EIT FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIONSTECHNIK

Laboratory

Digital Information Processing

Report

Names Harish Kongari [Mr. Nr. 214586]
Harikrishna Polavarapu [Mr. Nr. 218060]

Experiment Phase Vocoder

Date 12/04/2018

Contents

1. Content of the Experiment

The phase vocoder is a well-known digital signal process algorithms that performs frequency domain modification by using phase information to implement a variety of digital effects (e.g. time-streching, pitch-shifting, spectral image processing,etc). It can be categorized as analysis-synthesis techniques, which converts a time domain representation into time-frequency representation by applying STFT (short-time Fourier transform) and then performs amplitude and phase modification of specific frequency components, and finally resynthesize the frequency domain representation into the time domain by inverse-STFT.
In this experiment the basic principle of phase vocoder algorithms will be explained and examined. A traditional implementation of a phase-vocoder in Matlab will be given, from which two very important effects of phase-vocoder, namely 'Time-Streching without altering its pitch' and 'Pitch transposition' can be easily fulfilled with high fidelity.

The matlab code is based on the example written by Dan Ellis.

http://labrosa.ee.columbia.edu/matlab/pvoc/.

2. Preparatory Exercises

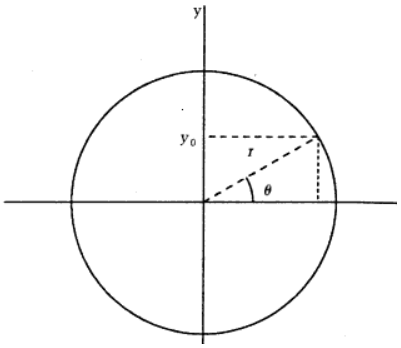
Exercise 2.1: Literature Research

Get familiar with the terms interpolation, polar coordinates, short time fourier transform, resampling.

Answer

Interpolation[1]: A method of constructing new data points within the range of a discrete set of known data points. We use linear interpolation in this phase vocoder.Linear interpolation works effectively by drawing a straight line between two neighboring samples and returning the appropriate points along that line.

Polar coordinate[2] : Radial position and angular position, also known as magnitude and phase. With polar coordinates we simply have a angular position and a constant radius. The relation between rectangular and polar coordinates:



$$r = \sqrt{x_0^2 + y_0^2} \quad \theta = \arctan(\frac{y_0}{x_0})$$

The Short-Time Fourier Transform (STFT) is a powerful general-purpose tool for audio signal processing. It defines a particularly useful class of time-frequency distributions which specify complex amplitude versus time and frequency for any signal. (See Exercise 3.1)

Resampling[3]: Change sampling rate by any rational factor

In Matlab: y = resample(x,p,q) resamples the sequence in vector x at p/q times the original sampling rate, using a polyphase filter implementation. p and q must be positive integers.

Exercise 2.2: Questions

What is a window function? Explain the constant overlap-add property for a window function w[n].

Answer:

Window function is used in digital signal processing. A window function is a mathematical function whose value is zero outside some chosen interval.
For example, A rectangular window is a window function whose value is constant over certain interval and remains zero outside the interval.

Matlab Code:

```
function rectangularwindow(w)      %Rectangular window
win = rectwin(w);                  %Defining the value
figure                             %Opens figure window
plot(win)                          %Plotting of window function
title('Rectangular Window')        %Adding title to plot
xlabel('n')                        %Labelling X-axis
ylabel('w(n)')                     %Labelling Y-axis
end
```

```
rectangularwindow(100);
```

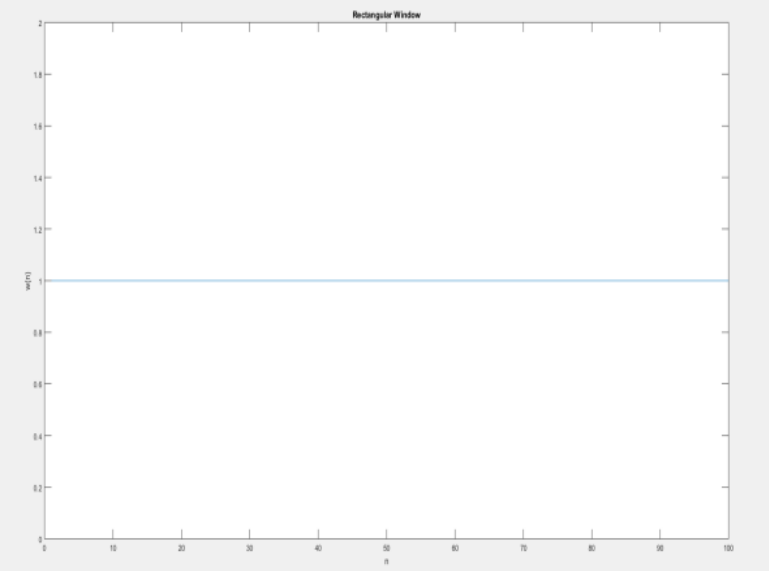


Figure 2.2.1:Rectangle window function.

Constant overlap − add property[2]

The STFT can be expressed by

$$X_m(\omega) = \sum_{m=-\infty}^{\infty} x(n)w(n - \mathrm{mR})e^{-j\omega n}$$

Where $X_m(\omega)$ stands for the DTFT of windowed data centered about time mR. If a window function w[n] fulfil constant overlap-add property,

$$\text{i.e. } \sum_{m=-\infty}^{\infty} w(n - \text{mR}) = 1\text{(R-hop size), then } X_m(\omega) = X(\omega).$$

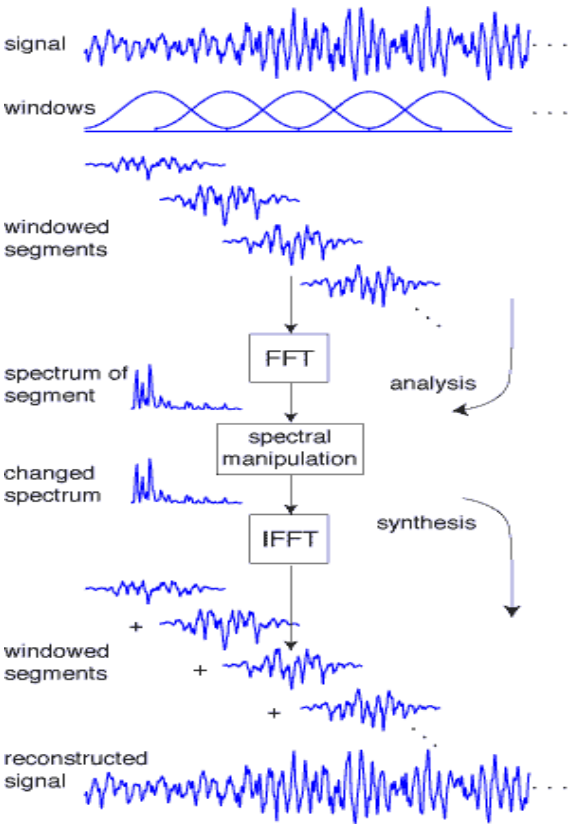
$$\text{Inverse STFT: } x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\sum_{m} X_m(\omega)e^{j\varpi n}d\omega = \sum_{m=-\infty}^{\infty} x_m(n).$$

How to generate a Hanning Window using Matlab[4]?

Hanning Window: win =0.5 + 0.5 cos [nπ/ (m+1)], for values of n between - m and +m, giving 2m + 1 sample points in total. The corresponding command function in Matlab is: win=hanning(n).

What is the general process of STFT in the phase vocoder[5]?

Phase Vocoder uses short-time Fourier transform (STFT) signal processor as an analysis/synthesis method that begins by windowing a signal into short segments. The FFT is applied to each segment separately and the resulting spectral snapshot can be manipulated in a variety of ways. After the desired spectral changes, the resynthesis is handled by the inverse FFT to return each segment to the time domain. The modified segments are then summed. For the special case where no spectral manipulations are made, the output of the STFT is identical to the input.



3. Experiments with Results

Exercise 3.1: Short-Time-Fourier Transformation

Write a function stft which performs short-time-fourier transform.

1. Generate a framing window "*win*" before calculating the STFT for a given window length *w*. Here Hanning window is taken as our framing window.

Matlab Code:

```
win= hanning(w)'; (window length:w)
```

2. Multiply the input sound signal with the series of the window function with a hop size *h* to get a successive windowed signal, and then perform an *f* -point's fast fourier transform on each windowed segment. The time domain representation of the input sound signal has now been transformed into time-frequency domain representation. Here the window size *w* equals the FFT length *f* .

Matlab Code:

```
function d = stft(x, f, w, h)     %define stft function
                                  %Sampled input signal-x
                                  %FFT length-f
                                  %Window size-n
                                  %Hop size-h
s = length(x);                    % length of input signal
d= zeros(w,floor((s-w)/h));       % pre-allocate output matrix d
win= hanning(w)';                 % generate hanning window
c = 1;                            % set a column index
for  b = 0:h:(s-w)
     u = win.*x((b+1):(b+f));     % get windowed segments
     t = fft(u);                  % perform FFT
     d(:,c) = t';
     c = c+1;
end;
```

Exercise 3.2: Phase Vocoder Sampling

Phase vocoder sampling builds a modified spectrogram array by re-sampling the original array at a fractional time values, interpolating the magnitudes and fixing-up the per-step phase as it goes along the new time path. From the STFT of Exercise 3.1, STFT matrix d was generated in forms of 'specgram', whose columns indicate the time and rows indicate frequency.

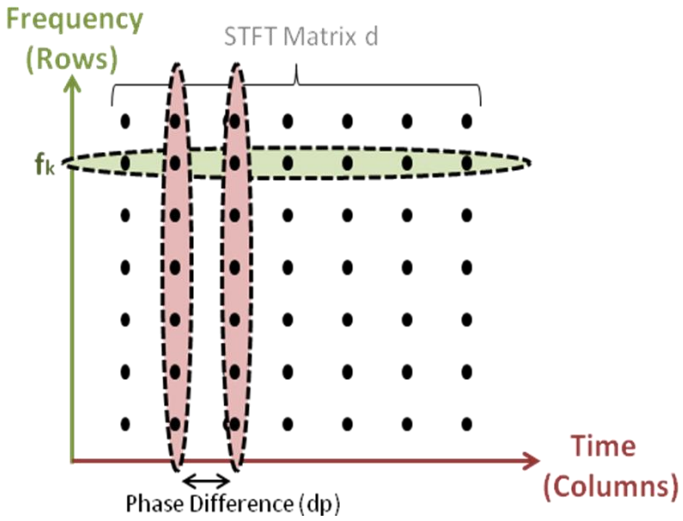


Figure 3.2.1 :Representation of STFT Matix d.

Step 1: take new samples according to the new time path and do linear magnitude interpolation.

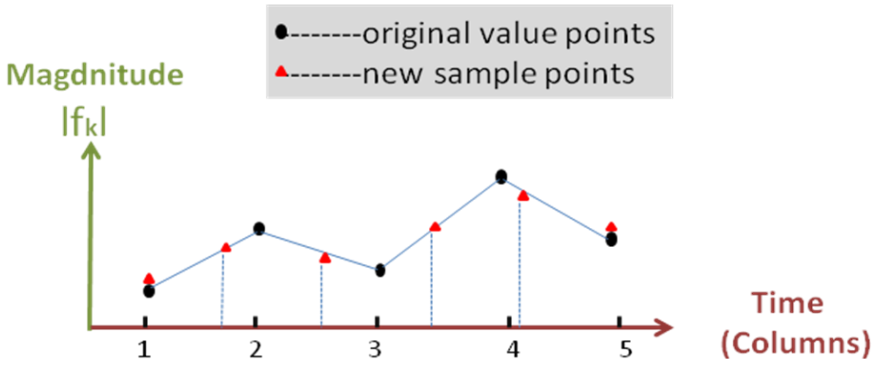


Figure 3.2.2 :Linear magnitude interpolation for frequency bin k.

In the Figure 3.2.2, the time ratio r = 0.8 was taken as an example, which means the speed of audio signal will be 0.8 times slower than the original one . The new sample points then follow the new time path (t): 0, 0.8, 1.6, 2.4, 3.2, 4, ------

Step 2: Calculate the phase difference of the original points in time t and t+1, then use this as the per-step phase difference for the new sample points.

Step 3: Save new sample points in polar coordinate with magnitude and phase information to generate the modified STFT Matrix C.

Write a Matlab function pvsample to perform phase vocoder sampling.

In the code below, d is an STFT array, of the form generated by 'specgram'. t is a vector of (real) time-samples, which specifies a path through the time-base defined by the columns of d. For each value of t, the spectral magnitudes in the columns of d are interpolated, and the phase difference between the successive columns of d is calculated; a new column is created in the output array c that preserves this per-step phase advance in each bin.

Matlab Code:

```
function c = pvsample(d, t, hop)
[rows,cols] = size(d);
ph = angle(d(:,1));          %phase of first frame as start accumulator
c= zeros(rows, length(t)); %pre-allocate the new output array c
ocol = 1;
for tt = t
  dcols = d(:,floor(tt)+[1 2]); %take new sample points
  tf = tt - floor(tt);              %magnitude interpolation
  dmag=(1-tf)*abs(dcols(:,1))+tf*abs(dcols(:,2));
  dp = angle(dcols(:,2)) – angle(dcols(:,1));  %calculate phase difference
  c(:,ocol) = dmag .* exp(j*ph);                %save columns in polar coordinates
  ph = ph + dp;                                 %Cumulate phase, ready for next frame
  col=ocol+1;
end
```

Exercise 3.3: Inverse Short-Time-Fourier-Transform

After executing pvsample in exercise 3.2, we get a complex matrix c, which is the modified STFT matrix in time-frequency domain. IFFT needs to be performed on every column. Afterwards the signal is multiplied by a hanning window and added-up for reconstruction.

Matlab Code:

```
function x = istft(c, f, w, h); %define inverse  stft function
                                 %c---Modified STFT Matrix
                                %f--- FFT length
                                 %w--Window size
                                 %h---Hop size
[rows,cols] = size(c);          %get the size of modified signal d
x = zeros(1, (cols-1)*h+w);   %pre- allocate output vector x
win = hanning(w)';               %generate hanning window
for b = 0:h:(h*(cols-1))
  px = real(ifft(c(:,1+b/h)'));   %perform inverse STFT
  x((b+1):(b+w)) = x((b+1):(b+w))+px.*win;
end
```

Exercise 3.4: Time scaling and pitch shifting effect based on phase vocoder

Define a function PVOC which combines all 3 steps:
STFT ⇨ pvsample⇨ ISTFT
to implement a complete phase vocoder. It should contain 3 parameters given by the user: x is an input sound signal, f is the FFT size, r is the time ratio compared to the original time.

Matlab Code:

```
function y = pvoc(x, r, f)  % define stft function
                            % Sampled input signal-x
                            % fft length-n
                            % Time ration-r
hop = f/4;                  % set hop size for 25% window length
X =stft(x', f, f, hop);     % perform STFT
[rows, cols] = size(X);
t = 0:r:(cols-2);           % calculate the new time path
                            % Have to stay two cols off end because (a) counting from zero
                            % (b) need col n and col n+1 to interpolate
X2 = pvsample(X, t, hop);   % perform the phase vocoder sampling
y = istft(X2, f, f, hop)';          % perform ISTFT
end
```

Exercise 3.5: Time scaling and pitch shifting effect based on phase vocoder

Use the example code below and try out parameter configurations that speed-up/slow-down and increase/decrease the pitch of the given example files.

a) Time scaling: Slowing down speed by factor 0.5

Matlab Code:

```
[x,Fs]=audioread('guitar.wav'); % Read the Audio File
y = pvoc(x,0.5,1024);           % Time scaling of the Signal using pvoc Function
soundsc(y,Fs);                  % Listen to the Time scaled audio file
figure(1);                      % Opens the figure of  input Signal
plot(x);                        % Plot of the input Signal
grid on;                        % Turns the gird ON
xlabel('time(s)');              % Labelling X-axis
ylabel('Amplitude');            % Labelling Y-axis
title('Original Input Signal')  % Adding title to the plot
figure(2);                      % Opens the figure of  Time scaled Signal
plot(y);                        % Plot of the output time scaled Signal
grid on;                        % Turns the gird ON
xlabel('time(s)');              % Labelling X-axis
ylabel('Amplitude');            % Labelling Y-axis
title('Time scaled signal by a factor of 0.5')          % Adding title to the plot
```

The time scaling function helps in speeding up or slowing down the signal with the amount of factor value. Less than the value of 1 increases the length of the signal and more than the factor value 1 decreases the length of the signal.

Here in this program we read the audio file ‘guitar.wav’ and apply the time scaling on this audio file. The time scaling is done by the pvoc function which is being called in this program file.

The output waveform of ‘guitar.wav’ is shown in the following two figures in which 3.5.1 represents the Original waveform without time scaling and 3.5.2 represents the waveform with time scaling factor of 0.5. Here we can observe the changes in the two waveforms clearly. The figure 3.5.2 has the length exactly double of the original waveform from figure 3.5.1

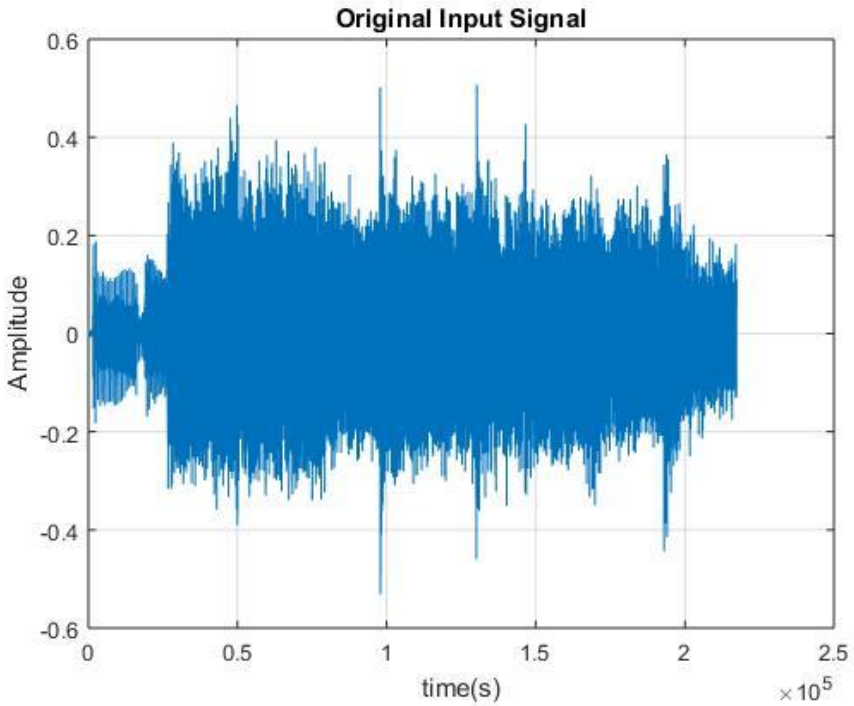


Figure 3.5.1 :Original input audio signal.

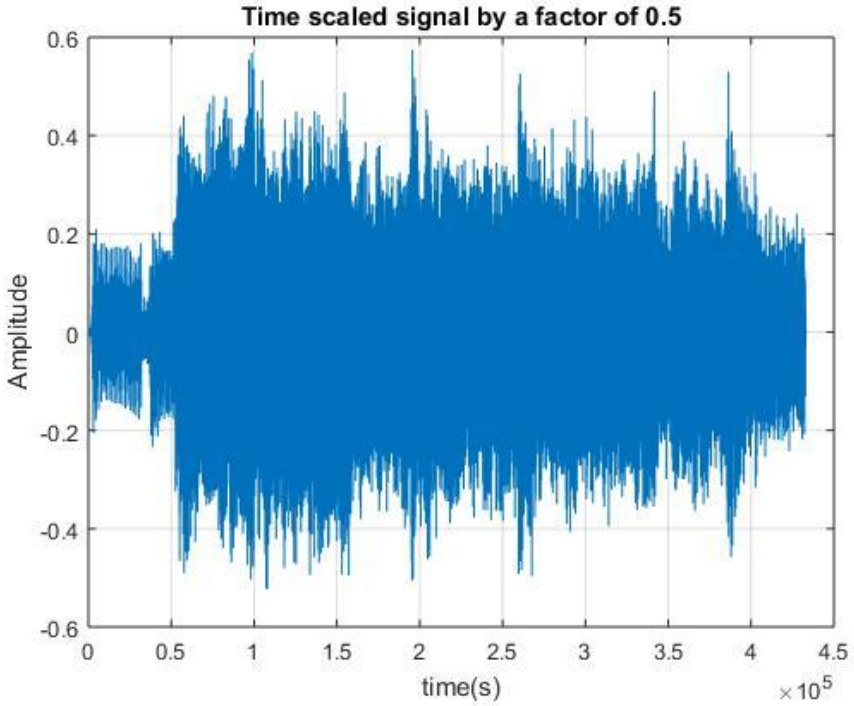


Figure 3.5.2 :'guitar.wav' time scaled by a factor of 0.5.

b) Pitch shifting: Hint: firstly, use matlab command *y=resample(x,p,q)* to generate a pitch shift by resampling signal *p/q* times the original sample rate, which means *y* is *p/q* times the length of signal *x*. Then use the time-scale modification from above to readjust the time scale to the original length. Here, *p=1* and *q=2*, hence *y* is half the length of *x* and must be stretched (slowed down) to the original length applying factor ½.

Matlab Code:

```
[x,fs]=audioread('guitar.wav');          % Read the Audio File
y=resample(x,1,2);                       % Pitch shifting by resampling the signal
z=pvoc(y,1/2,1024);                      % Time scaling of the Signal using pvoc Function
soundsc(z,fs)                            % Listen to the pitch shifted audio file
% Plot of input signal
figure(1);
plot(x)
grid on                                  % Turns the gird ON
title('Input')                           % Adding title to the plot
% Plot of pitch changed signal
figure(2);
plot(z)
grid on                                  % Turns the gird ON
title('Pitch shifting with factor 1/2')  % Adding title to the plot
```

The Pitch shifting function helps in increasing up or decreasing down the signal pitch with the amount of factor value. Less than the value of 1 for p/q increases the pitch of the signal and more than the factor value 1 lowers the pitch of the signal.

Here in this program we read the audio file ‘guitar.wav’ and apply the pitch sifting on this audio file. The pitch shifting is done by the resample function, in order to make no changes in time shift for original and 0.5 factor, the pvoc function is being called with a factor of 0.5 in this program file.

The output waveform of ‘guitar.wav’ is shown in the following two figures in which 3.5.3 represents the Original waveform without Pitch shifting and 3.5.4 represents the waveform with Pitch shifting factor of 0.5. Here we can observe the changes in the two waveforms clearly. The figure 3.5.4 has the pitch exactly double of the original waveform from figure 3.5.3

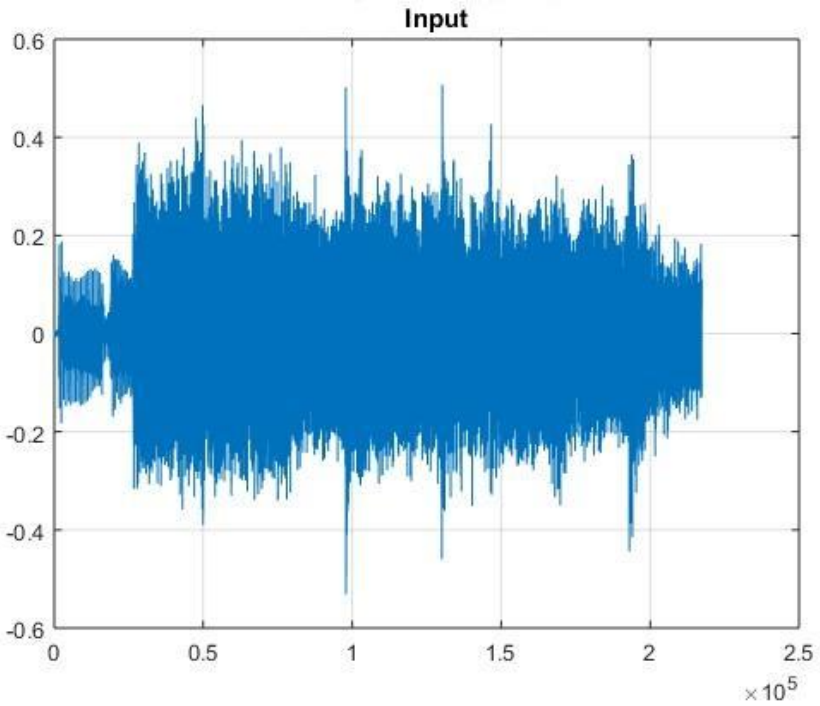


Figure 3.5.3 :Original input audio signal.

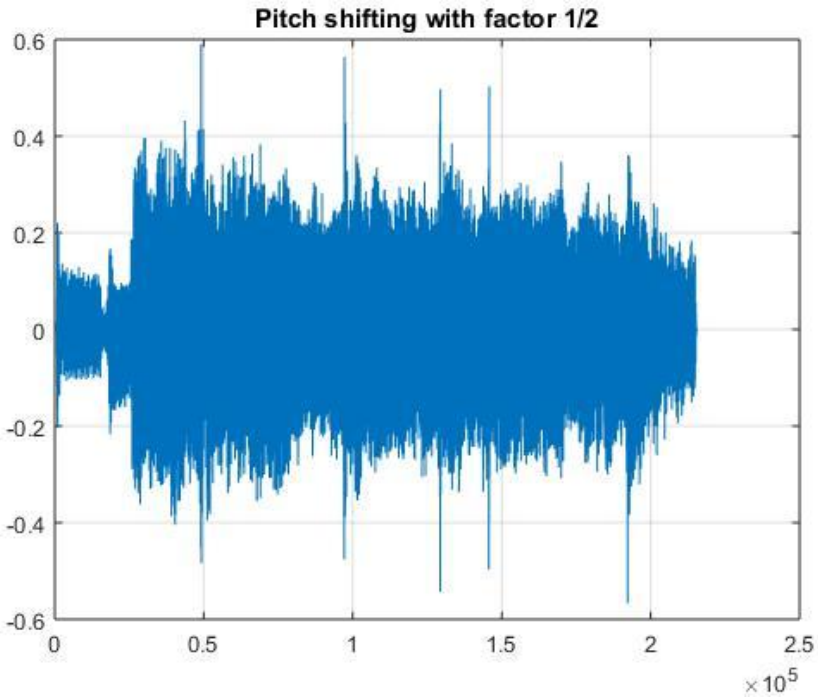


Figure 3.5.4 : 'guitar.wav' pitch shifted by a factor of 0.5.

Exercise 3.6: Vocoder Completion and Stereo Processing

Simplify the usage of the example code.

a) **Implement a matlab function adjustspeed(file, factor) with two parameters, one for the filename and the second one for the speed factor. A factor of 2 means twice as fast and one of 0.5 means half of the original speed. Test your function with the files named guitar.wav and voice.wav.**

Matlab Code:

```
function adjustspeed(filename, speed)  % Function defined to adjust the speed of an audio file
[x,fs] = audioread(filename);          % Reading audio file into matlab
y=pvoc(x,speed,1024);                  % Defining the pvoc function
time=(1:length(y))/fs;
%plots of speed variations
figure(1);
plot(time,y);
grid on;                               % Turning the grid ON
xlabel('time(s)')                      % Labelling X-axis
ylabel('Amplitude')                    % Labelling Y-axis
title('Original Speed')                % Adding title to the plot
soundsc(y,fs)                          % Listen to the audio file
end


% Execution of file 'voice.wav' in the function.
% Given values of adjustspeed  like {1,0.5,2}.
adjustspeed('voice.wav',1);
```

The adjustspeed function helps in speeding up or slowing down the signal with the amount of factor value. The input variable ‘filename’ refers to audio file and ‘speed’ variable is the factor for changing the speed of the signal, less than the value of 1 increases the length of the signal and more than the speed value 1 decreases the length of the signal.

Here in this program we read the audio file ‘voice.wav’ and ‘guitar.wav’ and apply the time scaling on this audio file. The time scaling is done by the pvoc function which is being called in this program file.

The output waveform of ‘voice.wav’ is shown in the following two figures in which 3.6.1 represents the Original waveform without time scaling and 3.6.2 represents the waveform with time scaling factor of 0.5. Here we can observe the changes in the two waveforms clearly. The figure 3.6.2 has the length exactly double of the original waveform from figure 3.6.1. 3.6.3 represents the waveform with time scaling factor of 2. Here we can observe the changes in the 3.6.3 and 3.6.1, figure 3.6.3 has only half the length of the original waveform.

The output waveform of ‘guitar.wav’ is shown in the following two figures in which 3.6.4 represents the Original waveform without time scaling and 3.6.5 represents the waveform with time scaling factor of 0.5. Here we can observe the changes in the two waveforms clearly. The figure 3.6.5 has the length exactly double of the original waveform from figure 3.6.4. 3.6.6 represents the waveform with time scaling factor of 2. Here we can observe the changes in the 3.6.6 and 3.6.4, figure 3.6.6 has only half the length of the original waveform.

```
% Execution of file 'voice.wav' in the function.
% Given values of adjustspeed  like {1,0.5,2}.
```

Calling the function in command window, adjustspeed('voice.wav',1);

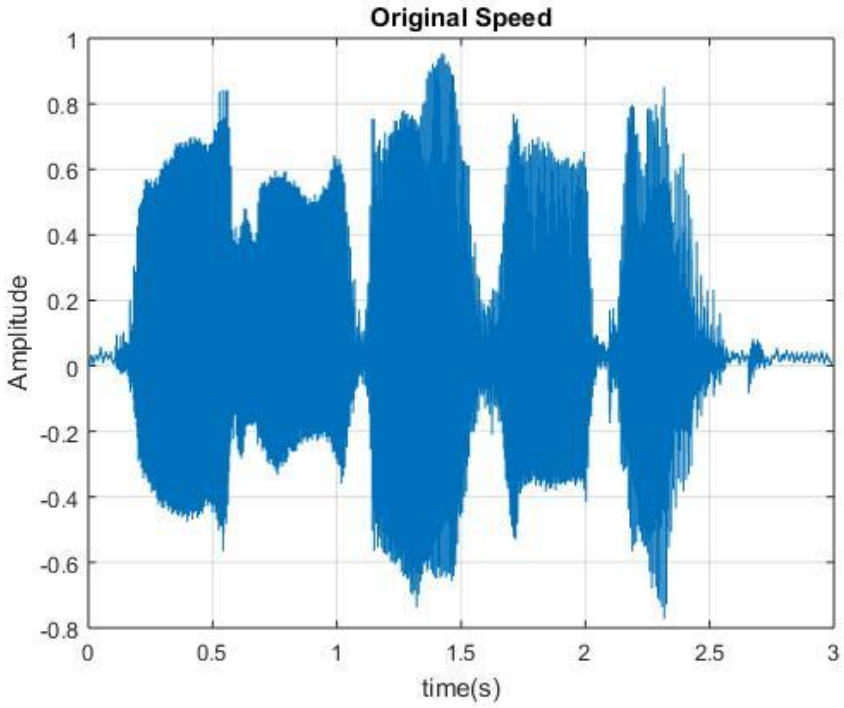


Figure 3.6.1 :Original signal speed for 'voice.wav'.

Calling the function in command window, adjustspeed('voice.wav',0.5);

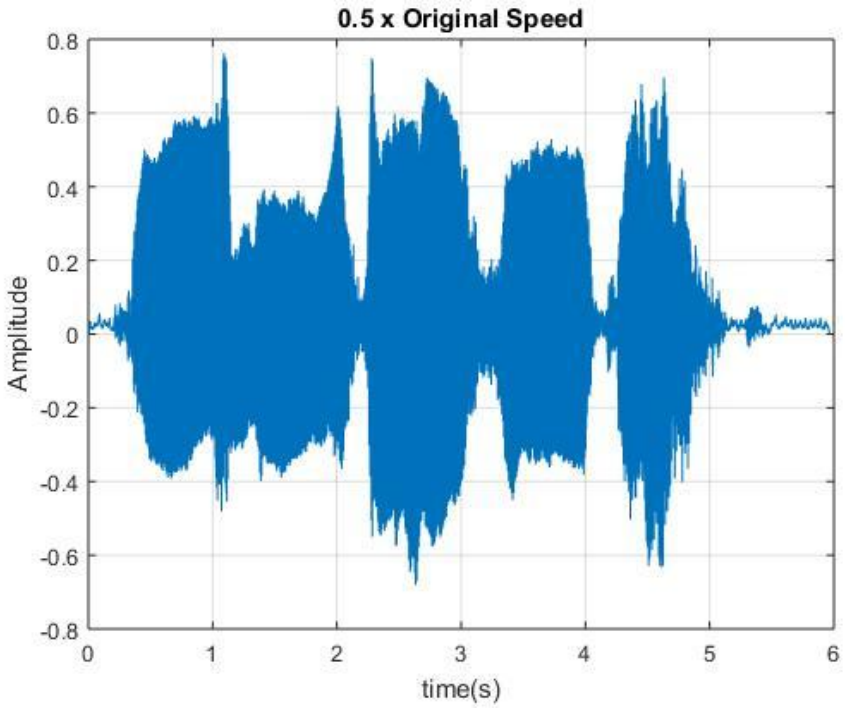


Figure 3.6.2 :0.5 x Original-reduced speed signal for 'voice.wav'.

Calling the function in command window, adjustspeed('voice.wav',2);

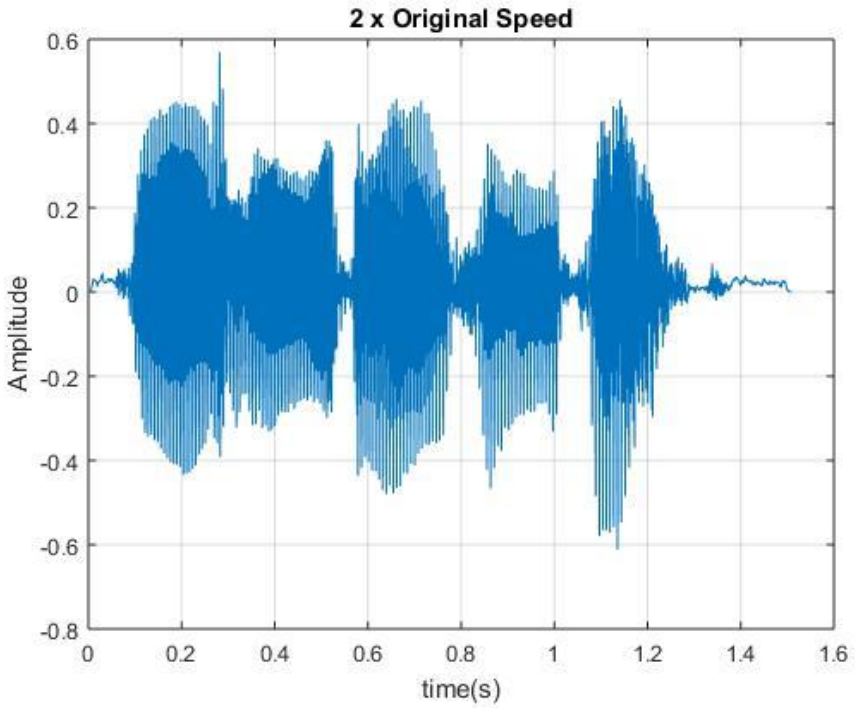


Figure 3.6.3 :2 x Original-increased speed signal for 'voice.wav'.

% Execution of file 'guitar.wav' in the function.
% Given values of adjustspeed like {1,0.5,2}.
Calling the function in command window, adjustspeed('guitar.wav',1);

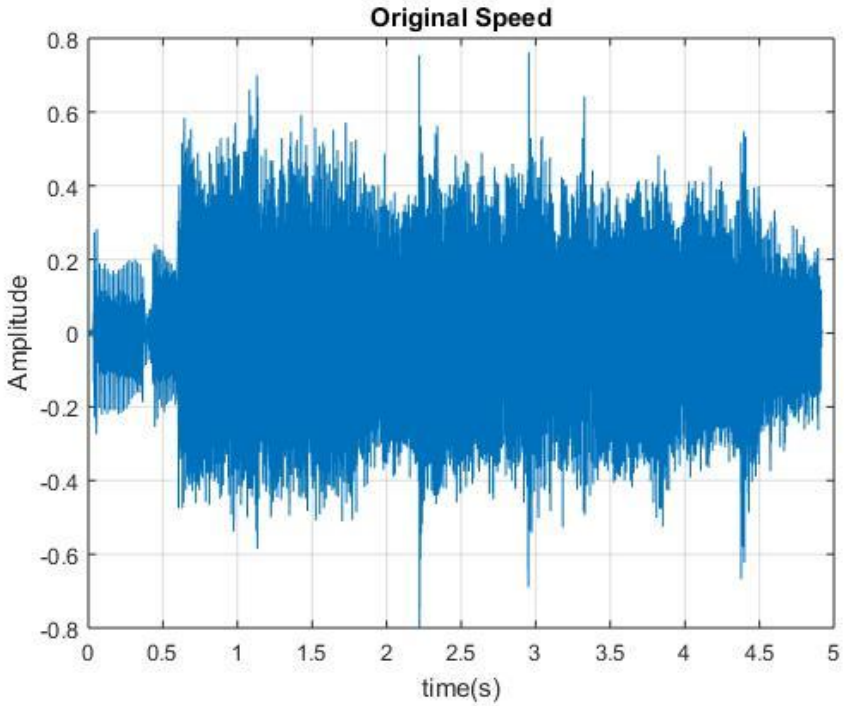


Figure 3.6.4 :Original speed signal for 'guitar.wav'.

Calling the function in command window, adjustspeed('guitar.wav',0.5);

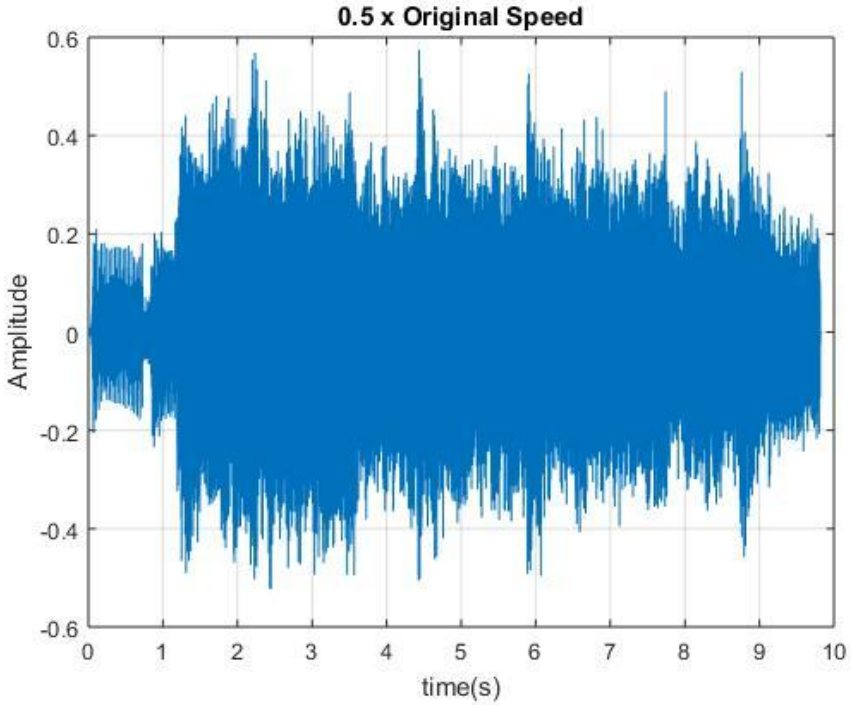


Figure 3.6.5 :0.5 x Original reduced speed signal for 'guitar.wav'.

Calling the function in command window, adjustspeed('guitar.wav',2);

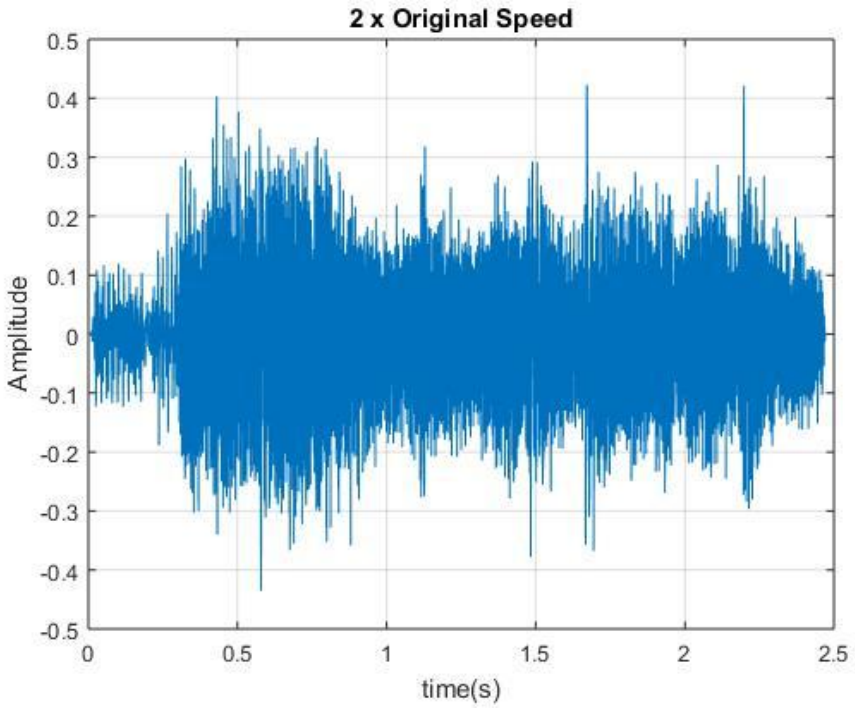


Figure 3.6.6 :2 x Original increased speed signal for 'guitar.wav'.

b) **Implement a matlab function adjustpitch(file, shift) with two parameters, one for the filename and the second one for the pitch shift. Here, a value of 1 shall mean an increase in pitch by one unit, while a value of -1 shall decrease the pitch by one unit. Further, find out how also decimal inputs like 1.5 for the shift can be implemented. Test your function with the files named guitar.wav and voice.wav.**

Matlab Code:

```
function [x,sf] = adjustpitch(file,speed)  % Function defined to adjust pitch audio file
speed = abs(speed);
speed = 2/(4*speed^2);
speed= speed * 1000;
speed = floor(speed);
[x,sf]=audioread(file);                    % Reading audio file into matlab
y=resample(x,speed,2000);                  % Pitch shifting by resampling the signal
z=pvoc(y,speed/2000,1024);                 % Defining the pvoc function
%plot of pitch varied signals
figure(1);
plot(z);
grid on;                                   % Turning the grid ON
xlabel('time(s)')                          % Labelling X-axis
ylabel('Amplitude')                        % Labelling Y-axis
title('Pitch decreased by one unit(guitar.wav)')   % Adding title to the plot
```

```
soundsc(z,sf);                                  % Listen to the audio file
end
```

The adjustpitch function helps in increasing up or lowers down the signal pitch with the amount of speed value. Less than the value of 1 for p/q increases the pitch of the signal and more than the factor value 1 lowers the pitch of the signal.

Here in this program we read the audio file ‘guitar.wav’ and ‘voice.wav’ for apply the pitch shifting on this audio file. The pitch shifting is done by the resample function and, the pvoc function is being called so that no time shifting takes place in this case..

The output waveform of ‘guitar.wav’ is shown in the following three figures in which 3.6.7 represents the pitch decreased by one unit and 3.6.8 represents the waveform with Pitch increased by one unit and 3.6.9 represents the waveform with pitch increased by 1.5 units. Here we can observe the changes in the three waveforms clearly.

The output waveform of ‘voice.wav’ is shown in the following three figures in which 3.6.10 represents the pitch decreased by one unit and 3.6.11 represents the waveform with Pitch increased by one unit and 3.6.12 represents the waveform with pitch increased by 1.5 units. Here we can observe the changes in the three waveforms clearly.

```
% Execution of file 'guitar.wav' in the function.
% Given values for adjustpitch like {-1,1,1.5}.
```
Calling the function in command window, adjustpitch('guitar.wav',-1);

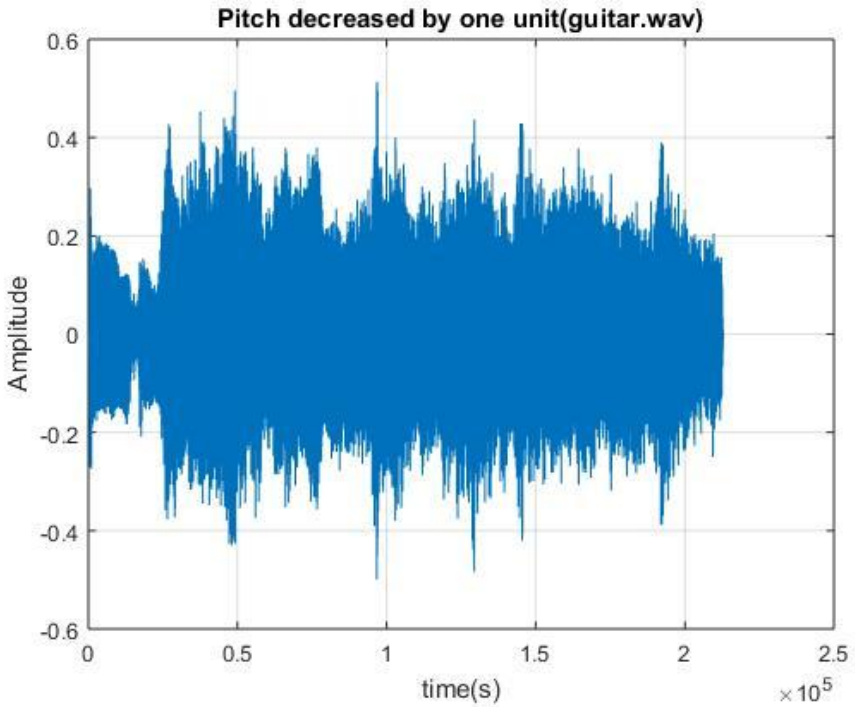


Figure 3.6.7 :Decreased pitch signal by one unit for 'guitar.wav'.

Calling the function in command window, adjustpitch('guitar.wav',1);

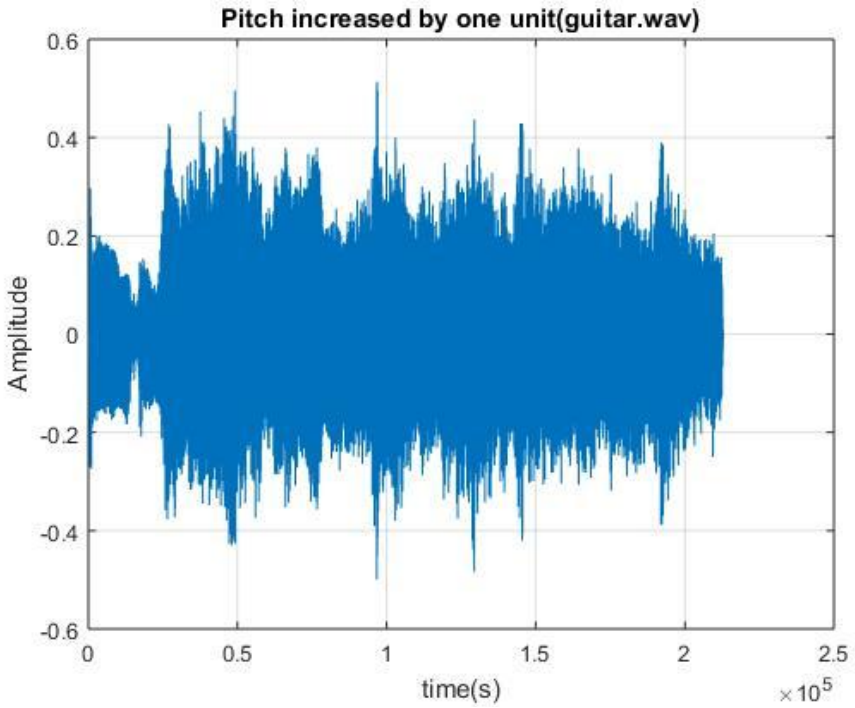


Figure 3.6.8 :Increased pitch signal by one unit for 'guitar.wav'.

Calling the function in command window, adjustpitch('guitar.wav',1.5);

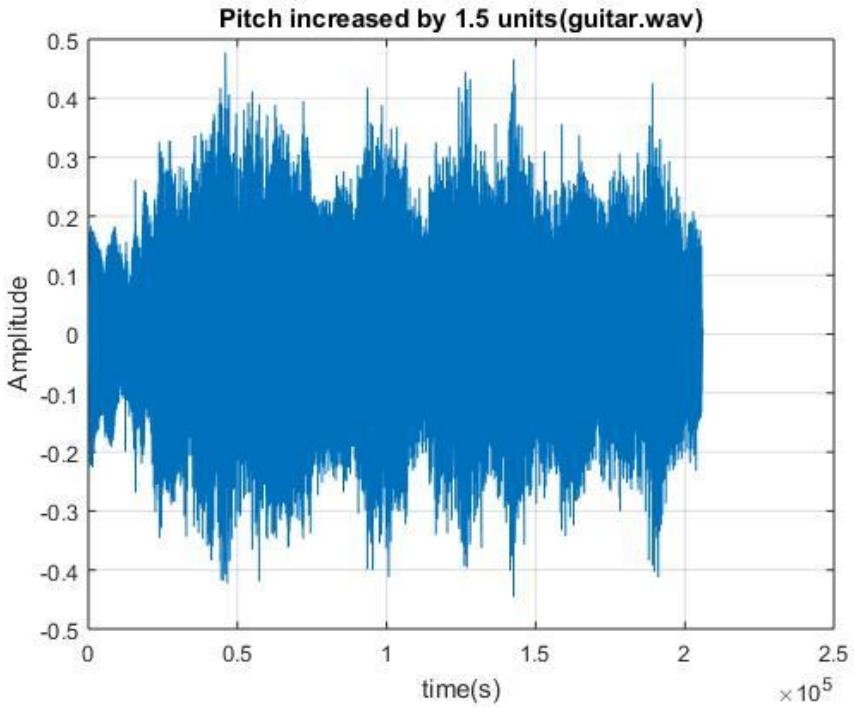


Figure 3.6.9 :Increased pitch signal by 1.5 unit for 'guitar.wav'.

% Execution of file 'voice.wav' in the function.
% Given values for adjustpitch like {-1,1,1.5}.
Calling the function in command window, adjustpitch('voice.wav',-1);

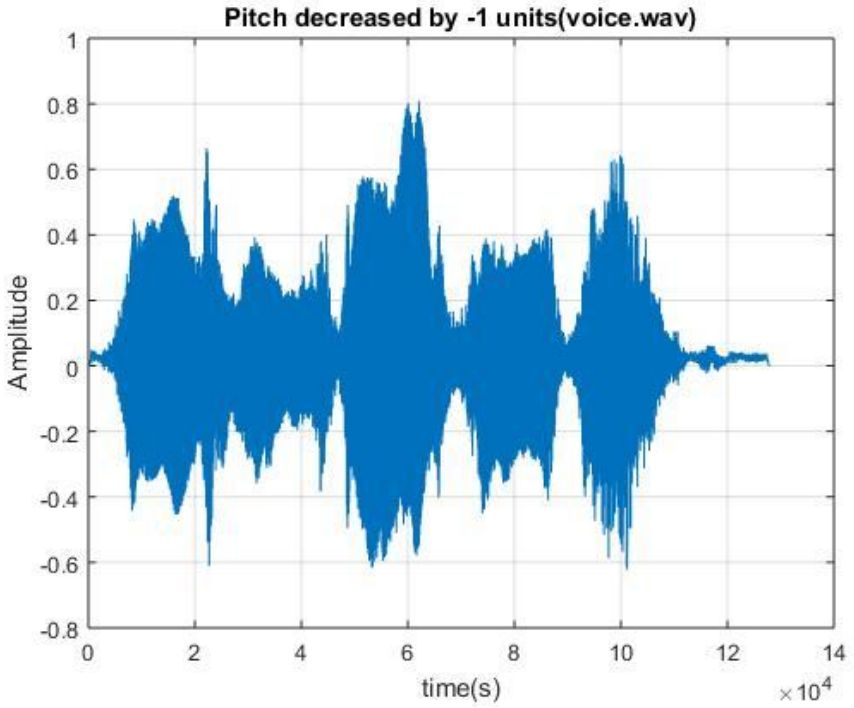


Figure 3.6.10 :Decreased pitch by one unit for 'voice.wav'.

Calling the function in command window, adjustpitch('voice.wav',1);

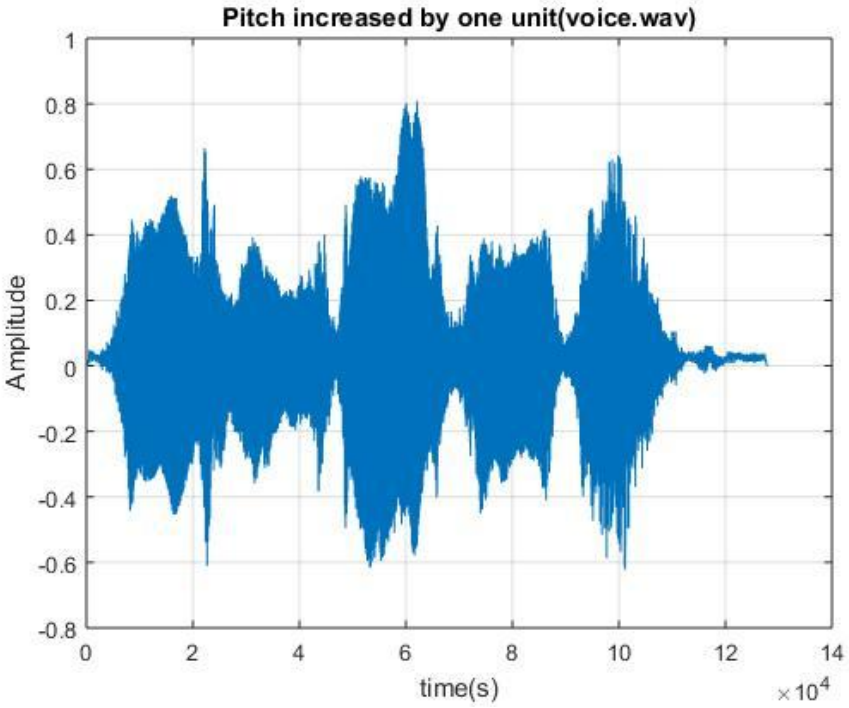


Figure 3.6.11 :Increased pitch signal by one unit for 'voice.wav'.

Calling the function in command window, adjustpitch('voice.wav',1.5);

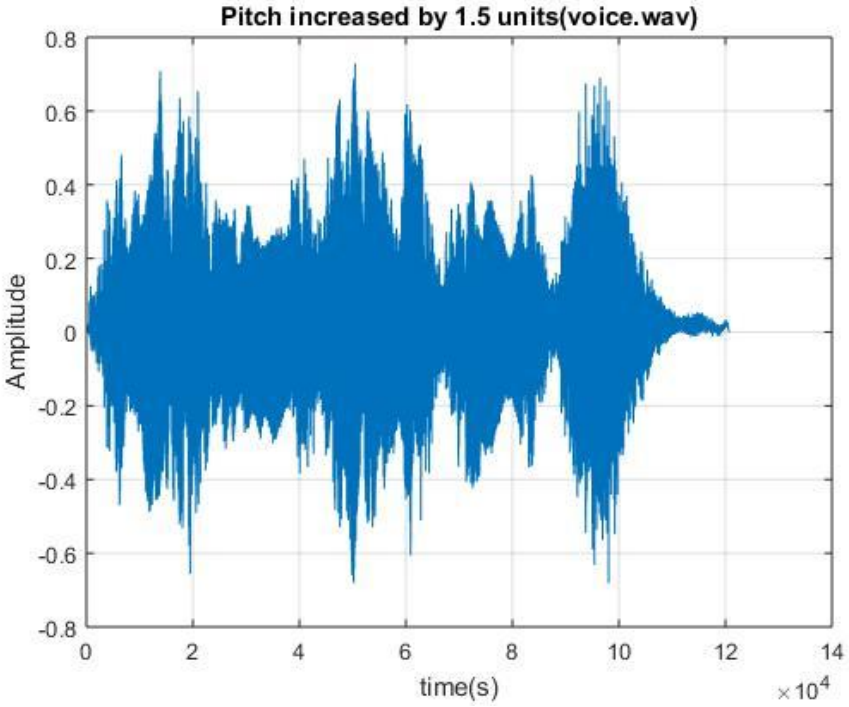


Figure 3.6.12 :Increased pitch signal by 1.5 unit for 'voice.wav'.

c) **If you call the functions from the previous two tasks with fahrstuhl.wav and jodeln.wav, the generated audio signal will sound incorrect in terms of the underlied speed factor or pitch shift. The reason is an incorrect processing of the signal *x* in the resample function, as it expects a mono recorded signal. However, fahrstuhl.wav and jodeln.wav were recorded in a stereo manner. Check out how the signal *x* looks like for stereo and for mono sounds and change the two functions from a) and b) in a way that is also possible to process stereo inputs.**

Matlab Code:

```
% Function defined to adjust the speed of a stereo file
function adjustspeedstereo(file,speed)
[input,sf] = audioread(file);              % Read the audio file
x = (input(:,1)+input(:,2))/2;
n = 40000;
y = pvoc(x,speed,n);
soundsc(y,sf);
% plot of the input signal
figure(1)
plot(x)
title('Original signal (fahrstuhl.wav)')
grid on;
% plot of the output signal
figure(2)
plot(y)
title('0.5x Original speed')
grid on;
xlabel ('time(s)')
ylabel ('amplitude')
end
```

The adjustspeedstereo function helps in speeding up or slowing down the signal with the amount of factor value. The input variable ‘filename’ refers to audio file and ‘speed’ variable is the factor for changing the speed of the signal, less than the value of 1 increases the length of the signal and more than the speed value 1 decreases the length of the signal.

Here in this program we read the audio file ‘fahrstuhl.wav’ and ‘jodeln.wav’ and apply the time scaling on this audio file. The time scaling is done by the pvoc function which is being called in this program file.

The output waveform of ‘fahrstuhl.wav’ is shown in the following two figures in which 3.6.13 represents the Original waveform without time scaling and 3.6.14 represents the waveform with time scaling factor of 0.5. Here we can observe the changes in the two waveforms clearly. The figure 3.6.14 has the length exactly double of the original waveform from figure

3.6.13. 3.6.15 represents the waveform with time scaling factor of 2. Here we can observe the changes in the 3.6.15 and 3.6.13, figure 3.6.15 has only half the length of the original waveform.

The output waveform of ‘jodeln.wav’ is shown in the following two figures in which 3.6.16 represents the Original waveform without time scaling and 3.6.17 represents the waveform with time scaling factor of 0.5. Here we can observe the changes in the two waveforms clearly. The figure 3.6.17 has the length exactly double of the original waveform from figure 3.6.16. 3.6.18 represents the waveform with time scaling factor of 2. Here we can observe the changes in the 3.6.18 and 3.6.16, figure 3.6.18 has only half the length of the original waveform.

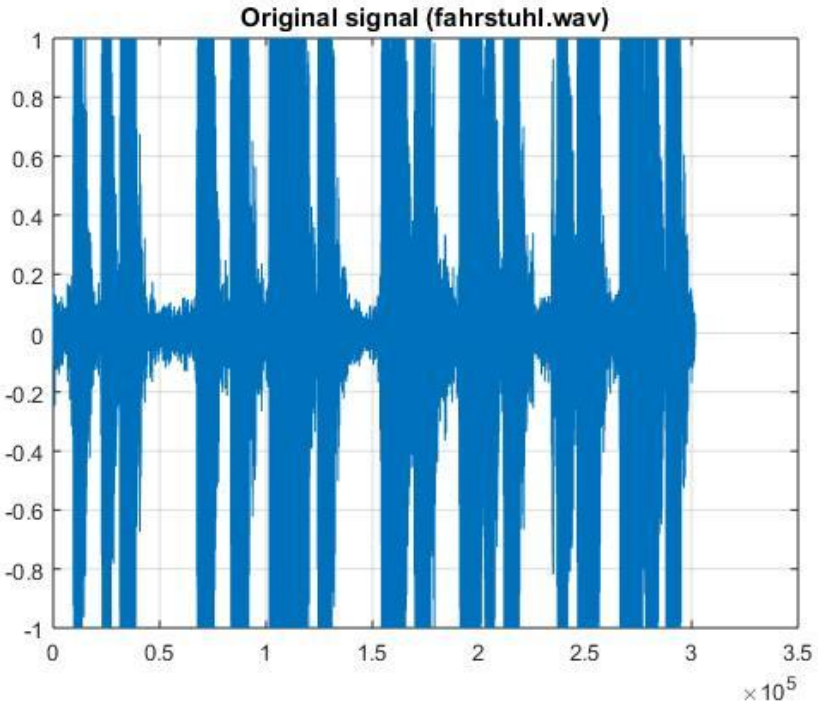


Figure 3.6.13 :Original signal for ‘fahrstuhl.wav’.

```
% Execution of file 'fahrstuhl.wav' in the function.
% Given values of speedstereo  like {0.5,2}.
```

Calling the function in command window, adjustspeedstereo('fahrstuhl.wav',0.5);

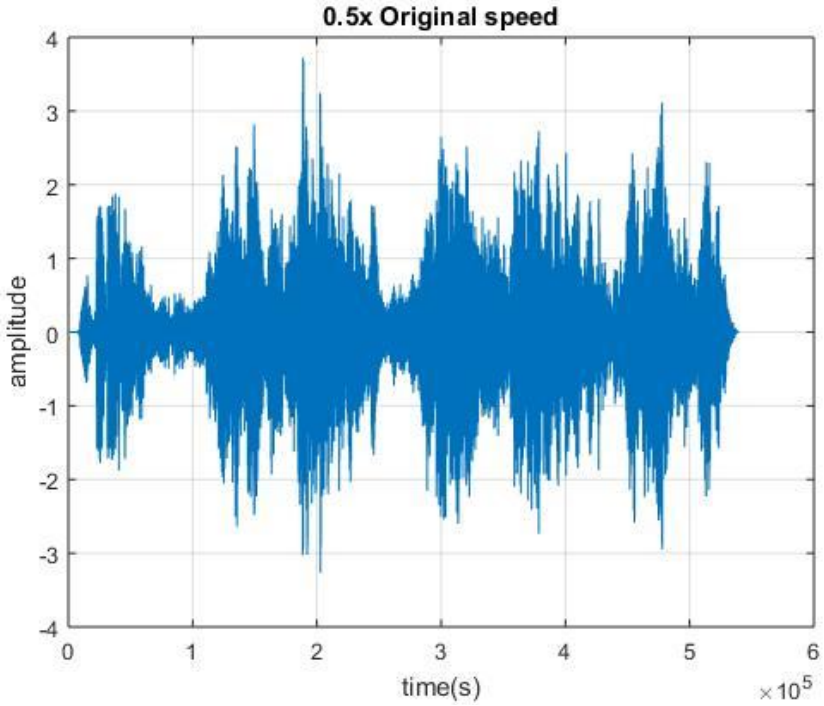


Figure 3.6.14 :0.5x Original speed signal of 'fahrstuhl.wav'.

Calling the function in command window, adjustspeedstereo('fahrstuhl.wav',2);

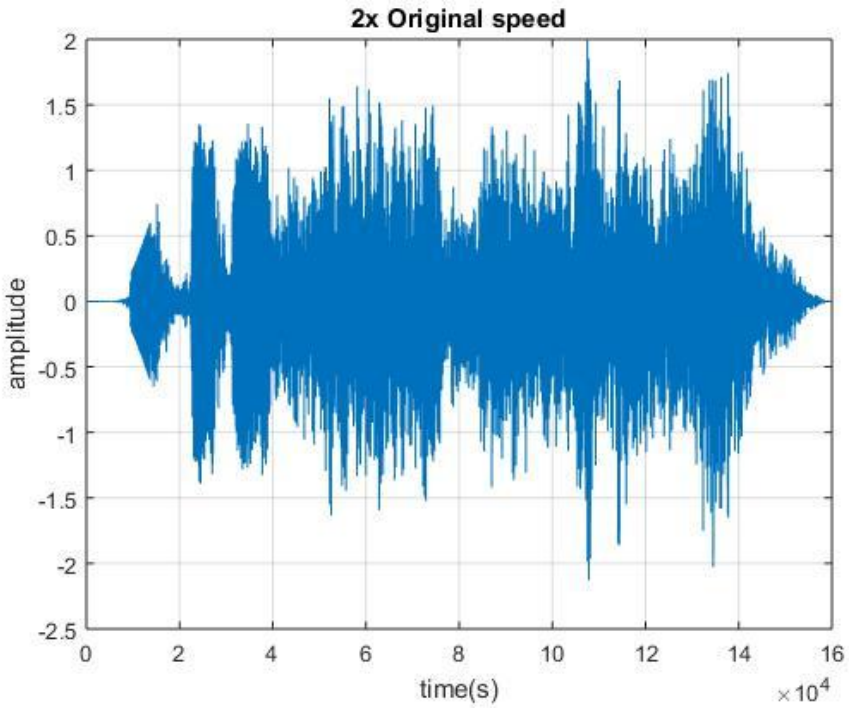


Figure 3.6.15 :2 x Original speed signal for 'fahrstuhl.wav'.

. % Execution of file 'jodeln.wav' in the function.

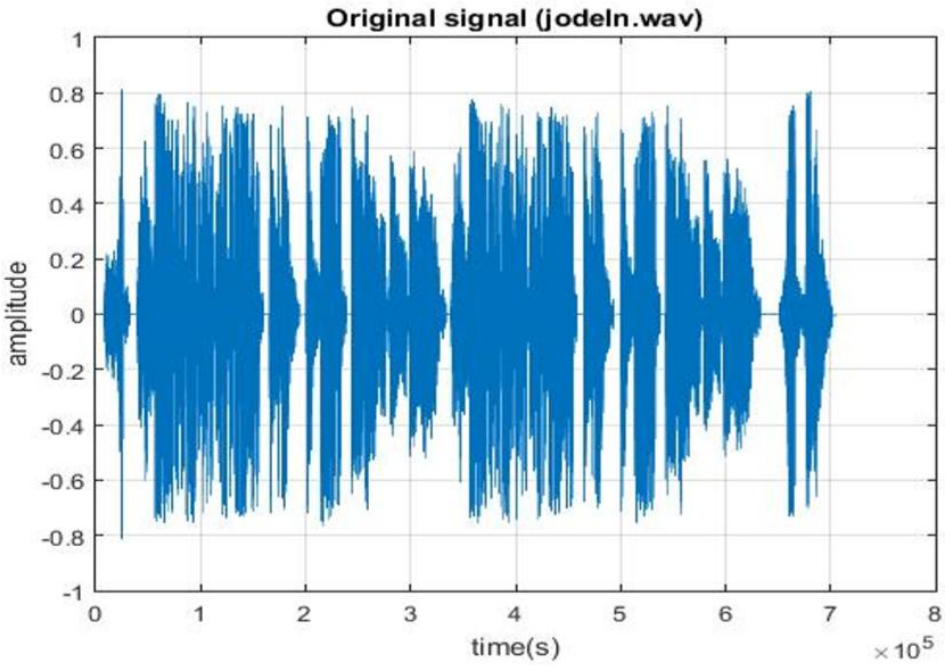


Figure 3.6.16 :Original stereosignal for 'jodeln.wav'.

```
% Execution of file 'jodeln.wav' in the function.
% Given values for adjustpitch like {0.5,2}.
```

Calling the function in command window, adjustspeedstereo('jodeln.wav',0.5);

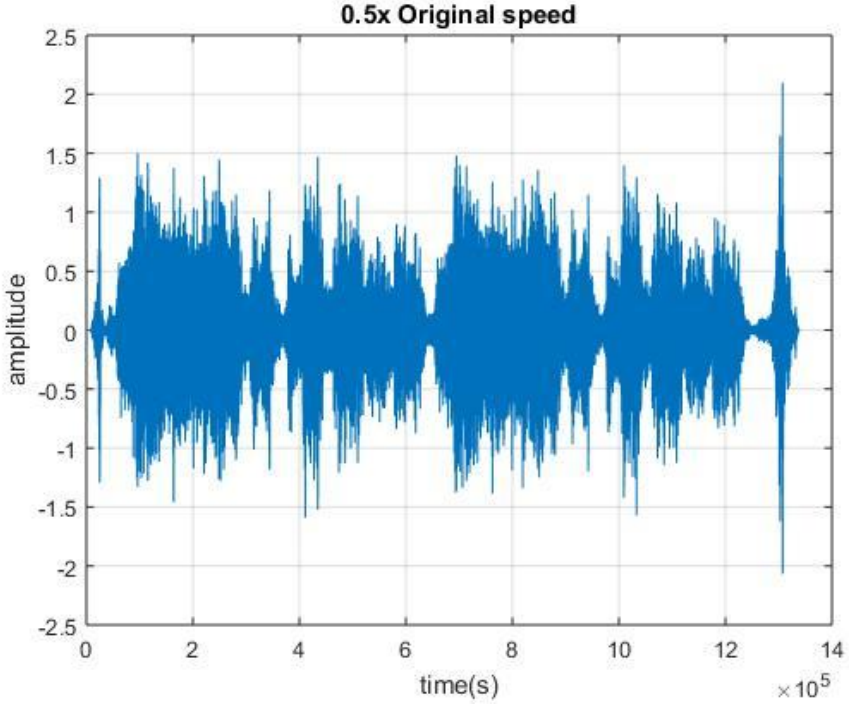


Figure 3.6.17 :0.5x speedstereo signal of 'jodeln.wav'.

Calling the function in command window, adjustspeedstereo('jodeln.wav',2);

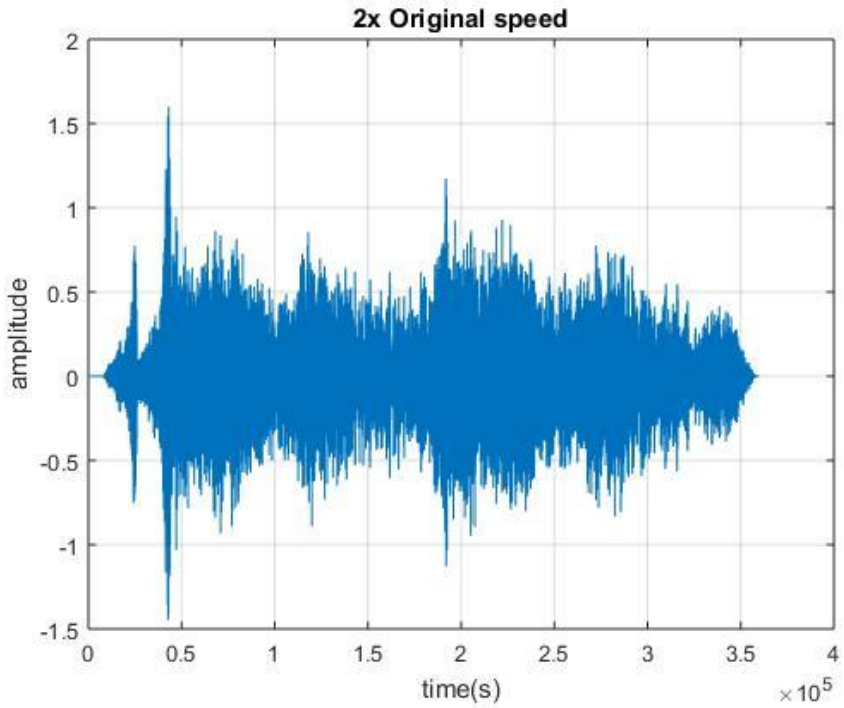


Figure 3.6.18 :2x original speedstereo signal for 'jodeln.wav'.

Matlab Code:

```
% Function defined to adjust the pitch of a stereo file
function [x,sf] = adjustpitchstereo(file,octave)
octave = abs(octave);
octave = 2/(4*octave^2);
octave = octave * 1000;
octave = floor(octave);
[x,sf] = audioread(file);          % Read the audio file
y=resample(x,octave*2,2000);
z(:,1)=pvoc(x(:,1),octave/2000,1024);
z(:,2)=pvoc(x(:,2),octave/2000,1024);
[m , n] = size(x);
soundsc(z,sf)
% plot of the output signal
figure(1);
plot(z);
grid on;
xlabel('time(s)')
ylabel('Amplitude')
title('Pitch increased by 1 unit')
end
```

The adjustpitchstereo function helps in increasing up or lowers down the signal pitch with the amount of speed value. Less than the value of 1 for p/q increases the pitch of the signal and more than the factor value 1 lowers the pitch of the signal.

Here in this program we read the audio file ‘fahrstuhl.wav’ and ‘jodeln.wav’ for apply the pitch shifting on this audio file. The pitch shifting is done by the resample function and, the pvoc function is being called so that no time shifting takes place in this case..

The output waveform of ‘fahrstuhl.wav’ is shown in the following three figures in which 3.6.19 represents the pitch decreased by one unit and 3.6.20 represents the waveform with Pitch increased by one unit and 3.6.21 represents the waveform with pitch increased by 1.5 units. Here we can observe the changes in the three waveforms clearly.

The output waveform of ‘jodeln.wav’ is shown in the following three figures in which 3.6.22 represents the pitch decreased by one unit and 3.6.23 represents the waveform with Pitch increased by one unit and 3.6.24 represents the waveform with pitch increased by 1.5 units. Here we can observe the changes in the three waveforms clearly.

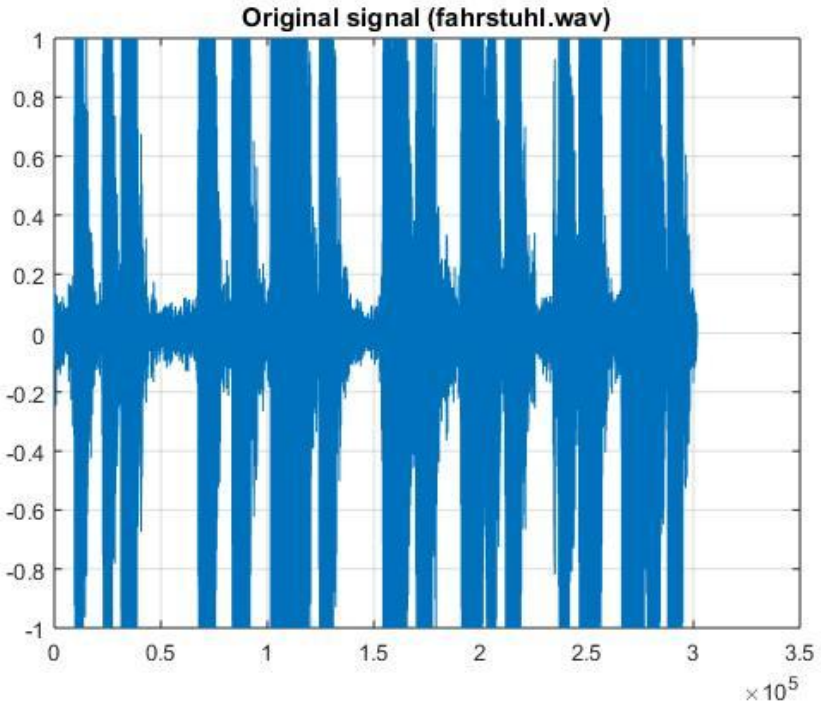


Figure 3.6.19 :Original stereosignal 'fahrstuhl.wav'.

% Execution of file 'fahrstuhl.wav' in the function.
% Given values for adjustpitch like {1,-1}.
Calling the function in command window, adjustpitchstereo('fahrstuhl.wav',1);

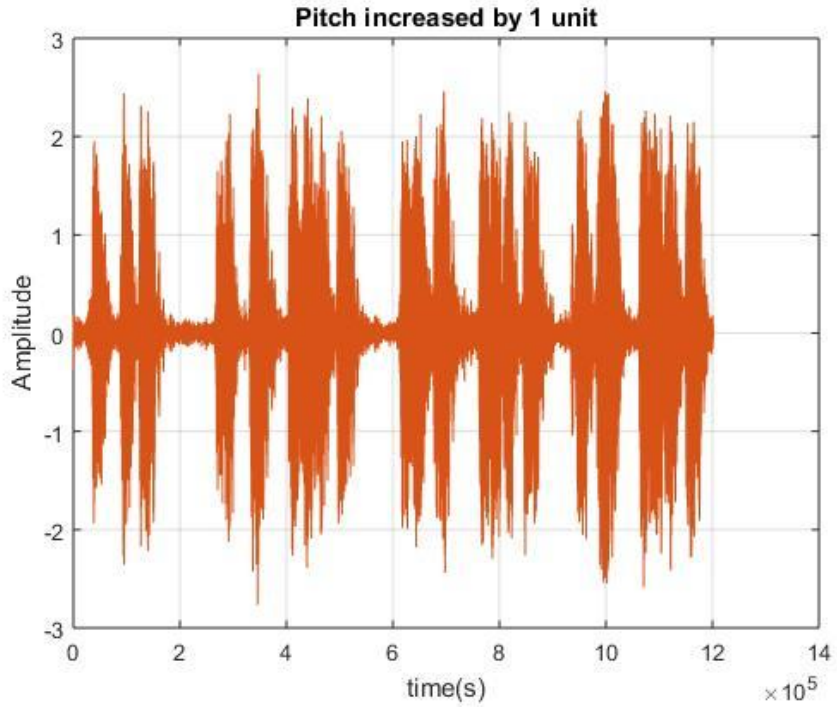


Figure 3.6.20 :Increased pitch signal by 1 unit for 'fahrstuhl.wav'.

Calling the function in command window, adjustpitchstereo('fahrstuhl.wav',-1);

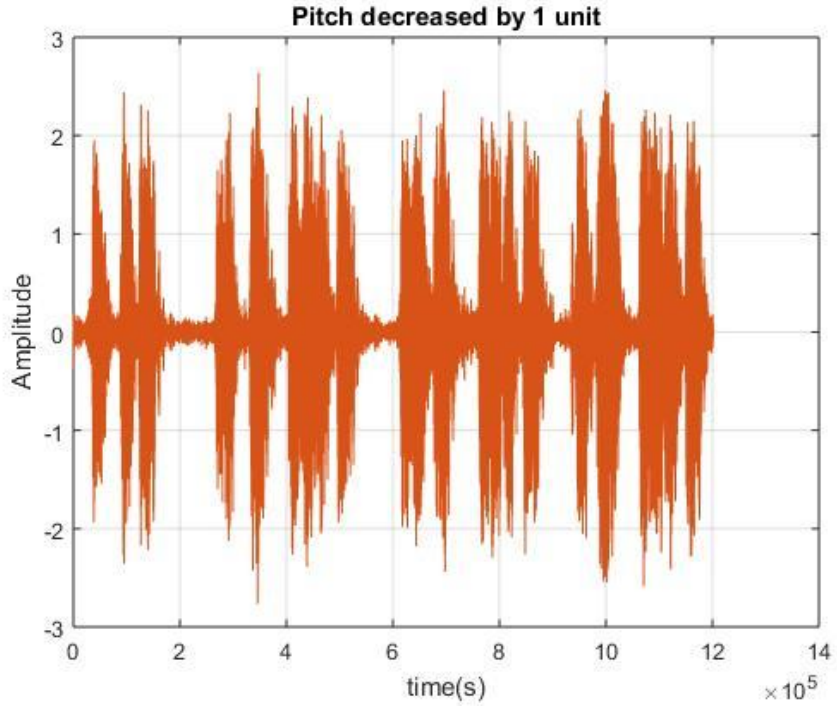


Figure 3.6.21 :Decreased pitch signal by 1 unit for 'fahrstuhl.wav'.

% Execution of file 'jodeln.wav' in the function.

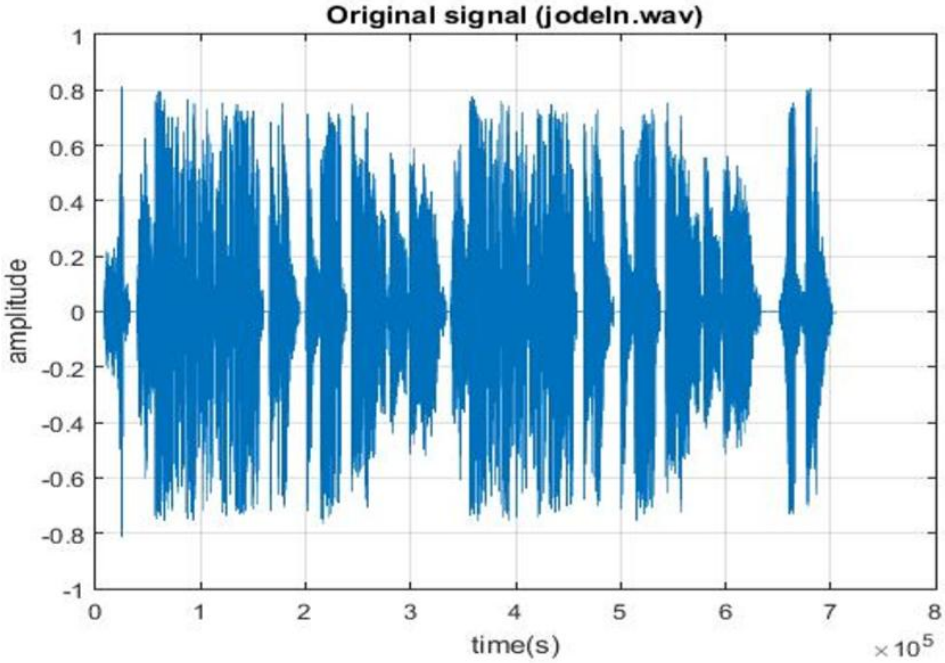


Figure 3.6.22 :Original stereosignal 'jodeln.wav'.

% Execution of file 'jodeln.wav' in the function.
% Given values for adjustpitch like {1,-1}.
Calling the function in command window, adjustpitchstereo('jodeln.wav',1);

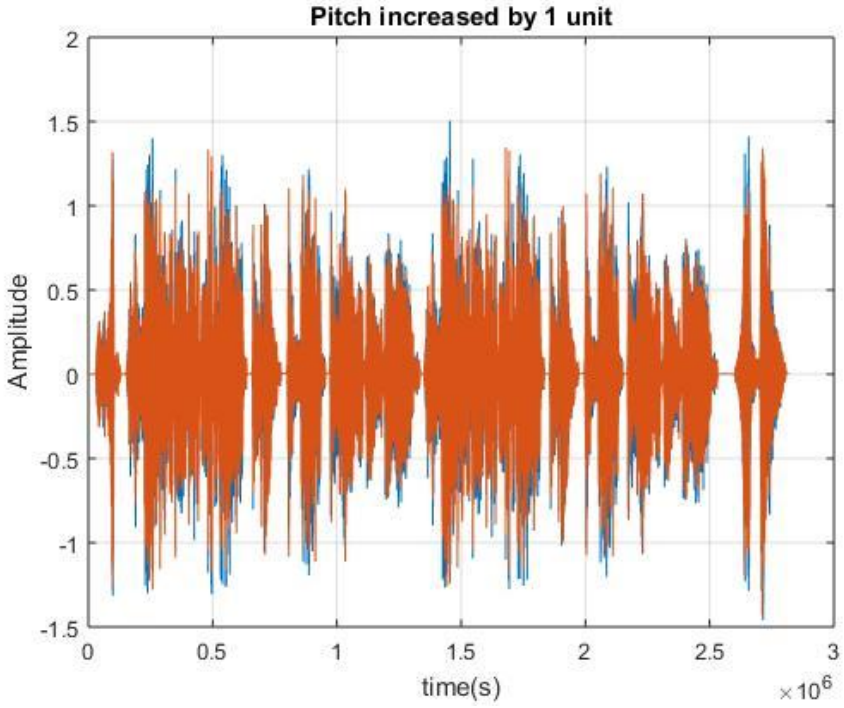


Figure 3.6.23 :Increased pitch signal by 1 unit for 'jodeln.wav'.

Calling the function in command window, adjustpitchstereo('jodeln.wav',-1);

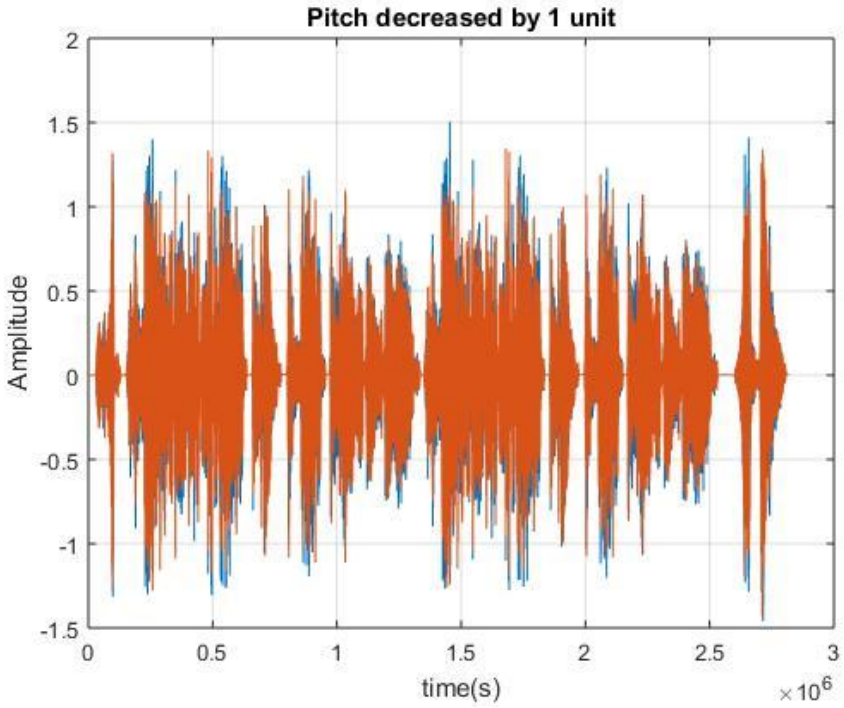


Figure 3.6.24 :Decreased pitch signal by 1 unit for 'jodeln.wav'.

4. References

[1] Julius Orion Smith, Linear Interpolation. http://www.dsprelatedcom/dspbooks/pasp/Linear_Interpolation.html

[2] Mark Dolson, The Phase Vocoder: A Tutorial, http://www.panix.com/~jens/pvoc-dolson.pa r

[3] Matlab Signal Processing Toolbox, Resampling

[4]Traditional Emplementations of a phase vocoder, Amalia De Götzen, Proceedings of the COST G-6 Conference on Digital Audio Effects(DAFX-00).

[5] Mark Dolson, The Phase Vocoder: A Tutorial, http://www.panix.com/~jens/pvoc-dolson.par