Table of Contents

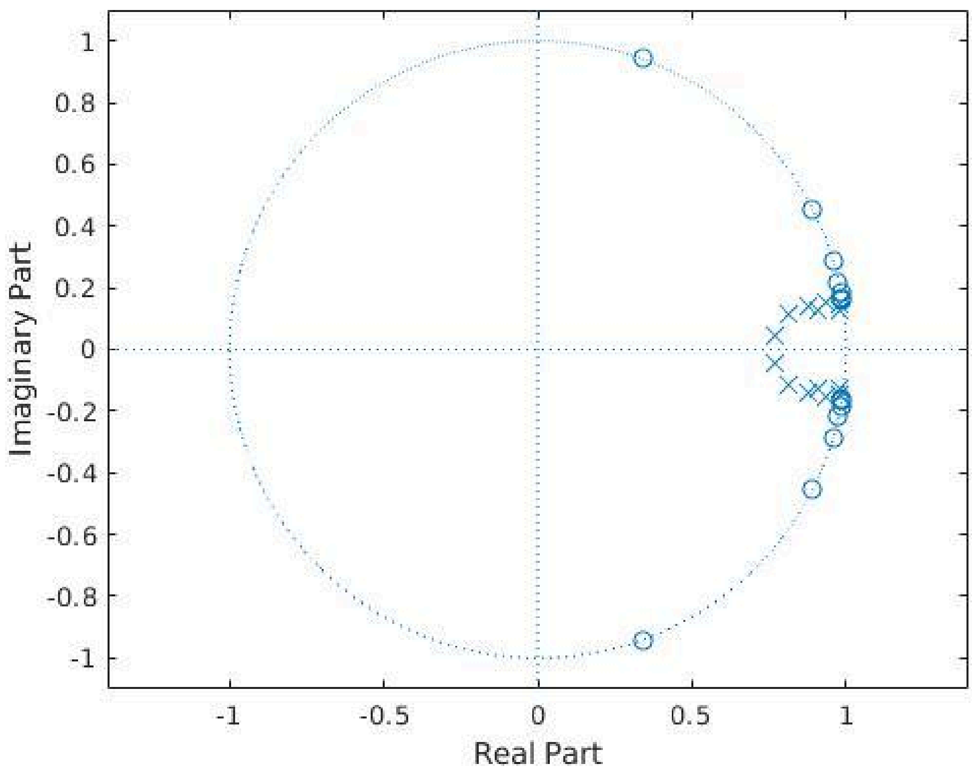
Exercise 8.1

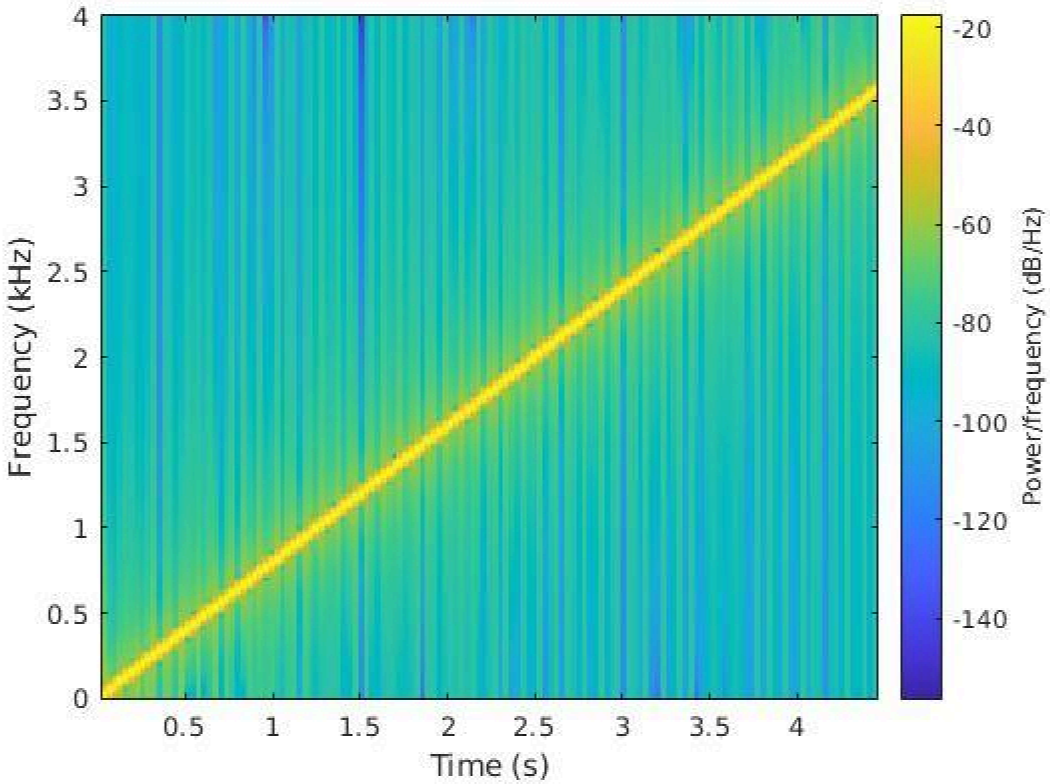
```
%a
fs = 8000;
tt = 0:1/fs:4.5;
x = cos(2*pi/20*fs*tt.^2);
figure;
spectrogram(x,512,256,512,fs,'yaxis');
%soundsc(x);

% The chirp sound is increasing in frequency, this is seen in the
% spectrogram as yellow line has a positive slope.

%b
% The resulting IIR filter has order 14
% transfer function
transfer_function = tf(b,a)
%pole zero plot
figure;
zplane(b,a);
%magnitude response
fvtool(b,a);

%c
figure;
xfiltered = filter(b,a,x);
spectrogram(xfiltered,512,256,512,fs,'yaxis');
%soundsc(xfiltered);

%The signal after the filter only plays the low frequency part of the
%signal loudly. This can be corelatted to the magnitude response of
 the
%signal as it can be seen that the magnitude response drops
 dramatically
%after the lower frequency.

%d
% the fir filter coefficients are stored in "c" to preserve the
 earlier a
% and b.

%The order of the fir filter is 474

%pole zero plot of fir filter
figure;
```

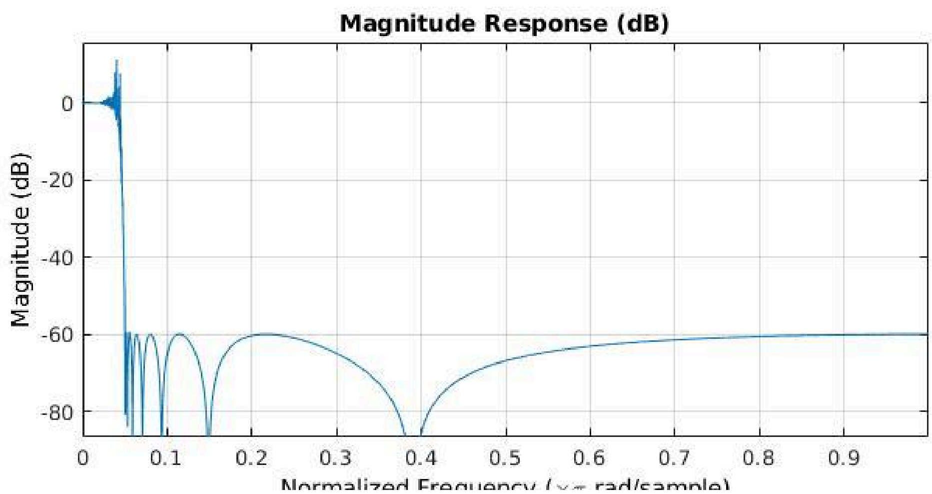
```
zplane(c);
%magnitude response
fvtool(c);

%c
figure;
xfiltered = conv(c,x);
spectrogram(xfiltered,512,256,512,fs,'yaxis');
%soundsc(xfiltered);

%The signal after the filter only plays the low frequency part of the
%signal loudly. This can be corelatted to the magnitude response of
 the
%signal as it can be seen that the magnitude response drops
 dramatically
%after the lower frequency.

% The IIR filter has a lower computational complexity,as it is of a
 much
% lower order. The fir filter has a more predictable audio quality, as
 can
% be seen in it's Magnitude resp
```

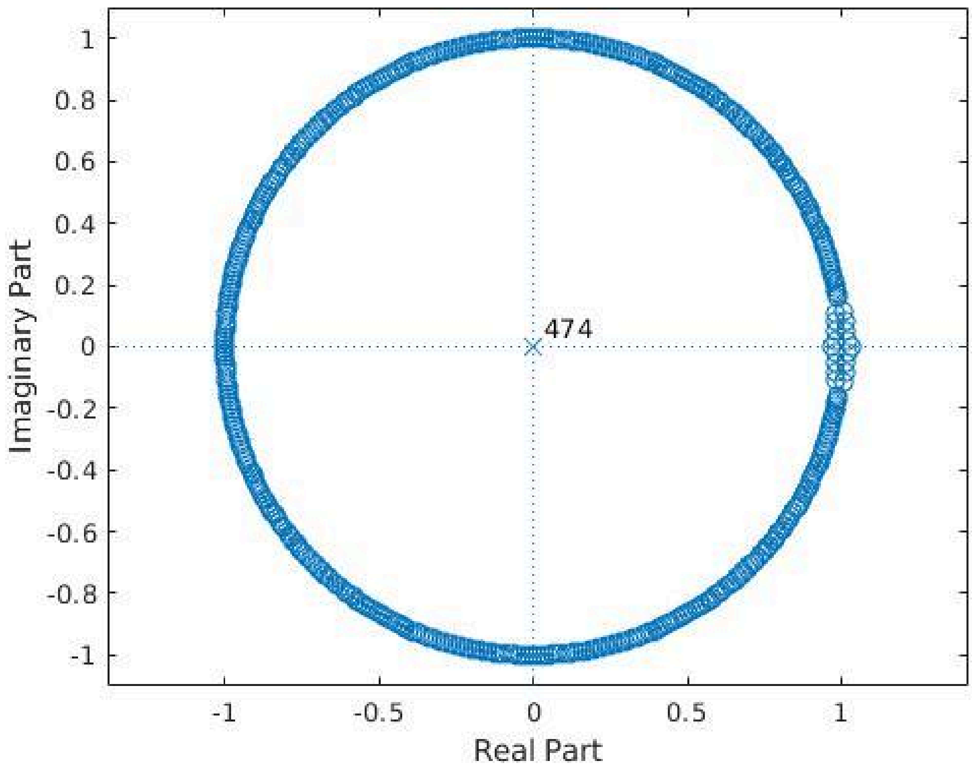
```
transfer_function =


  0.0007976 s^14 - 0.009764 s^13 + 0.05626 s^12 - 0.2028 s^11 + 0.5122
 s^10

         - 0.9618 s^9 + 1.388 s^8 - 1.565 s^7 + 1.388 s^6 - 0.9618
 s^5

         + 0.5122 s^4 - 0.2028 s^3 + 0.05626 s^2 - 0.009764 s +
 0.0007976


 --------------------------------------------------------------------------

   s^14 - 12.52 s^13 + 72.9 s^12 - 261.4 s^11 + 644.9 s^10 - 1158 s^9

          + 1562 s^8 - 1606 s^7 + 1265 s^6 - 760 s^5 + 342.7 s^4

                                 - 112.5 s^3 + 25.4 s^2 - 3.533 s +
 0.2283
```

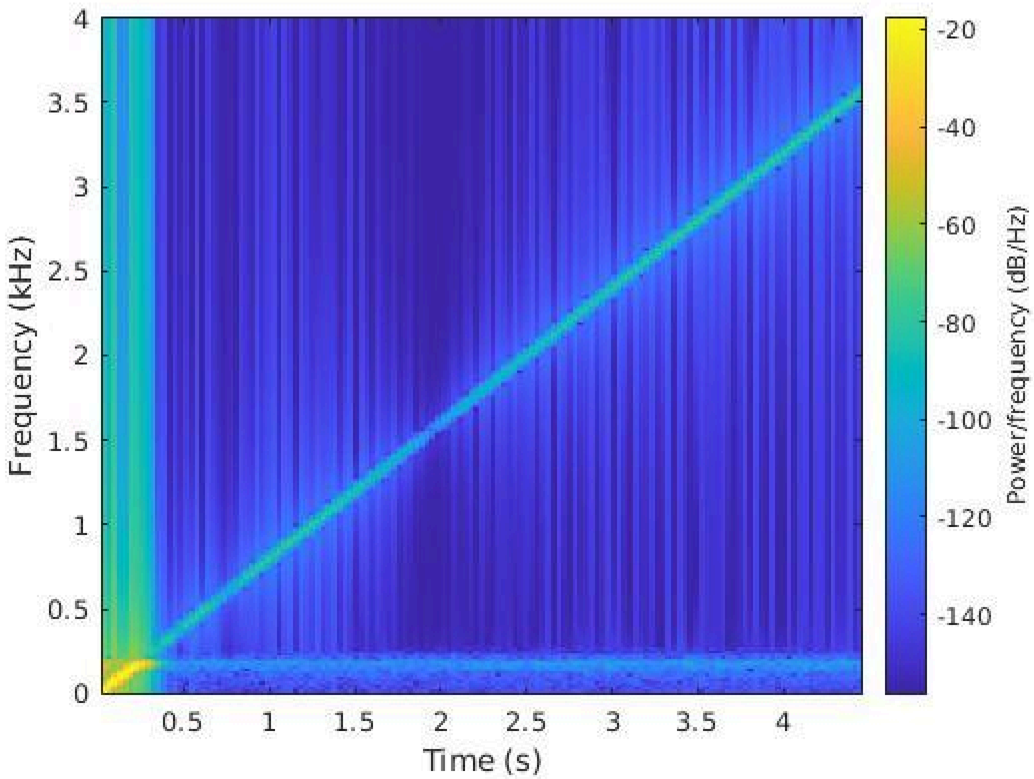
Continuous-time transfer function.

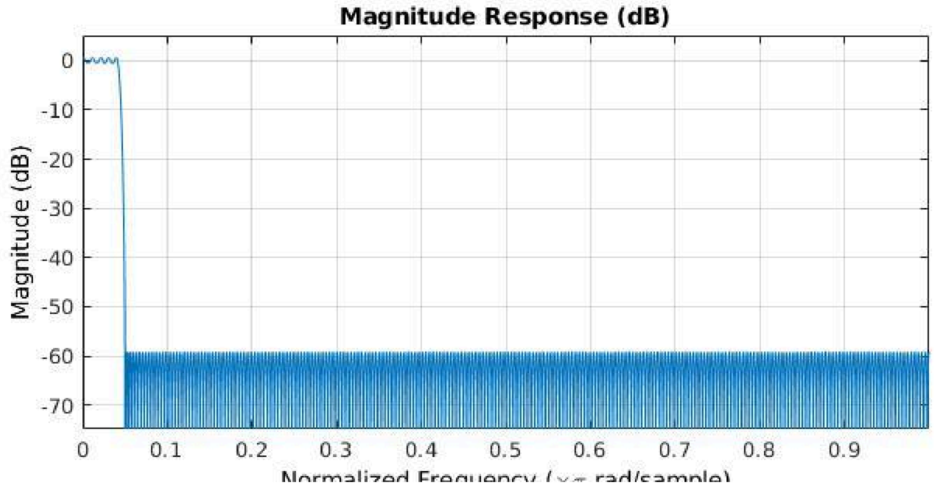


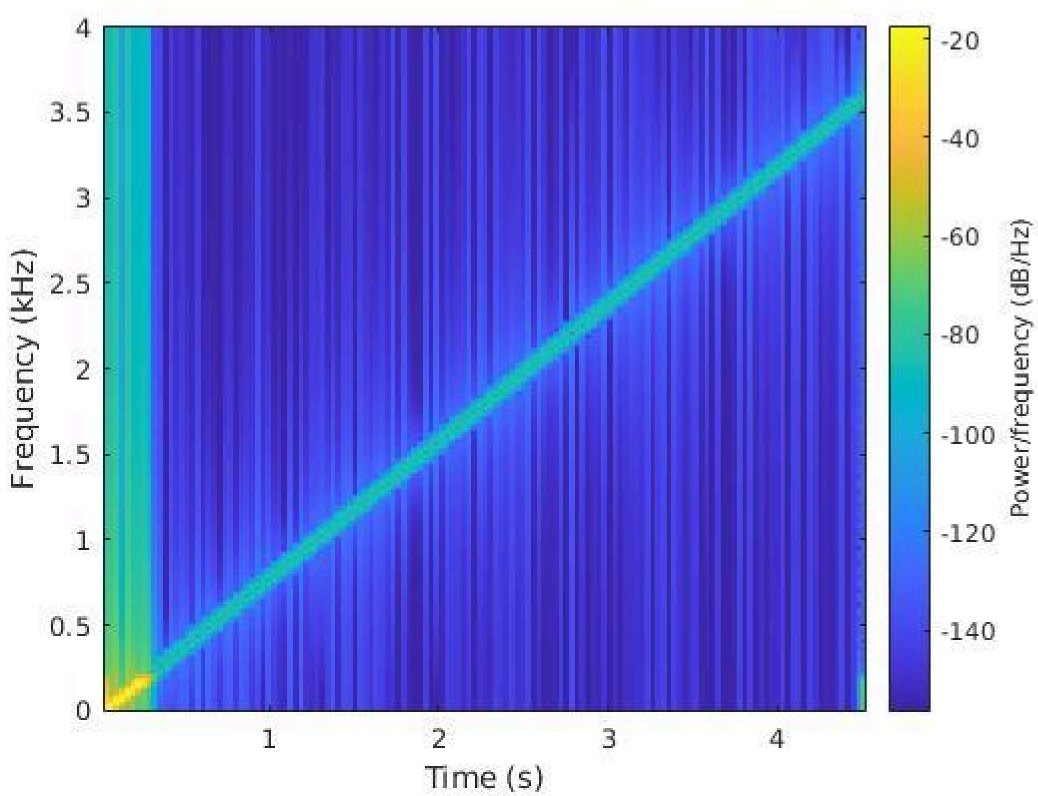

Frequency (kHz)
Time (s)
Power/frequency (dB/Hz)


Magnitude Response (dB)
Magnitude (dB)


474
Imaginary Part
Real Part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.5
0.5


Frequency (kHz)
Time (s)
Power/frequency (dB/Hz)
4
3.5
3
2.5
2
1.5
1
0.5
0
-20
-40
-60
-80
-100
-120
-140





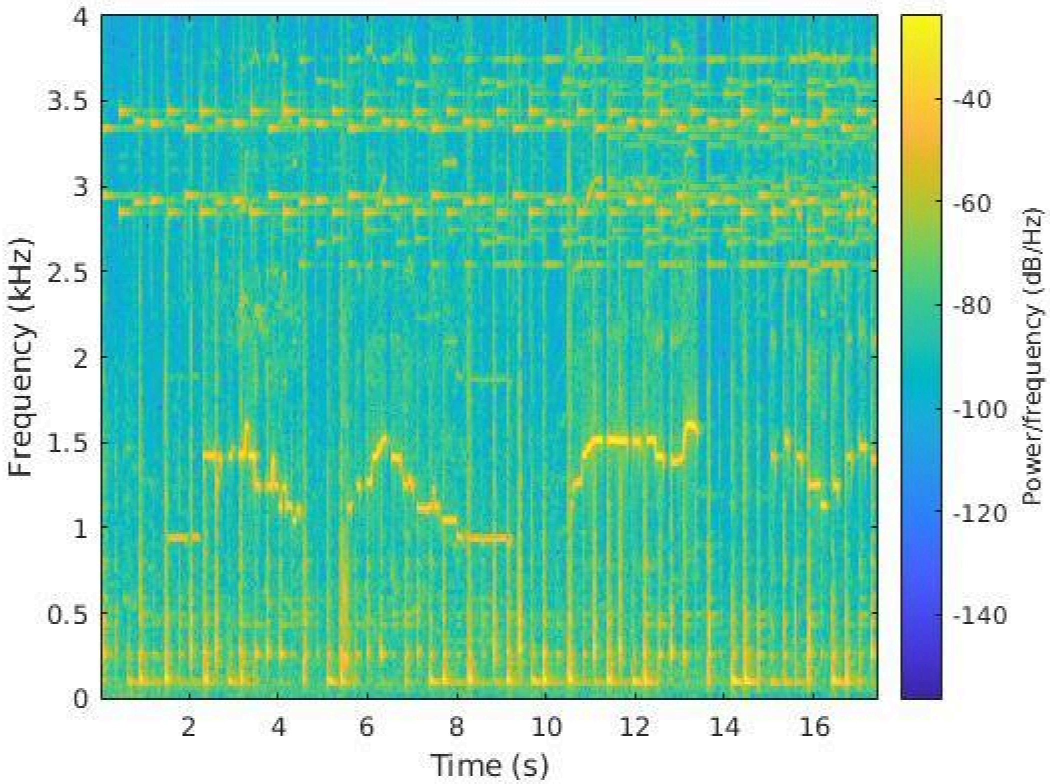
Exercise 8.2

```
[dnf,Fs] = audioread('noisy_drum_flute.wav');
figure;
spectrogram(dnf,512,256,512,fs,'yaxis');

% From the spectrogram, it can be seen that the drum frequency peaks
 at
% about 1.5 kHz, and the fulte frequency is localized above 2.5 kHz
% both have a minimum db/Hz of about 60
```

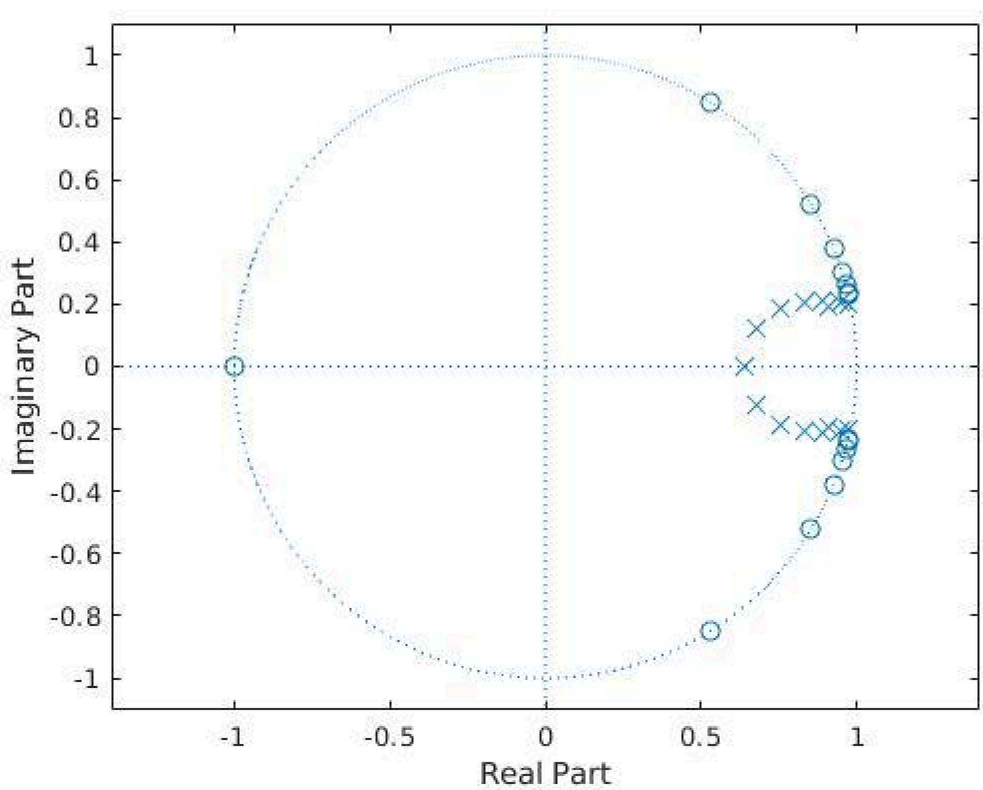
```
% the b and a of this filter are stored as d and e respectivley.


filtered = filter(d,e,dnf);
audiowrite('drums.wav',filtered,Fs);

% I created a low-pass filter, because we need to pass low frequencies
%The filter is an IIR filter to minimize the order
% Impulse response
impulse = [ zeros(1,100) 1 zeros(1,100)];
impulse_resp = filter(d,e,impulse);
figure;
stem(impulse_resp);
title('filter impulse response plot');

%Magnitude response
%used fvtools but had to export the images for proper publishing
% They are included at the bottom of the published file


%transfer function
transfer_function = tf(d,e)

%poles and zeros
zplane(d,e);

%The poles correlate to the points where the magnitude response grows
 or
%decays

% Dtft plot

w = -pi:pi/1000:pi;
input_dtft = dtft(dnf,w);
output_dtft = dtft(filtered,w);

figure;
subplot (2 , 1 , 1)
plot ((w/pi)*Fs/2,20*log10(abs(input_dtft)));
grid on ;
title ( 'Magnitude Response input dtft')
xlabel ( 'Frequency Hz') ;
ylabel ( ' Amplitude ') ;
subplot (2 , 1 , 2)
plot ((w/pi)*Fs/2,20*log10(abs(output_dtft))) ;
grid on ;
title ( 'Magnitude Response output dtft')
xlabel ( 'Frequency Hz') ;
ylabel ( ' Amplitude ') ;
```

```
% The magnitude response shows that the higher frequencies were
 removed
```

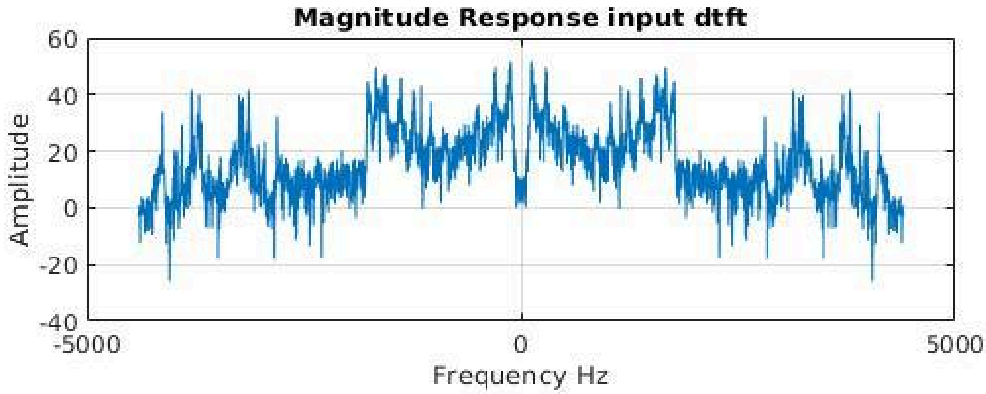
```
transfer_function =

  0.0008234 s^15 - 0.009341 s^14 + 0.04912 s^13 - 0.1574 s^12 + 0.3384
 s^11

         - 0.4996 s^10 + 0.4791 s^9 - 0.201 s^8 - 0.201 s^7 + 0.4791
 s^6

         - 0.4996 s^5 + 0.3384 s^4 - 0.1574 s^3 + 0.04912 s^2 -
 0.009341 s

                                                              +
 0.0008234

 --------------------------------------------------------------------------

   s^15 - 12.63 s^14 + 74.56 s^13 - 273.1 s^12 + 694.2 s^11 - 1297
 s^10

          + 1838 s^9 - 2014 s^8 + 1719 s^7 - 1143 s^6 + 587.7 s^5

                   - 229.2 s^4 + 65.66 s^3 - 13.04 s^2 + 1.606 s -
 0.09245

Continuous-time transfer function.
```

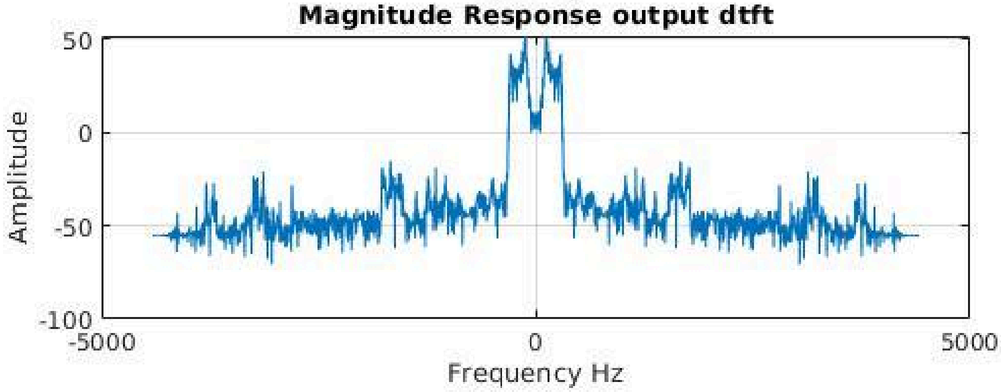


4
3.5
3
2.5
2
1.5
1
0.5
0
Frequency (kHz)
2
4
6
8
10
12
14
16
Time (s)
-40
-60
-80
-100
-120
-140
Power/frequency (dB/Hz)



1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Imaginary Part
-1
-0.5
0
0.5
1
Real Part





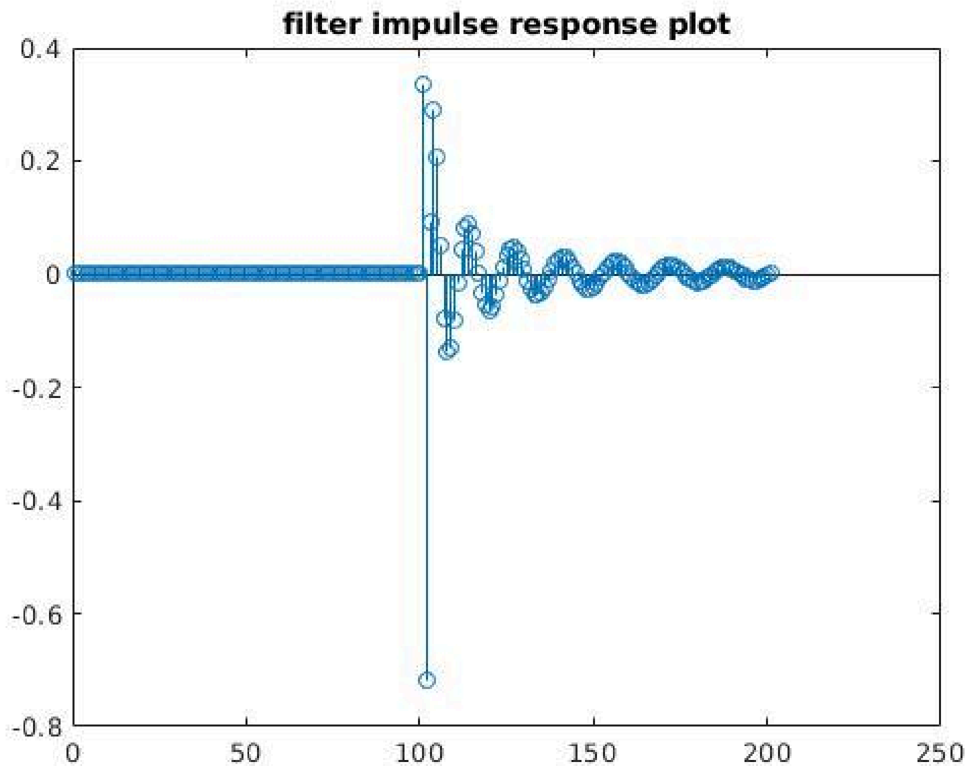
Exercise 8.3

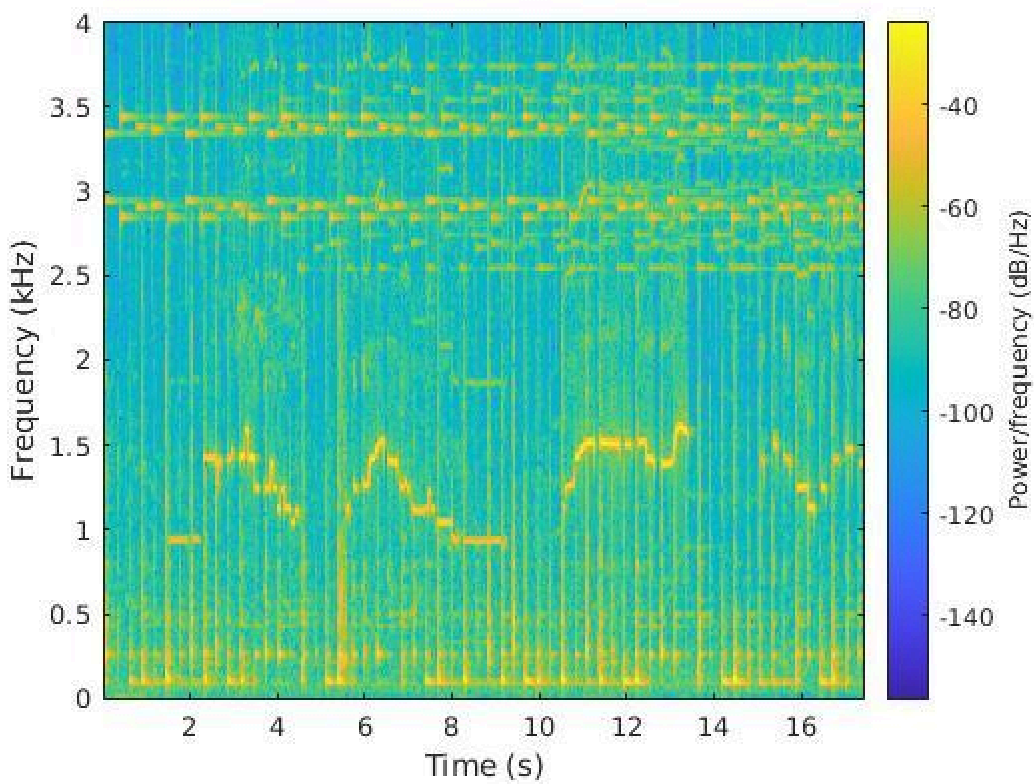
```
[dnf,Fs] = audioread('noisy_drum_flute.wav');
figure;
spectrogram(dnf,512,256,512,fs,'yaxis');

% From the spectrogram, it can be seen that the drum frequency peaks
 at
% about 1.5 kHz, and the fulte frequency is localized above 2.5 kHz
% both have a minimum db/Hz of about 60



% the b and a of this filter are stored as f and g respectivley.


filtered = filter(f,g,dnf);
audiowrite('flutes.wav',filtered,Fs);

% I created a high-pass filter, because we need to pass high
 frequencies
%The filter is an IIR filter to minimize the order
% Impulse response
impulse = [ zeros(1,100) 1 zeros(1,100)];
impulse_resp = filter(f,g,impulse);
figure;
```

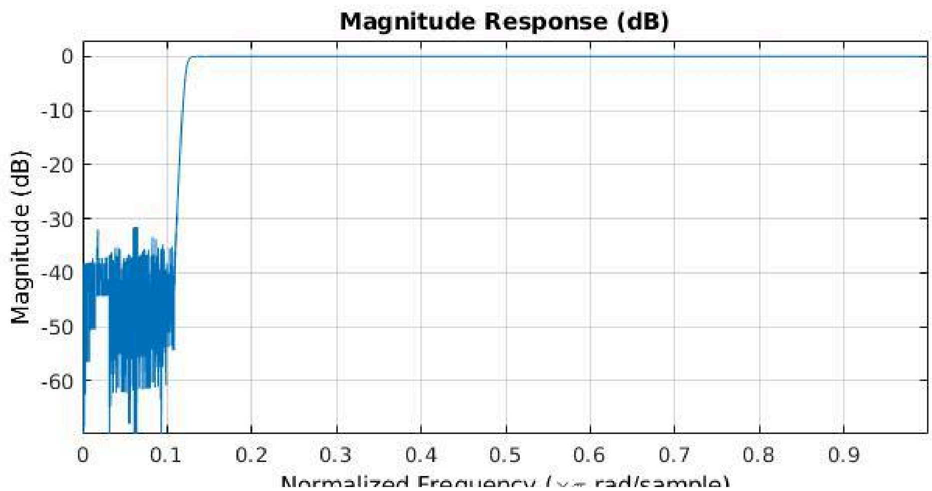
```
stem(impulse_resp);
title('filter impulse response plot');

%Magnitude response
%used fvtools but had to export the images for proper publishing
% They are included at the bottom of the published file
fvtool(f,g);


%transfer function
transfer_function = tf(f,g)

%poles and zeros
zplane(d,e);

%The poles correlate to the points where the magnitude response grows
 or
%decays

% Dtft plot

w = -pi:pi/1000:pi;
input_dtft = dtft(dnf,w);
output_dtft = dtft(filtered,w);

figure;
subplot (2 , 1 , 1)
plot ((w/pi)*Fs/2,20*log10(abs(input_dtft)));
grid on ;
title ( 'Magnitude Response input dtft')
xlabel ( 'Frequency Hz') ;
ylabel ( ' Amplitude ') ;
subplot (2 , 1 , 2)
plot ((w/pi)*Fs/2,20*log10(abs(output_dtft))) ;
grid on ;
title ( 'Magnitude Response output dtft')
xlabel ( 'Frequency Hz') ;
ylabel ( ' Amplitude ') ;

% The magnitude response shows that the lower frequencies were removed
```

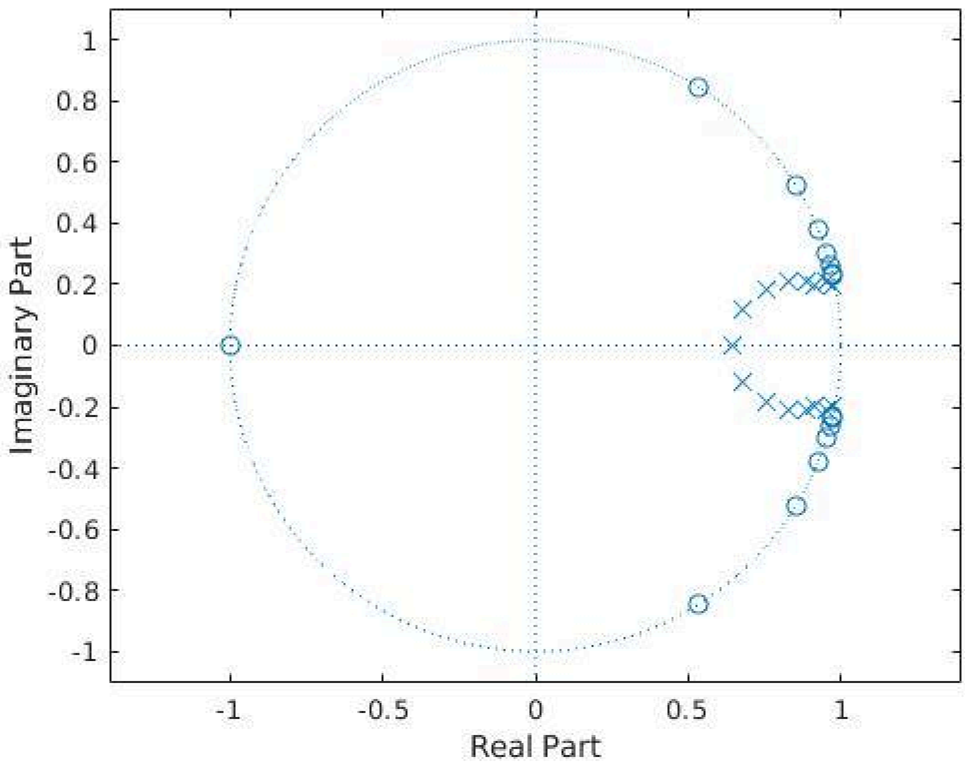
```
transfer_function =


  0.3358 s^17 - 5.556 s^16 + 43.41 s^15 - 212.7 s^14 + 732 s^13


          - 1878 s^12 + 3718 s^11 - 5803 s^10 + 7230 s^9 - 7230 s^8
```

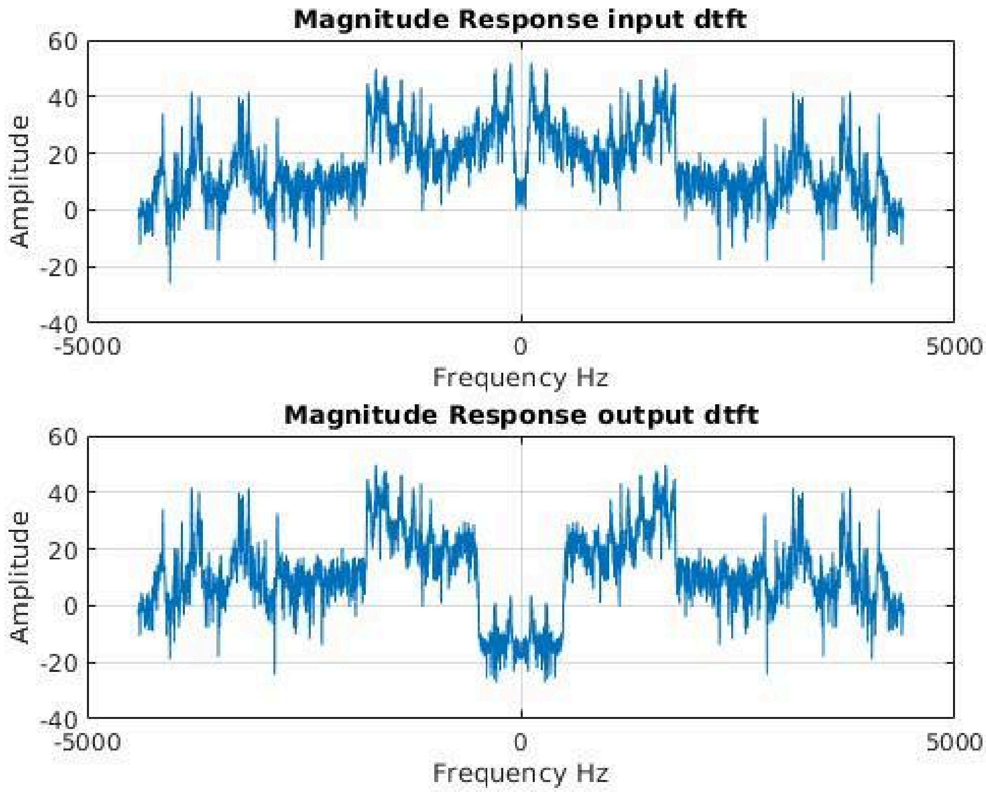
```
       + 5803 s^7 - 3718 s^6 + 1878 s^5 - 732 s^4 + 212.7 s^3

                                          - 43.41 s^2 + 5.556 s -
0.3358

  ----------------------------------------------------------------------

  s^17 - 14.41 s^16 + 98.18 s^15 - 420.3 s^14 + 1266 s^13 - 2845 s^12

       + 4942 s^11 - 6776 s^10 + 7423 s^9 - 6535 s^8 + 4623 s^7

       - 2612 s^6 + 1165 s^5 - 401.1 s^4 + 103 s^3 - 18.6 s^2

                                                  + 2.107 s -
0.1128

Continuous-time transfer function.
```


filter impulse response plot
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
50
100
150
200
250


Frequency (kHz)
Time (s)
Power/frequency (dB/Hz)
4
3.5
3
2.5
2
1.5
1
0.5
0
2
4
6
8
10
12
14
16
-40
-60
-80
-100
-120
-140


Magnitude Response (dB)
Magnitude (dB)
0
-10
-20
-30
-40
-50
-60
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Normalized Frequency (×π rad/sample)


Imaginary Part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
Real Part



Published with MATLAB® R2020a