Lab 1 - Fourier Analysis

ELECENG 3TR4

Gurleen Dhillon Benji Richler

dhillg25 richlerb

400301955 400296988

2023-02-12

Description

A second-order low-pass Butterworth filter is used to remove high-frequency noise and allows the low-frequency noise to remain, hence the name "low-pass". To design the filter, the cutoff frequency, the period of the input signal, and other filter coefficients need to be determined, as shown in figure 2. For this lab, we utilized two 10nF capacitors, three 8.2kΩ resistors, and one 4.7kΩ resistor, as well as an LM741 op-amp. The input of this filter is a square wave and in this lab, we will be comparing the calculated waveform with the measurements from the lab. The expected output is a high-amplitude sinusoid signal. The output can be determine in the time domain or the frequency domain, using $y(t) = h(t) \circledast x(t)$ and $Y(nf_0) = X(nf_0)H(nf_0)$, respectively.

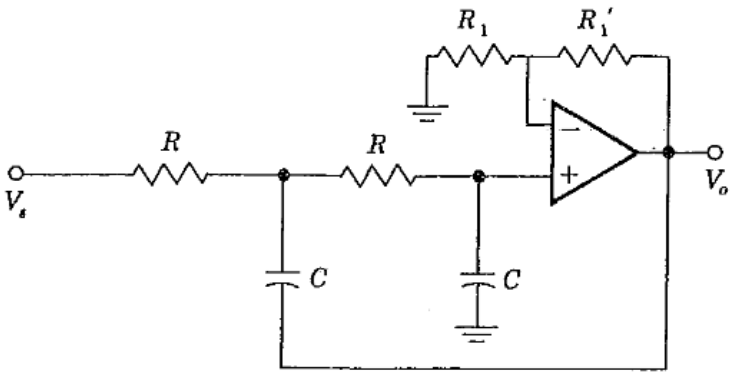


Figure 1: Second-Order Low-Pass Butterworth Filter

$K = \frac{\sqrt{2}}{2}$

$A_{V0} = 3 - 2K = 3 - 1.414 = 1.586$

$A_{V0} = \frac{R' + R_1}{R_1}$ $\quad R_1 = 8.2k$

$1.586 = \frac{R' + 8.2k}{8.2k}$

$R' = 4.8k \approx 4.7k$

$f_c = \frac{1}{2\pi RC} = 1941Hz$

$C = 10nF$

$R = 8.2k\Omega$

Figure 2: System component value calculations

MATLAB Code

```
%% triangular wave generator but making it into a square wave generator
clc
clear all
hold off

A=1; % amplitude of input signal

f0=1000;     %fundamental freq of input triangular wave
T0 = 1/f0;  %period
tstep = 0.005*T0;
no_sample = 3*T0/tstep + 1; %no. of samples  within  3*T0
no_sample1 = T0/tstep + 1; %no. of samples  within  T0

tt = -1.5*T0:tstep:1.5*T0;
tt1 = -0.5*T0:tstep:0.5*T0; % time vector for the period -0.5T0 to 0.5T0

gp1 =  A*sign(sin(2*pi*f0*tt1)); %input
gp_in = [gp1 gp1(2:no_sample1-1) gp1]; %3 cycles of the triangular wave
figure(1)
Hp1 = plot(tt,gp_in);
set(Hp1,'LineWidth',2)
Ha = gca;
set(Ha,'Fontsize',16)
title('input - time domain')
pause
```

Figure 3: The MATLAB code used to generate the input square wave

```
%% Fourier series representation of signal (Amplitude Spectrum)

K=1/(2*pi);
N=100; %no. of harmonics
nvec = -N:N;
c_in = zeros(size(nvec));
for n = nvec
    m = n+N+1;
    c_in(m) = 1i*K*((-1)^n)/n;

    if (n == 0)
      c_in(m) = 0.0;
    end
end
f = nvec*f0; %frequency vector
figure(2)
Hp1=stem(f,abs(c_in));
axis([-8*f0 8*f0 0 max(abs(c_in))])
set(Hp1,'LineWidth',2)
Ha = gca;
set(Ha,'Fontsize',16)
title('magnitude spectrum of input')
pause
```

Figure 4: The MATLAB code used to generate the magnitude spectrum of the input wave

```
%% Fourier series representation of signal (Phase Spectrum)

figure(3)
Hp1=stem(f,angle(c_in));
set(Hp1,'LineWidth',2)
Ha = gca;
set(Ha,'Fontsize',16)
axis([-0.1e4 0.1e4 -pi pi])
title('phase spectrum of input')
pause

```

Figure 5: The MATLAB code used to generate the phase spectrum of the input wave

```
%% Designing the 2nd order Butterworth filter

R=8.2e3;
C=10.0e-6;
fc=1/(2*pi*R*C)     %cutoff freq of filter

%Hf = 1 ./(1+1i*f/fc) ; % 1st order filter transfer function
Hf = 1 ./(((1i*f/fc).^2)+(1.414*1i*f/fc)+1); % 2nd order filter transfer function
c_out = c_in .* Hf; %Fourier coefficients of the filter output

figure(4)
stem(f,abs(c_in),'r','LineWidth',2);
hold on
stem(f,abs(c_out),'b','LineWidth',2);
hold off
axis([-8*f0 8*f0 0 max(abs(c_in))])
Ha = gca;
set(Ha,'Fontsize',16)
title('magnitude spectrum of filter output and input')
Ha = gca;
set(Ha,'Fontsize',16)
legend('input','output')
pause
```

Figure 6: The MATLAB code used to design the Butterworth filter displaying both input and output on the same graph

```
%% Construct the output signal from the Cout Fourier coefficient

A = zeros(2*N+1,ceil(no_sample));
for n = nvec
    m=n+N+1;
    A(m,:) = c_out(m) .* exp(1i*2*pi*n*f0*tt);
end
gp_out = sum(A);
figure(5)
Hp1 = plot(tt,real(gp_out),'b',tt,gp_in,'r');
set(Hp1,'LineWidth',2)
Ha = gca;
set(Ha,'Fontsize',16)
title('filter input and output-time domain')
set(Ha,'Fontsize',16)
legend('output','input')
return

```

Figure 7: The MATLAB code used to display the filter input and output results in the time domain

Simulation Results

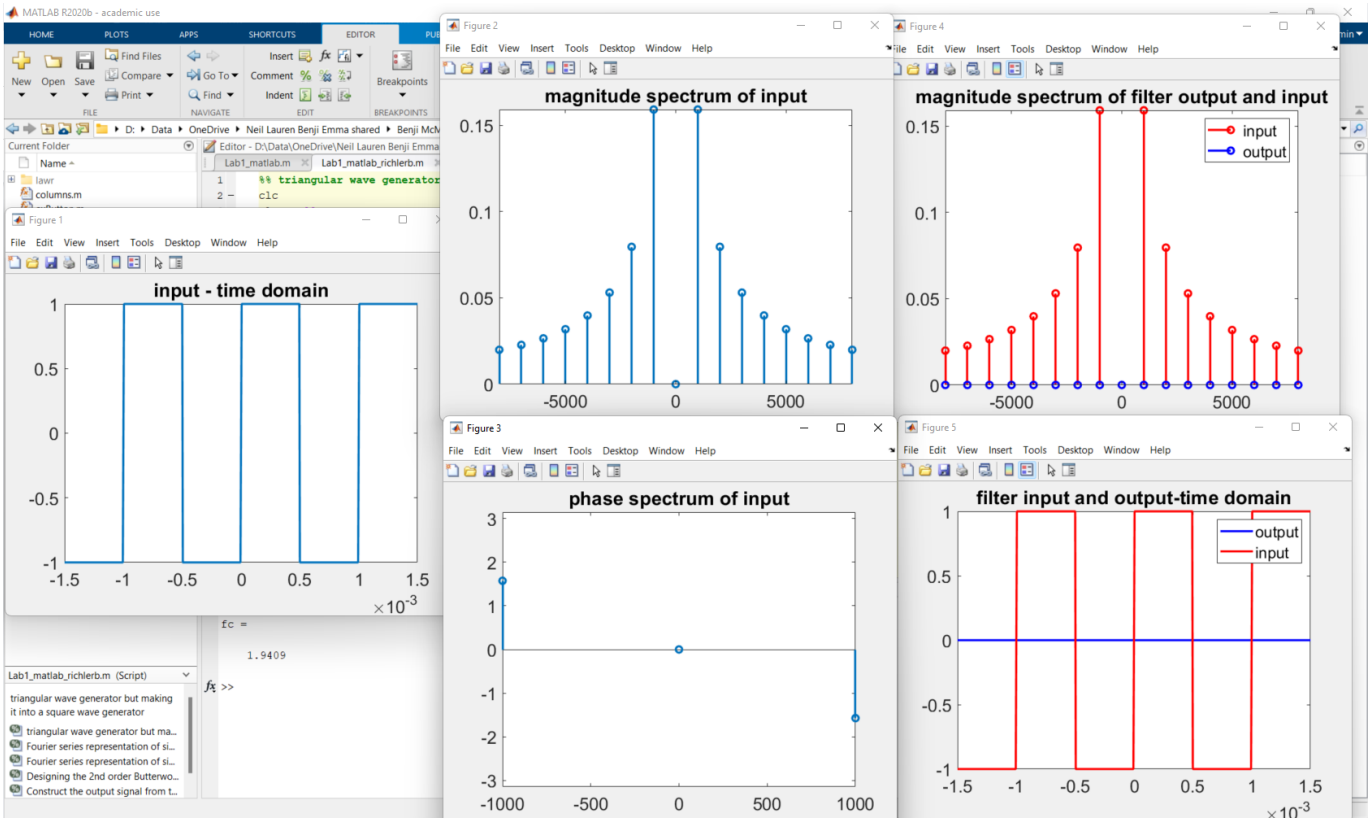


Figure 8: Matlab simulation results for the magnitude and phase of the output given a square input wave using a second order Butterworth filter

Part I

For the experiment section of this lab, we will be inputting a square wave with 1V Amplitude (2Vpeak-peak) with varying frequencies and measuring the corresponding relative amplitude and phase output. The output of the filter (as shown by channel 2), closely resembles a sine wave with a period very similar to its' input square wave (as shown in channel 1), which was anticipated. By increasing the frequency, the amplitude of the output would decrease. It was noted that most of the time, the frequency would increase by 100Hz in order to see a decrease of 0.5 in amplitude. As seen in figure 9, the real-time experiment results show that the FFT function has a 13dB attenuation when comparing its first peak (C_1) to its third (C_3). Knowing this, we can conclude that the following peaks are also decreasing in magnitude and that every harmonic would follow this pattern. When comparing the simulation data to the experimental data, there were little to no discrepancies visible.

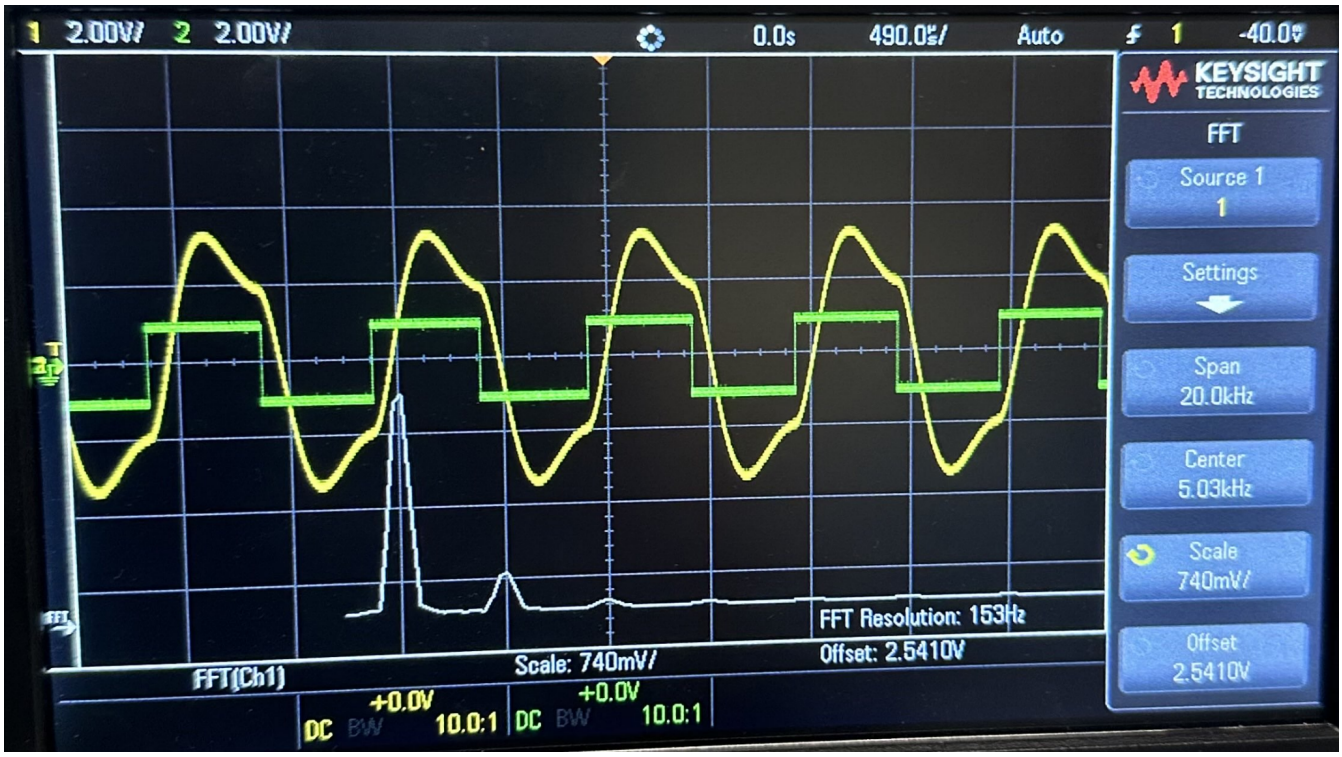
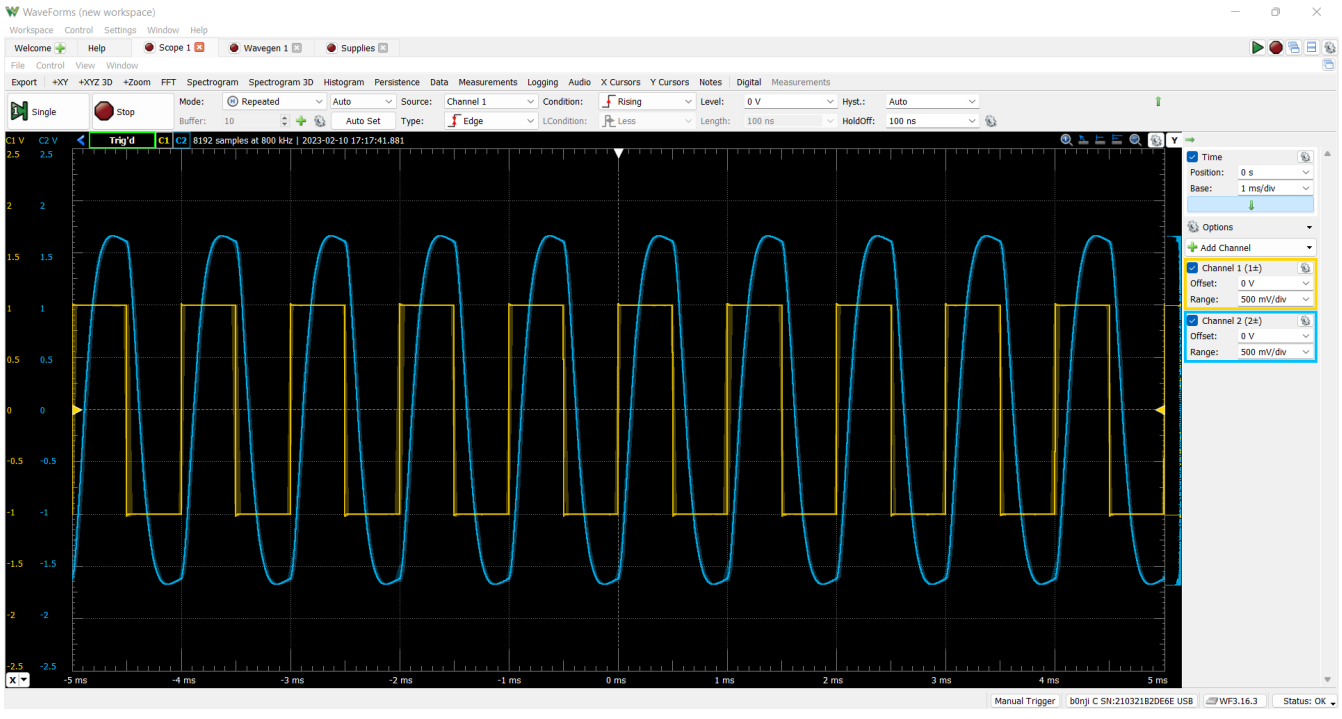
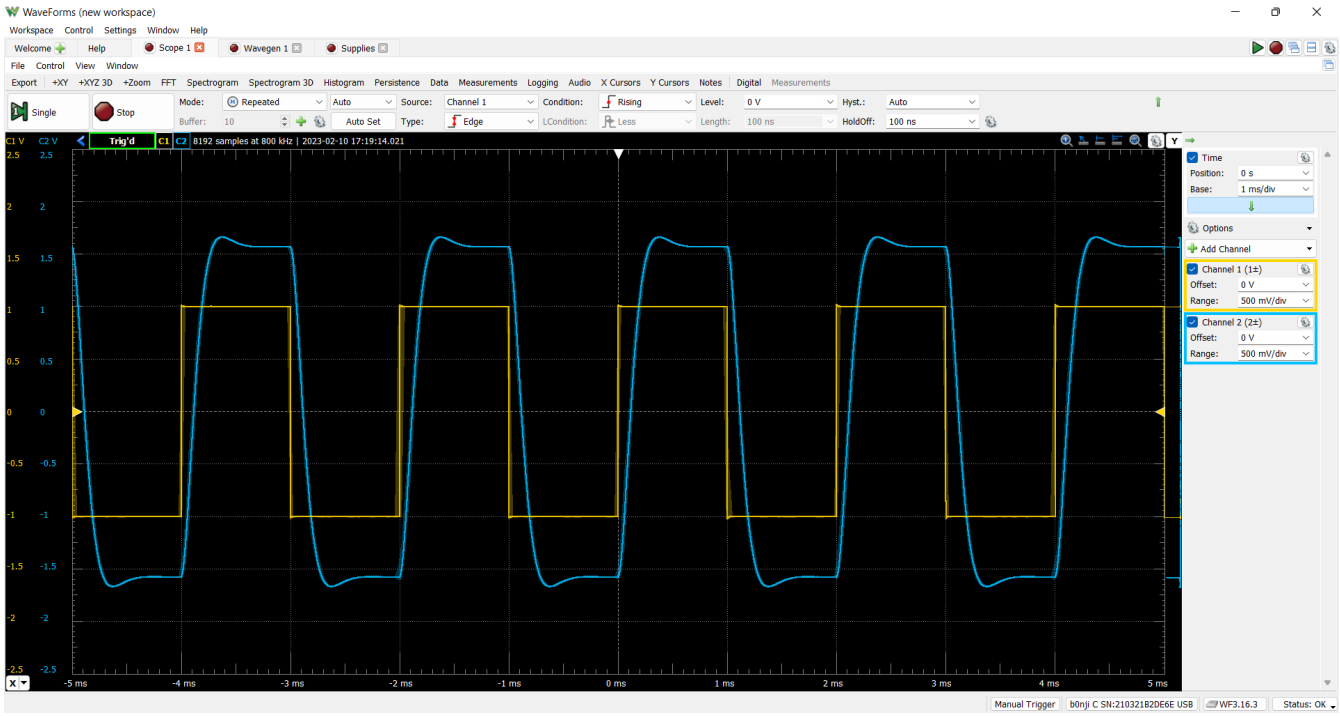


Figure 9: Experiment results produced by the circuit in the FFT function

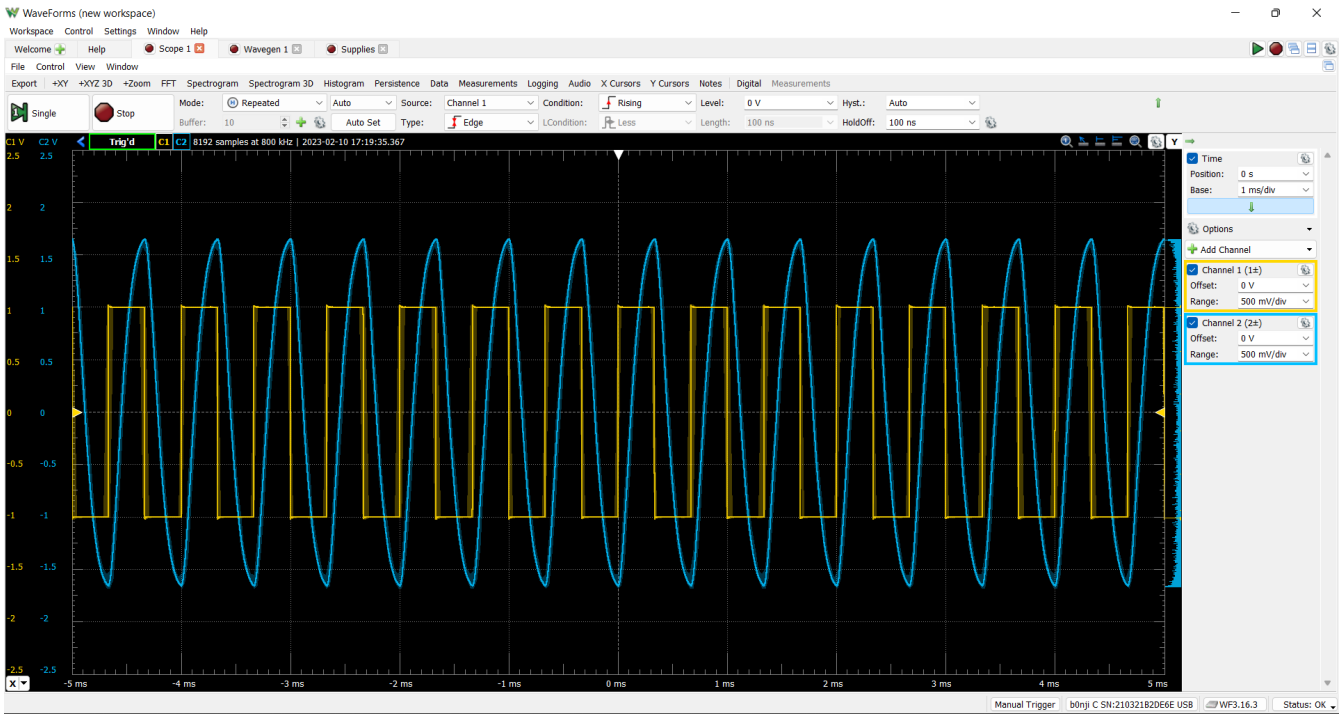
f = 1kHz:



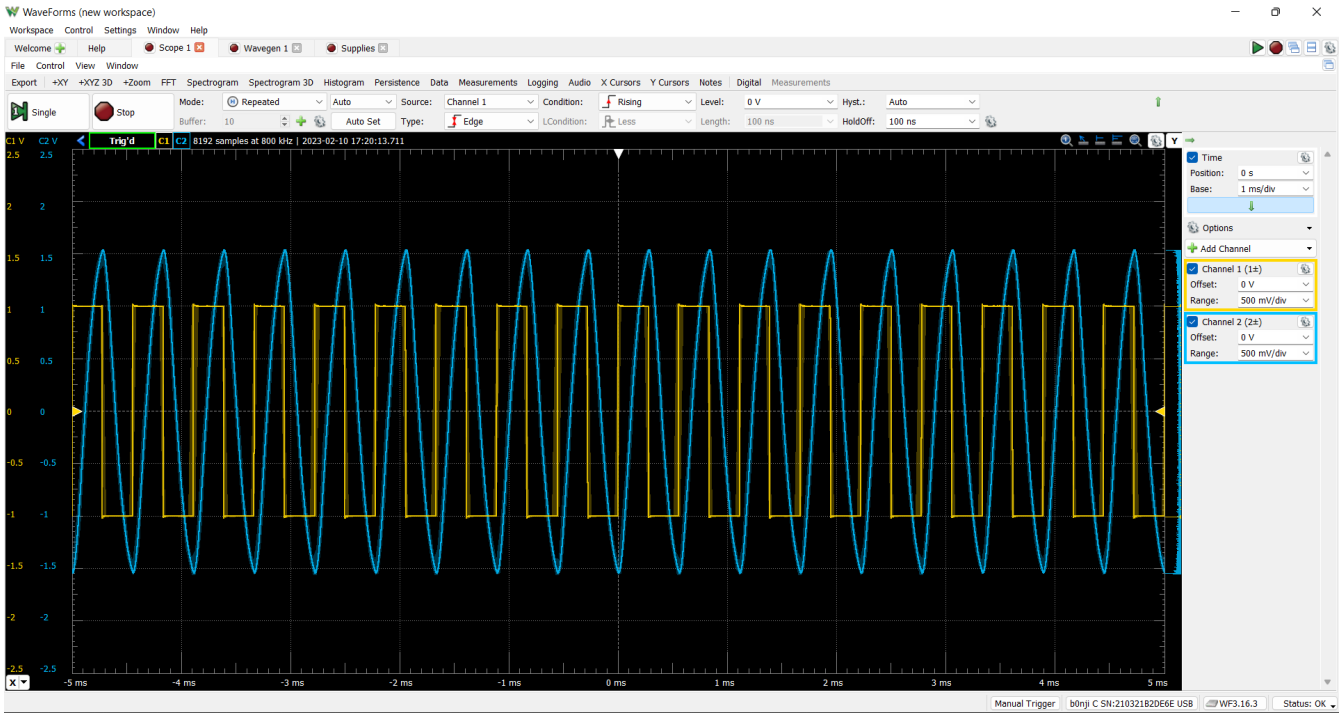
f = 500Hz:



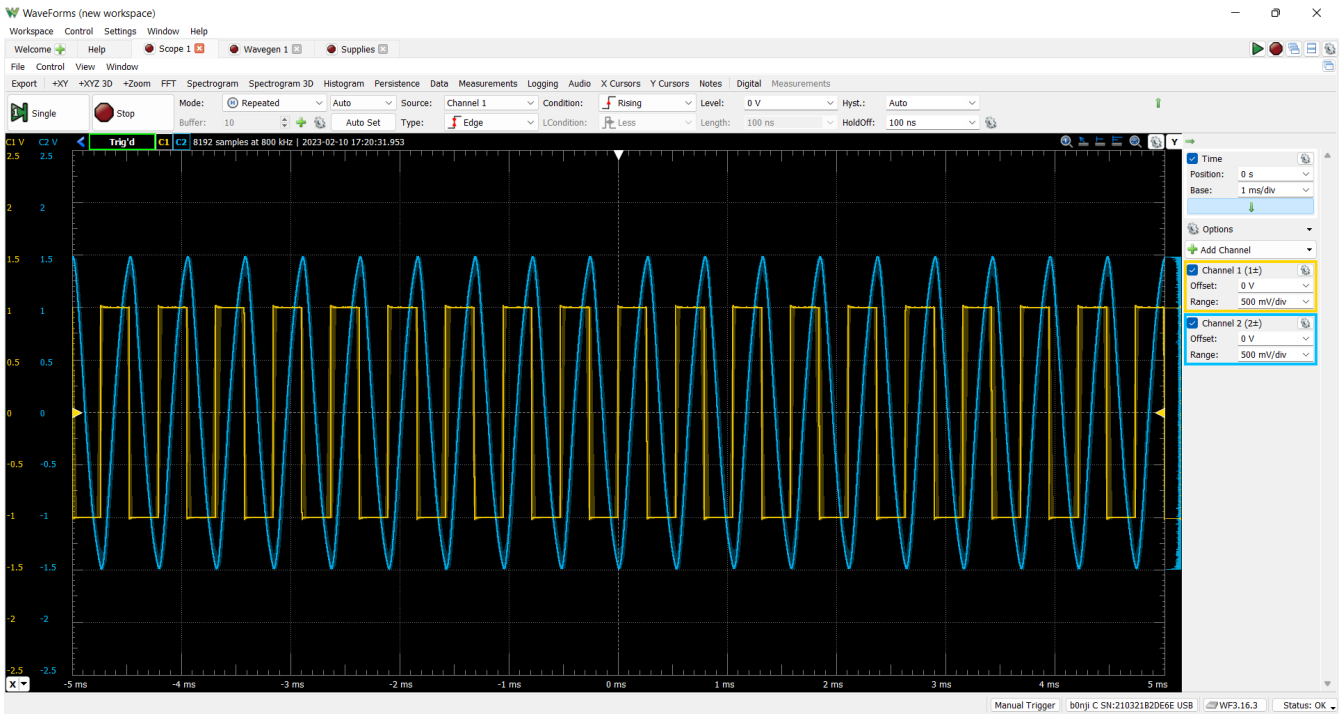
f = 1.5kHz:



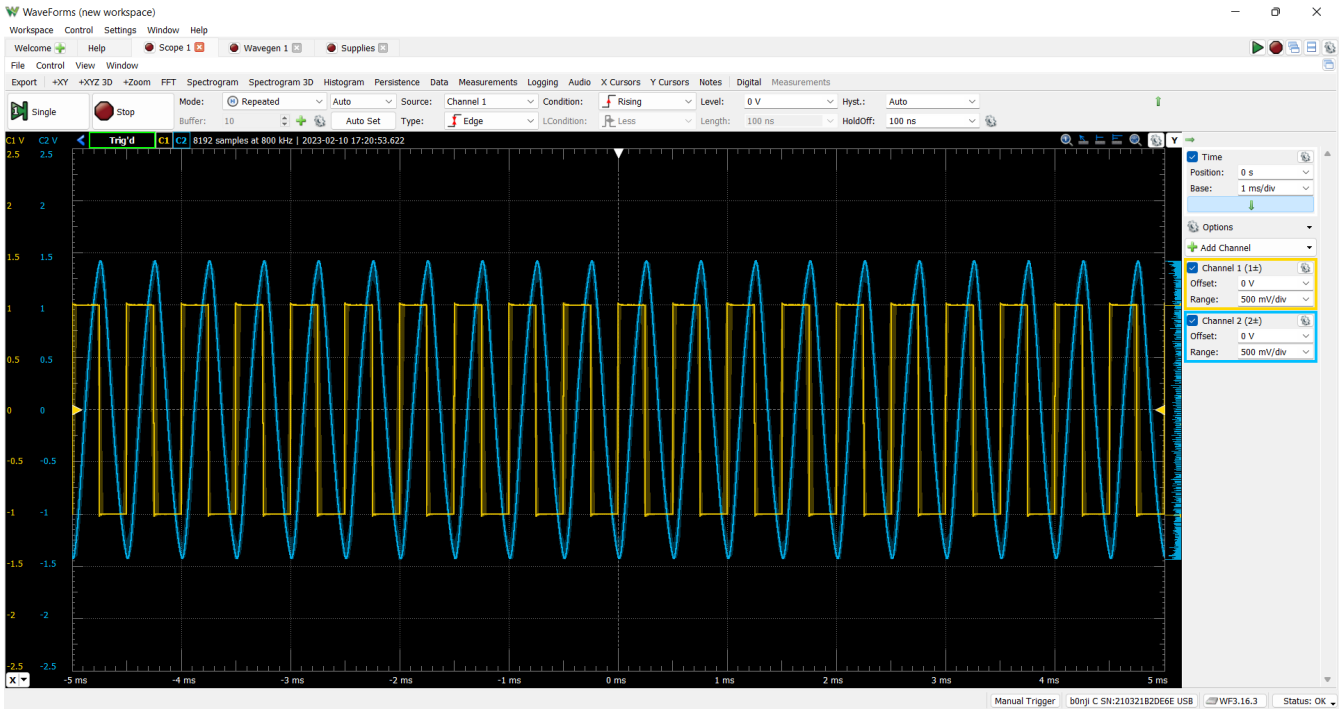
f = 1.8kHz:



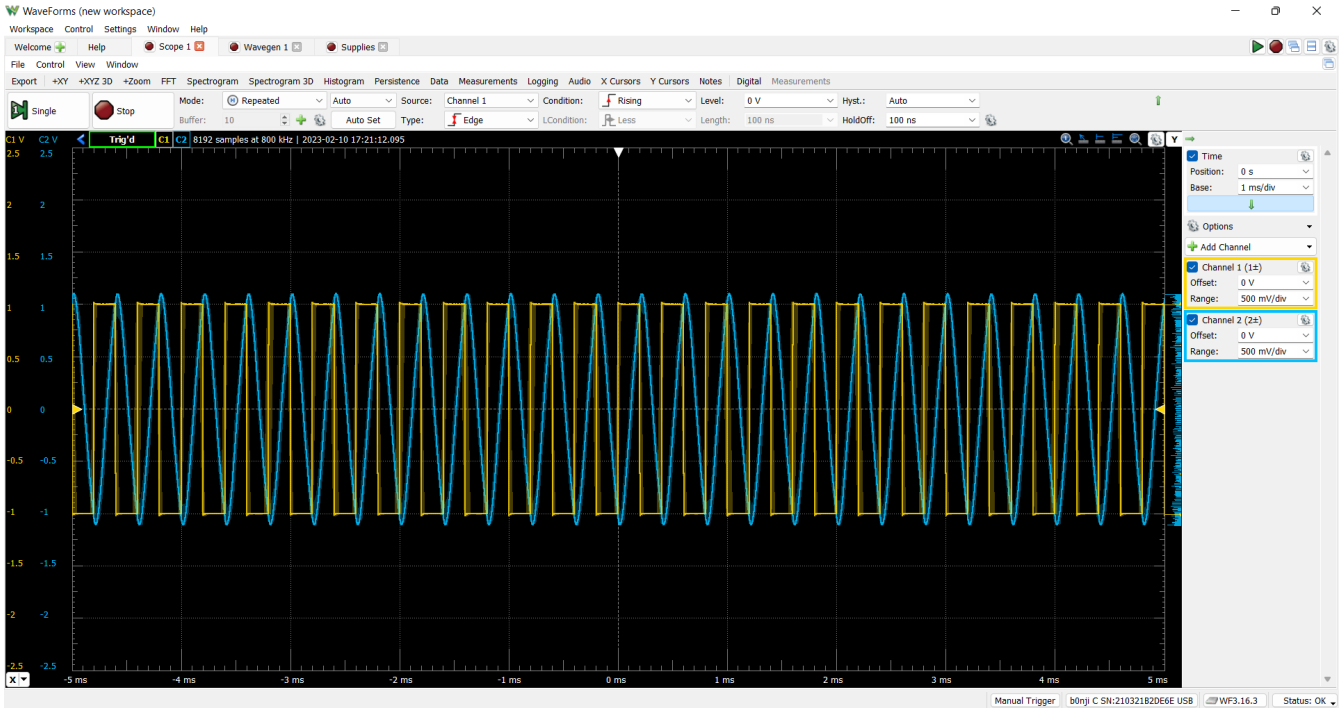
f = 1.9kHz:



f = 2.0kHz:



f = 2.5kHz:



f = 5kHz:

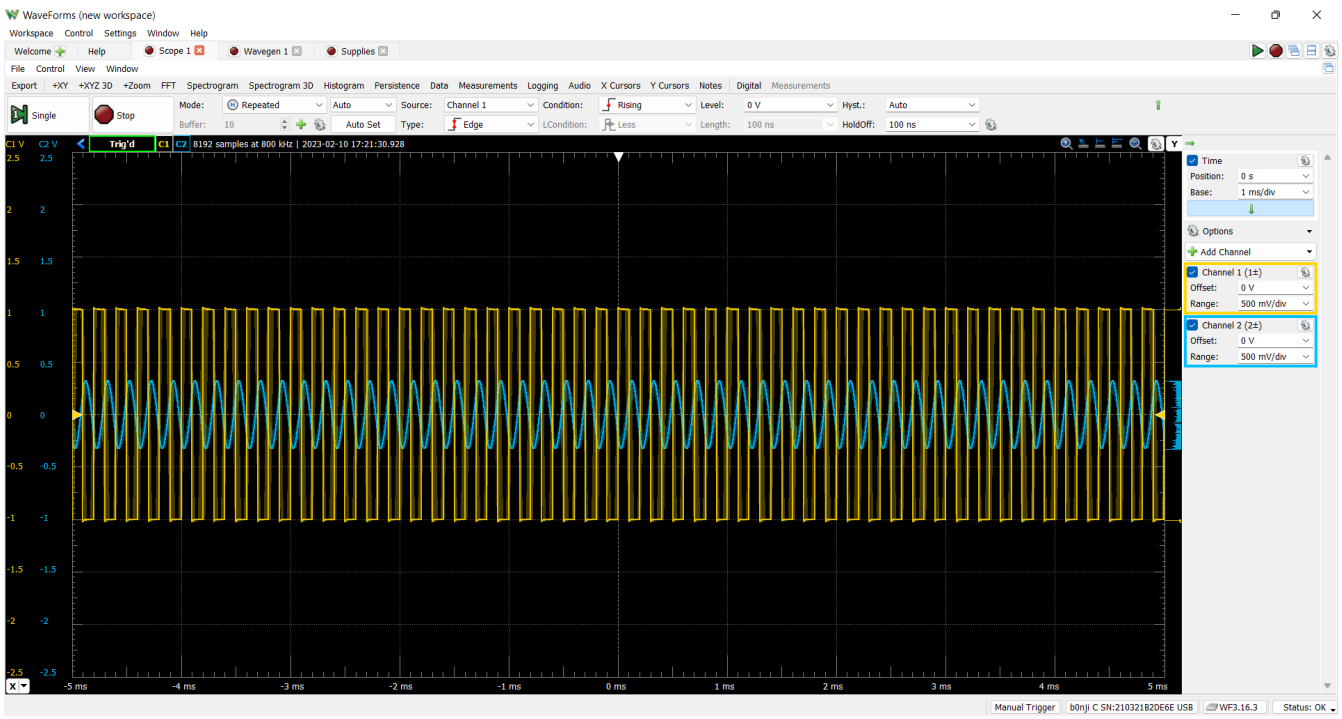


Table 1: Frequency input vs Amplitude output

Frequency (Hz)	Amplitude of output	Frequency (Hz)	Amplitude of output
100	1.7	2.8k	0.85
1k	1.65	2.9k	0.8
1.1k	1.6	3k	0.75
1.2k	1.55	3.2k	0.7
1.5k	1.5	3.4k	0.65
1.6k	1.45	3.5k	0.6
1.7k	1.4	3.7k	0.55
1.8k	1.35	3.8k	0.5
1.9k	1.3	4.3k	0.45
2k	1.25	4.7k	0.4
2.1k	1.15	5k	0.35
2.3k	1.1	5.6k	0.3
2.4k	1.05	6.9k	0.25
2.5k	1.00	7.5k	0.2

2.6k	0.95	10.1k	0.15
2.7k	0.9	13.7k	0.1

Part II

The low pass filtered speech sounds like it is highly muffled, as if someone is talking into a pillow. The higher pitches do not go that high and only the main bassier sounds (lower frequencies) are heard.

The band-pass filtered speech sounds like it is going through a standard telephone. All the parts of the speech can be heard and understood, but it is highly suppressed, in a telephone conversation type of sound.

Only the slightly painful, and higher frequencies can be heard in the high pass filtered speech. It also sounds like the speaker has a bit of a South african accent and the ‘el’ sound in “tell” sounds like a ‘k’ sound, making the word sound like “take”.

For the notch-filtered speech, the beep that is in the input signal is completely removed, although the speech sounds very close up, similar to the sound of the structures at playgrounds where you can speak to someone on the other side of the playground.