the signal. This problem can be solved with higher order Butterworth filters because we know that higher order Butterworth filter possesses a steeper falling edge which should fix this problem well. But here our design can just handle the required attenuation, so we do not need higher order filters.

Besides, the lab document asks us to realize the attenuation at 25 dB and this can be easily reached with output frequency at more than 38 kHz.

Simulation:

We managed to simulate in the similar way with the triangular_filtering.m which is provided in the lab folder on a2l and rebuild our own 2-nd order Butterworth Filter. Our own simulation script for this lab is named lab1.m.

Input Simulation:

According to the lab document, we are asked to generate a square wave with 50% duty cycle with a period of 0.1 ms and an amplitude of 1V. To meet the requirement, we used square function from MATLAB's Signal Processing Toolbox. By doing that, we get a square wave with period of 0.1 ms and amplitude of 1V in an interval from 0 ms to 0.3 ms.

```
10      % Input - Square Wave Signal
11 -    f0 = 10000; % asked period is .1 ms
12 -    T0 = 1/f0; % period
13 -    t_s = 0.001*T0;
14 -    samples = 6*T0/t_s + 1;
15 -    range_s = -3*T0:t_s:3*T0; % x-axis
16
17      % generate the sqaure wave
18 -    input_square = square(range_s*2*pi*f0, 50);
```

We generate magnitude and phase spectrums of the input signal in the same manner demonstrated in the triangular_filtering script. The Fourier series representation of the square wave was generated using the Fourier series coefficients of a square wave. The magnitude spectrum plot shows the absolute value of the Fourier series representation, while the phase spectrum plot shows the angles of the Fourier series representation.

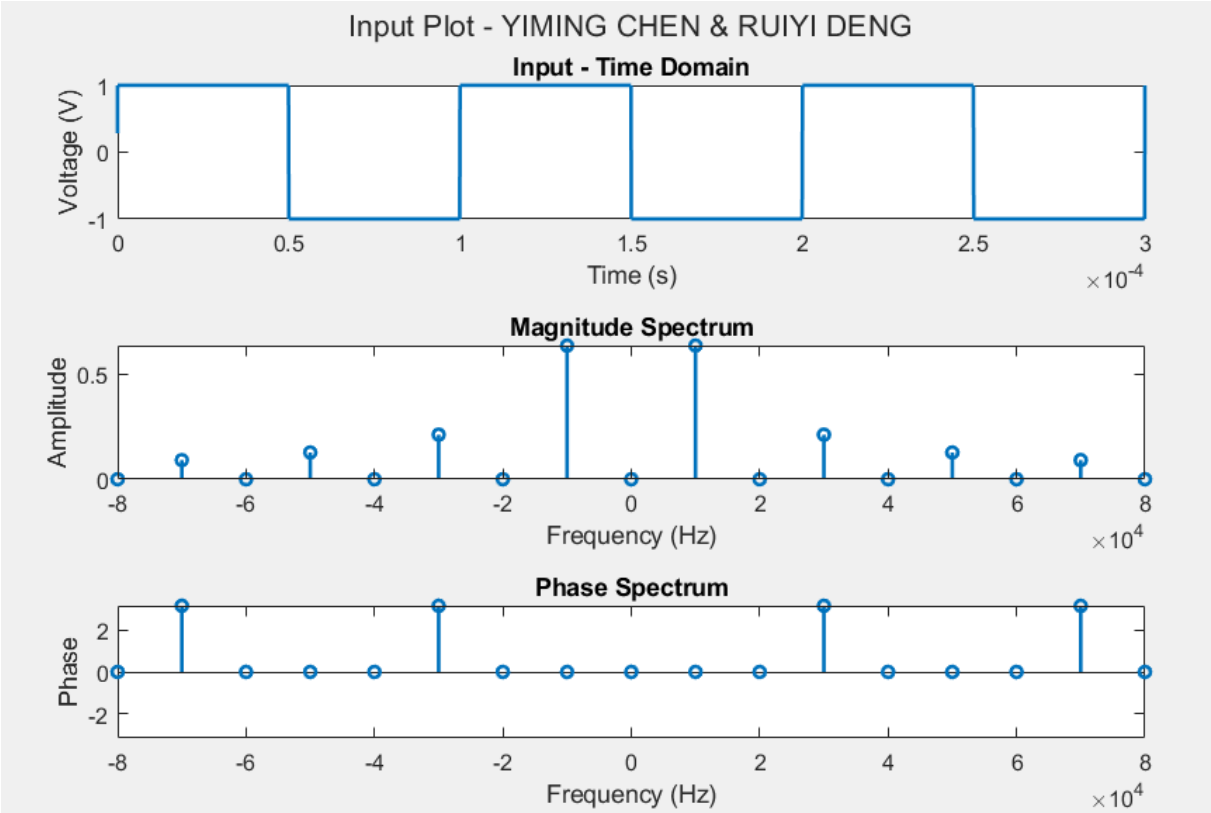
```
30 -    N=100; % number of harmonics
31 -    nvec = -N:N;
32 -    c_in = zeros(size(nvec));
33 -    for n = nvec
34 -        m = n+N+1;
35 -        if (mod(n,2))
36 -          c_in(m) = sinc(n/2); % fourier transformation of the sqaure wave
37 -        else
38 -          c_in(m) = 0.0;
39 -        end
40 -    end
41      % parameter
42 -    f = nvec*f0; % frequency vector
43 -    mag = abs(c_in);
44 -    phase = angle(c_in);
```

The magnitude spectrum shows the magnitude of the frequency response, we can see that there is an impulse at every odd harmonic (for each sinusoid that makes up the Fourier series representation of the square wave), and the fundamental sinusoid which is at frequency of 10 kHz contributes the most to the signal, with each subsequent sinusoid contributing a decreasing amount. These harmonics occur at every other multiple of the fundamental frequency (10 kHz, 30 kHz, 50 kHz, etc.).

Filter and Output Simulation:

We used the Fourier series representation method according to the lab

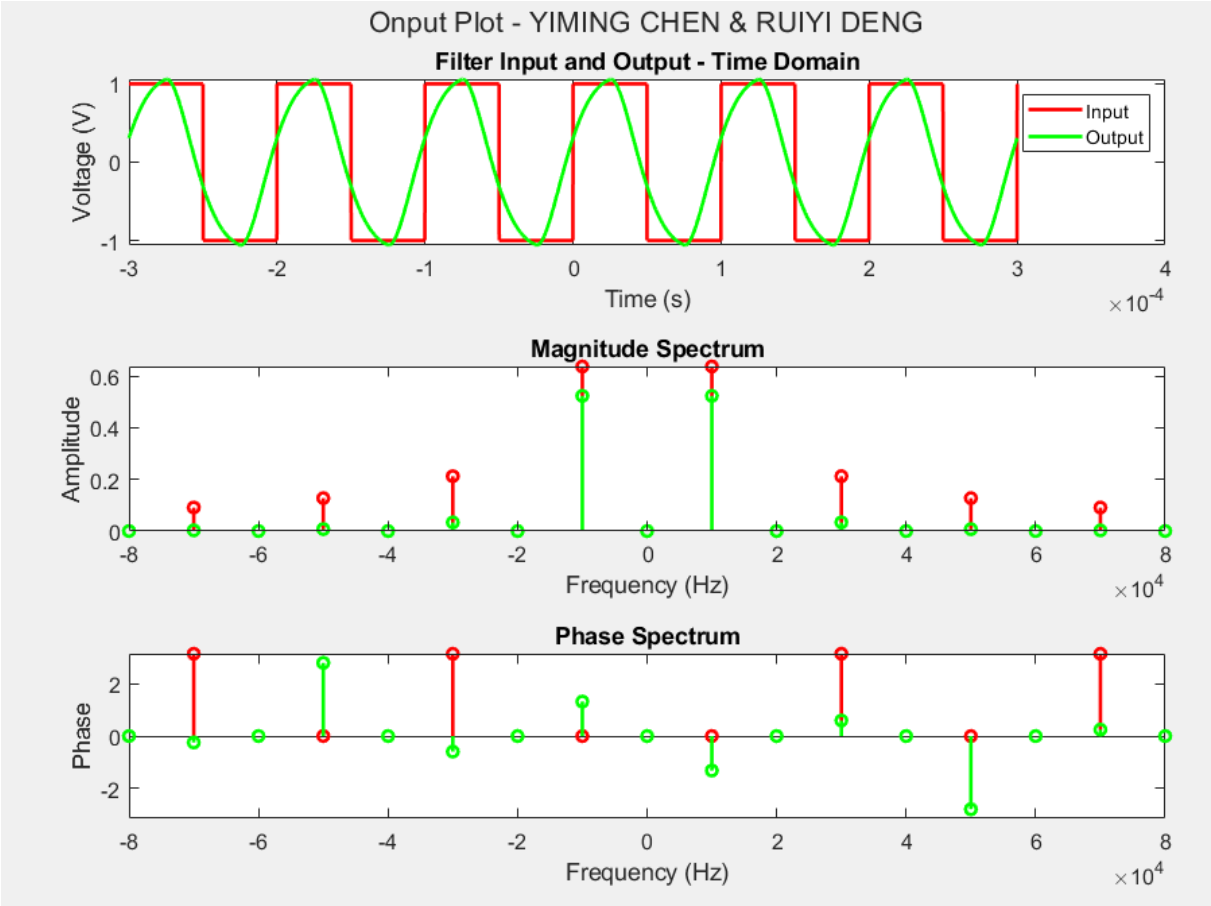
document.



The Fourier series representation of the input signal was multiplied by the transfer function evaluated at the corresponding frequencies. The cutoff frequency used when evaluating the transfer function was the minimum acceptable cutoff frequency that was previously calculated, 12 kHz. We generate the magnitude spectrums of the output signal using this new Fourier series representation in the same method applied on the input signal. The weighted sum of the Fourier series is calculated to generate the output signal in the time domain.

```
79      %% Implementation of the 2nd-order ButterWorth Filter
80 -    fc = 1.2*10e3; % cutoff frequency found from the previous section
81 -    Hf = 1 ./ (1+1.414*(1i*f/fc) + (1i*f/fc).^2); % transfer function
82 -    c_out = c_in .* Hf; % Fourier coefficients of the filter output
83
84      %% Fourier Coeff
85 -    A = zeros(2*N+1,ceil(samples));
86 -    for n = nvec
87 -        m=n+N+1;
88 -        A(m,:) = c_out(m) .* exp(1i*2*pi*n*f0*range_s);
89 -    end
90 -    gp_out = sum(A);
```

Below is the time and frequency domain plots of the output, compared with the corresponding plots of the input.



The output of the filter in the time domain closely resembles a pure sine wave with a period of 0.1 ms, which matched our expectation at the output. All harmonics other than at the fundamental frequency have been significantly weakened , leaving most of the Fourier series contribution to the sinusoid at the fundamental frequency (10 kHz). An evaluation of the attenuation in the magnitude 9 spectrum reveals that the filter satisfied the design constraints, with 1.96 dB attenuation at the fundamental frequency and 16.75 dB attenuation at the third harmonic. These values were also nearly identical to the values determined during the design process.

Reference: triangular_filter.m provided at a2l.