CmpE 362 Homework 3 Report

Berkay Kozan

May 2019

Contents

1 Introduction

In this homework, I implemented some simple frequency domain exercises with MATLAB.

2 Advanced Peak Finder

2.1 Explanation of code

In this part, I implemented an advanced peak filter. I used lowpass FIR filter for this exercise. I used *lowpass* method for the exercise. *y = lowpass(x,fpass,fs) specifies that x is sampled at a rate of fs hertz. fpass is the passband frequency of the filter in hertz.*[1] In these case, x is the data, fpass is the cutoff frequency and fs is frequency we obtained from data,

First, I read the file from directory. Then, I found peaks of data without any filter. Afterwards, I got number of peaks for each cut off frequency. Logically, if there is no filter, then there is no specified cutoff frequency. Therefore, I wrote no filter sample to "Fs". Because if there is no filter applied, then logically cutoff frequency is equal to sample's frequency.

You can see the code and plot below.

[1] https://www.mathworks.com/help/signal/ref/lowpass.html

2.2 Code

```
clear;
clc;
hfile = 'PinkPanther30.wav';  % This is a string, corresponding ...
    to the filename
clear y Fs  % Clear unneded variables

[y, Fs] = audioread(hfile); % Read the data back into MATLAB, ...
    and listen to audio.

peaks_without_filter= findpeaks(y); %Peaks of no filter.
cutoffs = [1000, 2000, 3000, 4000, Fs]; %Cutoff frequencies for ...
    each iteration.
number_of_cutoffs = length(cutoffs); %Number of cutoffs to iterate.
number_of_peaks = zeros(1,number_of_cutoffs); %Number of peaks ...
    for each iteration, it is array of zeros initially. We will ...
    change the values.

for i= 1: number_of_cutoffs-1
    z = lowpass(y,cutoffs(i),Fs); %z is lowpass filtered version ...
        of data, with specified cutoff frequency.
    peaks= findpeaks(z);
    n_peaks= length(peaks); %number of peaks for z.
    number_of_peaks(i)=n_peaks; %Save this to an array.
end

% Add no filter result to last index
number_of_peaks(number_of_cutoffs) = length(peaks_without_filter);

figure(1);
plot(cutoffs, number_of_peaks);

ax=gca;
ax.YRuler.Exponent = 0; %Set exponent of Y to zero to see whole ...
    number of peaks.
ax.XRuler.Exponent = 0; %Set exponent of X to zero to see whole ...
    frequency.

title('Number of peaks vs. Changing cut off frequencies (Hz)');
```

Listing 1: AdvancedPeakFreqFilter.m

2.3 Plot

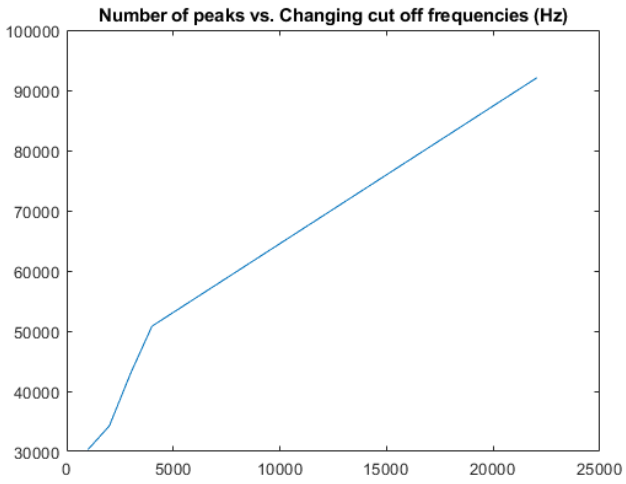

Figure 1: Advanced Peak Frequency Filter

2.4 Conclusion from Plot

As we see from Figure 1, when we increase cutoff frequencies, number of peaks increase as well. The reason is that, when we decrease cutoff frequency, multiple peaks that are over this frequency can be counted as 1. Therefore, when we increase cutoff frequency, we count more peaks.

3 Converting a Hubble Deep Space Image Into Space Sound

3.1 Explanation of code

In this part, we have a RGB image. For finding black and non-black pixels, I used 2 methods respectively. I converted the image to gray with *rgb2gray* method. Afterwards, I converted the gray image to black and white image with *imbinarize* method. This helped me a lot, since using defaults are generally better. As I see from converted pictures, they did a great job. You can see the images below.

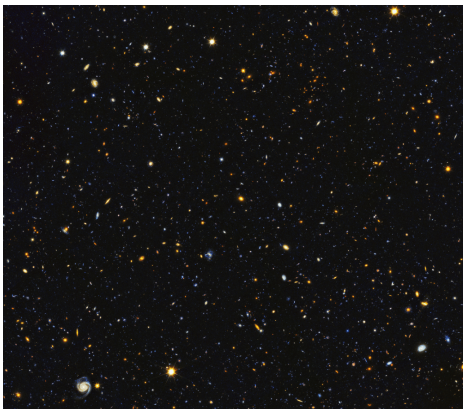

Figure 2: Original image

Figure 3: Gray image

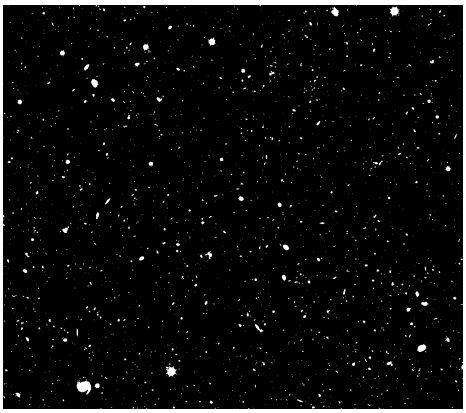

Figure 4: Binary image

So, I used the binary file from now on. I found dimensions of the image, they are 900 and 1024. Then, I take **Sample rate** with name **T**. T is an array with length 1001 and has elements from 0 to 1, with 0.01 difference between each elements. Afterwards, I started inspecting each column. For each item in the column, I tested if the item is black or non-black (a.k.a white, because I used binary image.) If the item is black, amplitude to be created is 0, therefore the wave to be added is 0.

If the item is non-black, then amplitude is collected from it's index, with formula `fix((j−1)/90)+1;`. If the item is in range of 1 to 90th element in column, then this gives Amplitude as 1.

For both black and non-black cases, I took samples from frequencies. I used `Amp*cos(2*pi*j*T)` instead of *sine* for calculating time domain values for each wave. I added these values, and have an array with length 1x1001 for each column. When I concatenate each summation of columns for each row, I got 1x1001*1024 file, that is the wave file that contains the *Space Sound*. Afterwards, I export the *Space Sound* to *SonifiedDeepSpace.wav*, which can be listened to. Also, it creates sound as well.

You can see code and example plots below.

3.2 Code

```
clear;
clc;
Image=imread('Hubble_Massive_Panorama.png'); %Take the image as ...
    input.
gray= rgb2gray(Image); %Gray image.
binaryimage= imbinarize(gray); %Binarize gray image, each pixel ...
    is either 1 or 0.
dimension_length= size(binaryimage);
column_length= dimension_length(2); %number of columns
row_length= dimension_length(1); %number of rows

T = (0:0.001:1);  %Sampling number.
wavefile=[];

for i=1:column_length %1024 times, for each column.
    sum=zeros(1,length(T));
    for j=1:row_length %900 times, for each row(length of column)
        if binaryimage(j,i)==1 %If the pixel is non_black.
            Amp=fix((j_1)/90)+1; %Amplitude's formula. j_1 is ...
                here, because at 90th item, index shouldn't ...
                change. It should change at next item.
            sum= sum+ (Amp*cos(2*pi*j*T)); %Summation of column.
        else
            Amp= 0;
            sum= sum+ (Amp*cos(2*pi*j*T)); %This will be 0, ...
                because Amp=0.
        end
    end
    wavefile= cat(2, sum, wavefile);
end
plot(wavefile);
title(strcat('Values of Wavefile, with sample rate: ', ...
    num2str(length(T))));
audiowrite("SonifiedDeepSpace.wav",wavefile,length(T));
sound(wavefile, length(T)); % Sounds the data.
```

Listing 2: SonifiedDeepSpace.m

3.3 Plots

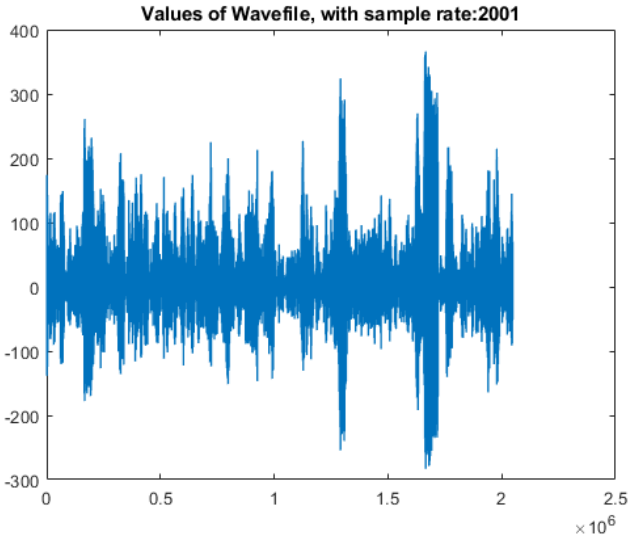

Figure 5: The wave file, with Sample Rate 2001

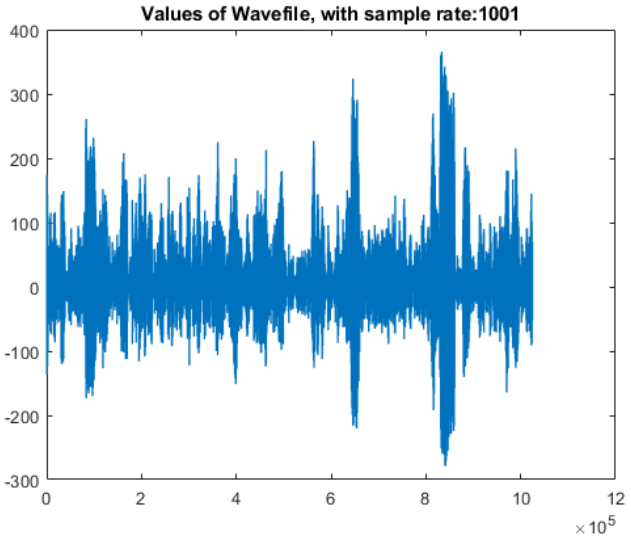

Figure 6: The wave file, with Sample Rate 1001

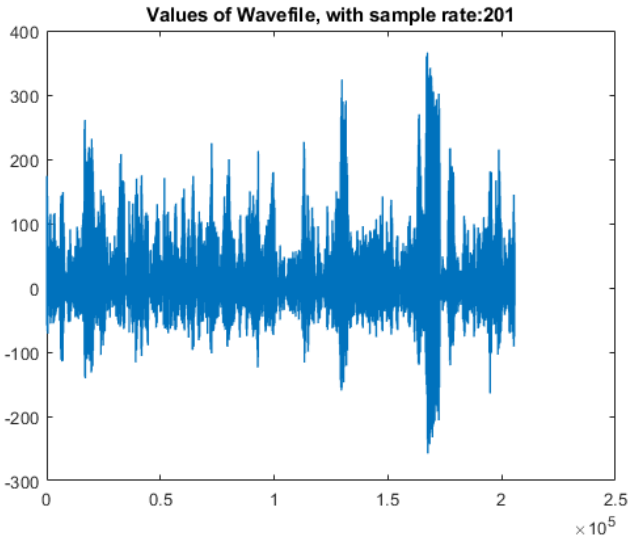

Figure 7: The wave file, with Sample Rate 201

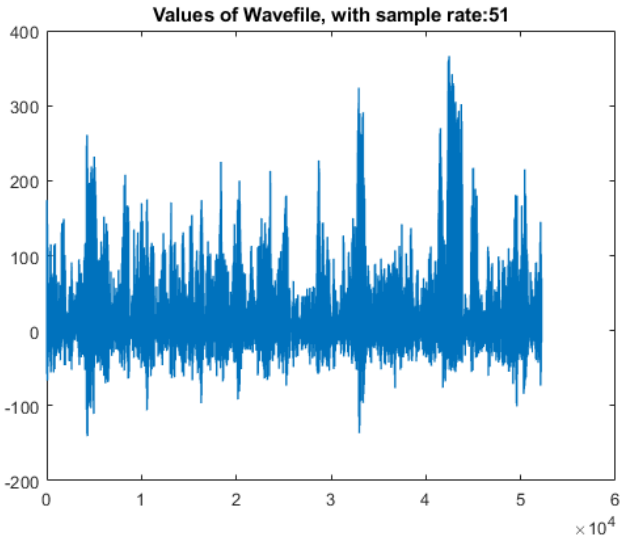

Figure 8: The wave file, with Sample Rate 51

3.4 Conclusion from Plots

As we can see, when we increase sample rate, then the shape of plot converges to a constant image. When the sample rate is low (Figure 8) , the image starts to converge to final version of it. As we see from Figure 6 and Figure 7, the images look really alike. So, we can conclude that the bigger the sample rate, the better the result is.