Lab 2 Report: Nonlinear Systems, Filters, and Convolution

Youssef Beltagy and Samuel Hunter
AUT21 BEE 235

January 7, 2022

1 Abstract

The lab introduces the concepts of rectifiers, filters, and convolutions. A rectifier is combined with a low-pass filter to extract the envelope of an audio signal. And a signal is convolved to filter out high frequencies.

2 Part 1 — Nonlinear rectifier and envelope extraction

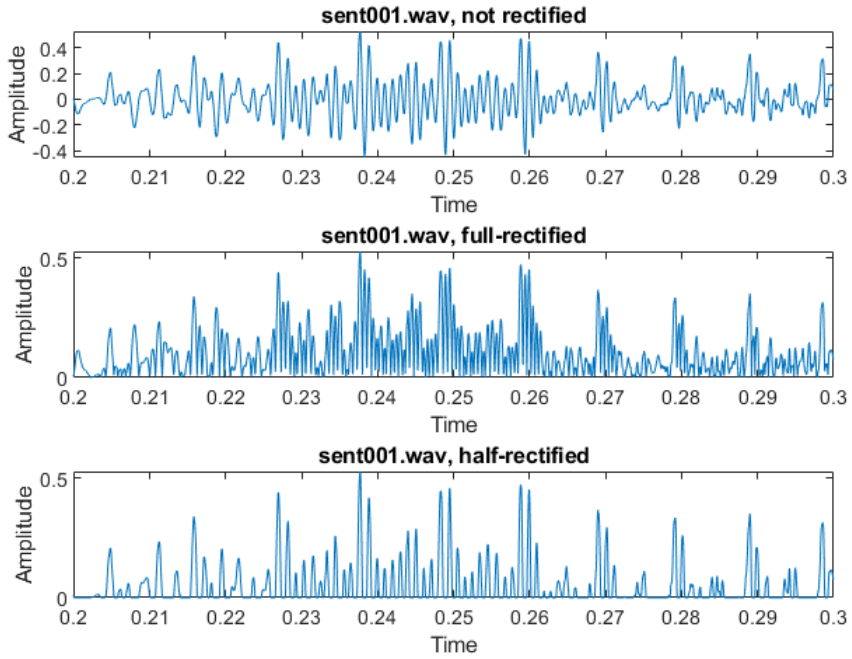
2.1 Exercise 1

The fully rectified signal sounded like a duck. There was a weird ringing sound at the end too.

The half rectified signal sounded a little weird, but not too abnormal.

```
% Samuel Hunter
% BEE 235A, Au 2021, Lab 2
% graph_rectifiers.m - Graph sent001.wav as-is and applied to both a
%                      half-wave and full-wave rectifier.

[signal, Fs] = audioread('sent001.wav');
full_rectified_signal = abs(signal);
half_rectified_signal = max(0, signal);

L = length(signal);
Ts = 1 / Fs;
t = 0:Ts:(L-1)*Ts;

% Play the sounds
sound(signal, Fs);
pause(4);
sound(full_rectified_signal, Fs);
pause(4);
sound(half_rectified_signal, Fs);

% Graph t=0.2 to t=0.3
t = t(0.2*Fs:0.3*Fs);
signal = signal(0.2*Fs:0.3*Fs);
full_rectified_signal = full_rectified_signal(0.2*Fs:0.3*Fs);
half_rectified_signal = half_rectified_signal(0.2*Fs:0.3*Fs);

% Graph raw
subplot(3,1,1);
plot(t, signal);
title('sent001.wav, not rectified');
xlabel('Time');
```

```
ylabel('Amplitude');

% Graph full-rectified-signal
subplot(3,1,2);
plot(t, full_rectified_signal);
title('sent001.wav, full-rectified');
xlabel('Time');
ylabel('Amplitude');

% Graph half-rectified signal
subplot(3,1,3);
plot(t, half_rectified_signal);
title('sent001.wav, half-rectified');
xlabel('Time');
ylabel('Amplitude');
```



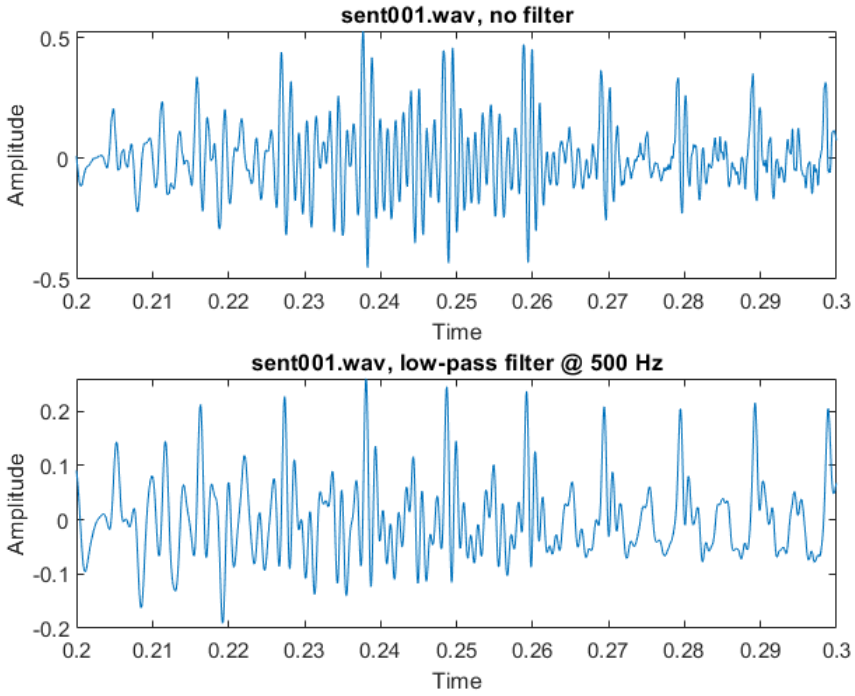
2.2 Exercise 2

After running *sent001.wav* through the 500 Hz low-pass filter, the audio gave a distorted, coming-from-another-room sound.

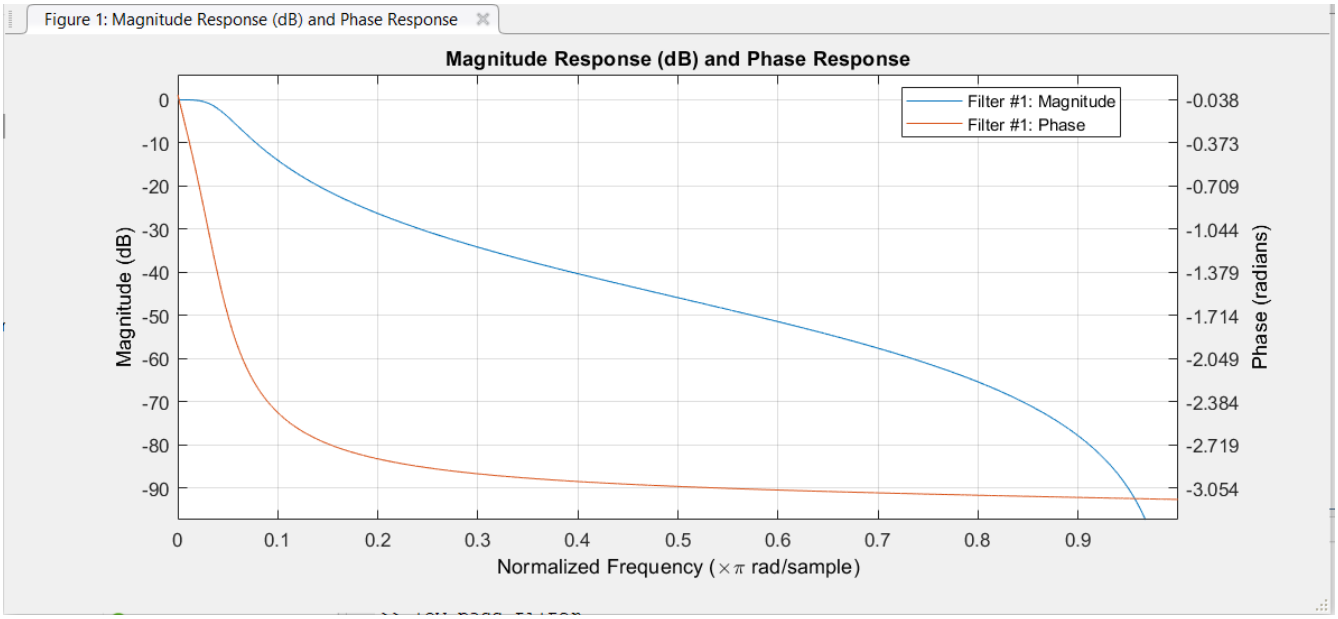
The filtered signal looks smoother than the original signal.

```
% Samuel Hunter
```

```
% BEE235A, Au 2021, Lab 2
% low_pass_filter.m - graph sent001.wav through a LPF

function low_pass_filter()

[signal, Fs] = audioread('sent001.wav');
[bb, aa] = butter(2, 500/(Fs/2)); % 2nd-order LPF @ 500 Hz
filtered_signal = filter(bb, aa, signal);

% Analyze the filter
fvtool(bb,aa);

% play the sounds
sound(signal, Fs);
pause(4);
sound(filtered_signal, Fs);

L = length(signal);
Ts = 1 / Fs;
t = 0:Ts:(L-1)*Ts;

% Slice for graphing for t=0.2 to t=0.3
t = t(0.2*Fs:0.3*Fs);
signal = signal(0.2*Fs:0.3*Fs);
filtered_signal = filtered_signal(0.2*Fs:0.3*Fs);

% Plot sent001.wav raw
subplot(2,1,1);
plot(t, signal);
title('sent001.wav, no filter');
xlabel('Time');
ylabel('Amplitude');

% Plot sent001.wav, filtered
subplot(2,1,2);
plot(t, filtered_signal);
title('sent001.wav, low-pass filter @ 500 Hz');
xlabel('Time');
ylabel('Amplitude');

end
```



Here is the magnitude and phase response of the filter. As you can see, the filter assigns more weight to lower frequencies. But high frequencies are attenuated.

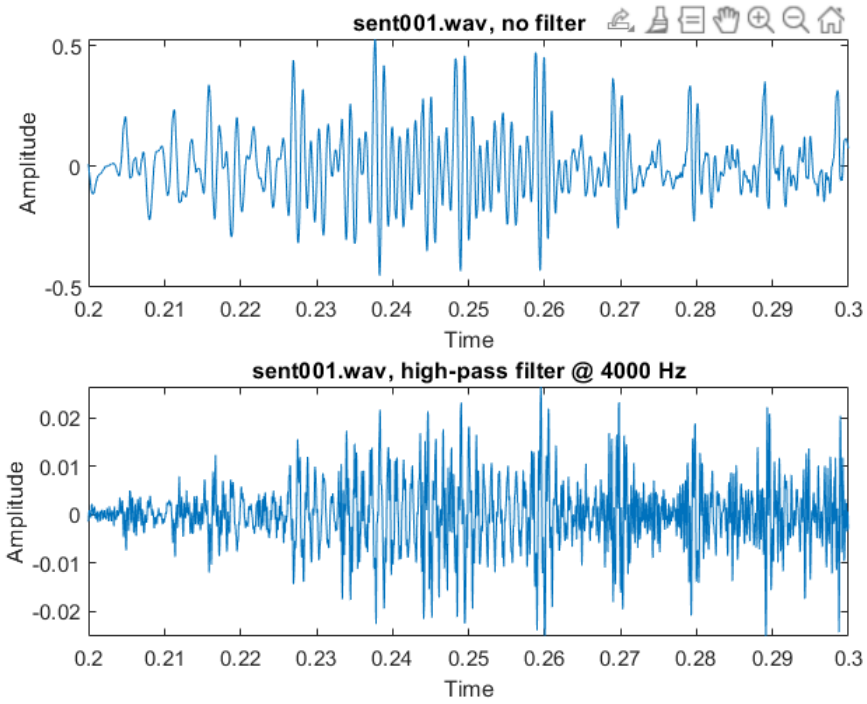


2.3 Exercise 3

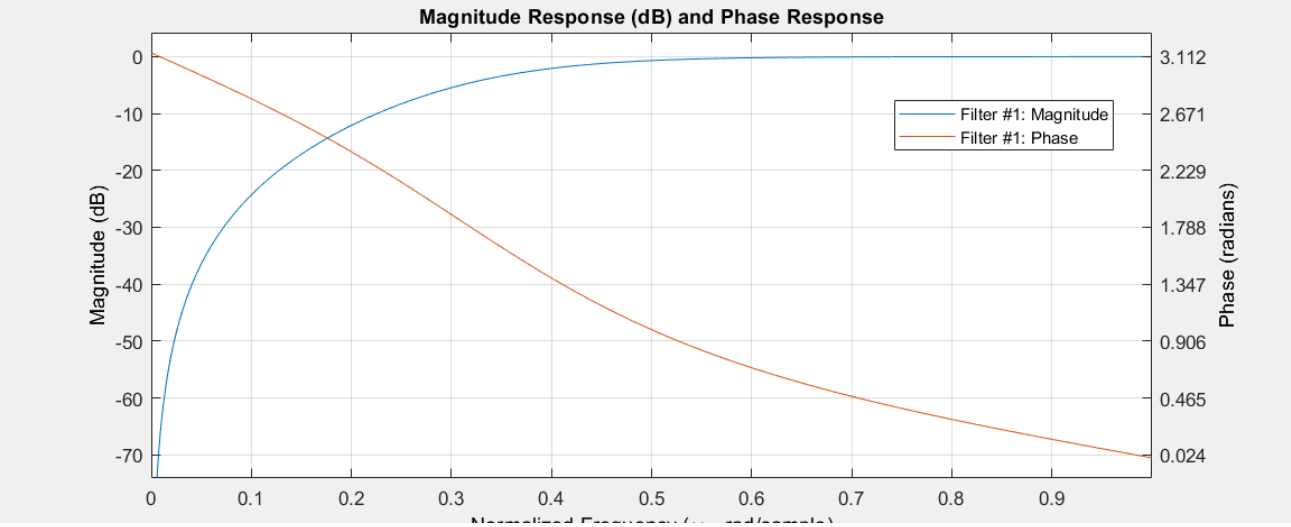
The high-pass filtered signal sounded quiet but very sharp. It was as if someone was whispering while stressing all s.

The filtered signal looks noisier than the original signal.

```
% Samuel Hunter
% BEE235A, Au 2021, Lab 2
% high_pass_filter.m - graph sent001.wav through a HPF
function high_pass_filter()
[signal, Fs] = audioread('sent001.wav');
[bb, aa] = butter(2, 4000/(Fs/2), 'high'); % 2nd-order LPF @ 4 kHz
filtered_signal = filter(bb, aa, signal);

% Analyze the filter
fvtool(bb,aa);

sound(signal, Fs);
pause(4);
sound(filtered_signal, Fs);

L = length(signal);
Ts = 1 / Fs;
t = 0:Ts:(L-1)*Ts;

% Graph t=0.2 to t=0.3
t = t(0.2*Fs:0.3*Fs);
signal = signal(0.2*Fs:0.3*Fs);
filtered_signal = filtered_signal(0.2*Fs:0.3*Fs);

% Plot sent001.wav raw
subplot(2,1,1);
plot(t, signal);
title('sent001.wav, no filter');
xlabel('Time');
ylabel('Amplitude');

% Plot sent001.wav, filtered
subplot(2,1,2);
plot(t, filtered_signal);
title('sent001.wav, high-pass filter @ 4000 Hz');
xlabel('Time');
ylabel('Amplitude');
end
```



Here is the magnitude and phase response of the filter. As you can see, the filter assigns more weight to higher frequencies and attenuates lower frequencies.



2.4 Exercise 4

The lower the frequency cutoff the smoother the envelope and vice versa.

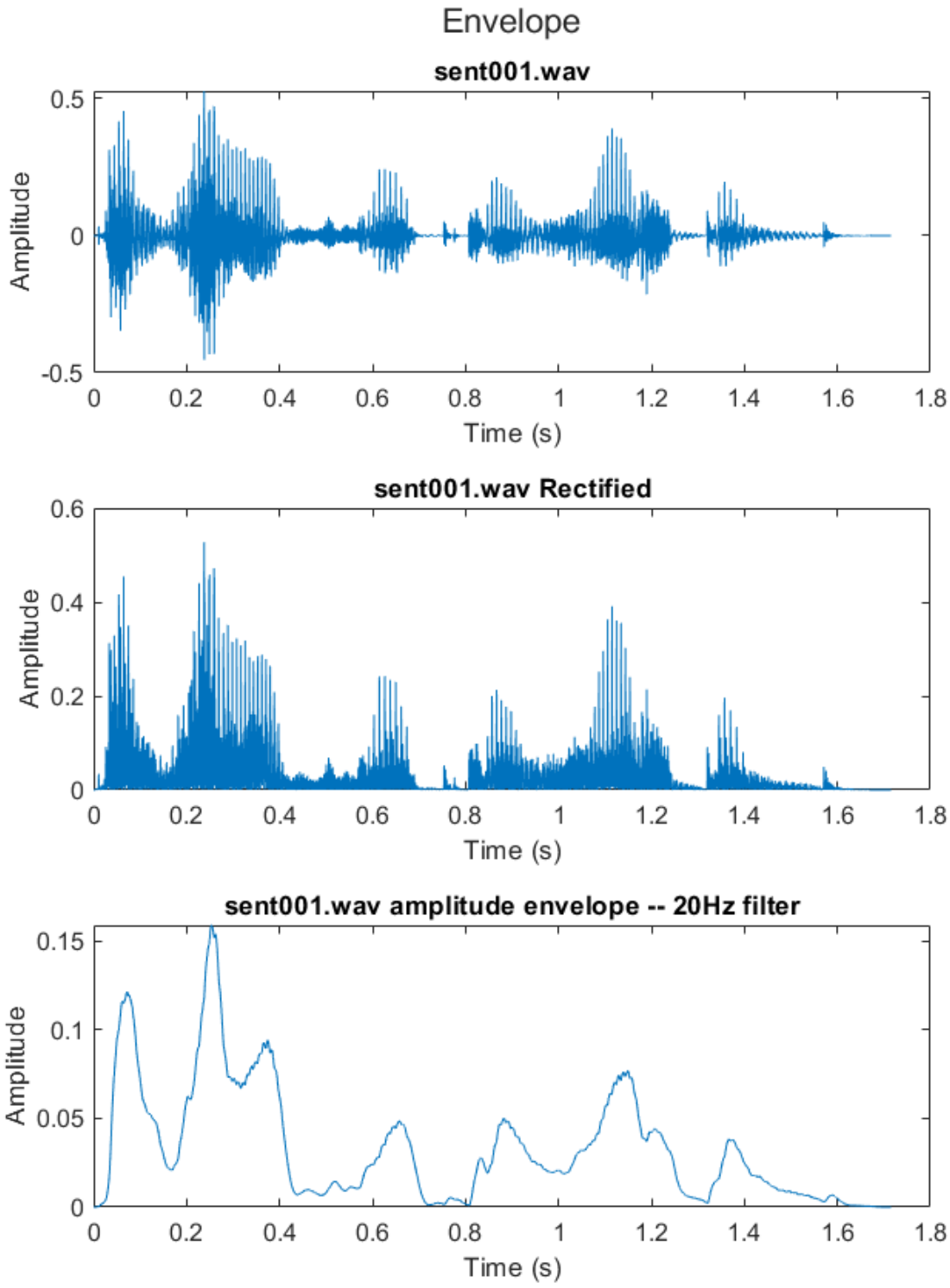
We used full-rectification. This means that though we used 20 Hz for our filter, we might be cutting off frequencies that are bigger than 10 but less than 20. As you know, full-rectification can cause the frequency of some signals to be doubled. This would cause frequencies that are originally in

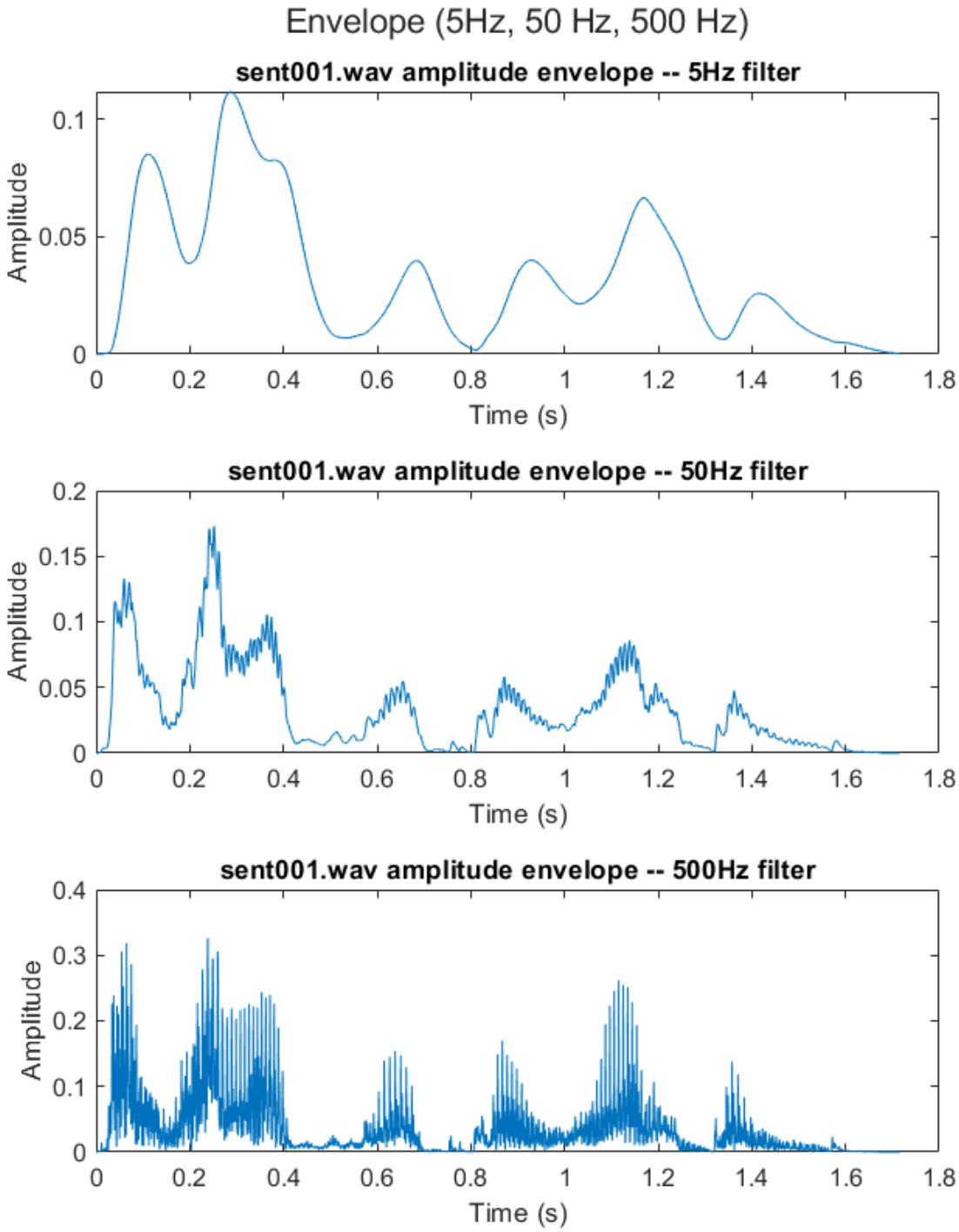
the range of [10,20] to be in the range of [20,40] and be attenuated.

We couldn't hear the 5, 20, and 50 Hz envelopes but the 500 Hz one sounded like a movie alien.

```
% Samuel Hunter
% BEE235A, Au 2021, Lab 2
% rectifier_envelope.m - transform sent001.wav to a peak envelope

function rectifier_envelope()
% load sent001.wav
[signal, Fs] = audioread('sent001.wav');

% full-rectification the speech signal
rectified_signal = abs(signal);

% pass it through a LPF @ 20 Hz
[bb, aa] = butter(2, 20/(Fs/2)); % 2nd-order LPF @ 20 Hz
filtered20_signal = filter(bb, aa, rectified_signal);

% Calculate time
L = length(signal);
Ts = 1 / Fs;
t = 0:Ts:(L-1)*Ts;

layout = tiledlayout(3,1);
title(layout, "Envelope");

% Plot sent001.wav raw
nexttile;
plot(t, signal);
title('sent001.wav');
xlabel('Time (s)');
ylabel('Amplitude');

% Rectified
nexttile;
plot(t, rectified_signal);
title('sent001.wav Rectified');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot sent001.wav envelope @ 20HZ
nexttile;
plot(t, filtered20_signal);
title('sent001.wav amplitude envelope -- 20Hz filter');
```

```
xlabel('Time (s)');
ylabel('Amplitude');


% More plots

% pass it through a LPF @ 5, 50, 500 Hz
[bb, aa] = butter(2, 5/(Fs/2)); % 2nd-order LPF @ 5 Hz
filtered5_signal = filter(bb, aa, rectified_signal);

[bb, aa] = butter(2, 50/(Fs/2)); % 2nd-order LPF @ 50 Hz
filtered50_signal = filter(bb, aa, rectified_signal);

[bb, aa] = butter(2, 500/(Fs/2)); % 2nd-order LPF @ 500 Hz
filtered500_signal = filter(bb, aa, rectified_signal);

figure();
layout = tiledlayout(3,1);
title(layout, "Envelope (5Hz, 50 Hz, 500 Hz)");

% Plot sent001.wav envelope @ 5HZ
nexttile;
plot(t, filtered5_signal);
title('sent001.wav amplitude envelope -- 5Hz filter');
xlabel('Time (s)');
ylabel('Amplitude');


% Plot sent001.wav envelope @ 50HZ
nexttile;
plot(t, filtered50_signal);
title('sent001.wav amplitude envelope -- 50Hz filter');
xlabel('Time (s)');
ylabel('Amplitude');


% Plot sent001.wav envelope @ 500HZ
nexttile;
plot(t, filtered500_signal);
title('sent001.wav amplitude envelope -- 500Hz filter');
xlabel('Time (s)');
ylabel('Amplitude');

end
```


Envelope
sent001.wav
Amplitude
Time (s)
sent001.wav Rectified
Amplitude
Time (s)
sent001.wav amplitude envelope -- 20Hz filter
Amplitude
Time (s)


Envelope (5Hz, 50 Hz, 500 Hz)
sent001.wav amplitude envelope -- 5Hz filter
Amplitude
Time (s)
sent001.wav amplitude envelope -- 50Hz filter
Amplitude
Time (s)
sent001.wav amplitude envelope -- 500Hz filter
Amplitude
Time (s)

3 Part 2 — Convolution

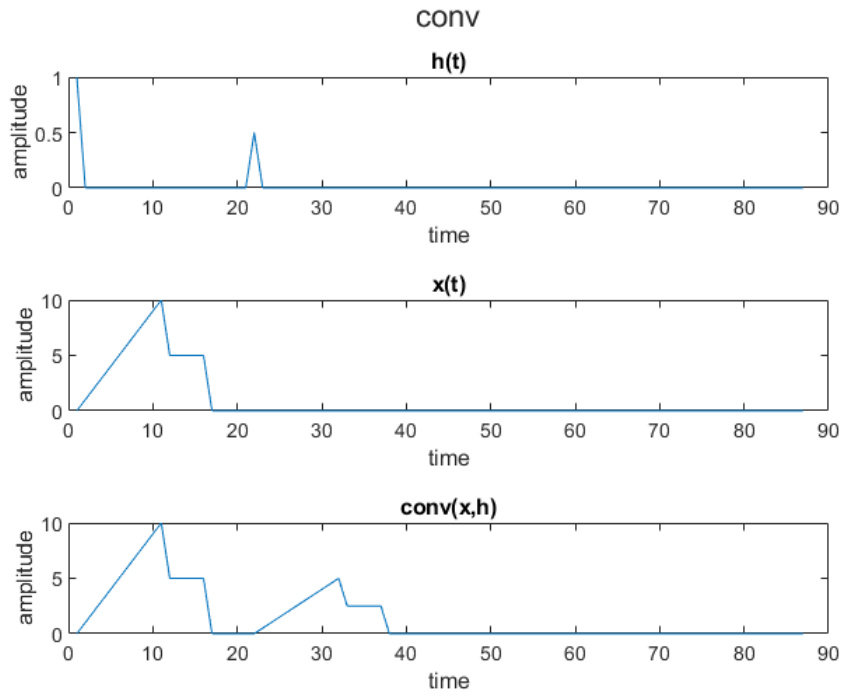
3.1 Convolution Demo

It seems there is a copy of the $x(t)$ in the output for every 1 in $h(t)$. If the coefficient in $x(t)$ is not one, then the output copy will be scaled by the coefficient.

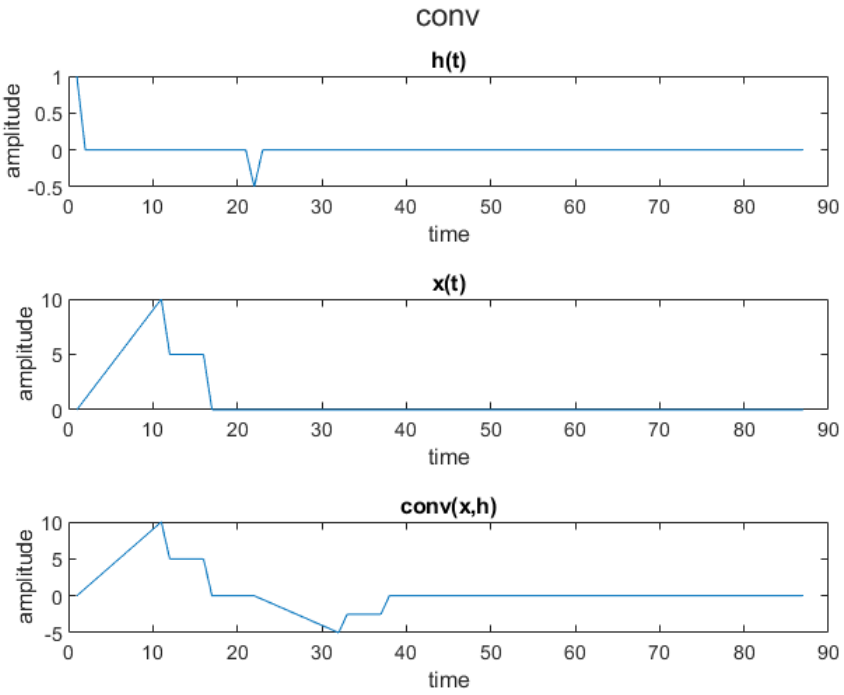
Thus, when the second coefficient is changed from 0.5 to -0.5, the corresponding echo in the output was reflected across the x-axis. Please compare the two images below.

```
% Youssef Beltagy
% BEE235A, Aut 2021, Lab 2
% convolution.m - Experimenting with Matlab Convolution

h = [1 zeros(1,20) .5 zeros(1,10)]; % impulses make triangles
x = [0 1:10 ones(1,5)*5 zeros(1,40)];
y = conv(x,h);

layout = tiledlayout(3,1);
title(layout, "Convolution");

nexttile
plot([h zeros(1, length(y) - length(h))]);
title("h(t)");
ylabel("amplitude");
xlabel ("time");

nexttile
plot([x zeros(1, length(y) - length(x))])
title("x(t)")
ylabel("amplitude");
xlabel ("time");

nexttile
plot(y)
title("conv(x,h)")
ylabel("amplitude");
xlabel ("time");
```

In this image, the second coefficient of $h(t)$ is 0.5.



In this image, the second coefficient of $h(t)$ is -0.5 which caused the second echo of the input to be reflected across the x-axis.

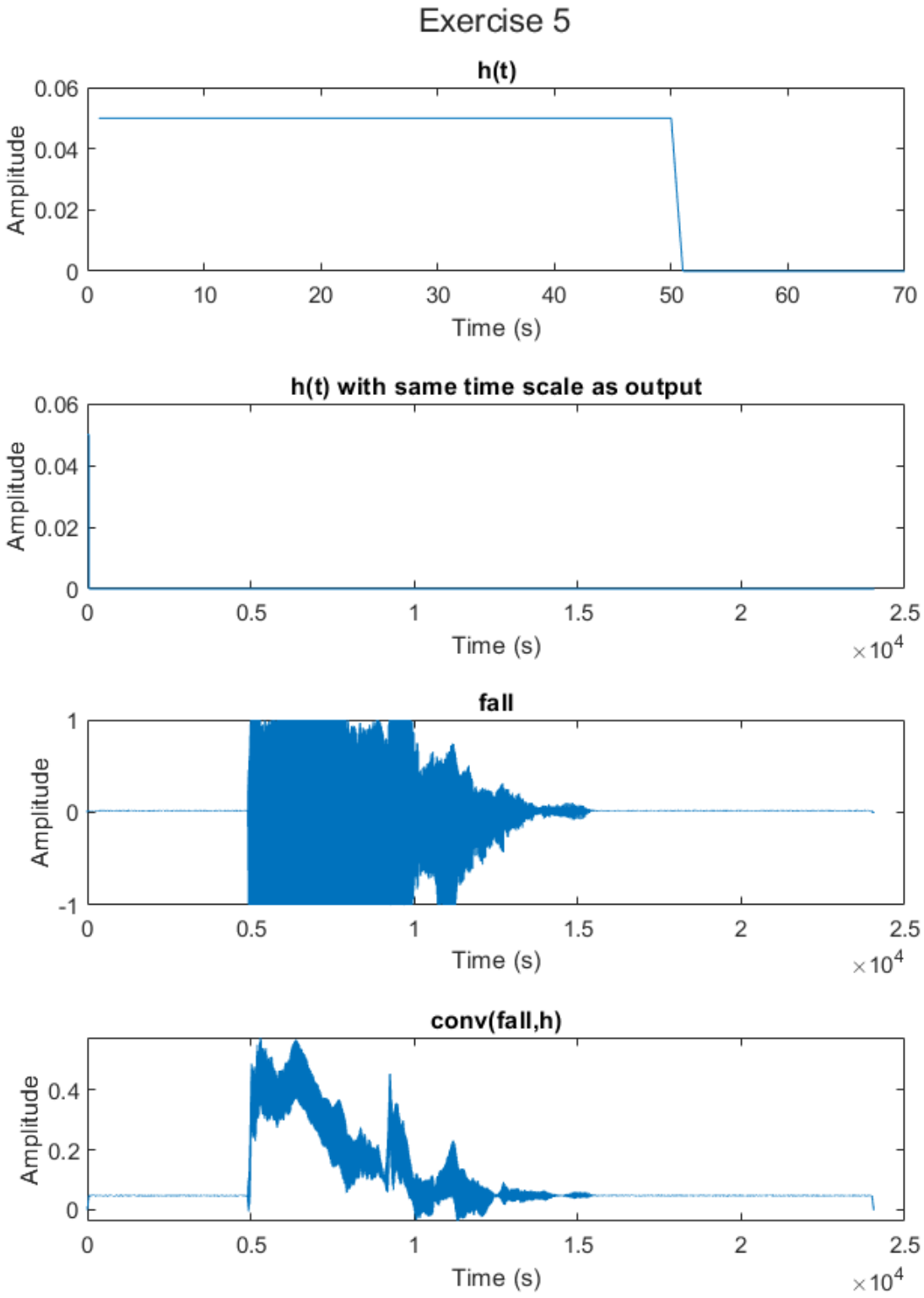


3.2 Exercise 5

The convolved signal sounded muffled and masked: quieter and somewhat dull. Apparently, this impulse response acts as filter so that may be why.

```
% Youssef Beltagy
% BEE235A, Au 2021, Lab 2
% exercise5.m - Convolving fall

load fall

fall = fall'; % convert fall into a row vector
h2 =[ones(1,50)/20 zeros(1,20)];
y2 = conv(fall, h2);

%plotting
layout = tiledlayout(4,1);
title(layout, "Exercise 5");

nexttile
plot(h2);
title("h(t)");
xlabel('Time (s)');
ylabel('Amplitude');

nexttile
plot([h2 zeros(1, length(y2) - length(h2))]);
title("h(t) with same time scale as output");
xlabel('Time (s)');
ylabel('Amplitude');

nexttile
plot([fall zeros(1, length(y2) - length(fall))])
title("fall")
xlabel('Time (s)');
ylabel('Amplitude');

nexttile
plot(y2)
title("conv(fall,h)")
xlabel('Time (s)');
ylabel('Amplitude');

% Play the sounds
sound(fall);
```

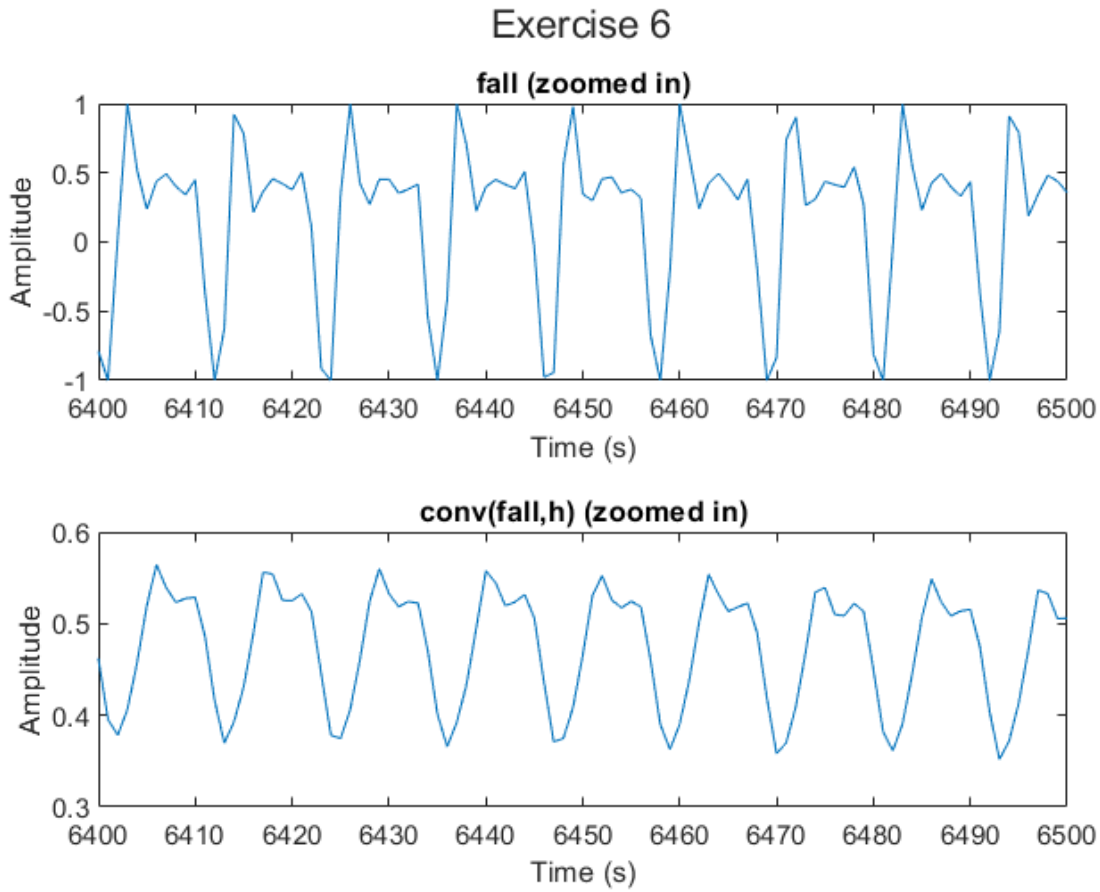
```
pause(4);
sound(y2); %muffled weaker sound
```


Exercise 5
h(t)
Amplitude
Time (s)
h(t) with same time scale as output
Amplitude
Time (s)
×10^4
fall
Amplitude
Time (s)
×10^4
conv(fall,h)
Amplitude
Time (s)
×10^4

3.3 Exercise 6

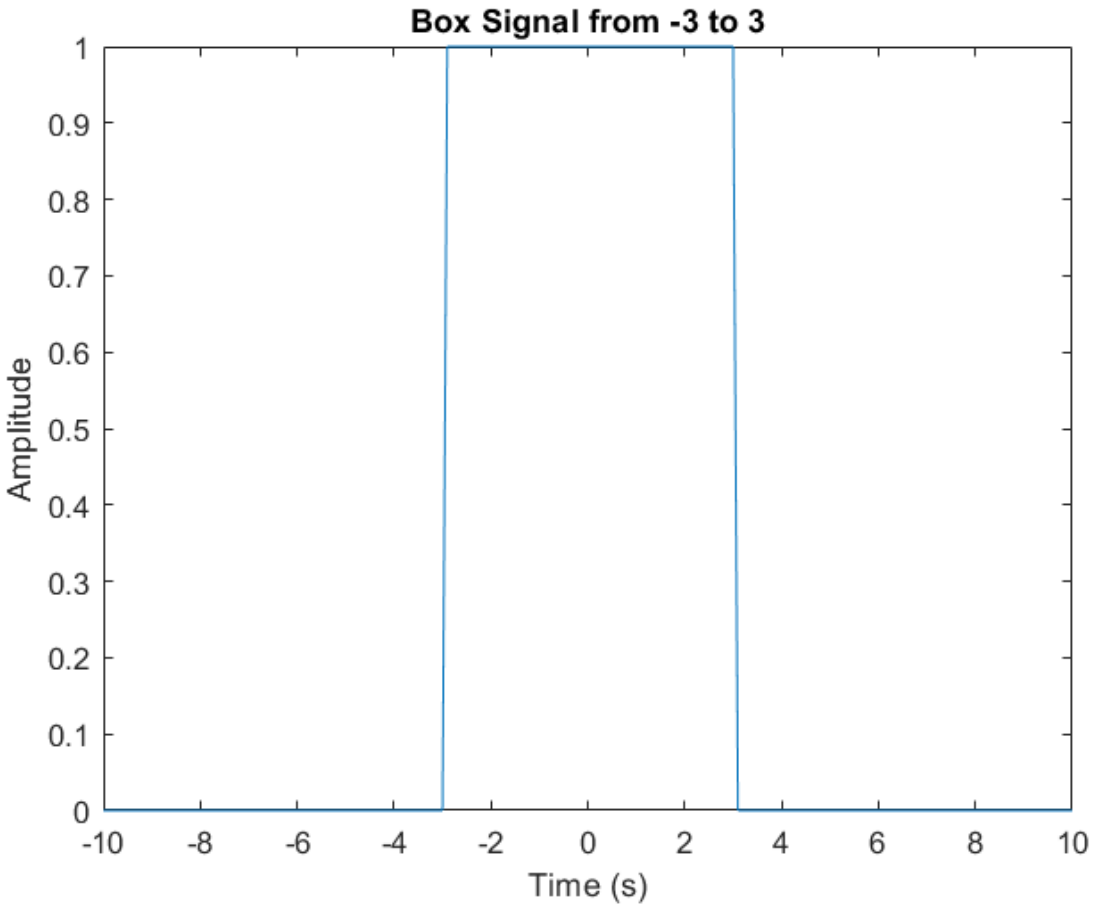
The convolution output really looks smoother (and missing data). It has shallower edges. It never goes below zero and the range of the output is around (0.39,0.55) compared to the input's greater range of (-1,1).

```
% Youssef Beltagy
% BEE235A, Au 2021, Lab 2
% exercise6.m - Zoom into convolved signal

load fall

fall = fall'; % convert fall into a row vector
h2 =[ones(1,50)/20 zeros(1,20)];
y2 = conv(fall, h2);

% Plotting
layout = tiledlayout(2,1);
title(layout, "Exercise 6");

nexttile
plot(6400:6500, fall(6400:6500));
title("fall (zoomed in)");
xlabel('Time (s)');
ylabel('Amplitude');

nexttile
plot(6400:6500, y2(6400:6500));
title("conv(fall,h) (zoomed in)")
xlabel('Time (s)');
ylabel('Amplitude');
```



3.4 Exercise 7

```
% Youssef Beltagy
% BEE235A, Au 2021, Lab 2
% exercise7.m - Generating a Box

t=-10:0.1:10;
boxsig = boxt(t,-3,3);

plot(t,boxsig);
title("Box Signal from -3 to 3");
xlabel('Time (s)');
ylabel('Amplitude');
```



3.5 Exercise 8

Yes, the difference between the three signals is only the slope of the rising and falling portions of the box. It seems the plot function draws points on the graph and then it connects the points with straight lines.

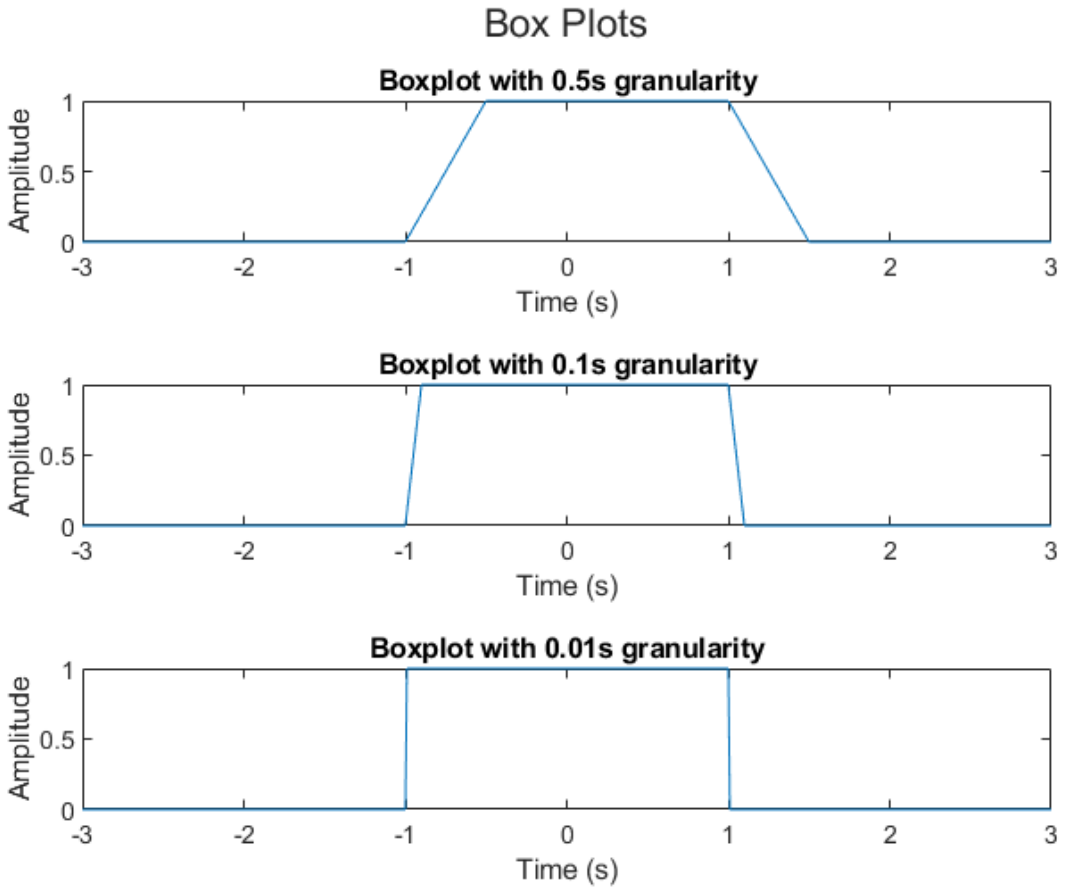
Since the data instantaneously rose, the slope is the rise over the step width: the smaller the step, the more steep the edges. It is hard to visually notice any difference when the step size is less than 0.01, though.

```
% Youssef Beltagy
% BEE235A, Au 2021, Lab 2
% boxtscript.m - boxplots with different time granularities

t05 = -3:0.5:3;
boxsig05 = boxt(t05,-1,1);

t01 = -3:0.1:3;
boxsig01 = boxt(t01,-1,1);
```

```
t001 = -3:0.01:3;
boxsig001 = boxt(t001, -1, 1);

% plotting
layout = tiledlayout(3,1);
title(layout, "Box Plots");

nexttile
plot(t05, boxsig05);
title("Boxplot with 0.5s granularity");
xlabel('Time (s)');
ylabel('Amplitude');

nexttile
plot(t01, boxsig01);
title("Boxplot with 0.1s granularity");
xlabel('Time (s)');
ylabel('Amplitude');

nexttile
plot(t001, boxsig001);
title("Boxplot with 0.01s granularity");
xlabel('Time (s)');
ylabel('Amplitude');
```

3.6 Exercise 9

Here are the box plots from exercise 8.



3.7 Exercise 10

In this exercise, the output had to be multiplied by the step size to get correct output values. This is because the convolve function was designed to work with discrete data so when two numbers are multiplied the width is assumed to be one and so the area is assumed to be value of the multiplication.

In our case, we are representing a continuos signal in a discrete array. So while the indices are of size 1, the step size (width) is only 0.001. So the output must be multiplied by the step size to correct the signal. If this correction was not in place, the output would go up to 2000.

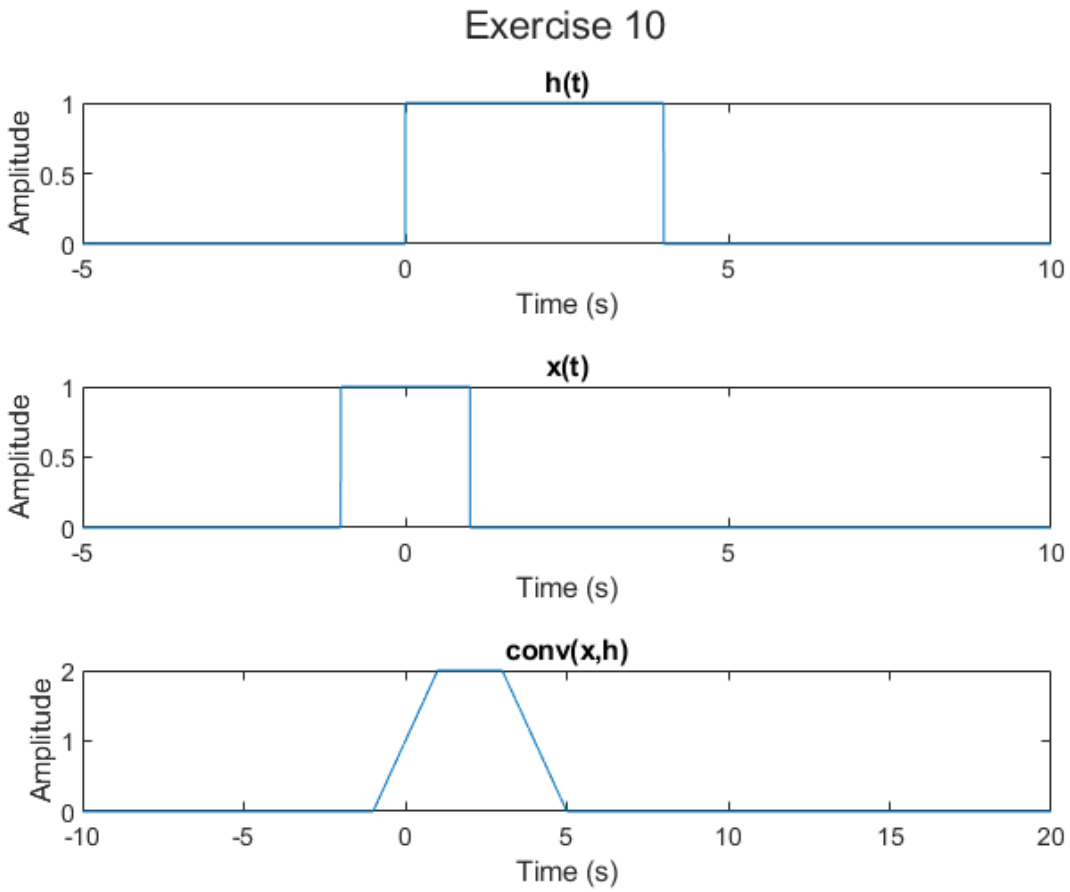
The output takes 2 seconds (-1,1) to rise from 0 to 2. This closely matches the theory. The output then remains at 2 for (1,3) and decreases to 0 again during (3,5). Again, this matches the theory.

Through visual inspection, the function behaves exactly as the theory when the granulation is small enough and the step size is accounted for. If we zoom into the plot, we will find that it doesn't rise

at exactly -1 and some other inconsistencies. However, these inconsistencies can be mitigated by increasing the granulariy (smaller steps).

```
% Youssef Beltagy
% BEE235A, Au 2021, Lab 2
% exercise10.m - Convolving two box plots

% Time Signal Parameters
del = 0.001;
st = -5;
ed = 10;

% Generate time signal
t = st:del:ed;

% Generate two box plots
h = boxt(t, 0, 4); % (0,4)
x = boxt(t, -1, 1); % (-1,1)

% Generate output
y = conv(x,h,'full')*del;
% multiply by del to compensate for the fact a continous signal
% is represented in discrete format


% Plotting
layout = tiledlayout(3,1);
title(layout, "Exercise 10");

nexttile
plot(t, h);
title("h(t)");
xlabel('Time (s)');
ylabel('Amplitude');

nexttile
plot(t,x)
title("x(t)")
xlabel('Time (s)');
ylabel('Amplitude');

nexttile
plot(2*st:del:2*ed,y)
title("conv(x,h)")
xlabel('Time (s)');
```

```
ylabel('Amplitude');
```



4 Conclusion

Exercise 1 demonstrated the use of *abs()* to create a full rectifier, and *max()* to create a half rectifier. Exercises 2 and 3 use *butter()* and the Nyquist frequency to create low-pass and high-pass filters at any given real cutoff frequency. Exercise 4 composes a full rectifier together with a low-pass filter to extract the envelope from an audio signal.

Exercises 5 and 6 showed how to use discrete signal convolutions in matlab and how they can be used as signal filters. Exercises 7, 8, and 9 demonstrated how to make box signals in matlab and how to make the edges of the signals as steep as possible. Exercise 10 demonstrated how to convolve continous signals even though they are discrete signals in matlab.