

## ECE 538 - Digital Signal Processing I

### Design Project 1

**Name** – Gowri Kurthkoti Sridhara Rao

#### 1. Abstract

Multirate Digital Signal Processing is a concept that is widely used in many applications. This project aims on implementing an application of multirate signal processing i.e., sampling rate conversion by a rational factor  $I/D$ . The sampling rate converter takes digital signal  $x(n)$  as input with sampling frequency  $F_s$  and outputs a signal  $x'(n)$  which has a sampling frequency of  $F_s \cdot (I/D)$ . This sampling rate converter that is implemented in MATLAB is tested using an input signal which has a sampling frequency of 6000Hz and is sampled for 3 seconds. The results and discussion about the implementation are discussed in this report.

#### 2. Introduction

Digital signal processing applications include converting sampling rate of signal often. Few applications include decreasing sampling rate while few include increasing the sampling rate. One of the methods of changing the sampling rate of the signal is using A/D converter and sampling the input signal to required rate and then carrying out D/A conversion. But the loss due to D/A conversion is high. Hence, multirate signal processing is used to reduce the losses during sampling rate conversion.

Systems that employ multiple sampling rate conversions in the processing of digital signals are called multirate digital signal processing. Multirate signal processing has three categories, interpolation, decimation, and resampling. Interpolation is increasing the sampling rate by given factor  $I$ . Decimation is reducing the sampling rate by a factor  $D$ . Resampling is carrying out sampling rate conversion by a rational factor  $I/D$ .

#### 3. Theory

In this project problem statement, the sampling rate conversion is by a rational factor  $I/D$ . Block diagram in Figure 1 represents the sampling rate conversion by a factor of  $I/D$ .

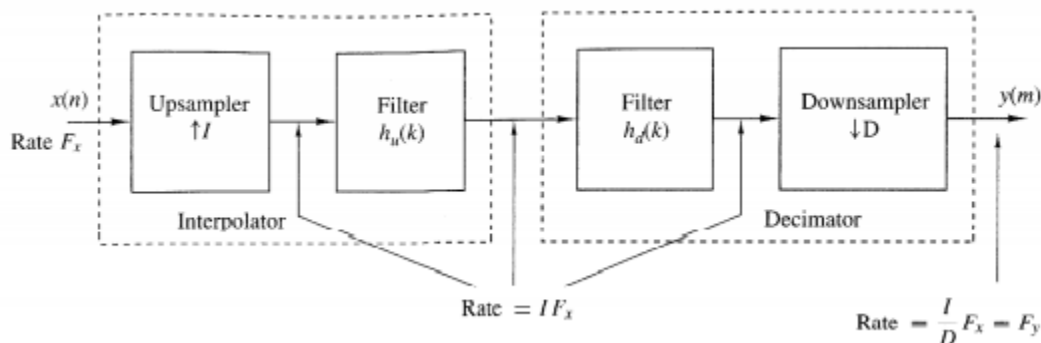


Figure 1: Sampling rate conversion by factor  $I/D$

Input signal  $x(n)$  with a sampling frequency  $F_x$  is presented to the sampling rate converter. Which first up samples by factor  $I$ . Filter  $h_u(k)$  filters the noise and discrepancies that is produced by upsampling. The sampling rate after interpolation is  $I \cdot F_x$ . After interpolation,

another filter  $h_d(k)$  is used to filter the signal. These two filters are cascaded to work as a single filter  $h(k)$  which performs operation of both these filters. Downsampler is used to decimate the signal by a factor of  $D$ . The frequency of the signal after downsampling is  $(I/D) * F_x$ . The interpolation is carried out before decimation to preserve the spectral characteristics of  $x(n)$ .

$$H(\omega_v) = \begin{cases} I, & 0 \leq |\omega_v| \leq \min(\pi/D, \pi/I) \\ 0, & \text{otherwise} \end{cases}$$

Figure 2: Frequency response of the filter

The frequency response of the filter  $h(k)$  that must be designed is given by the equation in figure 2. According to the equation, the frequencies below the cut-off frequency designed for the filter are interpolated by a factor  $I$ . The cut-off frequency in terms of  $I$  and  $D$  is as given in figure 2 and it is used in the project for calculation.

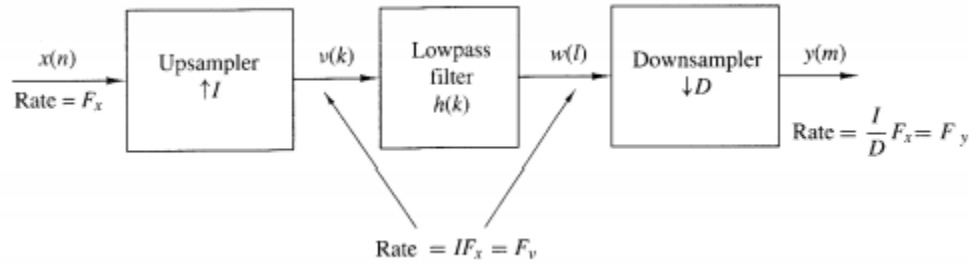


Figure 3: Functional block diagram of the process

The same filter that is designed can be used for decimation block too by using the following relation in figure 4. Even in this equation the cut-off frequency is as it was in figure 2. Hence that was chosen as the optimum cut-off frequency to design the filter.

$$\begin{aligned} V(\omega_v) &= H(\omega_v)X(\omega_v I) \\ &= \begin{cases} I X(\omega_v I), & 0 \leq |\omega_v| \leq \min(\pi/D, \pi/I) \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

Figure 4: Decimation filter response

The project required to design a sampling rate converter which converts any given digital signal with sampling rate  $F_s$  into a signal with sampling rate  $F_s * (I/D)$ . The FIR filter that is present between the upsampler and downsampler are defined between  $-60 < n < 60$

$$h(n) = 0.5 \frac{\sin(\omega_c n)}{\pi n}$$

In the project, the sampling rate converter was implemented using a function named `SamplingRateConverter()`. The MATLAB code of the function is as follows:

```
function [output_sg,out_samp_rate] =
SamplingRateConverter(input_sg,I,D,fs)
%SAMPLINGRATECONVERTER Multirate sampling rate converter
% This function converts the sampling rate of the input signal by a
% factor I/D. Input to the function is x(n), input signal whose sampling
% rate has to be converted. I is the interpolating factor, D is the
% decimating factor. fs is the original sampling frequency of the
% signal.
%% Determining cutoff frequency
wc= min((pi/I),(pi/D)); % cutoff frequency
```

```
%disp(wc);
%% Upsampling
sg_up = upsample(input_sg,I); %upsampling the input signal by a factor of
I
fs_u=fs*I; %Determining the sampling frequency after upsampling
figure(2);
plot(sg_up(1:1000),'red'); %Plotting 1000 samples of upsampled signal
title("Upsampled input signal, fs="+fs_u);
xlabel("n-->");ylabel("Amplitude");
%% Filtering
filtered=filter_sg(sg_up,wc); %Filtering the upsampled signal using
function filter_sg
figure(6);
plot(filtered(1:1000),'green'); %Plotting 1000 sampled of filtered signal
title("Filtered signal");
xlabel("n-->");ylabel("Amplitude");
%% Downsampling
output_sg=downsample(filtered,D); %Downsampling the filtered signal by
factor D
out_samp_rate= fs_u/D; %Determining output sampling rate

end
```

The above function implements the sampling rate conversion as mentioned in the comments next to the code. This function calls another function called filter\_sg () which implements filtering of the signal with the desired filter characteristics. The MATLAB code of the function is as follows:

```
function [output] = filter_sg(input_sg,wc)
%FILTER_SG Filter the input signal and plot the frequency and impulse
%response
% The input to this function is a digital signal and its' cutoff frequency
which is convoluted with
% a signal having impulse function  $h(n)=0.5*\sin(wc*n)/\pi*n$ . The impulse
% response of the filter is plotted along with its frequency and phase
% response.
n=-60:1:60; %Impulse response of filter is defined between [-60 60]
filter=0.5*(sin(wc.*n)./(pi.*n)); %Definition of filter
filter(61)=0.5*(wc/pi); %Setting the 0th sample to a definite value
%% Frequency response of the filter
freq_filt=fft(filter);
[h,w]=freqz(filter,1); %Using freqz function to find H(w) from h(n)
dB = mag2db(abs(h));
%disp(dB);
%figure(8);plot(dB);
%%
figure(3);
plot(n,filter); %Plotting the impulse response
title("Filter response in time domain");
%%
figure (4);
subplot(2,1,1);plot(w/pi,20*log10(abs(h))); %Plotting frequency response
title("Magnitude response, Wc="+wc/pi));
xlabel("w");ylabel("H(w) in dB");
```

```

subplot(2,1,2);
plot(w/pi,unwrap(angle(h))*(180/pi)); %Plotting phase response
title("Phase response");
xlabel("w"); ylabel("angel");
%% Filtering the input signal
output=conv(input_sg,filter); %Convolution of singal and filter

end

```

The function implements a low-pass filter which has a cut-off frequency of  $w_c$  defined using I and D factors. The filter characteristics in time domain and frequency domain are presented in detail in the results section of the report.

#### 4. Results

The problem statement provided in the project requires the system to be tested using a signal. The problem statement is as follows:

To test your code, create the following analog signal

$$x(t) = 10\cos(2\pi \cdot 890t) + 5\cos(2\pi \cdot 385t) + 4\cos(2\pi \cdot 1450t) + \eta(t)$$

where  $\eta(t)$  is random zero mean Gaussian noise with standard deviation 1. The signal is digitized at a sampling rate of 6000 Hz, and the length of the resulting sequence is 18,000 samples. Plot a portion of the resulting digital signal. Then perform sampling rate conversion by the factor 6/2 and plot the resulting signal.

This signal was generated, and sampling rate converted by calling the SamplingRateConverter () function. The code used to carry out this process is as given below:

```

%% Sampling the input signal
% The input signal is sampled with the initial sampling frequency of 6000Hz
% and a part of the input signal which is interfered with noise is plotted.
I=6; D=2; %Example I and D values
fs=6000; %Initial sampling frequency
t=0:1/fs:3; %Since the resulting number of samples is 3*sampling frequency
Noise = 1*randn(size(t)) + 0; %random noise with zero mean and standard
deviation equal to 1
sampled_inp=10*cos((2*pi*890).*t)+5*cos((2*pi*385).*t)+4*cos((2*pi*1450).*
t)+Noise; %Example signal
figure(1);
plot(sampled_inp(1:1000)); %Plotting 1000 samples of the input signal
title("Part of the input signal, fs="+fs);
xlabel("n-->");ylabel("Amplitude");
%% Sampling rate conversion
% The sampling rate of the input signal is converted by a factor of I/D.
% Part of the output signal with sampling frequency fs*(I/D) is plotted.
[output_sg,fs_out]=SamplingRateConverter(sampled_inp,I,D,fs); %Calling the
function sampling rate converter
figure(7);
plot(output_sg(1:1000)); %Plotting 1000 samples of output signal
title("Part of the output signal, fs="+fs_out);
xlabel("n-->");ylabel("Amplitude");

```

The results obtained after carrying out the sampling rate conversion is presented in the following section.

- The input digitized signal has around 18,000 samples. A part of the input signal is presented in the figure 5.

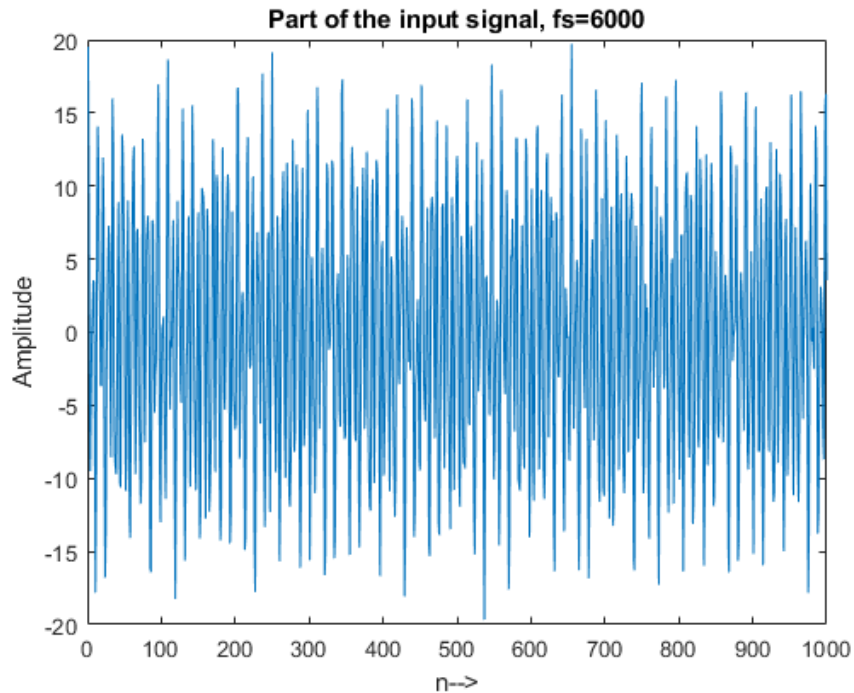


Figure 5: Part of the input signal

- The digital input signal is presented as input to the sampling rate converter in which the signal is initially upsampled. The resulting sampling frequency raises to 36000 due to multiplication by factor  $I=6$ .

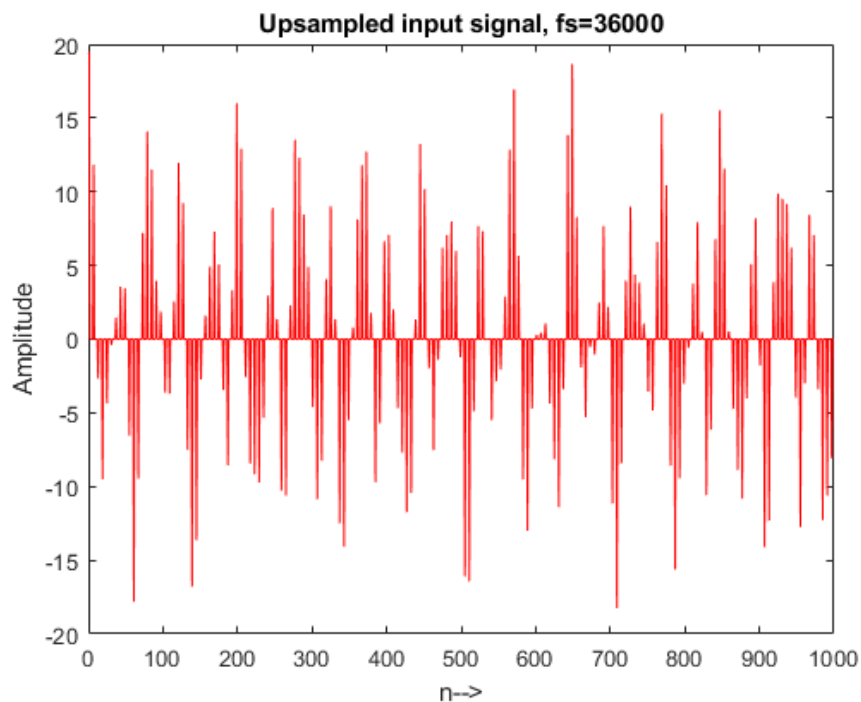


Figure 6: Upsampled signal

- The signal is then filtered using the filter defined in filter\_sg. The time domain impulse response, frequency domain magnitude and phase response of the filtered signal is as given below.

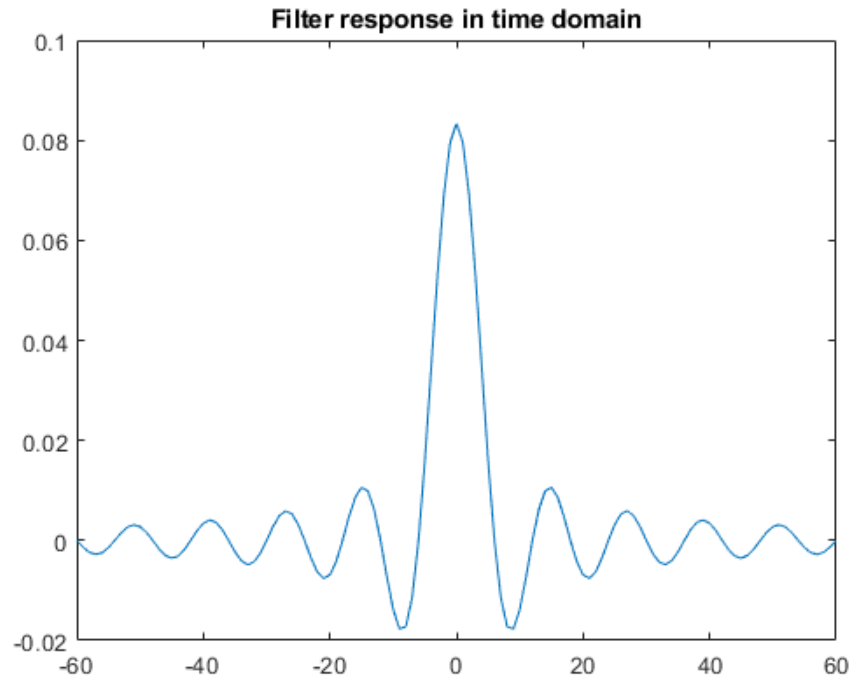


Figure 7: Time-domain impulse response of filter

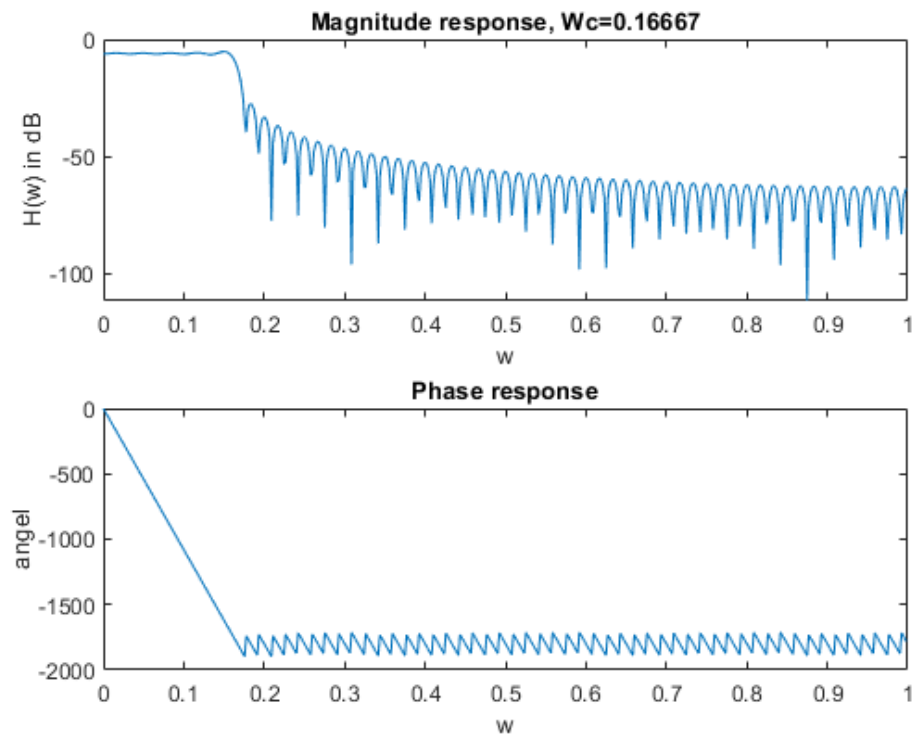


Figure 8: Magnitude and phase response of filter in frequency domain

- A part of the filtered signal looks like the one presented in following figure.

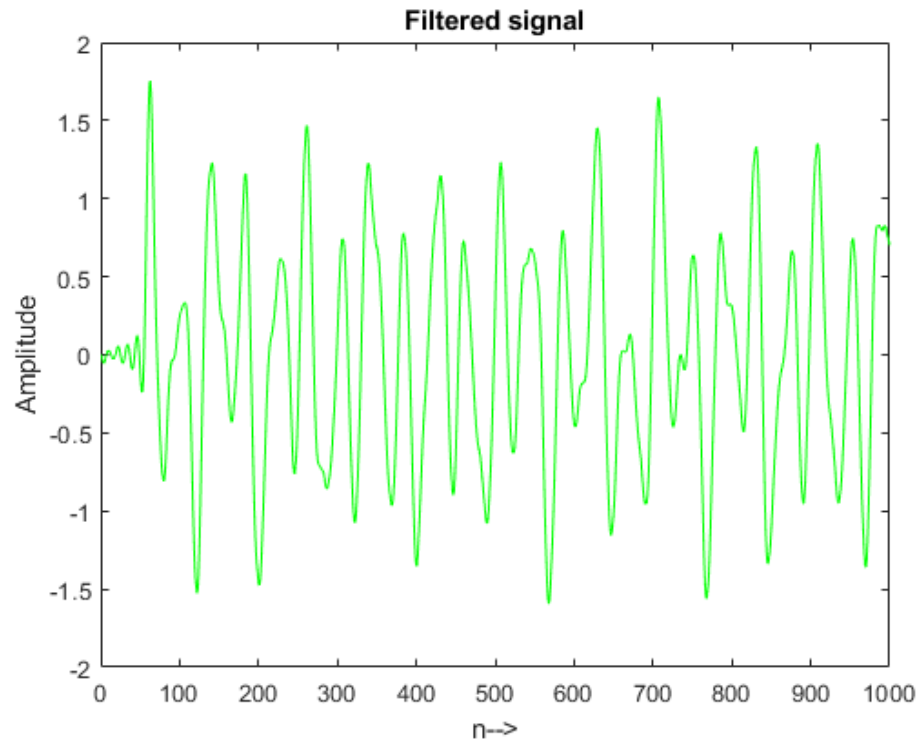


Figure 9: Filtered signal

- The filtered signal is then downsampled with a factor of  $D=2$ . Which is also the output of the function and project. The final sampling frequency is 18000Hz which is  $(6/2) = 3 \cdot f_s$ .

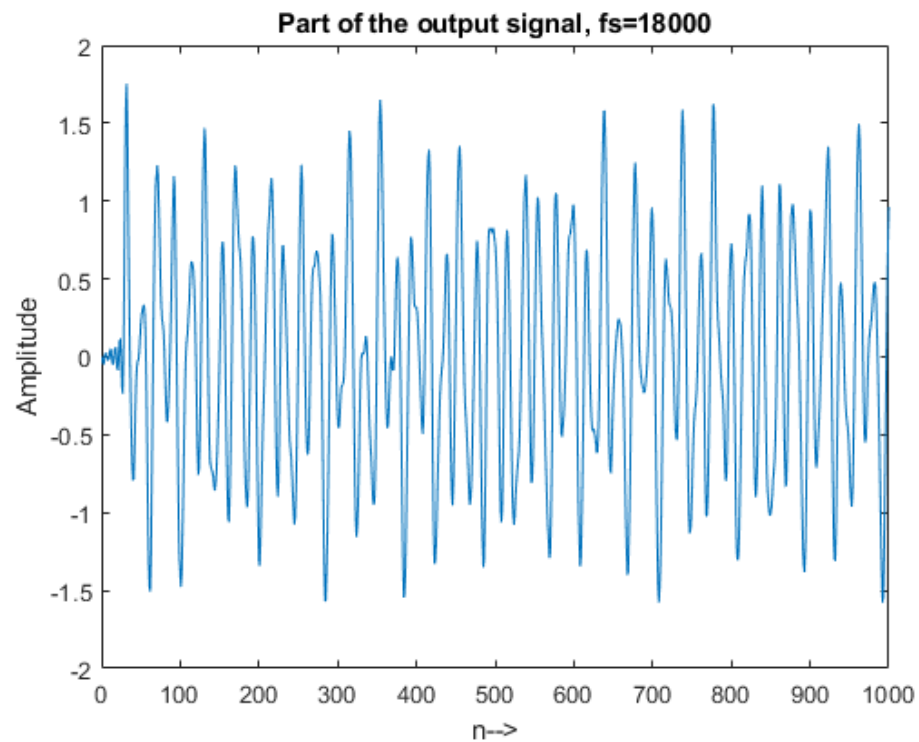
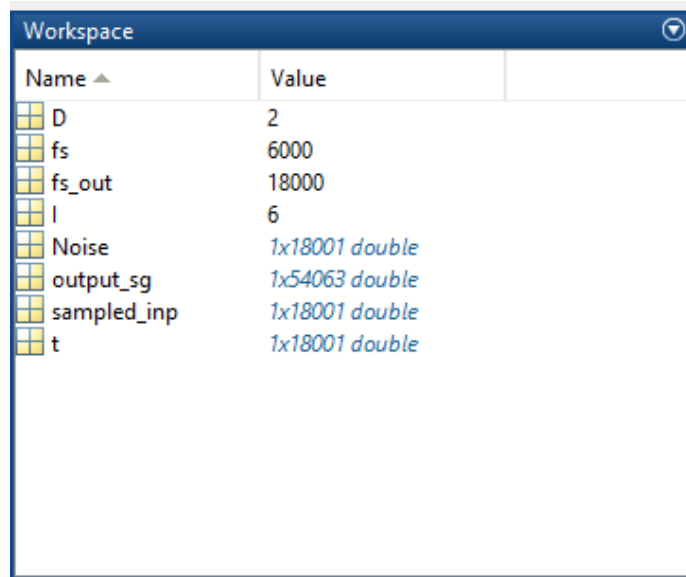


Figure 10: Part of the output signal

## 5. Discussion

The project included sampling rate conversion of a signal with 6000hz sampling frequency having 18,000 samples to a signal with sampling frequency 18,000Hz and 54,000 samples. This was achieved successfully with the MATLAB codes as mentioned above. Here is a screenshot of the workspace in the MATLAB that defines the number of variables and their size.



Name	Value
D	2
fs	6000
fs_out	18000
I	6
Noise	1x18001 double
output_sg	1x54063 double
sampled_inp	1x18001 double
t	1x18001 double

Figure 11: Workspace tab in MATLAB

By dividing the implementation into different functions reduced the number of variables that were statically allocated in the code. Although the variables were allocated values during the function call, once the end appears all the values are released making efficient use of the space. Sampled input is the input signal with 18001 samples. Output\_sg is the output signal with 54,063 samples achieving the major objective of the project of sampling rate conversion.

The filter that is used in this project has a normalized cutoff frequency of 0.1667. The ripples in the magnitude response of the filter is seen in both transition and in stop band. Due to presence of ripples, some of the higher frequencies are not completely filtered which can be observed in the figure 9. The filter is a higher order filter as it has a sharp/narrow transition bandwidth. But the filter still performs the major filtering of high frequency noise and hence the filter is considered stable and reliable for this application.

This sampling rate converter can be used for several applications where the conversion of sampling rates is necessary for examples when connecting from speakers to headphones. A simple sampling rate conversion must be carried out as the headphones might be operating at a different sampling frequency configuration.

## 6. Conclusion

Multirate Digital Signal Processing is one of the fundamental concepts of digital signal processing which is used in most of the applications. Sampling rate conversion in the digital domain was the major goal of this project. The sampling rate converter was implemented on MATLAB using several in-built MATLAB commands and functions. Sampling rate converter was provided with a digital input, I, D, and sampling frequency which outputted the signal whose sampling rate was modified by ratio I/D. From this project knowledge about



sampling rate conversion in digital domain by avoiding A/D and D/A converter was acquired. This project also involved filter designing and analyzing the frequency response of the filter. To conclude, this project involved implementing and testing the sampling rate conversion by a rational factor  $I/D$ .