# CmpE 362- Homework 3

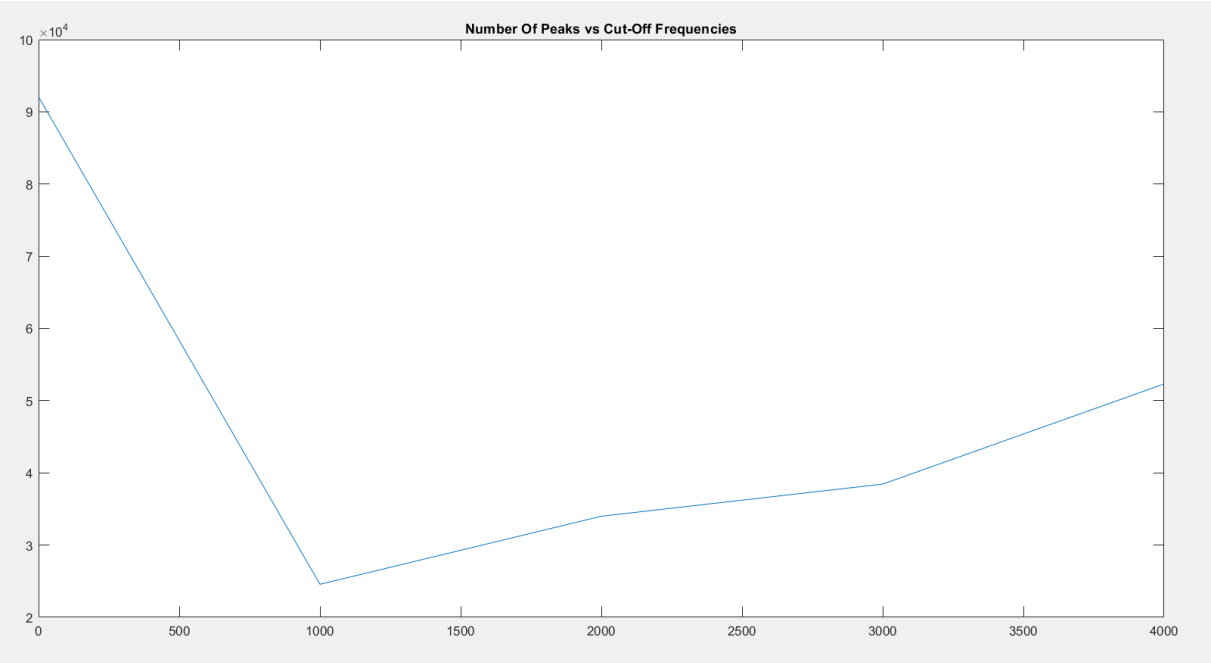
BARAN DENİZ KORKMAZ
2015400183

---

## QUESTION 1: ADVANCED PEAK FINDER

```
%{
   QUESTION 1: ADVANCED PEAK FINDER
%}

%{
   Clears old variables from console and workspace to avoid some
   possible errors.
%}
clear;clc;
clear y Fs

%This is a string, corresponding to the filename
audioFile = 'PinkPanther30.wav';

%The sound has been converted into an array by audioread built-in function
[y, Fs] = audioread(audioFile);

%To check if the data has been read correctly
sound(y, Fs);

%The number of peaks of the original signal is calculated
numOfPeaksNoFilter=numel(findpeaks(y));

%Creating the cut-off frequencies array for forward-use
%0 Hz Frequency indicates the original signal to enable
%a simple notation
cutOffFrequencies = [0,1000,2000,3000,4000];

%Creating the array that will hold the number of peak values for
%corresponding frequency values
numOfPeaksForEachCutOff = zeros(1,5);

%The number of peaks of the original signal is added into the
%corresponding index which is the index number 0 which will later be
%denotes as 0 Hz Frequency in the plot.
numOfPeaksForEachCutOff(1) = numOfPeaksNoFilter;

%The loop will iterate for the indexes corresponding to the cut-off freqs
for i = 2:5
    % Defining low pass filter by designfilt method
    lowPassFilter = designfilt('lowpassiir', ...
             'PassbandFrequency', cutOffFrequencies(i), ...
             'FilterOrder',8, ...
             'PassbandRipple',0.2, ...
             'SampleRate', 22050);

    % Apply low pass filter by filter method
    result = filter(lowPassFilter,y);

    % Finding the number of peaks for the data which is the result of
    % low-pass filter
```

```
    numOfPeaksCutOff = numel(findpeaks(result));

    % Adding the number of peaks into the corresponding index
    numOfPeaksForEachCutOff(i) = numOfPeaksCutOff;
end

% Plotting the figure
figure;
plot(cutOffFrequencies, numOfPeaksForEachCutOff);
title('Number Of Peaks vs Cut-Off Frequencies');
```




## On Question 1

In question 1, i have created an array which holds the cut-off frequency values with an element of 0 at the index 0 which indicates the original signal. Inside the for loop which iterates from the indexes 2 to 5 which correspons to the related cut-off frequencies, i designed a low-pass iir filter with designfilt function and then get the filtered signal by filter function which operates on the low-pass filter that i've designed before. For each corresponding frequency result, i obtained the resulting signals and the number of peaks for these signals. When the for loop ends, i have obtained the array containing to the number of peaks for the corresponding cut-off frequency levels with the number of peaks of the original signal at the index of 0. Then the resulting plot has been resulted at the very end of the script.

# QUESTION 2: CONVERTING A HUBBLE DEEP SPACE IMAGE INTO SPACE SOUND

```
%{
   QUESTION 2: CONVERTING A HUBBLE DEEP SPACE IMAGE INTO SPACE SOUND
%}

%{
   Clears old variables from console and workspace to avoid some
   possible errors.
%}
clear;clc;
% Sets the file name in order to be able to read it via imread command.
path='Hubble-Massive-Panorama.png';

% Reads the png file.
hubble=imread(path);

% Converts the rgb image into the grayscale form.
hubblegray=rgb2gray(hubble);

% Converts the rgb image into the black-white form. The default threshold
% value is 0.5.
hubbleBW=imbinarize(hubblegray);

% The sound array that will hold the concatenated sounds with duration of
% 1-sec
soundArray=0;

% The duration has been specified.
duration=1;

% The sampling rate has been specified. The sampling rate is selected
% according to the Nyquist Rate.
Fs=2000;

% The array that will be used in the creation of the sound which holds
% values from 0 to duration (1) with an interval of 1/Sampling Rate.
t=0:1/Fs:duration;

% The 900*1024 array will be processed inside the nested loops.
for i = 1:1024 % The number of columns is indicated by i which is 1024.
    myWave=0; % The wave of the column processed has been initialized.
    for j = 900:-1:1 % The number of rows is indicated by j which is 900
        Amplitude=0; % The amplitude of the wave has been initialized.

        %{
            If the current cell is marked as white, Then the amplitude
            according to the number of the pixel will be assigned and
            the wave created by the pixel will be added into the myWave
            variable which holds the overall signal for a given column.
        %}
        if hubbleBW(j,i)==1
            if j>=1 && j<=90
                Amplitude=10;
                myWave=myWave+Amplitude*sin(2*pi*j*t);
            end
            if j>=91 && j<=180
                Amplitude=9;
                myWave=myWave+Amplitude*sin(2*pi*j*t);
            end
            if j>=181 && j<=270
                Amplitude=8;
```

```
                myWave=myWave+Amplitude*sin(2*pi*j*t);
            end
            if j>=271 && j<=360
                Amplitude=7;
                myWave=myWave+Amplitude*sin(2*pi*j*t);
            end
            if j>=361 && j<=450
                Amplitude=6;
                myWave=myWave+Amplitude*sin(2*pi*j*t);
            end
            if j>=451 && j<=540
                Amplitude=5;
                myWave=myWave+Amplitude*sin(2*pi*j*t);
            end
            if j>=541 && j<=630
                Amplitude=4;
                myWave=myWave+Amplitude*sin(2*pi*j*t);
            end
            if j>=631 && j<=720
                Amplitude=3;
                myWave=myWave+Amplitude*sin(2*pi*j*t);
            end
            if j>=721 && j<=810
                Amplitude=2;
                myWave=myWave+Amplitude*sin(2*pi*j*t);
            end
            if j>=811 && j<=900
                Amplitude=1;
                myWave=myWave+Amplitude*sin(2*pi*j*t);
            end
        end
    end
    %Before a new column has been processed, the overall wave of the
    %current column will be concatenated into the soundArray which holds
    %the sounds created by each column.
    soundArray=cat(2,soundArray,myWave);
end

% The wav file has been created.
audiowrite("SonifiedDeepSpace.wav",soundArray,2000);
% The wav file has been read to enable sound function.
[y,Fs]=audioread("SonifiedDeepSpace.wav");
% Plays the created sound.
sound(y,Fs);
```

## On Question 2

In question 2, after the image has been converted into the gray-scale version, then the black-white version of the signal has become available to obtain by imbinarize built-in function. The overall signal for each column has been created inside the nested for loop and concatenated into the soundArray array which holds the overall signals corresponding to each columns. Finally, by audiowrite function, i have created the wav file from the array and by audioread function, the sound is ready to play.