Case Study 1

Table of Contents

Authors: Lauren Lynch, Xinzhu Lin, and Chinh Mach Class: Signals and Systems Date: 3/17/2023

Part 0: Reset Matlab

```
clear;
close all;
```

Part 1A: Treble Boost Example

```
% Our group has attempted to design our equalizer to be as adaptable and
% user friendly as possible, though we assume that the user is familiar
% with signals and systems content.

% The numbers in the brackets refer to the cutoff frequency of the filter.
% In our example, we have implemented a high pass filter with a cutoff
% frequency of 800 hertz with a gain of 50.

% The volume parameter adds an additional gain to all of the filters
% in case the combination of the filters brings the volume of the speaker
% down too drastically.

% The boolean parameter at the end of the function determines whether the
% user would like the associated plots with their code

lp = {};
gain_lp = {};

hp = {800};
gain_hp = {50};

lp_hp = {};
gain_lp_hp = {};

rlc_elements = {};
gain_rlc = {};
```

```
volume = [1];
plotting = false;
spectogram = [];

equalizer_lsim("m-violin-and-
siren.wav",lp,gain_lp,hp,gain_hp,lp_hp,gain_lp_hp,rlc_elements,gain_rlc,volume,plotting,sp
```

```
successful: Imported Sound
successful: Created Parameters
successful: Implemented Low Pass Filters
successful: 1 Transfer Function Added from HP 1
successful: Implemented High Pass Filters
successful: Implemented Low Pass and High Pass Filters
successful: RLC Filters
successful: Played Sound
```

Part 1B: Bass Boost Example

```
% The same approach would be used to implement a low pass filter in the
% code. In the following example, we have implemented a low pass filter
% with a cutoff frequency of 500 hertz and a gain of 2.

lp = {[ones(10)]};
gain_lp = {[0.1.*ones(length(lp))]};

hp = {};
gain_hp = {};

lp_hp = {};
gain_lp_hp = {};

rlc_elements = {};
gain_rlc = {};

volume = [1];
plotting = false;
spectogram = [2000];

equalizer_lsim("m-violin-and-
siren.wav",lp,gain_lp,hp,gain_hp,lp_hp,gain_lp_hp,rlc_elements,gain_rlc,volume,plotting,sp
```

```
successful: Imported Sound
successful: Created Parameters
successful: 1 Transfer Function Added from LP1
successful: Implemented Low Pass Filters
successful: Implemented High Pass Filters
successful: Implemented Low Pass and High Pass Filters
successful: RLC Filters
successful: Played Sound
```

Part 1C: Unity Boost Example

```
% To amplify a signal frequency, we can implement a low pass and a high
% pass filter with the same cutoff frequency and add a gain.

lp = {};
gain_lp = {};

hp = {};
gain_hp = {};

lp_hp = {[0.1.*ones(20)]};
gain_lp_hp = {(5*10^5)*[ones(length(lp_hp{1}))]};

rlc_elements = {};
gain_rlc = {};

volume = [1];
plotting = false;
spectogram = [3000];

equalizer_lsim("m-violin-and-siren.wav",lp,gain_lp,hp,gain_hp,lp_hp,gain_lp_hp,rlc_elements,gain_rlc,volume,plotting,sp
```

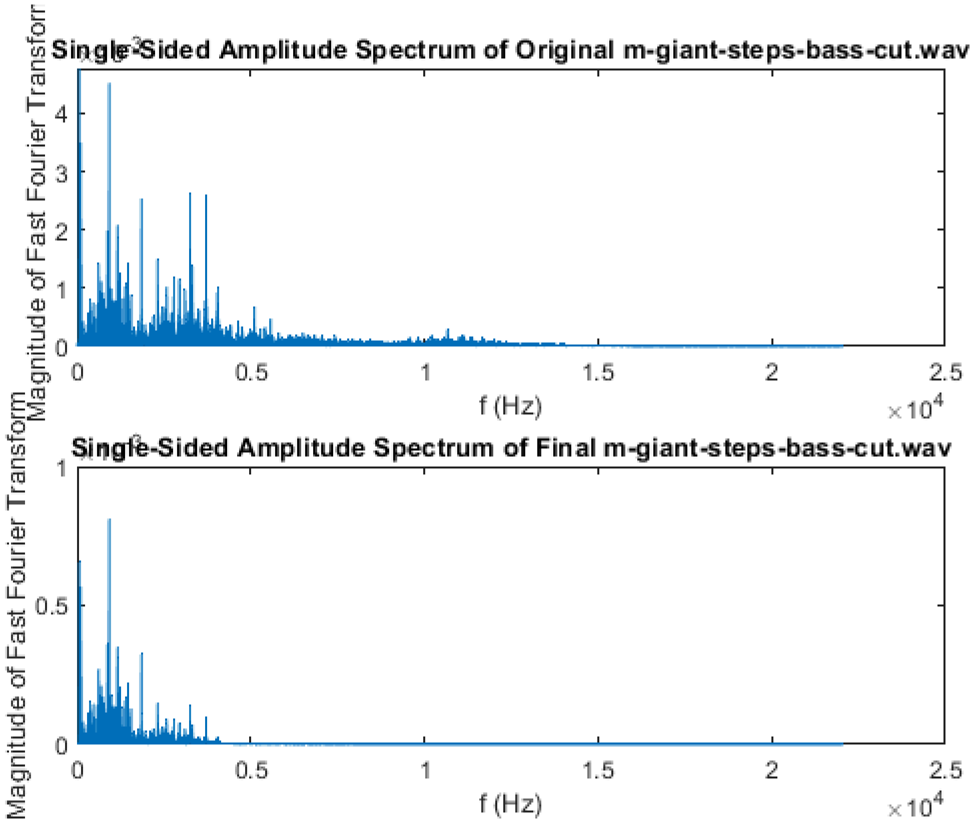
```
successful: Imported Sound
successful: Created Parameters
successful: Implemented Low Pass Filters
successful: Implemented High Pass Filters
successful: 1 Transfer Function Added from LP_HP 1
successful: Implemented Low Pass and High Pass Filters
successful: RLC Filters
successful: Played Sound
```

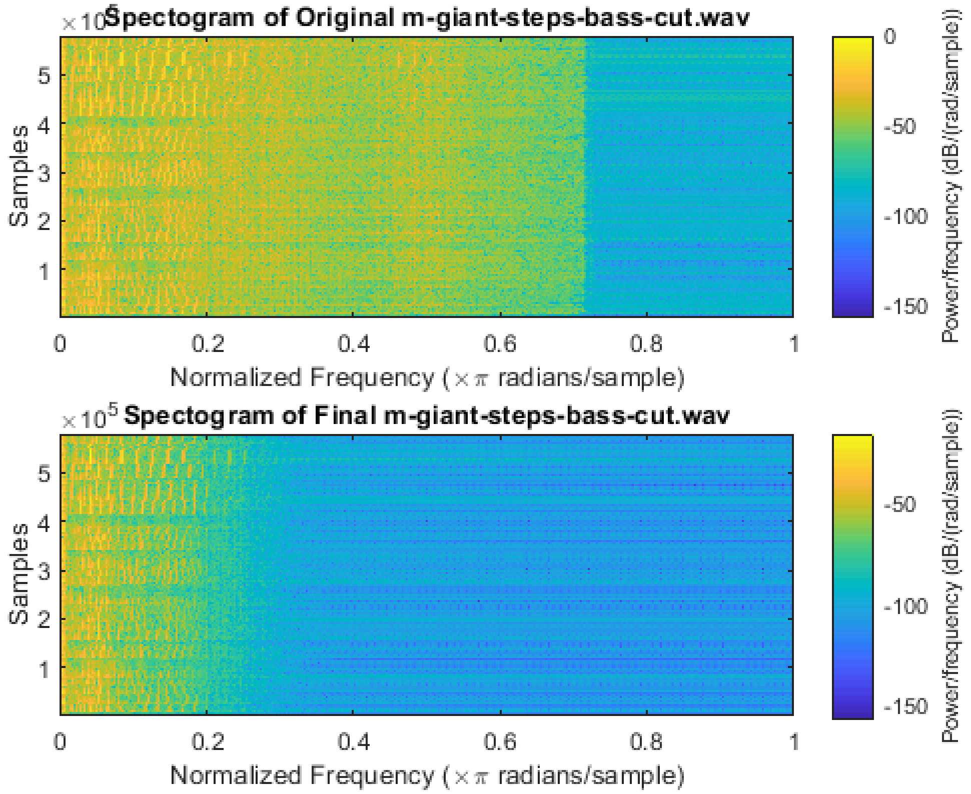
Part 2A: Processing Giant Steps By John Coltrane -- Reducing Extra Noise

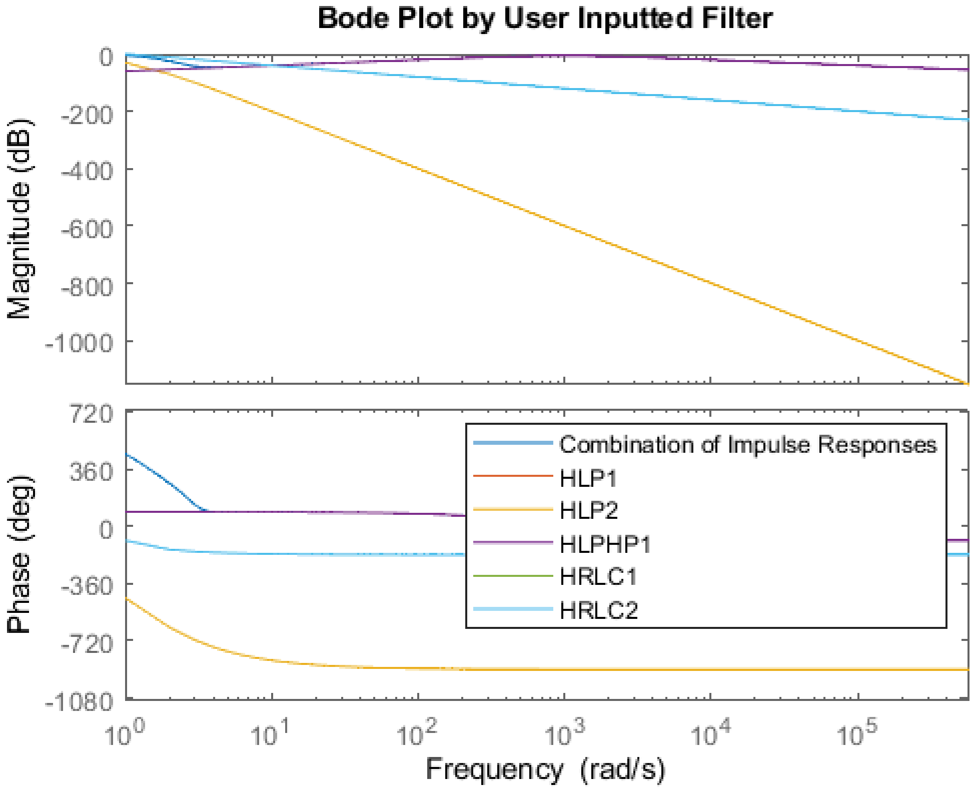
```
lp = {[ones(10)] [2.*ones(10)]};
gain_lp = {[ones(length(lp{1}))] [ones(length(lp{2}))]};

hp = {};
gain_hp = {};

lp_hp = {[1000.*ones(length(2))]};
gain_lp_hp = {[10000.*ones(length(lp_hp{1}))]};

rlc_elements = {[1 1 1] [10 10 10]};
gain_rlc = {[2] [1]};

volume = [1];
plotting = true;
spectogram = [1500];

equalizer_lsim("m-giant-steps-bass-cut.wav",lp,gain_lp,hp,gain_hp,lp_hp,gain_lp_hp,rlc_elements,gain_rlc,volume,plotting,spec
```

```
successful: Imported Sound
```

```
successful: Created Parameters
successful: 1 Transfer Function Added from LP1
successful: 1 Transfer Function Added from LP2
successful: 1 Transfer Function Added from LP3
successful: 1 Transfer Function Added from LP4
successful: Implemented Low Pass Filters
successful: Implemented High Pass Filters
successful: 1 Transfer Function Added from LP_HP 5
successful: 1 Transfer Function Added from LP_HP 6
successful: Implemented Low Pass and High Pass Filters
successful: 1 iteration RLC
successful: 1 Transfer Function Added from RLC 7
successful: 2 iteration RLC
successful: 1 Transfer Function Added from RLC 8
successful: 1 iteration RLC
successful: 1 Transfer Function Added from RLC 9
successful: 2 iteration RLC
successful: 1 Transfer Function Added from RLC 10
successful: RLC Filters
successful: Played Sound
```



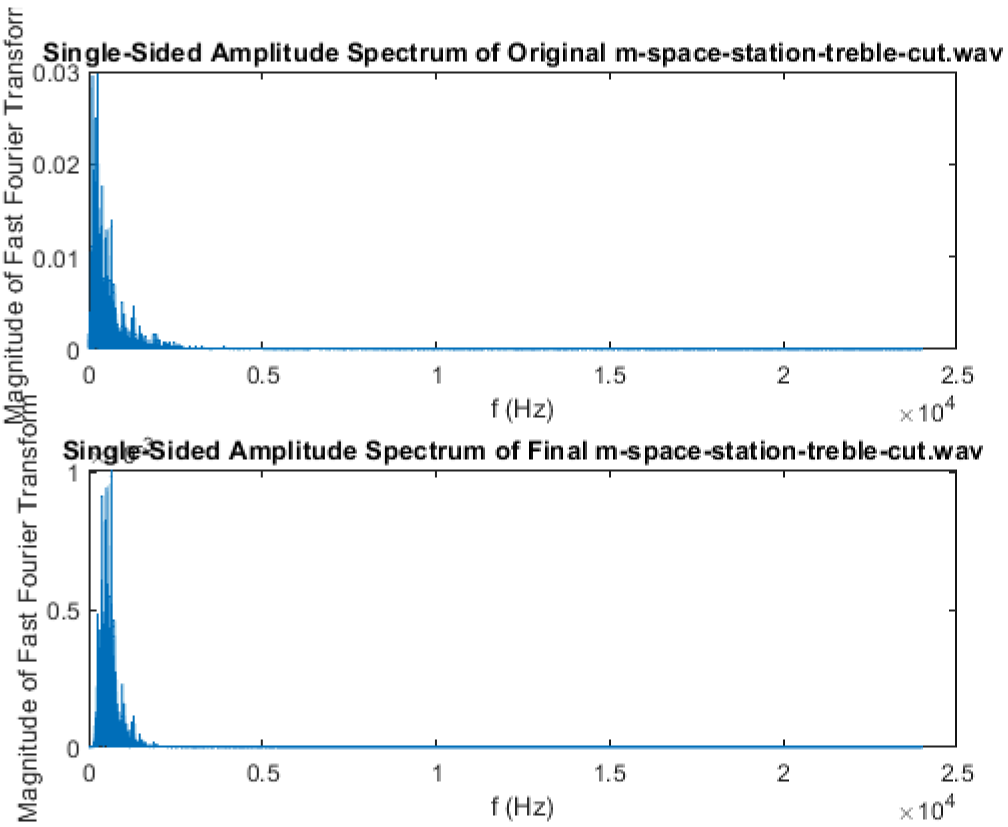

Spectogram of Original m-giant-steps-bass-cut.wav
×10^5
Samples
5
4
3
2
1
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π radians/sample)
0
-50
-100
-150
Power/frequency (dB/(rad/sample))
Spectogram of Final m-giant-steps-bass-cut.wav
×10^5
Samples
Normalized Frequency (×π radians/sample)
-50
-100
-150
Power/frequency (dB/(rad/sample))

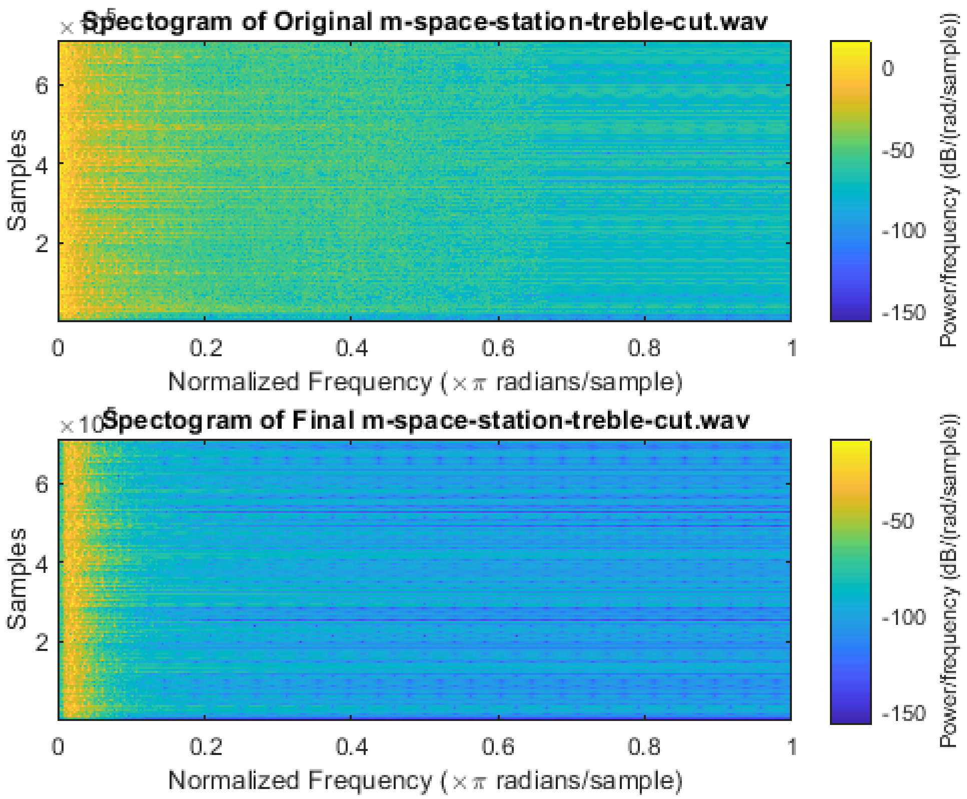

Bode Plot by User Inputted Filter
Magnitude (dB)
0
-200
-400
-600
-800
-1000
Phase (deg)
720
360
0
-360
-720
-1080
Combination of Impulse Responses
HLP1
HLP2
HLPHP1
HRLC1
HRLC2
10^0
10^1
10^2
10^3
10^4
10^5
Frequency (rad/s)

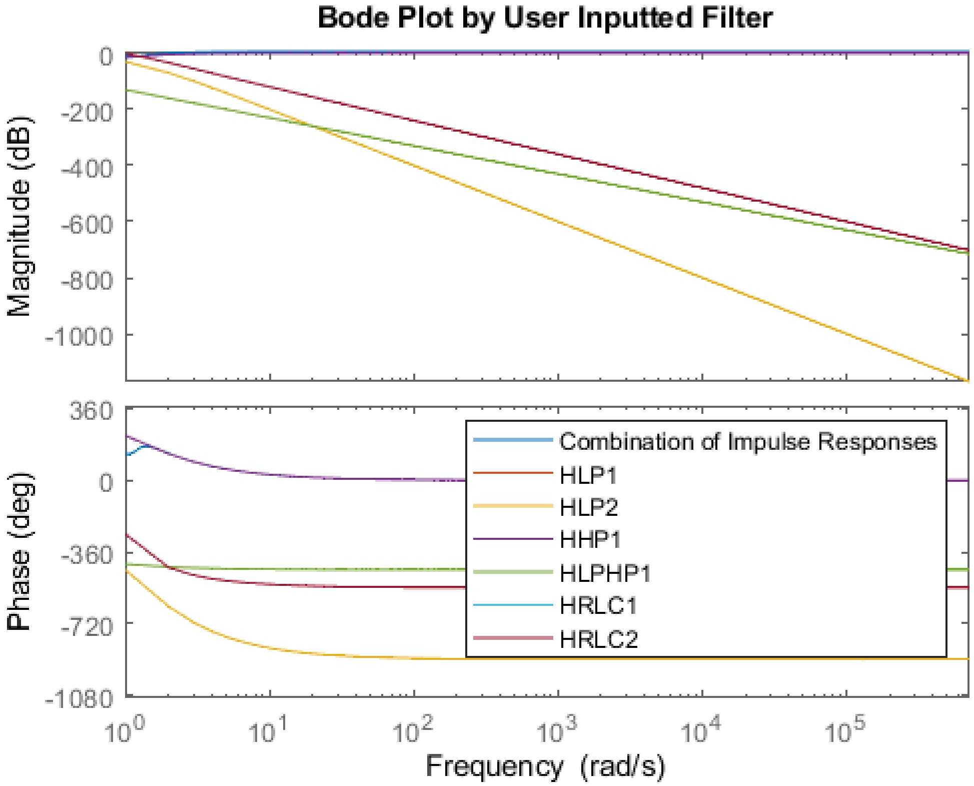
Part 2B: Processing Space Station by Art Farmer -- Emphasize the Piano

```
lp = {[ones(10)] [2.*ones(10)]};
gain_lp = {[ones(length(lp{1}))] [ones(length(lp{2}))]};

hp = {[ones(5)]};
gain_hp = {[ones(length(hp))]};

lp_hp = {[0.05.*ones(5)]};
gain_lp_hp = {[(5*10^3)*ones(length(lp_hp))]};

rlc_elements = {{[1 1 1],[3]} [1 1 1]};
gain_rlc = {[1] [2]};

volume = [30];
plotting = true;
spectogram = [2000];

equalizer_lsim("m-space-station-treble-
cut.wav",lp,gain_lp,hp,gain_hp,lp_hp,gain_lp_hp,rlc_elements,gain_rlc,volume,plotting,spec
```

```
successful: Imported Sound
successful: Created Parameters
successful: 1 Transfer Function Added from LP1
successful: 1 Transfer Function Added from LP2
successful: 1 Transfer Function Added from LP3
successful: 1 Transfer Function Added from LP4
successful: Implemented Low Pass Filters
successful: 1 Transfer Function Added from HP5
successful: 1 Transfer Function Added from HP6
successful: Implemented High Pass Filters
successful: 1 Transfer Function Added from LP_HP 7
successful: 1 Transfer Function Added from LP_HP 8
successful: Implemented Low Pass and High Pass Filters
successful: 1 iteration RLC
successful: 1 Transfer Function Added from RLC 9
successful: 2 iteration RLC
successful: 1 Transfer Function Added from RLC 10
successful: 1 iteration RLC
successful: 1 Transfer Function Added from RLC 11
successful: 2 iteration RLC
successful: 1 Transfer Function Added from RLC 12
successful: RLC Filters
successful: Played Sound
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
```


Single-Sided Amplitude Spectrum of Original m-space-station-treble-cut.wav
Magnitude of Fast Fourier Transforr
0.03
0.02
0.01
0
0
0.5
1
1.5
2
2.5
f (Hz)
×10^4
Single-Sided Amplitude Spectrum of Final m-space-station-treble-cut.wav
×10^-3
Magnitude of Fast Fourier Transform
1
0.5
0
0
0.5
1
1.5
2
2.5
f (Hz)
×10^4


Spectogram of Original m-space-station-treble-cut.wav
×10^5
Samples
6
4
2
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π radians/sample)
0
-50
-100
-150
Power/frequency (dB/(rad/sample))
Spectogram of Final m-space-station-treble-cut.wav
×10^5
Samples
6
4
2
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π radians/sample)
-50
-100
-150
Power/frequency (dB/(rad/sample))

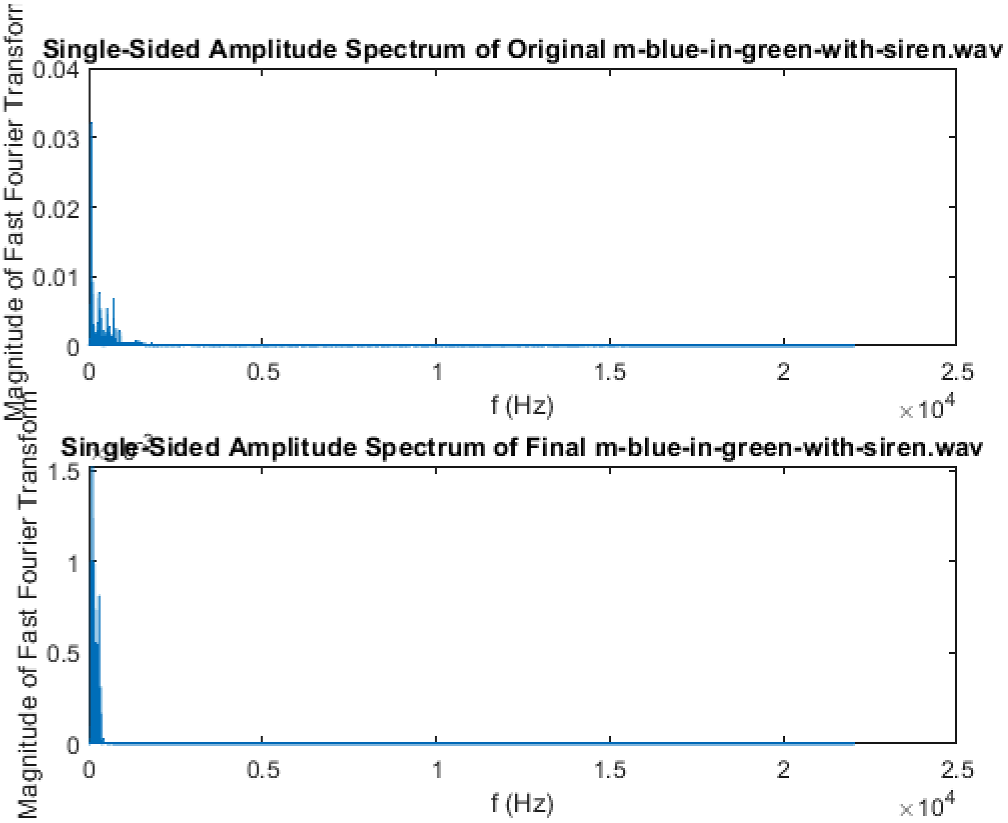


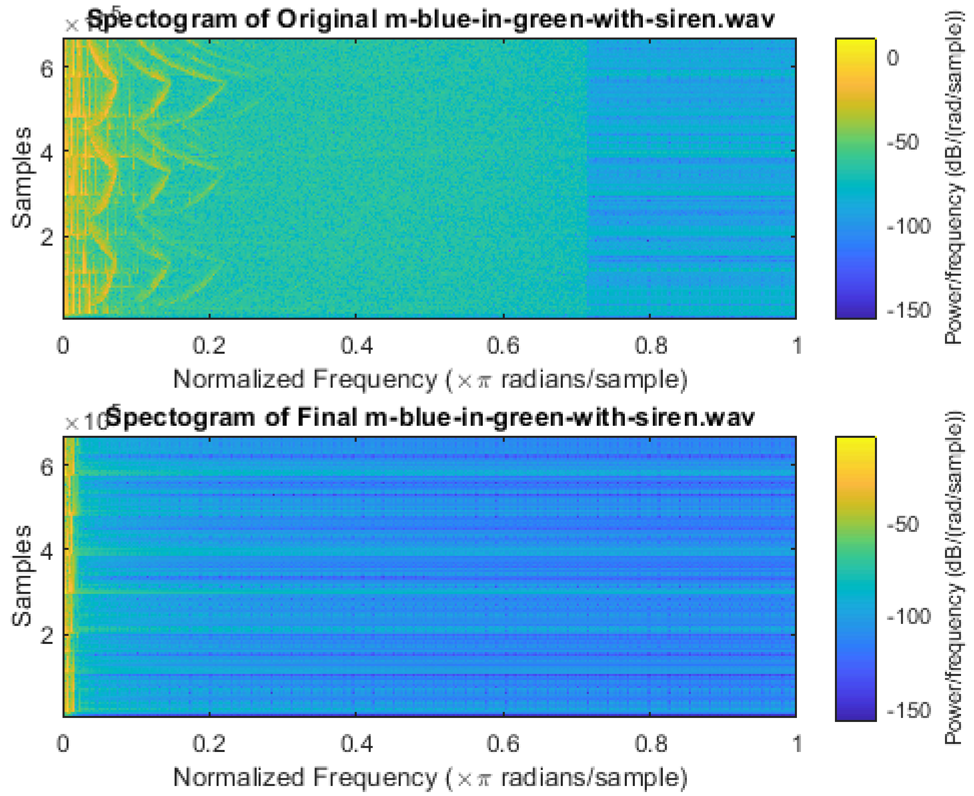
Part 3A: Processing Blue in Green by Miles Davis

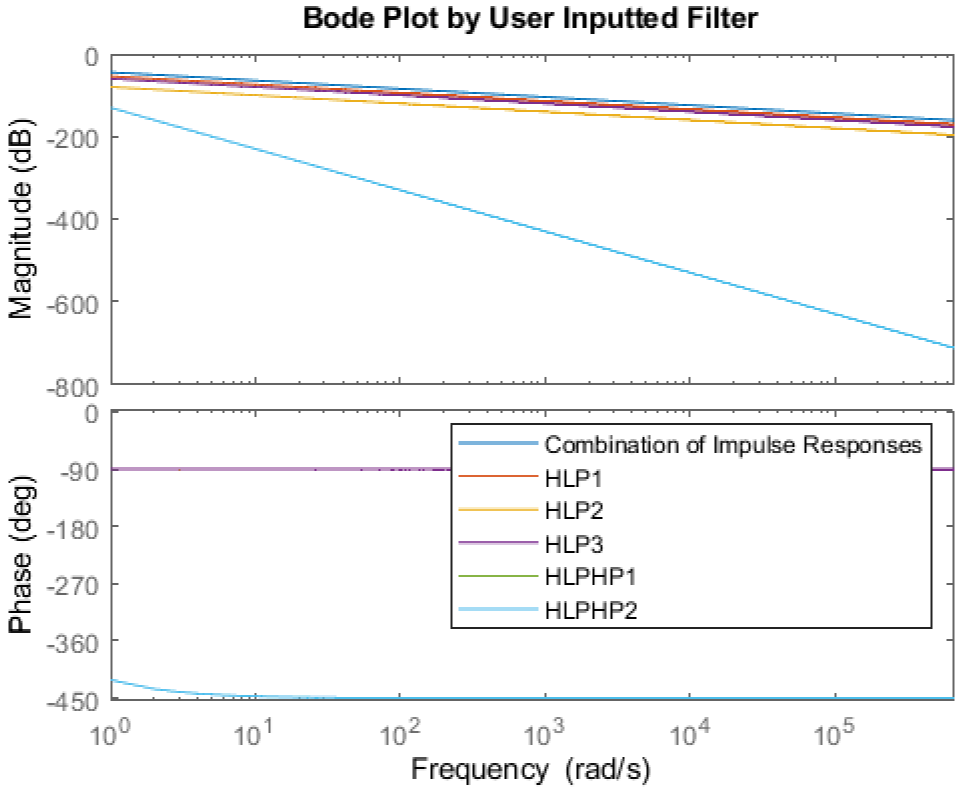
```
lp = {[0.002.*ones(1)] [0.001.*ones(1)] [0.0001.*ones(1)]};
gain_lp = {[50.*ones(length(lp{1}))] [10.*ones(length(lp{2}))]
 [200.*ones(length(lp{3}))]};

hp = {};
gain_hp = {};

lp_hp = {[0.05.*ones(5)] [0.025.*ones(5)]};
gain_lp_hp = {[30*ones(length(lp_hp))] [30.*ones(5)]};

rlc_elements = {};
gain_rlc = {};

volume = [1];
plotting = true;
spectogram = [3000];

equalizer_lsim("m-blue-in-green-with-
siren.wav",lp,gain_lp,hp,gain_hp,lp_hp,gain_lp_hp,rlc_elements,gain_rlc,volume,plotting,sp
```

```
successful: Imported Sound
successful: Created Parameters
```

```
successful: 1 Transfer Function Added from LP1
successful: 1 Transfer Function Added from LP2
successful: 1 Transfer Function Added from LP3
successful: 1 Transfer Function Added from LP4
successful: 1 Transfer Function Added from LP5
successful: 1 Transfer Function Added from LP6
successful: Implemented Low Pass Filters
successful: Implemented High Pass Filters
successful: 1 Transfer Function Added from LP_HP 7
successful: 1 Transfer Function Added from LP_HP 8
successful: 1 Transfer Function Added from LP_HP 9
successful: 1 Transfer Function Added from LP_HP 10
successful: Implemented Low Pass and High Pass Filters
successful: RLC Filters
successful: Played Sound
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
```




Spectogram of Original m-blue-in-green-with-siren.wav
× 10^5
Samples
6
4
2
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π radians/sample)
0
-50
-100
-150
Power/frequency (dB/(rad/sample))
Spectogram of Final m-blue-in-green-with-siren.wav
× 10^5
Samples
6
4
2
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (×π radians/sample)
-50
-100
-150
Power/frequency (dB/(rad/sample))


Bode Plot by User Inputted Filter
Magnitude (dB)
0
-200
-400
-600
-800
Phase (deg)
0
-90
-180
-270
-360
-450
Combination of Impulse Responses
HLP1
HLP2
HLP3
HLPHP1
HLPHP2
10^0
10^1
10^2
10^3
10^4
10^5
Frequency (rad/s)

Part 4A: Creative Portion -- Mr Postman -- Getting Rid of Claps and Amplifying the Alto

```
% Youtube Link Here: https://youtu.be/fXgw948RwYs

% Original Strategy
% lp = {[(1/829).*ones(1)] [(1/1140).*ones(1)]};
% gain_lp = {[ones(length(lp{1}))] [ones(length(lp{2}))]};
%
% hp = {[(1/1657).*ones(1)] [(1/2390).*ones(1)]};
% gain_hp = {[ones(length(hp{1}))] [ones(length(hp{2}))]};
%
% lp_hp = {[(1/1465).*ones(1)]};
% gain_lp_hp = {[20.*ones(length(lp_hp{1}))]};

lp = {1500 500};
gain_lp = {10 10};

hp = {};
gain_hp = {};

lp_hp = {[0.1.*ones(2)] [0.15.*ones(2)] [0.20.*ones(2)]};
gain_lp_hp = {[50*ones(2)] [ones(2)] [50*ones(2)]};

rlc_elements = {{[1 5 1],[2]}};
gain_rlc = {[0.001]};

volume = [1];
plotting = true;
spectogram = [4000];

equalizer_lsim("m-mr-
postman.wav",lp,gain_lp,hp,gain_hp,lp_hp,gain_lp_hp,rlc_elements,gain_rlc,volume,plotting,
```

```
successful: Imported Sound
successful: Created Parameters
successful: 1 Transfer Function Added from LP1
successful: 1 Transfer Function Added from LP2
successful: 1 Transfer Function Added from LP3
successful: 1 Transfer Function Added from LP4
successful: Implemented Low Pass Filters
successful: Implemented High Pass Filters
successful: 1 Transfer Function Added from LP_HP 5
successful: 1 Transfer Function Added from LP_HP 6
successful: 1 Transfer Function Added from LP_HP 7
successful: 1 Transfer Function Added from LP_HP 8
successful: 1 Transfer Function Added from LP_HP 9
successful: 1 Transfer Function Added from LP_HP 10
successful: Implemented Low Pass and High Pass Filters
successful: 1 iteration RLC
successful: 1 Transfer Function Added from RLC 11
successful: 1 iteration RLC
```

```
successful: 1 Transfer Function Added from RLC 12
successful: RLC Filters
successful: Played Sound
```

Published with MATLAB® R2022b