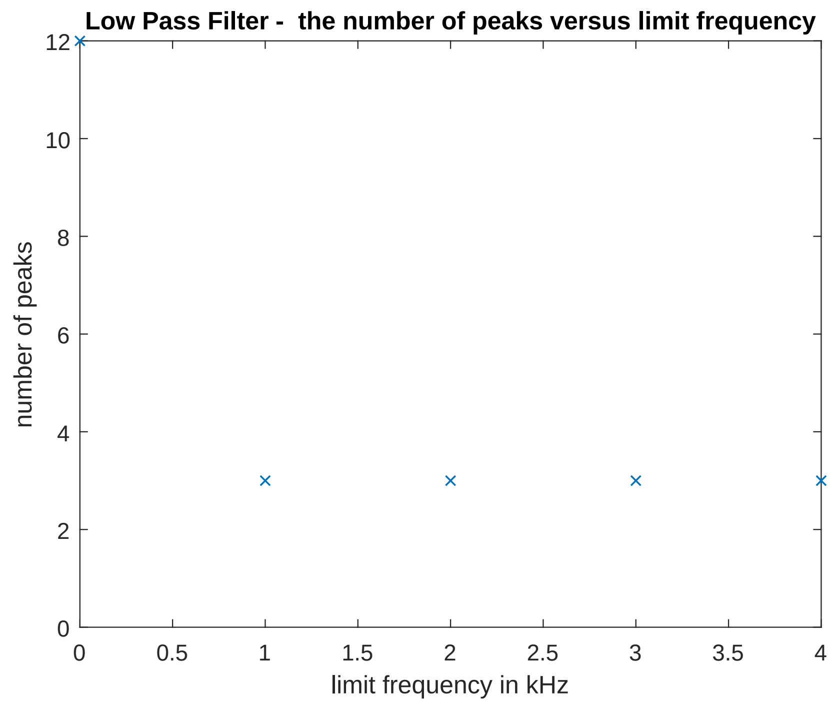
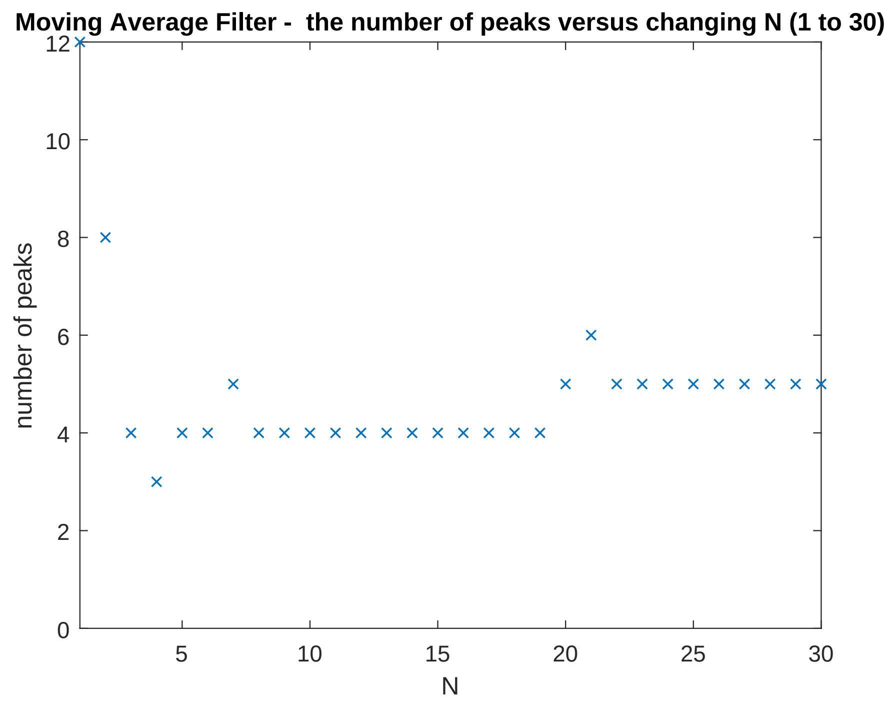
CmpE 362
2018 Spring
Homework 2

Baran Kılıç
2014400123

April 10, 2018

1 Peak Finder





Low pass filter removes the high frequency components. The resulting signal becomes smooth. Therefore, the number of peaks I found decreased when I applied the low pass filter. The moving average filter takes average of signal N points and reduces noise. Since the noise is reduced, I found less peaks when I applied the moving average filter.

1.1 Code

```
prompt = 'Enter the folder name in the working directory (e.g. signal.csv): ';
foldername = input(prompt,'s');

allcvsfiles = dir( strcat(foldername,'/**/*.csv') );

for file = allcvsfiles'
    csvdata = csvread(strcat(file.folder, "\", file.name));

    % without filter
    [pks,locs,w,p] = findpeaks(csvdata);
    [pks,locs] = findpeaks(csvdata,'MinPeakProminence', ...
        mean(p)+std(p),'MinPeakWidth',mean(w)+std(w));

    numOfPeaks = length(locs);

    % low pass filter
    for limitFreq=1000:1000:4000
        LPF = dsp.LowpassFilter('PassbandFrequency',limitFreq);
        csvdata2 = LPF(csvdata);

        [pks,locs,w,p] = findpeaks(csvdata2);
        [pks,locs] = findpeaks(csvdata2,'MinPeakProminence', ...
            mean(p)+std(p),'MinPeakWidth',mean(w)+std(w));

        numOfPeaks(end+1)=length(locs);

    end

    figure;
    plot(0:length(numOfPeaks)-1,numOfPeaks,'x');

    yl = ylim; % Get current limits.
    ylim([0, yl(2)]); % draw y from 0

    xl = xlim; % Get current limits.
    xlim([0, xl(2)]); % draw x from 0

    title({'Low Pass Filter -  the number of peaks versus limit frequency'})
    xlabel('limit frequency in kHz')
    ylabel('number of peaks')
end

for file = allcvsfiles'
    csvdata = csvread(strcat(file.folder, "\", file.name));

    numOfPeaks = zeros(1,30);

    for i=1:30
        mvAvg = dsp.MovingAverage(i);
        csvdata2 = mvAvg(csvdata);

        [pks,locs,w,p] = findpeaks(csvdata2);
        [pks,locs] = findpeaks(csvdata2,'MinPeakProminence', ...
            mean(p)+std(p),'MinPeakWidth',mean(w)+std(w));

        numOfPeaks(i) = length(locs);
    end

    figure;
    plot(numOfPeaks,'x');
    yl = ylim; % Get current limits.
    ylim([0, yl(2)]); % Replace lower limit only with a y of 0.

    xl = xlim; % Get current limits.
    xlim([1, xl(2)]); % Replace lower limit only with a x of 0.

    title({'Moving Average Filter - the number of peaks versus changing N'})
    xlabel('N')
    ylabel('number of peaks')
end
```

2 Frequency(Pitch) of a Sound

Exercise 2 and 4 sounds similar. “Change the pitch” example and exercise 3 sound similar. But there is a small difference. In “Change the pitch” example, since we delete data, some information get lost but if we increase the sampling frequency the data is the same but it is played faster. In exercise 2, we duplicate data (we rearrange data). But in exercise 4, we decrease sampling frequency and the sound is player slower.

2.1 Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   CMPE 362 Homework II-b   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%

                            % Fs is the frequency = number of samples per second
                            % y is the actual sound data
hfile = 'laughter.wav';     % This is a string, corresponding to the filename
clear y Fs                  % Clear unneded variables

%% PLAYING A WAVE FILE

[y, Fs] = audioread(hfile);  % Read the data back into MATLAB, and listen to audio.
                             % nbits is number of bits per sample
sound(y, Fs);                % Play the sound & wait until it finishes

duration = numel(y) / Fs;   % Calculate the duration
pause(duration + 2)        % Wait that much + 2 seconds

%% CHANGE THE PITCH

sound(y(1:2:end), Fs);    % Get rid of even numbered samples and play the file


%% EXERCISE I
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Re-arrange the data so that   %
%   the frequency is quadrupled and play the file   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pause(duration/2 + 2)
sound(y(1:4:end), Fs); % Get every fourth sample and play file


%% EXERCISE II
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Re-arrange the data so that   %
%   the frequency is halved and play the file  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pause(duration/4 + 2)
sound(repelem(y,2), Fs); % Repeat the same elements
```

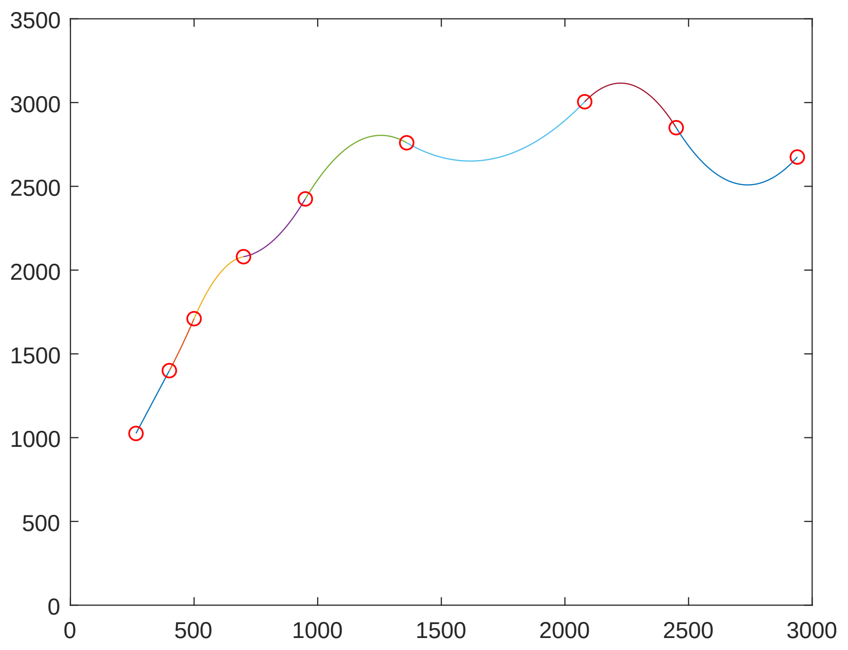
```
%% EXERCISE III
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Double Fs and play the sound  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pause(duration*2 + 2)
sound(y, 2*Fs);

%% EXERCISE IV
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Divide Fs by two and play the sound  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pause(duration/2 + 2)
sound(y, Fs/2);
```

3 Spline Interpolation



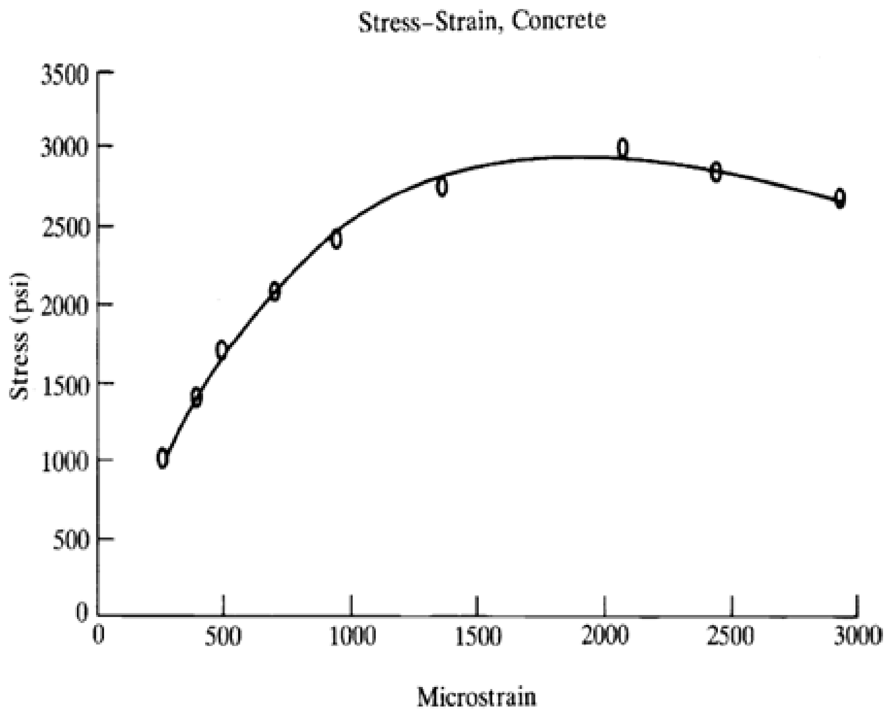


FIGURE 2.8 Stress-strain characteristic for a concrete block.

The graph looks similar for the points on the left. But for the points on the right it is not so similar. The actual function is concave everywhere. However,the interpolated graph has part that are convex. For example, the last spline or the third spline from the right.

3.1 Code

```
x = [265 400 500 700 950 1360 2080 2450 2940];
y = [1025 1400 1710 2080 2425 2760 3005 2850 2675];

A = zeros(24);

% spline pass through start point
for i=1:8
    A(i,(i-1)*3+1) = x(i).*x(i);
    A(i,(i-1)*3+2) = x(i);
    A(i,(i-1)*3+3) = 1;
end

% spline pass through end point
for i=2:9
    A(i+7,(i-2)*3+1) = x(i).*x(i);
    A(i+7,(i-2)*3+2) = x(i);
    A(i+7,(i-2)*3+3) = 1;
end

% slope at intersection of splines is equal to each other
for i=2:8
    A(i+15,(i-2)*3+1)= 2*x(i);
    A(i+15,(i-2)*3+2)= 1;
    A(i+15,(i-2)*3+4)= -2*x(i);
    A(i+15,(i-2)*3+5)= -1;
end

% first spline has 0 as coefficient for the x^2
A(24,1) = 1;

B = zeros(24,1);

for i=1:8
    B(i) = y(i);
end

for i=2:9
```

```
    B(i+7) = y(i);
end

Res = linsolve(A,B);

% plot each spline
for i=1:8
    xspline = x(i):1:x(i+1);
    yspline = Res((i-1)*3+1).*xspline.^2 + Res((i-1)*3+2).*xspline ...
     + Res((i-1)*3+3);
    plot(xspline,yspline);
    hold on;
end

plot(x,y,'o','MarkerEdgeColor','r'); % plot real points

yl = ylim; % Get current limits.
ylim([0, yl(2)]); % Replace lower limit only with a y of 0.

xl = xlim; % Get current limits.
xlim([0, xl(2)]); % Replace lower limit only with a x of 0.
```