ECE310 – Project 1

Jonathan Lam

September 29, 2020

Problem description

Design a sample rate converter that efficiently converts an input audio signal, sampled at 11025Hz, to an output audio signal sampled at 24000Hz. Record the number of floating-point operations (FLOs) performed.

The resulting system should approximately look like a low-pass filter, and it should meet the following specifications:

passband cutoff frequency	$\omega_p = 11025\pi/24000$
passband ripple	$R_p < \pm 0.1\text{dB}$
stopband frequency	$\omega_s = 1.2\omega_p$
stopband attenuation	$R_s > 70\text{dB}$
phase constraints (in passband)	$\lvert\max\{\text{grpdelay}\} - \min\{\text{grpdelay}\}\rvert < 720$

Notes on implementation

- A three-stage system was used here. The ratio 24000/11025 can be reduced to the whole-number ratio 320/147, which can be expressed as the product $5/3 \times 8/7 \times 8/7$. These (small) integers were used to make reasonable upsample/filter/downsample blocks, where each system is implemented using a polyphase decomposition of the interpolation. (The polyphase implementation may also have been performed on the decimation, but here all of the interpolation ratios are larger than the decimation ratios, so this is more efficient.)
- Similar to the textbook, only FLOs relating to the polyphase implementation (convolution and additions) were counted.
- I decided to report the FLOs as floating point operations per input sample (how we should report efficiency was not specified in the problem description). If we were processing an input sample in real-time (which we are not), this can be translated into FLOPS by dividing by the input audio sample rate (11025Hz). To get the total number of FLOs, multiply the floating-point operations per sample by the number of samples.
- The LPFs were designed to be lowest-order Chebyshev filters using `cheb2ord` and `cheby2`.
- Greatly appreciated help was received from Dan Kim and Tony Belladonna.

Source code

Sample rate converter

```
% efficiently converts sampling rate from 11.025kHz to 24kHz
% (or any other resampling with a ratio of 320/147)
% returns the samplerate-converted signal and the number of FLOs per sample
function [x, flo] = srconvert(x)
    % break down into three stages
    [x, flo1] = efficientresample(x, 5, 3);
    [x, flo2] = efficientresample(x, 8, 7);
    [x, flo3] = efficientresample(x, 8, 7);

    % number of FLOs per sample
    flo = flo1 + flo2 + flo3;
end

% uses polyphase decomposition to efficiently up- and down-sample x
function [res, flo] = efficientresample(x, L, M)
    % design the LPF
    % source: https://www.mathworks.com/help/signal/ref/cheb2ord.html
    Wp = min(1/L, 1/M);
    Ws = Wp * 1.2;
    Rp = 0.1;
    Rs = 85;
    [n, Ws] = cheb2ord(Wp, Ws, Rp, Rs);
    [b, a] = cheby2(n, Rs, Ws, 'low');
    h = impz(b, a).';

    % decompose h into its polyphase components
    hc = poly1(h, L);

    % do all the convolutions
    res = zeros(1, L * length(x));
    for i = 1:L
        component = upsample(fftfilt(hc(i, :), x), L);
        res = res + [zeros(1, i-1) component(1:length(component)+1-i)];
    end

    % decimation and rescaling
    res = L * downsample(res, M);

    % counting calculations; same calculation as in textbook;
    % L * (N/L) = N multiplications/(sample period) in convolutions
    % L * (N/L - 1) = N - L adds/(sample period) in convolutions
    % L - 1 additions when adding polyphase components together
    % total = N + (N - L) + (L - 1) = 2 * N + 1 FLO per sample
    flo = 2*length(h) + 1;
end
```

poly1.m (provided in assignment)

```
function E=poly1(h,M)
%
% Performs type I polyphase decomposition of h in M components.
% The ith row of E corresponds to the ith polyphase component.
% Assumes that the first point of h is index 0.
%
h = [h zeros(1, ceil(length(h)/M)*M-length(h))];
E = reshape(h, M, length(h)/M);
```

verify.m (provided in assignment)

```
% Verifies that h meets all the specifications given in the
% handout. Returns a 1 if all specifications are met.
function yes = verify(h)
    [R,G,A]=examlpf(h,147/320,147/320*1.2);

    % passband magnitude
    yes         = R <=0.1;
    subplot(3,1,1)

    if R <=0.1
        title(['Passband Response,  Ripple = ' num2str(R) ' dB, OK']);
    else
        title(['Passband Response,  Ripple = ' num2str(R) ' dB, too big']);
    end

    % Group delay
    subplot(3,1,2)
    if G <= 720
        title(['Group Delay in Passband, max-min = ' num2str(G) ', OK'])
    else
        title(['Group Delay in Passband, max-min = ' num2str(G) ', too big'])
    end
    yes         = yes*(G <= 720);

    % Stopband magnitude
    subplot(3,1,3)
    if A <=-70
        title(['Overall Response , Attenuation = ' num2str(A) ' dB, OK']);
    else
        title(['Overall Response , Attenuation = ' num2str(A) ' dB, too small']);
    end

    yes         = yes * (A <= -70);

    sprintf('Passband Ripple:       %5.3f dB \n',R)
    sprintf('Groupdelay Variation:  %5d   samples \n',G)
    sprintf('Stopband Attenuation:  %5.3f dB \n',A)
return
```

Results

Testing sample rate converter on an audio file

```
% testing wagner; resample and write to new file
[x, fs] = audioread('Wagner.wav');
x = x.';
[y, flo] = srconvert(x);
fprintf('FPOs per sample: %d\n', flo);
audiowrite('resampledWagner.wav', y, 24000);

% plot x and y
subplot(2, 1, 1);
t = linspace(0, length(x)/fs, length(x));
plot(t, x);
title(sprintf('Original, sampled at 11025Hz (%d samples)', length(x)));
xlabel('t (s)');
ylim([-1 1]);
xlim([0 length(x)/fs]);
subplot(2, 1, 2);
t = linspace(0, length(x)/fs, length(y));
plot(t, y);
title(sprintf('Sample rate-converted to 24000Hz (%d samples)', length(y)));
xlabel('t (s)');
ylim([-1 1]);
xlim([0 length(x)/fs]);
figure();
```

Output:

```
FLOs per sample: 3477
```

See (Figure 1) for the generated plots.

Testing sample rate converter using `verify()`

```
ir = srconvert([1 zeros(1,3000)]);
verify(ir);
```

Output:

```
Passband Ripple:       0.001 dB
Groupdelay Variation:  1.585250e+01   samples
Stopband Attenuation:  -73.208 dB
```

See (Figure 2) for the generated plots.

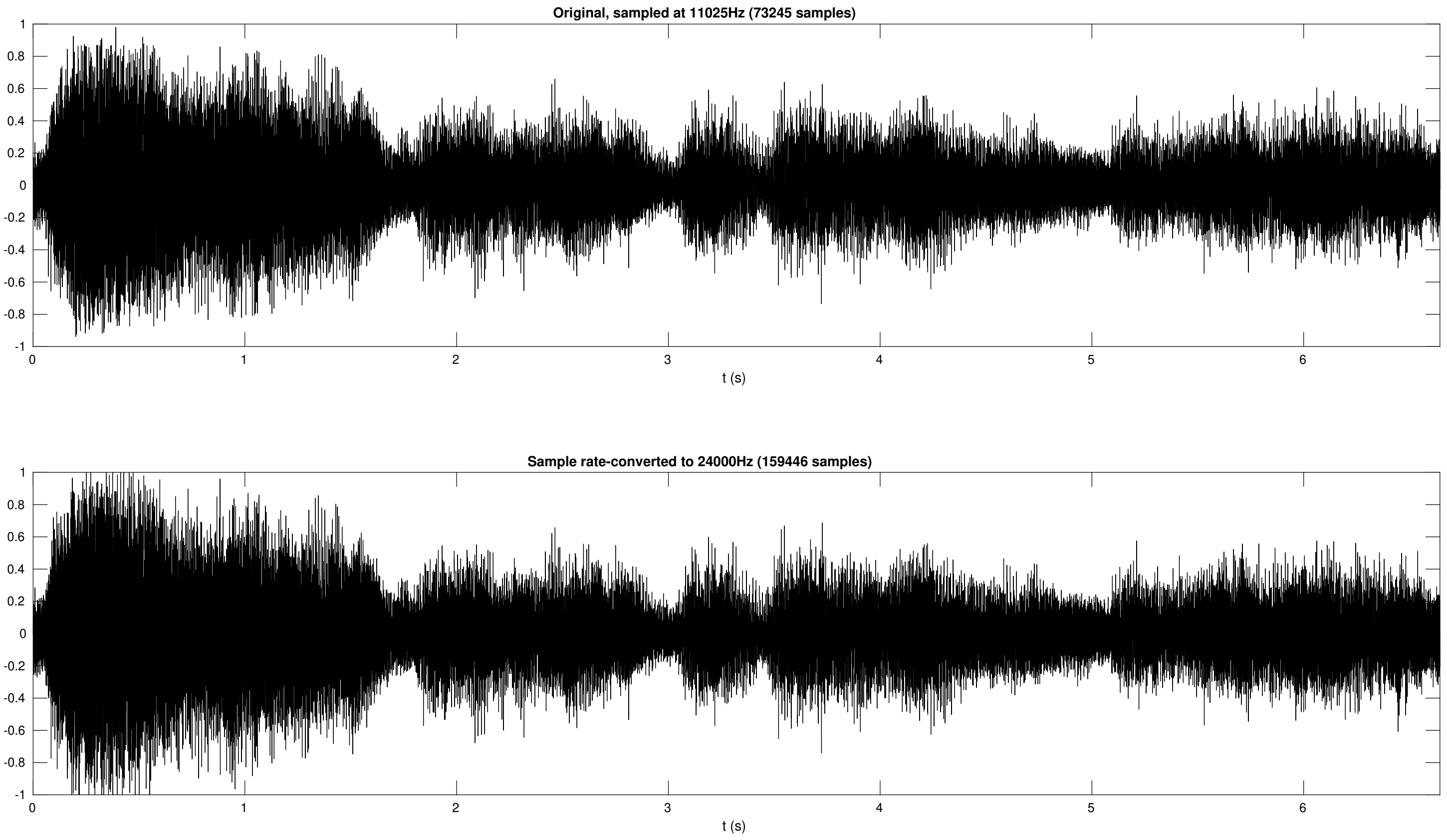

Figure 1: Comparison of original Wagner.wav with sample rate-converted signal. Both the visual quality (from these graphs) and the audio quality (from listening to the two audio files) are comparable.

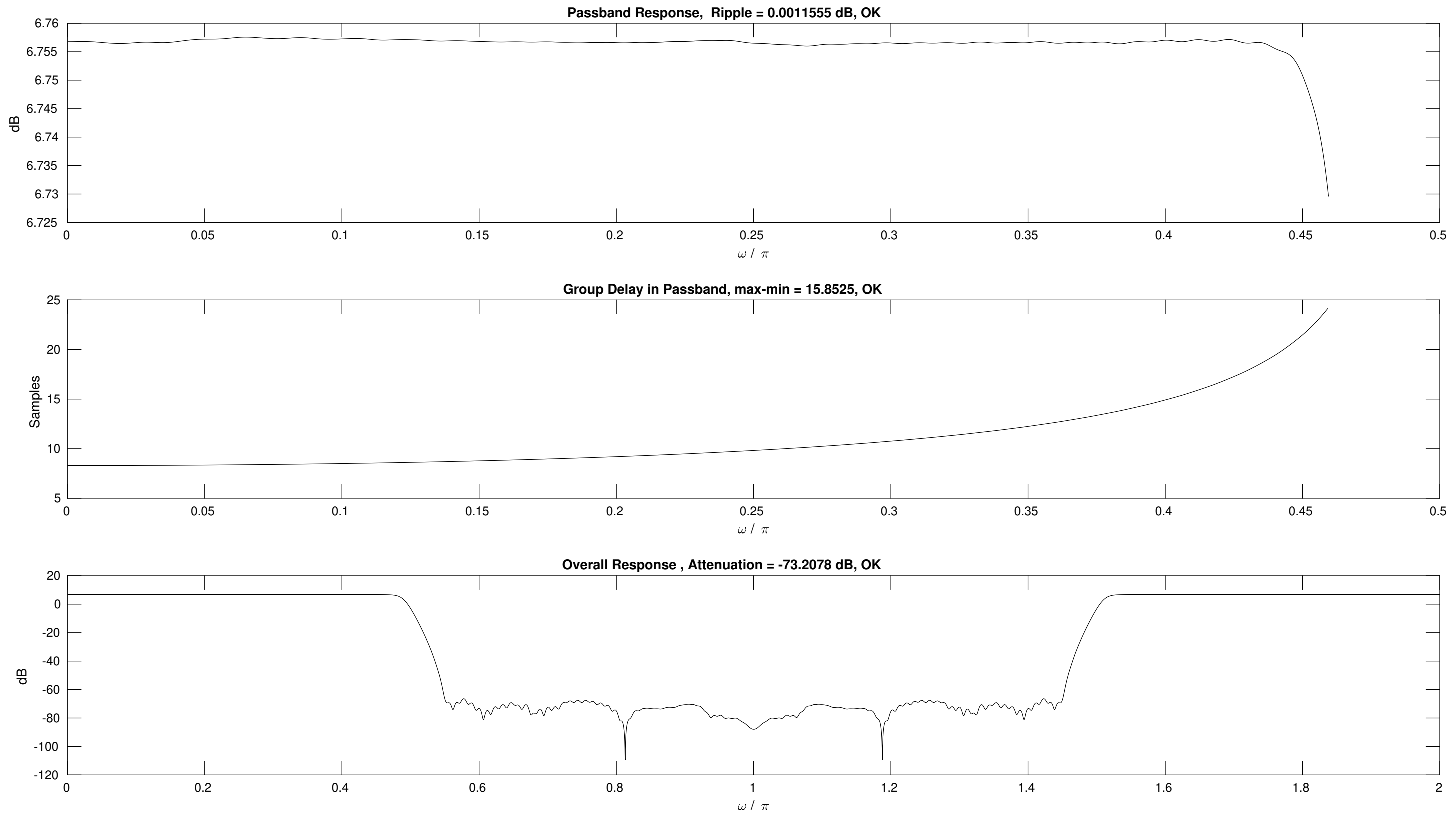

Figure 2: Verification that the system meets the specifications.