Electrical Engineering 3TR4
Laboratory 1 Fourier Analysis
2024-02-1

Instructor: Dr. Jun Chen

Zihan Wang– C01 – wangz726 – 400369195

Xiaoyan Li– C01 – li1675 – 400387478

1. Design and Simulation

We designed a second order Butterworth filter, as Av0 = (R1+R1')/R1=3-2k,2k=1.414, 0.586R1 = R1', We set the R1 = 5870 ohms and R1' = 10kohms. As for the parameter R, we set it is equal to 10kohms and we used two 10nF capacitors. Cutoff frequency *fc* = 1/（2*pi*R*C） = 16 kHz.

Figure 2: This figure shows a square wave as the input signal.

Figure 3: The magnitude spectrum shows the frequency components of the input square wave. As expected for a square wave, there are a series of harmonic spikes at odd multiples of the fundamental frequency.

Figure 4: This phase spectrum graph shows the phase of the signal's frequency components.

Figure 5: This graph compares the magnitude spectra of the input and output signals. It should be noted the high frequencies are attenuated, as shown by the reduced spikes in the output, indicating the filter is a low-pass filter that has attenuated the higher harmonics of the input square wave.

Figure 6: This graph shows the input and output signals in the time domain, which means that the high-frequency components of the square wave have been filtered out, resulting in a signal that rises and falls more gently compared to the sharp transitions of the input square wave.

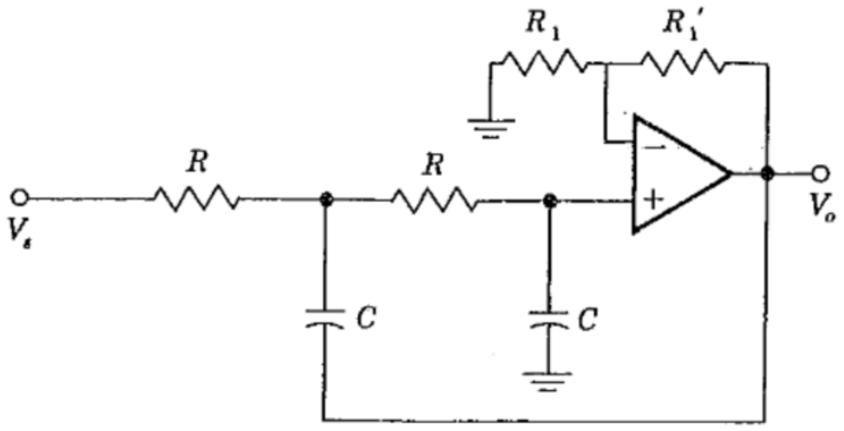


Figure 1. Second-order low-pass filter

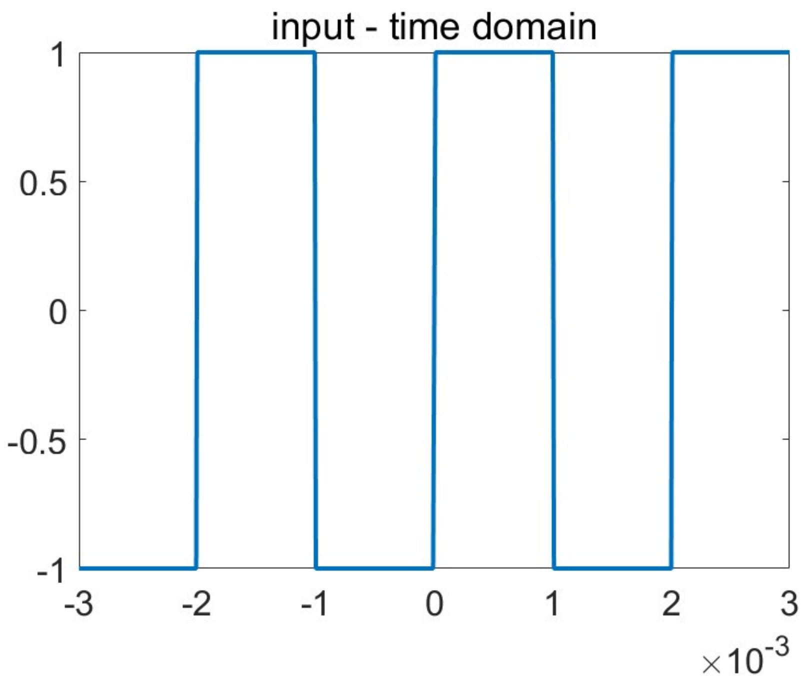


Figure 2. Input – time domain in Matlab

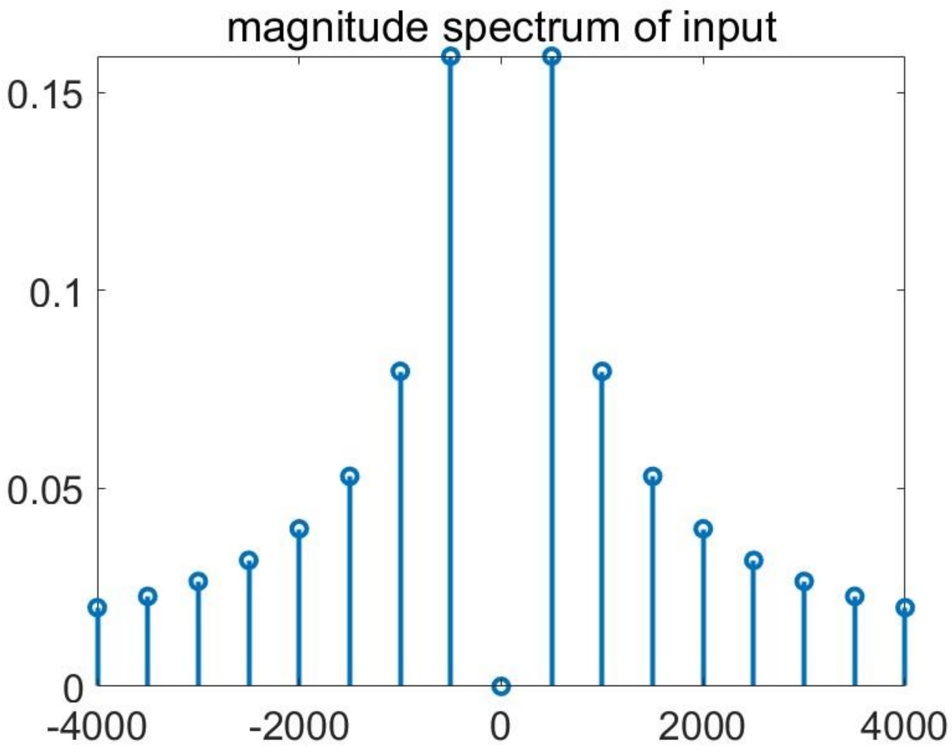


Figure 3. Magnitude Spectrum of Input in Matlab

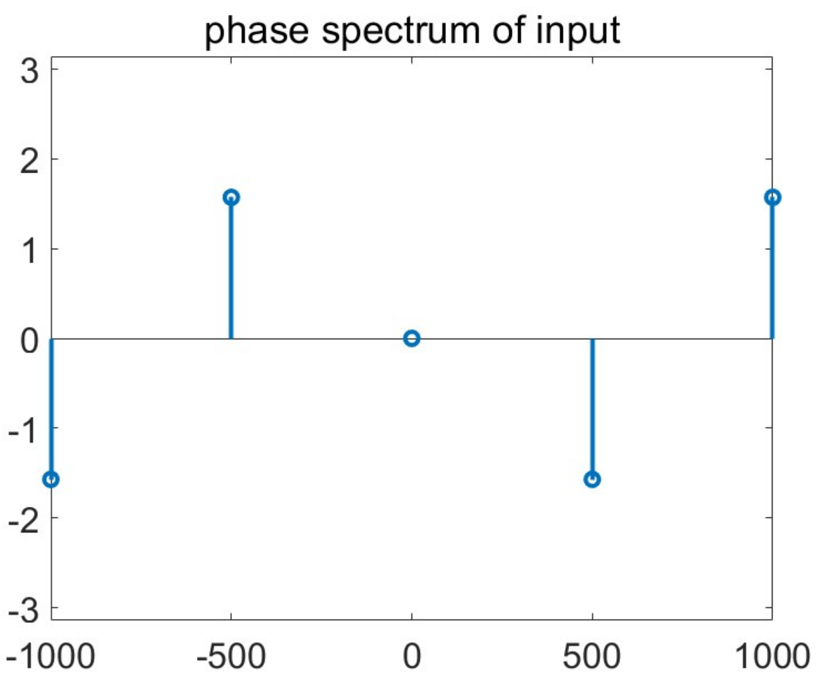


Figure 4. Phase Spectrum of Input in Matlab

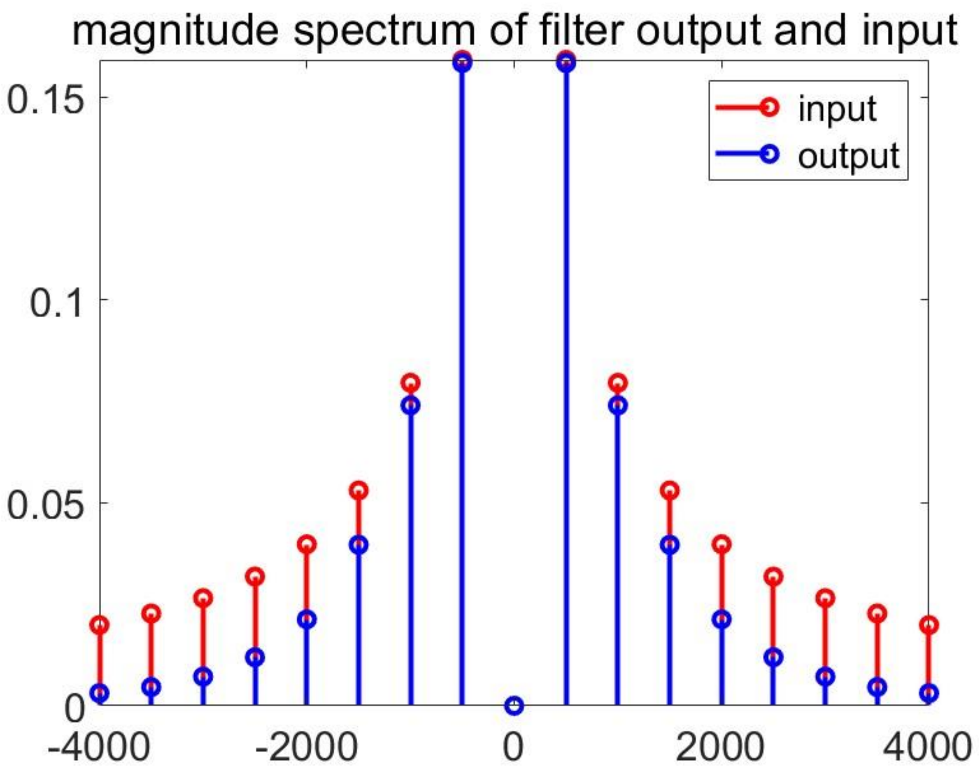


Figure 5. Magnitude Spectrum of Filter Output and Input in Matlab

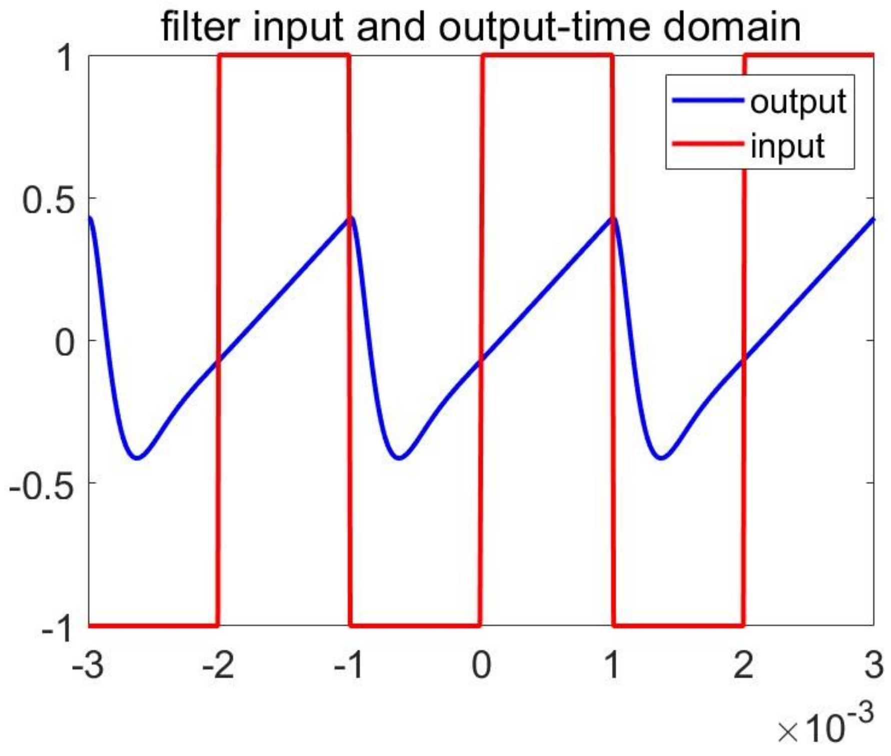


Figure 6. Filter Input and Output – Time Domain

2. **Part I: Butterworth Filter Experiments**

The experiment used a 1 Volt amplitude square wave to analyze both the frequency and time domains of the output signal. In the time domain, the output signal was expected to resemble a sine wave with a period matching the input square wave, which was indeed observed. This similarity in periods between the input and output signals aligns with the simulated results. A significant finding was the 16 dB decrease in the amplitude of the output signal across the harmonics, especially noticeable in the odd harmonics. This observation is consistent with the characteristic attenuation behavior of a low-pass filter in the frequency domain, demonstrating the relationship between increased frequency and greater attenuation. In conclusion, the experimental results closely match the simulations.

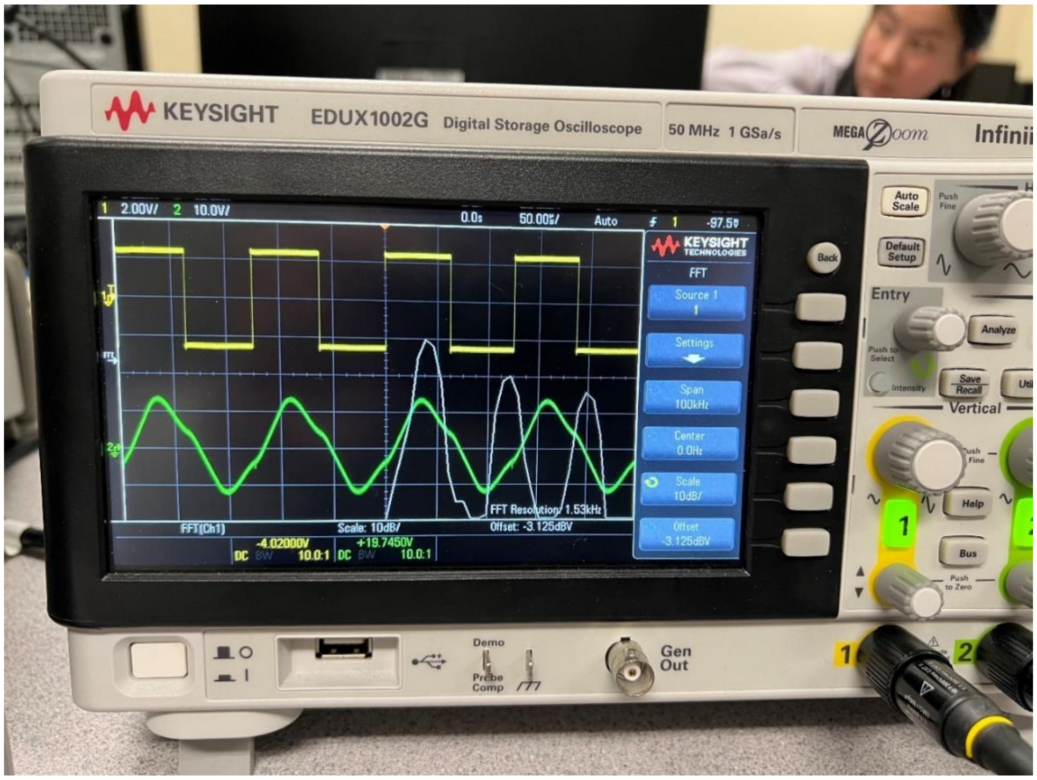


Figure 7. Experimental Result of Input Signal

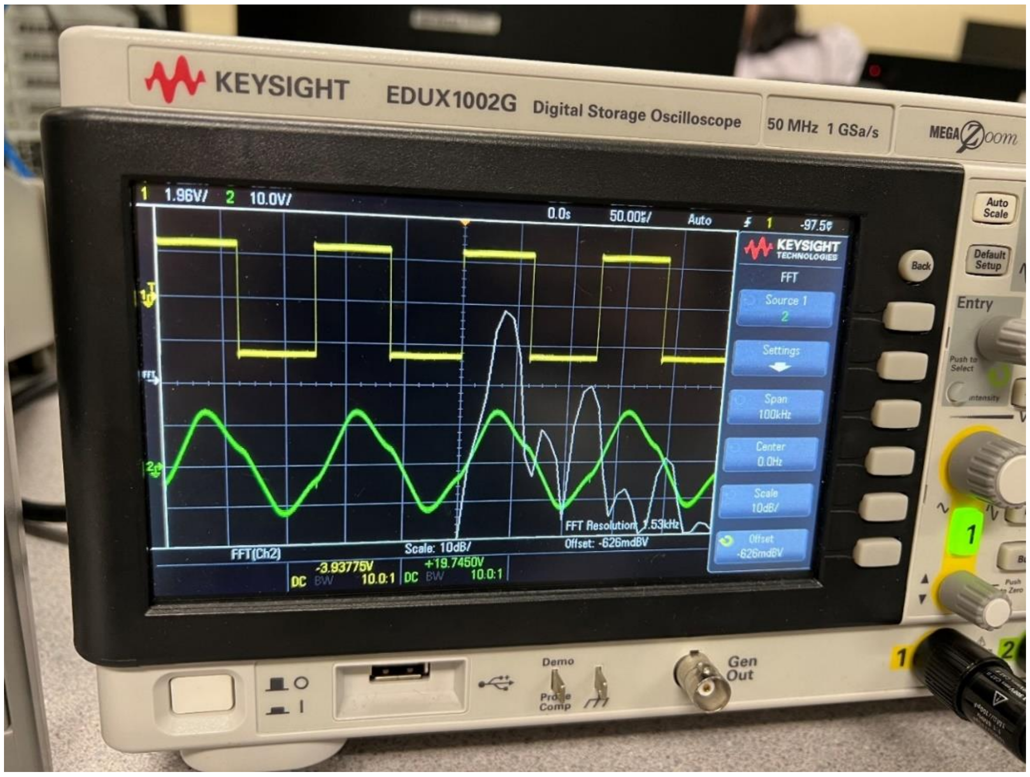


Figure 8. Experimental Signal of Output Signal

3. **Part II: Audio Signal Filtering**
    a. Low Pass Filter:

Audio Output: The low pass filter will remove high-frequency components from the speech. This results in a more muffled sound, as sharp sounds like 's' or 't' are attenuated.

Plot Output: The plot of the filtered signal will show reduced power in higher frequencies. The frequency response plot of the filter will illustrate a curve that allows low frequencies to pass while attenuating those above a certain cutoff frequency.

    b. Band Pass Filter:

Audio Output: This filter allows only a specific range of frequencies to pass. The speech will sound like it's coming through a narrow frequency band, possibly making it sound nasal or tinny, depending on the frequency range allowed.

Plot Output: The power spectrum plot after filtering will show significant power within the allowed frequency band and reduced power outside of it. The filter's frequency response plot will have peaks in the allowed band and dips elsewhere.

    c. High Pass Filter:

Audio Output: The high pass filter will remove lower frequencies, which might make the speech sound thinner or tinier, as it predominantly allows high frequencies to pass through.

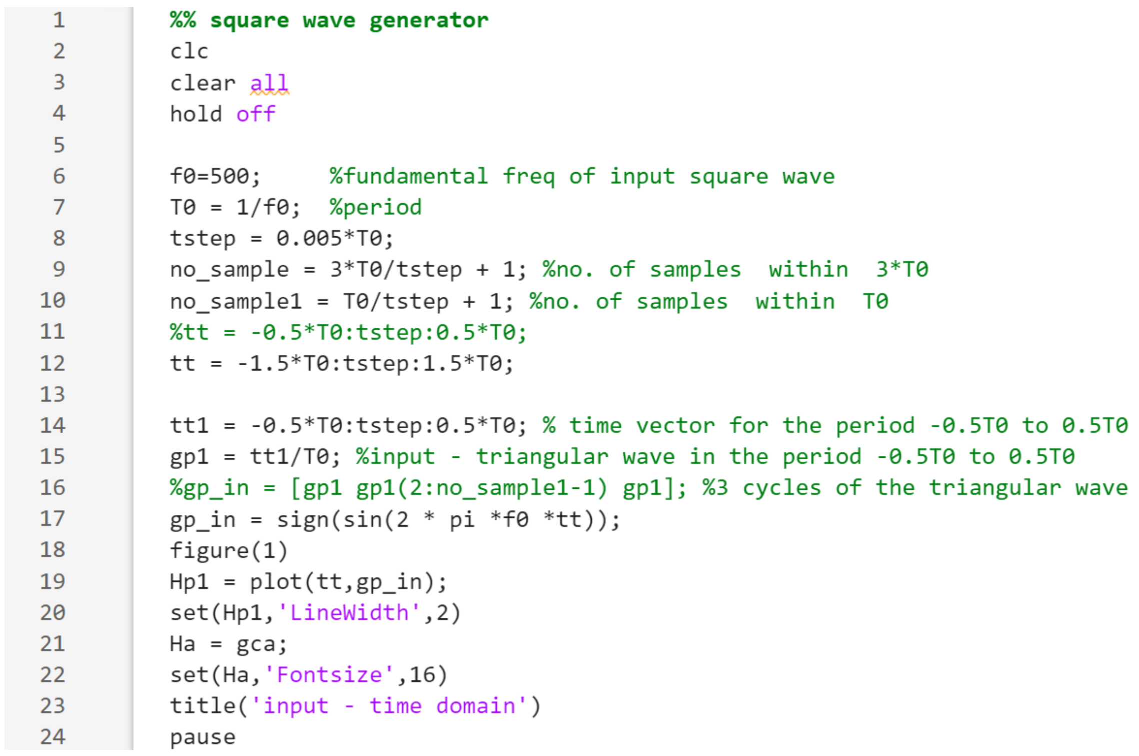
Plot Output: The power spectrum will display reduced power in the lower frequencies and relatively unchanged power in the higher frequencies. The filter's frequency response plot will show a curve that rises at the cutoff frequency, allowing high frequencies to pass.

    d. Notch Filter:

Audio Output: A notch filter attenuates a specific narrow frequency range. Depending on the frequency range being notched out, this could result in the removal of a certain hum or buzz in the speech.

Plot Output: The frequency response of the notch filter will show a dip in the notched frequency range. The power spectrum of the filtered speech will similarly show a dip or reduction in power at the notched frequencies.

4. **Matlab code**

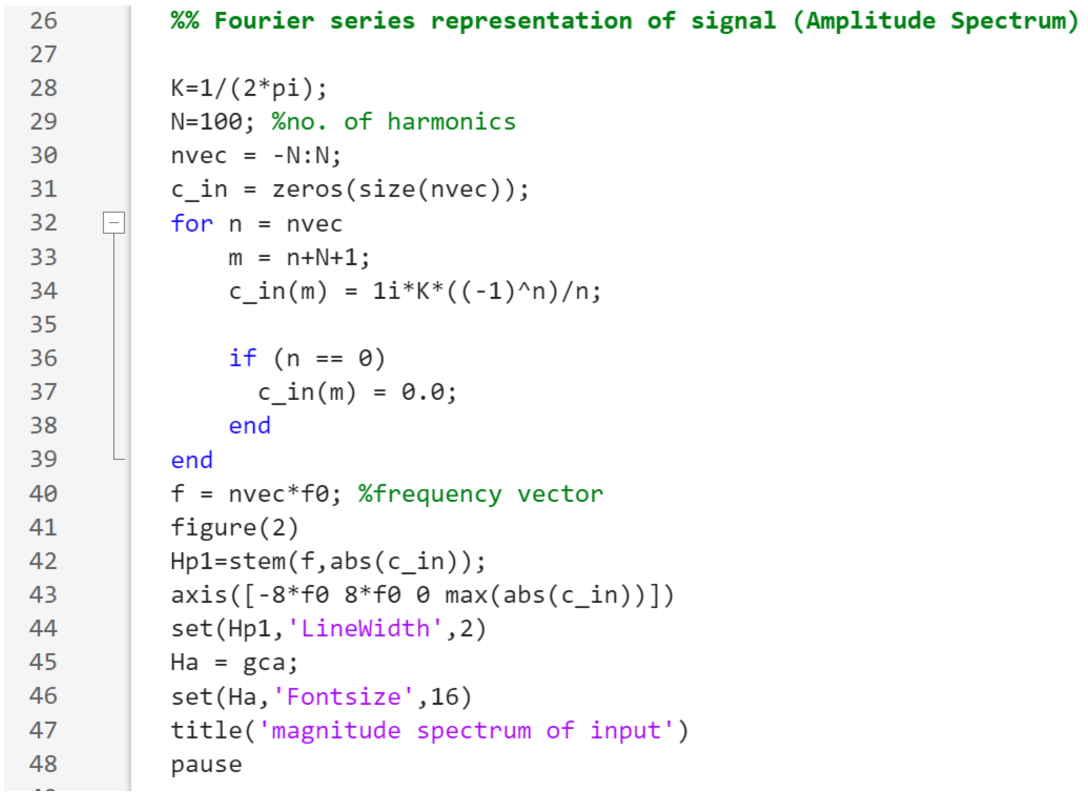


```
%% square wave generator
clc
clear all
hold off

f0=500;     %fundamental freq of input square 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 = -0.5*T0:tstep:0.5*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 = tt1/T0; %input - triangular wave in the period -0.5T0 to 0.5T0
%gp_in = [gp1 gp1(2:no_sample1-1) gp1]; %3 cycles of the triangular wave
gp_in = sign(sin(2 * pi *f0 *tt));
figure(1)
Hp1 = plot(tt,gp_in);
set(Hp1,'LineWidth',2)
Ha = gca;
set(Ha,'Fontsize',16)
title('input - time domain')
pause
```

Figure 9. Matlab Code for Generating Square Wave

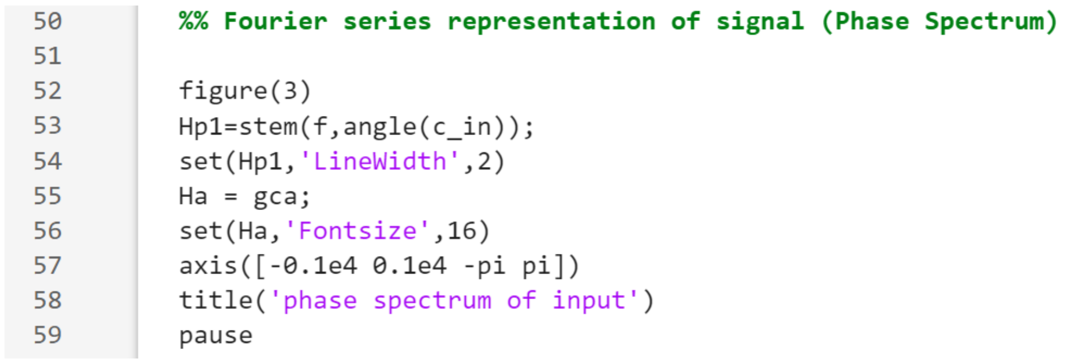


```
%% Fourier series representation of signal (Amplitude Spectrum)

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
```

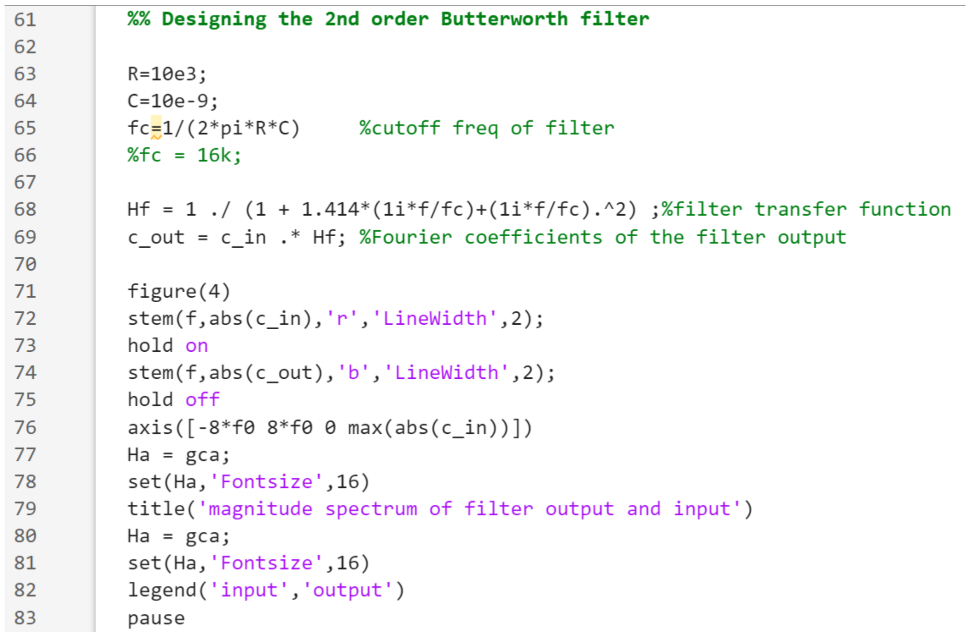
Figure 10. Matlab Code for Amplitude Spectrum



```matlab
%% 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 11. Matlab Code for Phase Spectrum



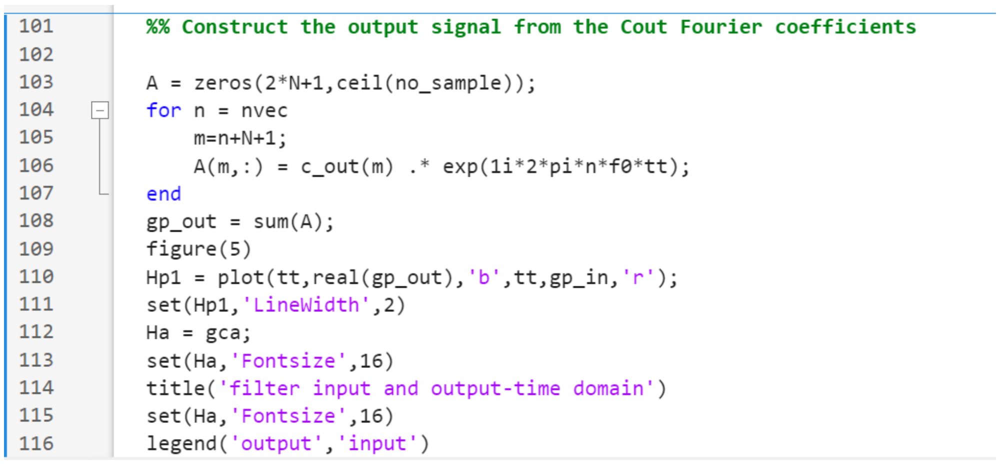
```matlab
%% Designing the 2nd order Butterworth filter

R=10e3;
C=10e-9;
fc=1/(2*pi*R*C)     %cutoff freq of filter
%fc = 16k;

Hf = 1 ./ (1 + 1.414*(1i*f/fc)+(1i*f/fc).^2) ;%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 12. Matlab Code for Second-order Butterworth Filter



```matlab
%% Construct the output signal from the Cout Fourier coefficients

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')
```

Figure 13. Matlab Code for Output Signal