Table of Contents

Don't change this section

```
clc; clear all; clf;

% read audiofile
[y, Fs] = audioread('Purdue__Hail_Purdue_ver2.mp3', [325000, 390000]);

% compute ak's
ak = ifft(y);

% compute power
power = abs(ak).^2;
dt = 1/Fs;
nu = length(y);

% find frequencies corresponding to ak's
freq = fftfreq(nu, dt);

% find magnitude of ak's, expressed in log-scale, Decibels
dB = 20*log10(abs(ak));
```

You only have to change this section

```
% ideal band stop filter, change the center and bandwidth to reduce
% the volume of the singing from the Purdue fight song
center = 2200;  % TODO center frequency in Hz for band-stop, set this
bandwidth = 1350; % TODO bandwidth in Hz for the band-stop, set this

% find all indicies within the band-stop range
k_kill = abs(abs(freq) - center) < bandwidth; % TODO, find all k's you want to
 remove
  % note, to use a bandstop filter you can use logic like this
  %  k_kill = abs(abs(freq) - center) < bandwidth
  % notice that this finds k and -k, so handles the conjugates for you
```

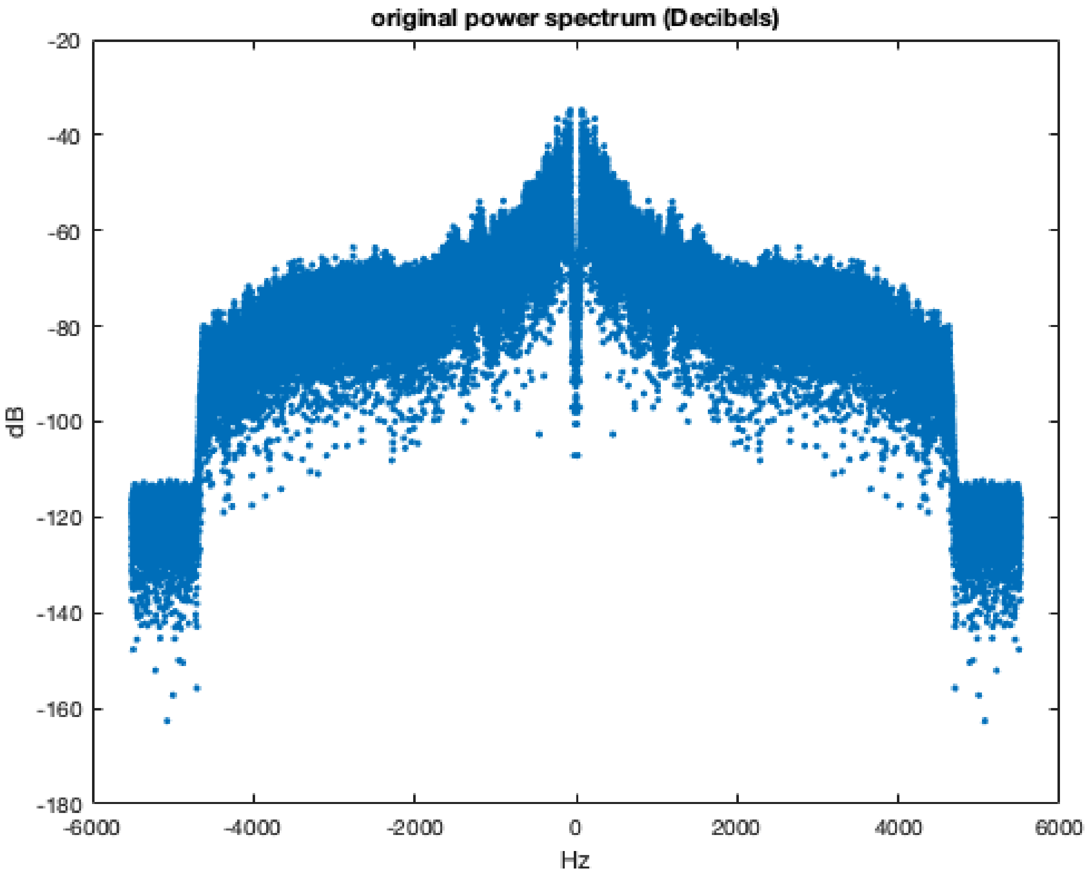
Don't change below here

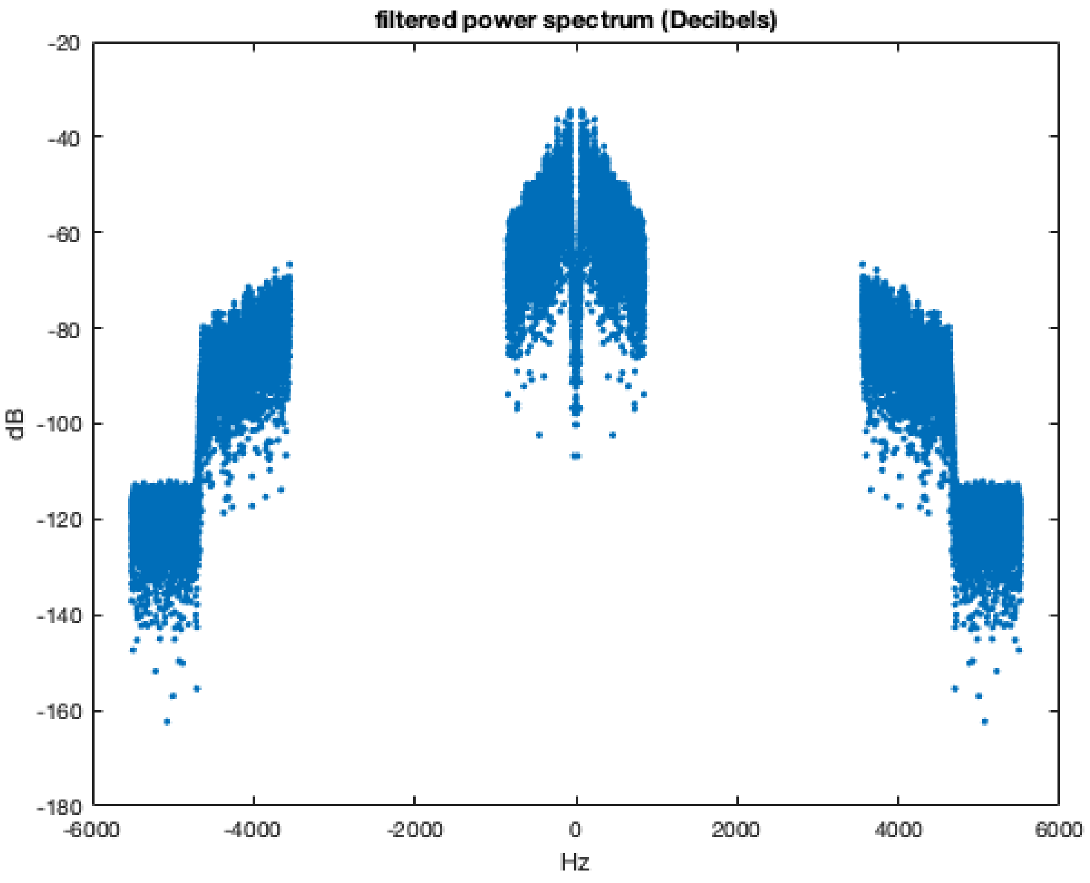
```
% initially copy ak into ak_filt (filtered ak's)
ak_filt = ak;

% eliminate ak's for frequencies we don't want
ak_filt(k_kill) = 0;
```

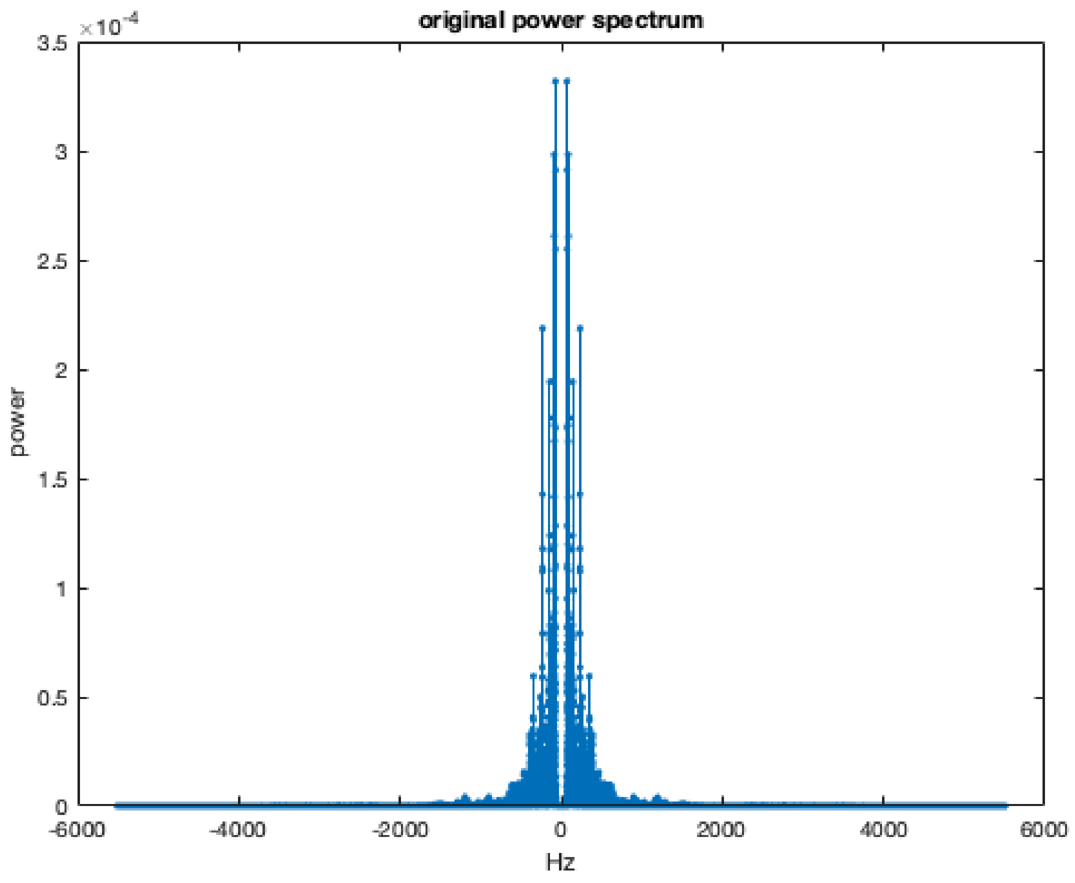
```
% renormalize to keep volume level the same
ak_filt = ak_filt*norm(ak)/norm(ak_filt);
y_filt = fft(ak_filt);

% prevent any signal outside range -1 to 1
y_filt = y_filt/max(y_filt);

% make sure that we didn't make any mistakes
% eliminating ak's
disp('norm of imaginary part should be small');
disp(norm(imag(y_filt)));
y_filt = real(y_filt);

% instead of power, we will plot the magnitude of the
% ak on the log scale in Decibels, since human hearing
% also operates on a log scale
dB_filt = 20*log10(abs(ak_filt));
power_filt = abs(ak_filt).^2;

figure(1)
plot(freq, dB, '.');
xlabel('Hz')
ylabel('dB')
title('original power spectrum (Decibels)')

figure(2)
plot(freq, dB_filt, '.');
xlabel('Hz')
ylabel('dB')
title('filtered power spectrum (Decibels)')

figure(3)
stem(freq, power, '.');
xlabel('Hz')
ylabel('power')
title('original power spectrum')

figure(4)
stem(freq, power_filt, '.');
xlabel('Hz')
ylabel('power')
title('filtered power spectrum')

% append audio signals together, first play
% filtered, then unfiltered fight song
y_comb = real([y_filt; y]);
p = audioplayer(y_comb, Fs);
play(p);

% function to compute frequencies
function freq = fftfreq(nu, dt)
    if mod(nu, 2) == 0
        k_vals = [(0:nu/2-1), (-nu/2:-1)];
    else
```

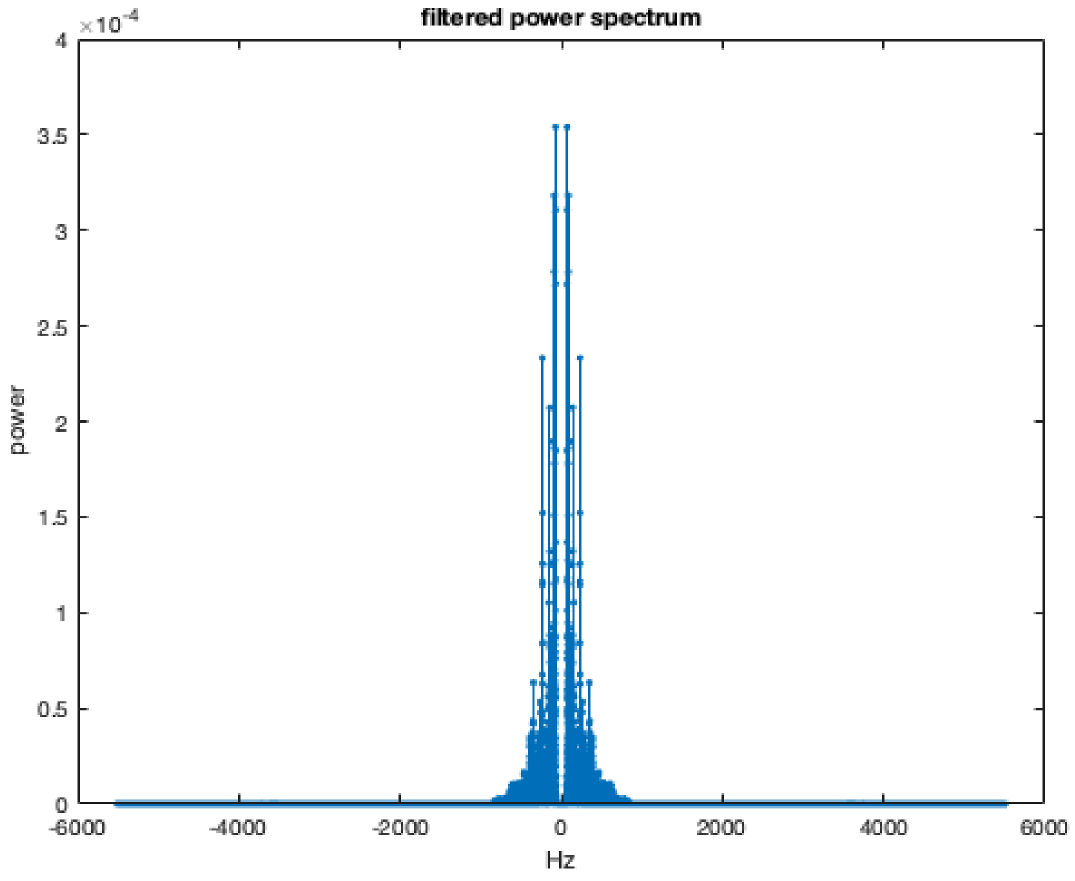
```
        k_vals = [(0:(nu-1)/2), (-(nu-1)/2:-1)];
    end
    freq = k_vals./(dt.*nu);
end
```

```
norm of imaginary part should be small
   2.8408e-14
```




filtered power spectrum (Decibels)
dB
Hz
-20
-40
-60
-80
-100
-120
-140
-160
-180
-6000
-4000
-2000
0
2000
4000
6000


original power spectrum
×10^-4
3.5
3
2.5
2
1.5
1
0.5
0
power
-6000
-4000
-2000
0
2000
4000
6000
Hz



Published with MATLAB® R2021b