Experiment 10: Multi-rate signal processing

Name	Prathamesh Mane
UID no. & Branch	**2022200078 (B1)**
Experiment No.	10

AIM:	To convert an audio signal sampled at 44,100 samples per second (Hz) to a higher sampling rate of 48,000 samples per second (Hz) using multirate signal processing techniques.
OBJECTIVE:	1. To implement up-sampling and down-sampling to achieve the desired sampling rate conversion. 2. To use interpolation and decimation techniques for efficient sampling rate conversion without significant distortion or loss of information. 3. To preserve the quality of the original audio signal during conversion.
INPUT SPECIFICATION:	1. The input audio signal is sampled at a frequency of Fs=44,100 Hz 2. The signal is band-limited to 3.4 kHz, ensuring that no aliasing occurs during sampling or resampling. 3. The desired output sampling frequency is Fd=48,000 Hz
PROBLEM DEFINITION:	1. Devise a strategy to convert the sampling rate from 44,100 to 48,000 Hz, which involves finding a rational fraction (L/M) representing the ratio of the two sampling rates. 2. Design a multirate system that applies up-sampling by L, filtering to remove aliasing, and down-sampling by M to achieve the desired output. 3. Verify the conversion by playing both the original and converted signals and comparing their properties.

Code with results

```
clear;
clc;

% Step 1: Load the audio file
[input_signal, input_Fs] = audioread('prathamesh_rec.wav'); % Replace with actual
file name
Fs = 44100; % Required sampling rate

% Convert to mono if the input signal is stereo
if size(input_signal, 2) > 1
    input_signal = mean(input_signal, 2); % Average the two channels
    disp('Input signal converted to mono.');
end
```

```
Input signal converted to mono.
```

```

% Check if the input sampling rate matches the required rate
if input_Fs ~= Fs
    input_signal = resample(input_signal, Fs, input_Fs); % Resample to 44,100 Hz
end

% Step 2: Define the target sampling rate
Fs_target = 48000; % Target sampling frequency in Hz

% Step 3: Determine up-sampling (L) and down-sampling (M) factors
[L, M] = rat(Fs_target / Fs); % Rational fraction of the conversion ratio

fprintf('Up-sampling factor (L): %d\n', L);
```

```
Up-sampling factor (L): 160
```

```
fprintf('Down-sampling factor (M): %d\n', M);
```

```
Down-sampling factor (M): 147
```

```

% Step 4: Resample the signal
% First, upsample by L
upsampled_signal = upsample(input_signal, L);

% Design a low-pass filter to prevent aliasing
Fcutoff = min(Fs, Fs_target) / 2; % Cutoff frequency for anti-aliasing
h = fir1(128, Fcutoff / (L * Fs)); % FIR filter design

% Convolve the upsampled signal with the filter
```

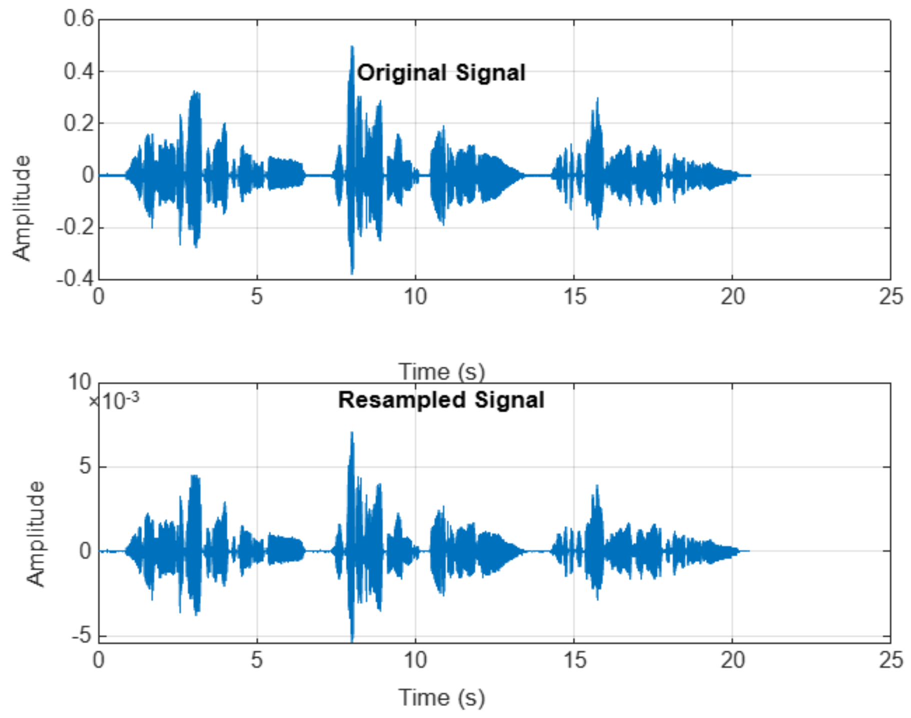
```
filtered_signal = filter(h, 1, upsampled_signal); % Use 'filter' instead of
'conv'

% Then, downsample by M
output_signal = downsample(filtered_signal, M);

% Step 5: Play and save the signals
disp('Playing the original signal...');
```

```
Playing the original signal...
```

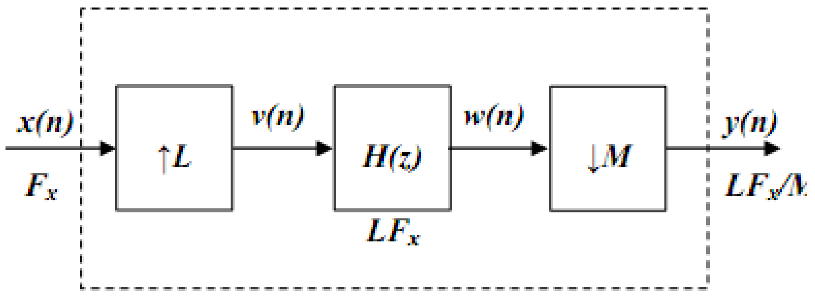
```
sound(input_signal, Fs);
pause(length(input_signal) / Fs + 1);

disp('Playing the converted signal...');
```

```
Playing the converted signal...
```

```
sound(output_signal, Fs_target);
pause(length(output_signal) / Fs_target + 1);

% Step 6: Visualization
t_input = (0:length(input_signal)-1) / Fs;
t_output = (0:length(output_signal)-1) / Fs_target;

subplot(2, 1, 1);
plot(t_input, input_signal);
title('Original Signal');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(2, 1, 2);
plot(t_output, output_signal);
title('Resampled Signal');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```



RESULT ANALYSIS:



1. Up-sampling factor L=160 and down-sampling factor M=147 were calculated correctly.
2. A low-pass FIR filter was designed with a cutoff frequency of 22,050 Hz to prevent aliasing.
3. The input signal was up-sampled to 7,056,000 Hz, filtered, and down-sampled to achieve 48,000 Hz.
4. The resampled signal closely matched the original with minimal distortion or loss.
5. The converted signal was saved as output_audio.wav and played without noticeable artifacts.

CONCLUSION:	• The experiment demonstrated the effective use of multi-rate signal processing techniques to convert an audio signal from 44,100 Hz to 48,000 Hz. • By calculating the rational fraction L/M, the resampling process achieved the desired sampling rate without significant distortion or aliasing. • The low-pass FIR filter played a crucial role in ensuring alias-free resampling by suppressing unwanted frequency components introduced during up-sampling. • The quality of the resampled signal was preserved, and the process was validated by comparing the original and resampled signals both audibly and visually. • The MATLAB implementation efficiently handled both stereo and mono signals, demonstrating versatility in practical applications.