EE5900 Programming Assignment 3

Gautam Singh
CS21BTECH11018

1) The code for efficient polyphase filter implementation is given in Source Code 1. Here, the filter coefficients are split into their polyphase coefficients. All filters operate at a rate $F = \frac{F_s}{M} = 12$ kHz. The results are shown in Figure 1. Notice that this filter will not work for signals whose spectral content lies outside the Nyquist done of $F_N = \frac{F_s}{2M} = 6$ kHz. Thus, a 3 kHz signal has been used.
   Notice that image bands do exist, however they attenuate in further Nyquist zones. If more coefficients of the filter are taken, they will attenuate quicker. The output can be passed through a low-pass filter to obtain the required signal without its harmonics.

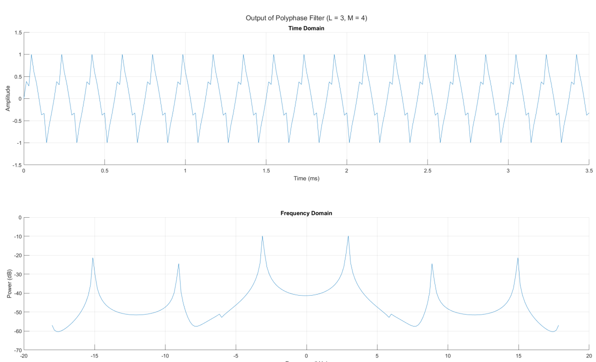

Fig. 1: Resampling Using a Polyphase Filter.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name        : Gautam Singh                              %
% Roll Number : CS21BTECH11018                            %
% Date        : 2023-11-11                                %
% File        : ee5900_assign_3.m                         %
% Purpose     : Implement a computationally efficient     %
%               polyphase filter to resample signals with %
%               a factor of 3/4.                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
clear
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% List of constants
F = 3e3;     % Frequency of signal
N = 300;     % Number of samples
Fs = 48e3;   % Initial sampling frequency
Ff = 36e3;   % Final sampling frequency
L = 3;       % Upsampling factor
M = 4;       % Downsampling factor
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Sampling interval
Ts = 1/Fs;

% Timestamps
t = 0:Ts:(N-1)*Ts;

% Create samples of signal at rate Fs
x = sin(2*pi*F*t);

% Output
y = zeros(1,N*L/M);
n = length(y);

% Filter coefficients
mx = max(L,M);
h = sinc(0:1/mx:N-1/mx);

% Subfilters hij, 0 <= i < L, 0 <= j < M
for i = 0:1:L-1
    for j = 0:1:M-1
        % Get start index
        st = L - i - j;
        while st <= 0
            st = st + M;
```

```
        end
        % Get decimated samples to be filtered for this branch
        xij = x(st:M:end);
        % Subfilter for this branch is R(i, j)
        % Start coefficient of subfilter
        st_subf = L - i + M*j;
        % Get subfilter coefficients
        rij = h(st_subf:L*M:end);
        % Apply the subfilter
        yij = filter(rij,1,xij);
        % Accumulate the output after upsampling
        st_y = L - i;
        y(st_y:L:end) = y(st_y:L:end) + yij;
    end
end

% Plot the outputs (time domain and frequency domain)
tlo = tiledlayout(2,1);
title(tlo, ['Output of Polyphase Filter (L = ', num2str(L), ...
            ', M = ', num2str(M), ')']);
nexttile
hold on
grid on
xp = 0:1:n-1;
plot(xp*1e3*L*Ts/M,y);
xlabel('Time (ms)');
ylabel('Amplitude');
title('Time Domain');

nexttile
hold on
grid on
Yf = fftshift(fft(y))/(L*N/M);
f = (-n/2:n/2-1)*L*Fs/(M*1e3*n);
plot(f, 20*log10(abs(Yf)));
xlabel('Frequency (kHz)');
ylabel('Power (dB)');
title('Frequency Domain');
```

Source Code 1: MATLAB Code for Question 1.