1 $r_s[n]$ analysis

Student number: 7 and 4. Hence we use `projsignalm.mat` where $m = 1$, carrier frequency $F_c = 12.288$ kHz.

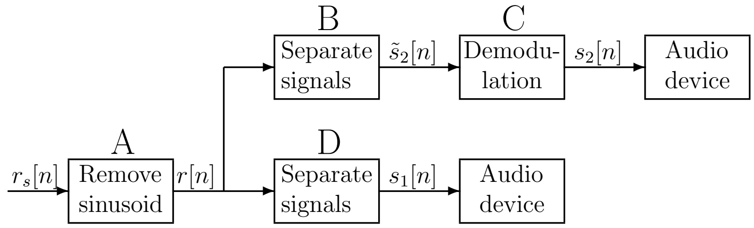

Figure 1: Signal removal and separation

In Figure 1, the signal $r_s[n]$ is made up of three components

$$r_s[n] = r_1[n] + r_2[n] + w[n] \tag{1}$$

$r_1[n]$ and $r_2[n]$ are the two important signal components and they occupy two separate frequency bands. $r_1[n]$ has a bandwidth of 4096 Hz, and $r_2[n]$ is a DSB-SC (double side band, suppressed carrier) modulated signal. The carrier frequency is F_c kHz, and the bandwidth of the original signal is 4096 Hz. $w[n]$ is a sinusoidal disturbance signal. The sampling frequency of $r_s[n]$ is 32.768 kHz.

In block A the sinusoid $w[n]$ is removed. The two signal components of $r[n]$ are then separated in block B and D such that output of block B, $\tilde{s}_2[n]$ contains the high frequency signal, and the output of block D contains the low frequency signal. The high frequency signal $\tilde{s}_2[n]$ is demodulated in block C, so that the original signal is recovered.

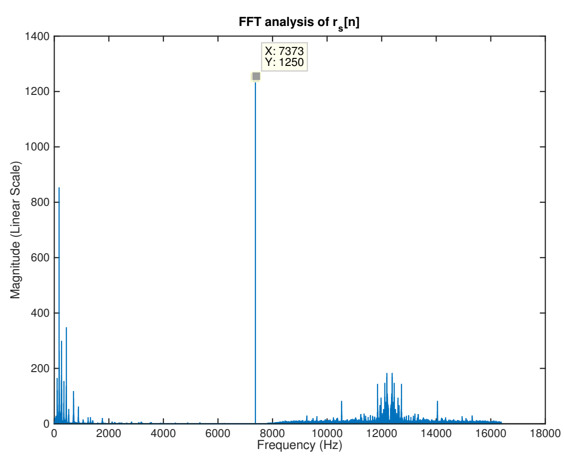

Figure 2: $r_s[n]$ in frequency domain (Linear scale)

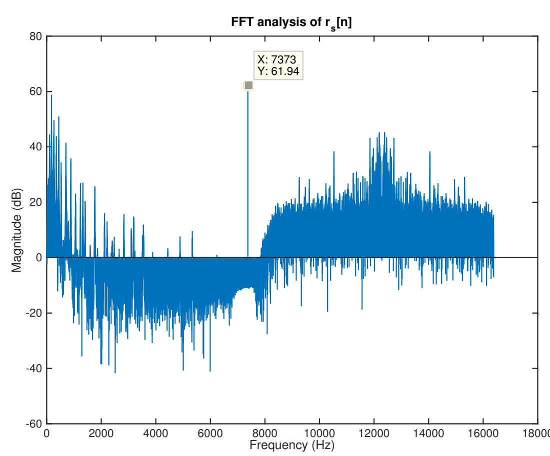

Figure 3: $r_s[n]$ in frequency domain (dB)

It can be clearly seen in Figure 2, the frequency of $w[n]$ is 7372.8 Hz (0.45π rad/sample).

Assuming $r_2(t)$ is the DSB-SC modulated signal of $m(t)$

$$r_2(t) = m(t)\cos(2\pi F_c t) \leftrightarrow R_2(f) = \frac{1}{2}[M(f + F_c) + M(f - F_c)] \tag{2}$$

Hence, $r_2[n]$ has a bandwidth of $[F_c - 4096 \text{ Hz}, F_c + 4096 \text{ Hz}]$.

$$B_w = [8192 \text{ Hz}, 16384 \text{ Hz}] \tag{3}$$

Note that: the carrier signal with the phase ϕ, $v_c(t) = \cos(2\pi F_c t + \phi)$ will have the same magnitude - frequency spectra. (The phase - frequency spectra are different.)

Define

$$y(t) := r_2(t)\cos(2\pi F_c t) \tag{4}$$

Similar to Eq. 2

$$y(t) \leftrightarrow Y(f) = \frac{1}{2}[R_2(f + F_c) + R_2(f - F_c)] \tag{5}$$

$$Y(f) = \frac{1}{2}M(f) + \frac{1}{4}M(f + 2F_c) + \frac{1}{4}M(f - 2F_c) \tag{6}$$

Apparently, multiplying the DSB-SC signal with the carrier signal yields a scaled version of the original message signal plus a higher frequency term.

To sum up, block A is a notch filter ($w[n]$ frequency: 7372.8 Hz, i.e. 0.45π rad/sample); block D is a lowpass filter ($r_1[n]$ frequency band: [0, 4096 Hz]); block B is a highpass filter ($r_2[n]$ frequency band: [8192 Hz, 16384 Hz]); block C contains a lowpass filter (bandwidth of interest: [0, 4096 Hz]).

The demodulation oscillator's phase ϕ_2 must be exactly the same as modulation oscillator's ϕ_1, otherwise, attenuation will occur. To see this effect, take demodulation signal with small phase deviations θ from the modulation signal: $\cos(2\pi F_c t + \theta)$.

The resultant signal is attenuated by a constant factor $\cos(\theta)$.

$$\begin{aligned}
&m(t)\cos(2\pi F_c t)\cos(2\pi F_c t + \theta) \\
&= \frac{1}{2}m(t)\cos(\theta) + \frac{1}{2}m(t)\cos(2 \cdot 2\pi F_c t + \theta) \\
&\xrightarrow{\text{After low pass filter}} \frac{1}{2}m(t)\cos(\theta)
\end{aligned}$$

Our goal is to make $\cos(\theta) \to 1$, in other words, $\theta \to 0$.

Define the demodulation carrier signal:

$$v_c(t) = \cos(2\pi F_c t + \phi_2) \tag{7}$$

In order to obtain the ϕ_2 that minimizes $|\theta| = |\phi_1 - \phi_2|$, different ϕ_2 will be tested until the largest amplitude of demodulated signal is found. Function `find_phi(s2_tilde)` is programmed to find optimal ϕ_2 (available on page 13).

2 FIR filters design

2.1 Block A Notch filter

2.1.1 Specification

Method: Parks-McClellan optimal FIR filter design
Passband: [0, 6872.8 Hz] and [7872.8 Hz, $+\infty$)
Stopband: [7371.8 Hz, 7373.8 Hz]
Passband ripple: 0.005 (smaller than 0.01 for further cascade)
Stopband ripple: 0.001 (-60 dB)

The lowest order can be calculated by `firpmord()` function.

$$N = 184 \tag{8}$$

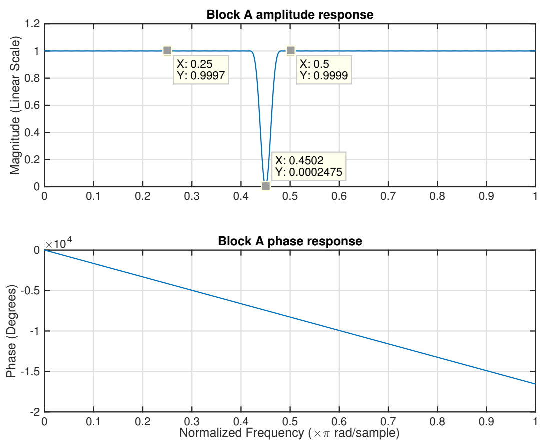

Figure 4: Block A response

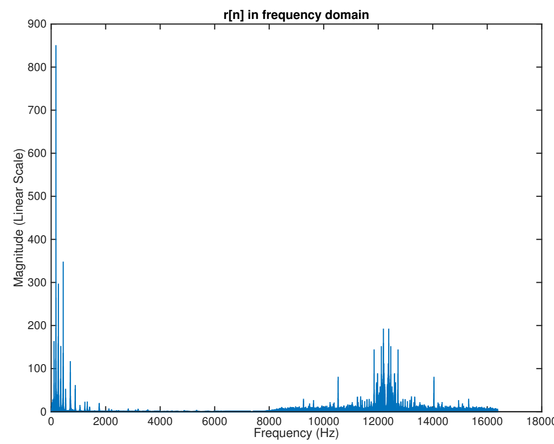

Figure 5: $r[n]$ in frequency domain

As can be seen in Figure 4 and Figure 5, $w[n]$ at 7372.8 Hz (0.45π rad/sample) has been attenuated by $20\log_{10}(0.0002475)$ = -72.1285 dB < -60 dB. Also, the notch filter has linear phase.

2.2 Block D Lowpass filter

2.2.1 Specification

Method: **Parks-McClellan** optimal FIR filter design
Passband: [0, 4096 Hz]
Stopband: [4596 Hz, $+\infty$)
Passband ripple: 0.005 (smaller than 0.01 because of cascade)
Stopband ripple: 0.01 (-40 dB)

The lowest order can be calculated by `firpmord()` function.

$$N = 141 \tag{9}$$

2.2.2 Frequency Response & Output

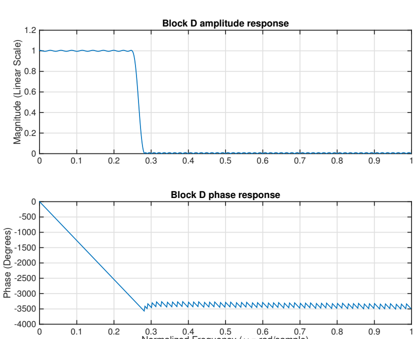

Figure 6: Block D response

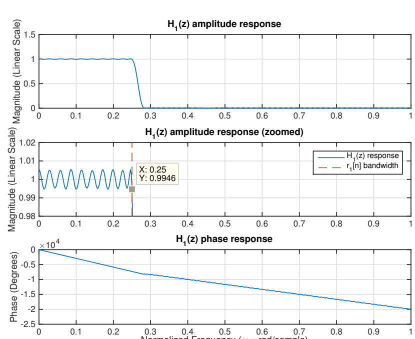

Figure 7: $H_1(z)$ Response

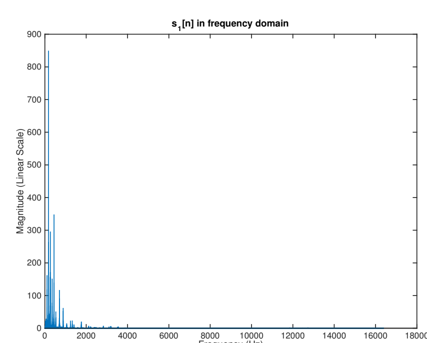

Figure 8: $s_1[n]$ in frequency domain

In Fig. 6, Block D has linear phase. In Fig. 7, $H_1(z)$ has ripple less than 0.01. In Fig. 8, high frequency components have been successfully filtered.

$H_1(z)$ is the filter obtained by cascading the filters in block A and D.

2.3 Block B Highpass filter

2.3.1 Specification

Method: **Parks-McClellan** optimal FIR filter design
Stopband: [0, 7692 Hz]
Passband: [8192 Hz, $+\infty$)
Stopband ripple: 0.01 (-40 dB)
Passband ripple: 0.005 (smaller than 0.01 because of cascade)

The lowest order can be calculated by `firpmord()` function.

$$N = 140 \tag{10}$$

2.3.2 Frequency Response & Output

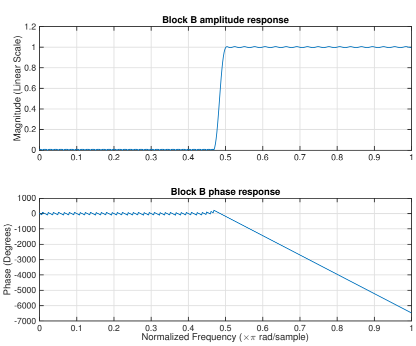

Figure 9: Block B response

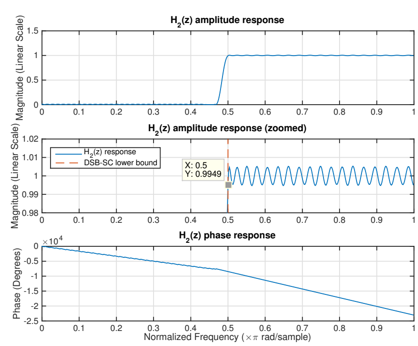

Figure 10: $H_2(z)$ Response

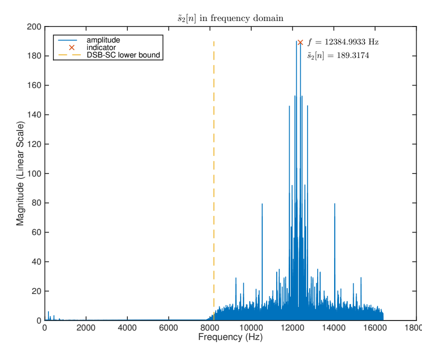

Figure 11: $\tilde{s}_2[n]$ in frequency domain

In Fig. 9, Block B has linear phase. In Fig. 10, $H_2(z)$ has ripple less than 0.01. In Fig. 11, low frequency components have been successfully filtered.

$H_2(z)$ is the filter obtained by cascading the filters in block A and B.

2.4 Block C Lowpass filter

2.4.1 Demodulation phase shift

Based on the theoretical discussion on page 2, ϕ_2 in Eq.7 can be calculated by `find_phi(s2_tilde)` function.

$$\phi_2 = 0.25\pi \tag{11}$$

For instance, the spike at 12384.99 Hz in Fig. 11 corresponds to the spike at 96.99 Hz in Fig. 12 (96.99 Hz + F_c = 12384.99 Hz). Their magnitudes are nearly equal $189.32 \approx 189.6$ ($\frac{189.32-189.6}{189.32} = -0.15\%$ negligible difference). Hence, signal has been successfully demodulated.

2.4.2 Specification

Method: **Parks-McClellan** optimal FIR filter design
Passband: [0, 4096 Hz]
Stopband: [4596 Hz, $+\infty$)
Passband ripple: 0.005 (smaller than 0.01 because of cascade)
Stopband ripple: 0.01 (-40 dB)

The lowest order can be calculated by `firpmord()` function.

$$N = 153 \tag{12}$$

2.4.3 Frequency Response & Output

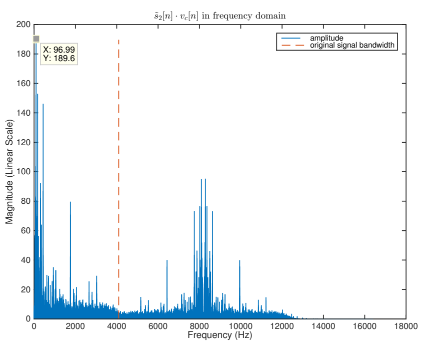

Figure 12: $\tilde{s}_2[n]$ multiplied by carrier

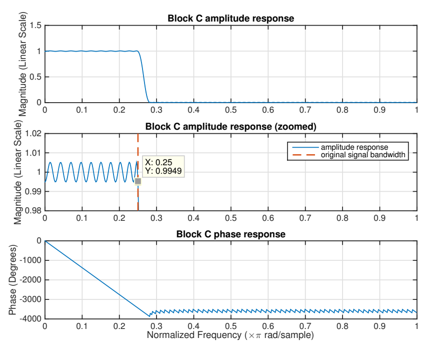

Figure 13: Block C response

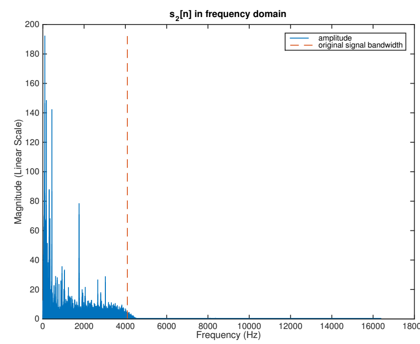

Figure 14: $s_2[n]$ in frequency domain

In Fig. 13, Block C has linear phase. In Fig. 13, block C has ripple less than 0.01. In Fig. 14, high frequency components have been successfully filtered.

3 FIR filters optimization

A filter has linear phase if its frequency response can be written as

$$H(e^{j\omega}) = e^{-j\frac{N}{2}\omega} e^{j\beta} \breve{H}(\omega) \tag{13}$$

where N is the filter order and $\breve{H}(\omega)$ is a real function of ω.

$$\theta(\omega) = \beta - \frac{N}{2}\omega \tag{14}$$

Group delay

$$\tau_g(\omega) = -\frac{d\theta(\omega)}{d\omega} = \frac{N}{2} \tag{15}$$

3.1 Total group delays

$$H_1(z) : N = 184 + 141 = 325 \longrightarrow \tau_g = \frac{325}{2} = 162.5$$
$$\text{ABC cascade} : N = 184 + 140 + 153 = 477 \longrightarrow \tau_g = \frac{477}{2} = 238.5$$

3.2 Total group delays minimization

Minimizing total filter orders is to minimize total group delays because of the proportional relationship.

Filter order can be reduced from two aspects.
1. expand transition region
2. make full use of the ripple range

3.3 New Specifications

3.3.1 Block A

Passband: [0, 6472.8 Hz] and [8272.8 Hz, $+\infty$)
Stopband: [7371.8 Hz, 7373.8 Hz]
Passband ripple: 0.005
Stopband ripple: 0.001 (-60 dB)

3.3.2 Block B

Stopband: [0, 4096 Hz]
Passband: [8192 Hz, $+\infty$)
Stopband ripple: 0.01 (-40 dB)
Passband ripple: 0.005

3.3.3 Block C & D

Passband: [0, 4096 Hz]
Stopband: [8192 Hz, $+\infty$)
Passband ripple: 0.01 (-40 dB)
Stopband ripple: 0.005

3.4 Minimal total group delays

$$H_1(z) : N = 102 + 16 = 118 \longrightarrow \tau_g = \frac{118}{2} = 59$$
$$\text{ABC cascade} : N = 102 + 16 + 18 = 136 \longrightarrow \tau_g = \frac{136}{2} = 68$$

3.5 Frequency response

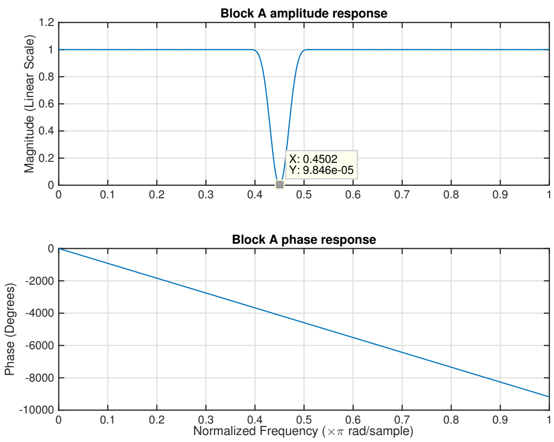

Figure 15: Block A response

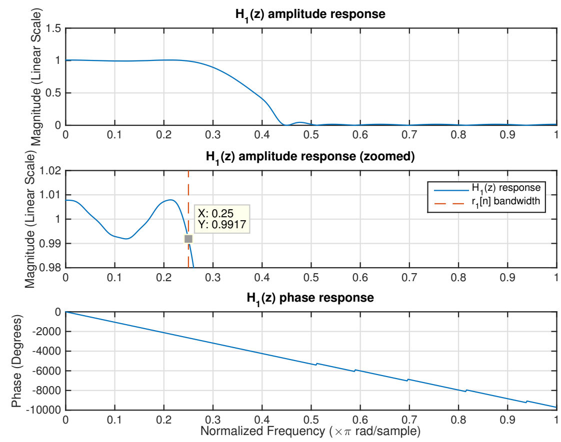

Figure 16: $H_1(z)$ Response

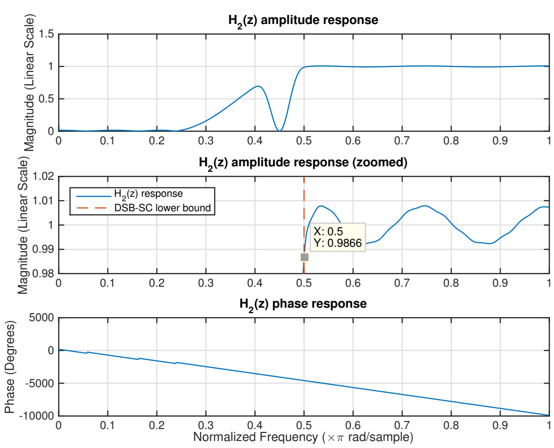

Figure 17: $H_2(z)$ Response

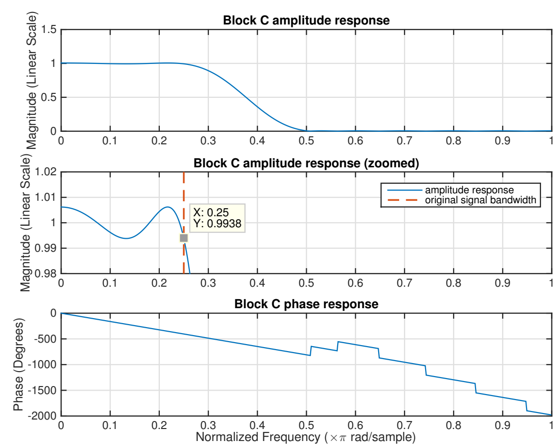

Figure 18: Block C response

In Fig. 15, attenuation $20\log_{10}(9.846 \times 10^{-5})$ = -80.1348 dB < -60 dB. In Fig. 16, the maximal ripple of $H_1(z)$ is $1 - 0.9917 = 0.0083 < 0.01$. In Fig. 17 and Fig. 18, even in the worst situation, the maximal ripple of ABC cascade is $1 - 0.9866 \times 0.9938 = 0.0195 < 2\%$.

4 IIR filters design

4.1 Block A 2nd order notch filter

$$H_{BS}(z) = \frac{1+\alpha}{2}\frac{1-2\beta z^{-1}+z^{-2}}{1-\beta(1+\alpha)z^{-1}+\alpha z^{-2}} \tag{16}$$

We have known the notch frequency $\omega_0 = 0.45\pi$ rad/sample,

$$\beta = \cos(\omega_0) = \mathbf{0.156434} \tag{17}$$

We set

$$B_w = \cos^{-1}(\frac{2\alpha}{1+\alpha^2}) = 0.005\pi \text{ rad/sample} \tag{18}$$

Given $0 < \alpha < 1$

$$\alpha = \frac{1}{\cos(B_w)} - \sqrt{\frac{1}{(\cos(B_w))^2} - 1} = \mathbf{0.984414} \tag{19}$$

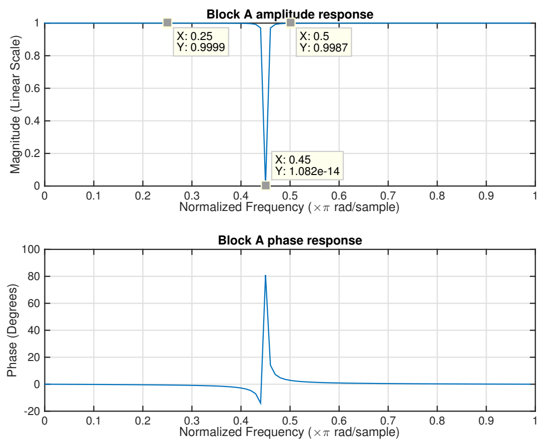

Figure 19: Block A response

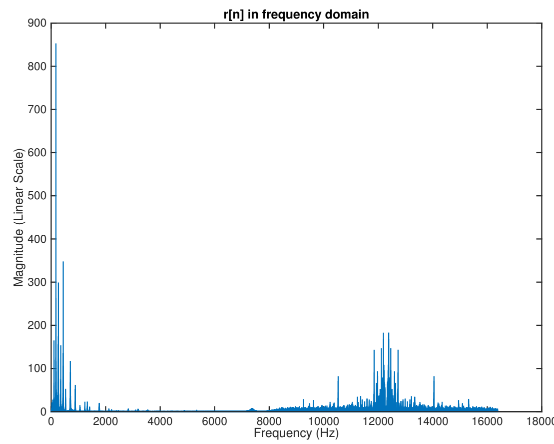

Figure 20: $r[n]$ in frequency domain

As can be seen in Figure 19 and Figure 20, $w[n]$ at 7372.8 Hz (0.45π rad/sample) has been attenuated by $20\log_{10}(1.082\times10^{-14})$ = -279.3155 dB, hence the design satisfies the -60 dB gain requirement. In addition, the notch filter has gain of 0.9999 and 0.9987 at 0.25π rad/sample and 0.5π rad/sample respectively.

$$A_{0.25\pi} = 0.9999 \tag{20}$$

$$A_{0.5\pi} = 0.9987 \tag{21}$$

Due to the monotonicity, in the bands of [0, 4096 Hz] and [8192 Hz, $+\infty$), these two ripples are maxima.

4.2 Block D Lowpass filter

4.2.1 Specification

Model: **Chebyshev Type I**
Passband ripple: 0.0099 (explained by Eq. 22)
Stopband attenuation: 0.001 (-60 dB)
Passband: [0, 0.25π rad/sample]
Stopband: [0.255π rad/sample, π rad/sample)

The total ripple for the frequency band occupied by $r_1[n]$ must be less than 0.01. The gain of block A at 0.25π rad/sample is 0.9999 (Eq. 20). Hence, the ripple of block D is

$$1 - \frac{1-0.01}{0.9999} = 0.0099 \tag{22}$$

The lowest order can be calculated by `cheb1ord()` function.

$$N = 15 \tag{23}$$

4.2.2 Frequency Response & Output

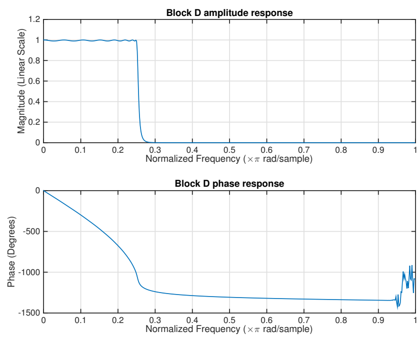

Figure 21: Block D response

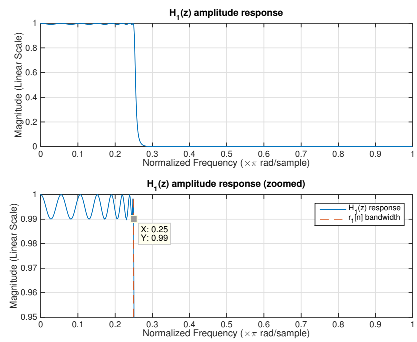

Figure 22: $H_1(z)$ Response

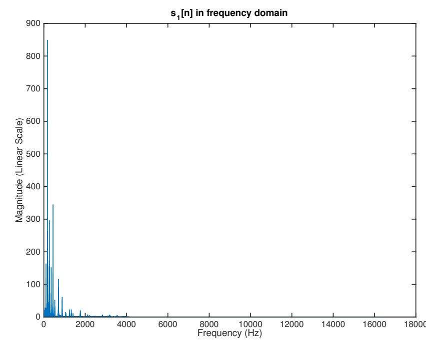

Figure 23: $s_1[n]$ in frequency domain

In Fig. 22, $H_1(z)$ has ripple less than 0.01. In Fig. 23, high frequency components have been successfully filtered.

4.3 Block B Highpass filter

4.3.1 Specification

Model: **Butterworth**
Stopband attenuation: 0.001 (-60 dB)
Passband ripple: 0.0088 (explained by Eq. 24 and Eq. 25)
Stopband: [0, 0.45π rad/sample]
Passband: [0.5π rad/sample, π rad/sample)

$H_2(z)$ is the filter obtained by cascading the filters in block A and B.

We specify the ripple of $H_2(z)$ and the ripple of block C are both 0.01.

$$1 - (1 - 0.01) \times (1 - 0.01) = 0.0199 < 2\% \tag{24}$$

As a result, the total ripple for the cascade of the filters in block A, B and C is such that the the magnitude spectrum of the demodulated signal $s_2[n]$ is within 2% of the magnitude of the original signal (the one which was modulated) for each frequency.

Given the gain of block A at 0.5π rad/sample is 0.9987 (Eq. 21), the ripple of block B is

$$1 - \frac{1 - 0.01}{0.9987} = 0.0088 \tag{25}$$

The lowest order can be calculated by `buttord()` function.

$$N = 57 \tag{26}$$

4.3.2 Frequency Response & Output

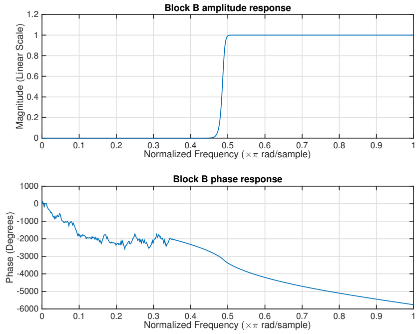

Figure 24: Block B response

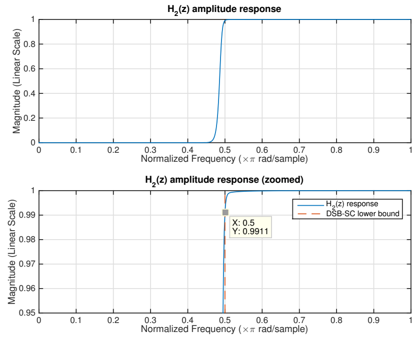

Figure 25: $H_2(z)$ Response

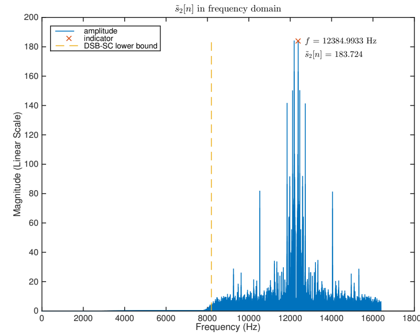

Figure 26: $\tilde{s}_2[n]$ in frequency domain

In Fig. 25, $H_2(z)$ has ripple less than 0.01. In Fig. 26, low frequency components have been successfully filtered.

4.4 Block C Lowpass filter

4.4.1 Demodulation phase shift

Based on the theoretical discussion on page 2, ϕ_2 in Eq.7 can be calculated by `find_phi(s2_tilde)` function.

$$\phi_2 = 0.45\pi \tag{27}$$

For instance, the spike at 12384.99 Hz in Fig. 26 corresponds to the spike at 96.99 Hz in Fig. 27 (96.99 Hz + F_c = 12384.99 Hz). Their magnitudes are nearly equal $183.724 \approx 183.8$ ($\frac{183.724-183.8}{183.724} = -0.041\%$ negligible difference). Hence, signal has been successfully demodulated.

4.4.2 Specification

Model: **Elliptic**
Passband ripple: 0.01 (Eq. 24)
Stopband attenuation: 0.001 (-60 dB)
Passband: [0, 0.25π rad/sample]
Stopband: [0.255π rad/sample, π rad/sample)

The lowest order can be calculated by `ellipord()` function.

$$N = 11 \tag{28}$$

4.4.3 Frequency Response & Output

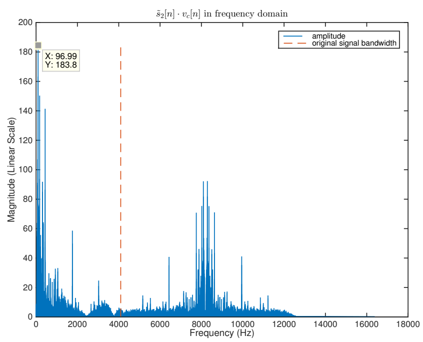

Figure 27: $\tilde{s}_2[n]$ multiplied by carrier

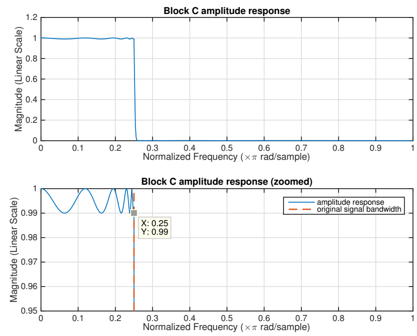

Figure 28: Block C response

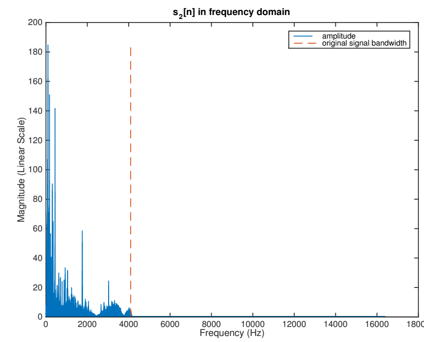

Figure 29: $s_2[n]$ in frequency domain

In Fig. 28, block C has ripple less than 0.01. In Fig. 29, high frequency components have been successfully filtered.

5 FIR and IIR comparison

5.1 Filter parameters comparison

	FIR	IIR
Passband edge frequency	Yes	Yes
Stopband edge frequency	Yes	Yes
Passband desired amplitudes	Yes	
Stopband desired amplitudes	Yes	
Stopband attenuation		Yes
Passband ripple	Yes	Yes
Stopband ripple	Yes	

Table 1: Parameters Comparison

FIR can control the desired amplitude more accurately and flexibly at the cost of more parameters required. Besides, FIR requires one more parameter to control stopband ripple.

5.2 Sound quality comparison

In order to compare, we recorded some speech and compared the outputs of FIR and IIR in a continuous loop. We realized output from IIR filters was more close to the original sound. However, IIR filtering was more time-consuming due to larger filter orders. We thought IIR filters have inherent advantages in terms of sound quality because of **linear phase**.

When input is $e^{j2\pi f_0 t}$, output is

$$|H(f_0)|e^{j2\pi f_0 t + \angle H(f_0)} = |H(f_0)|e^{j2\pi f_0 (t + \frac{\angle H(f_0)}{2\pi f_0})} \tag{29}$$

i.e. output is proportion to input delayed by $-\frac{\angle H(f_0)}{2\pi f_0}$ seconds = *phase delay*.

If phase response is not linear, this depends on f_0.
i.e. system delays Fourier components of inputs by different amounts, depending on frequency. Signal will be distorted.

6 IIR filters optimization

6.1 Methods to minimize filter orders

Filter order can be reduced from three aspects.
1. change filter type
2. expand transition region
3. make full use of the ripple range

6.2 New Specifications

6.2.1 Block B

Model: **Elliptic**
Stopband: $[0, 0.25\pi$ rad/sample$]$
Passband: $[0.5\pi$ rad/sample, π rad/sample$)$
Stopband attenuation: 0.001 (-60 dB)
Passband ripple: 0.0088

6.2.2 Block C & D

Model: **Elliptic**
Passband: $[0, 0.25\pi$ rad/sample$]$
Stopband: $[0.5\pi$ rad/sample, π rad/sample$)$
Passband ripple: 0.0100 (C), 0.0099 (D)
Stopband attenuation: 0.001 (-60 dB)

6.3 Minimal total filter orders

$$H_1(z) : N = 2 + 5 = 7$$
$$\text{ABC cascade} : N = 2 + 5 + 5 = 12$$

6.4 Frequency response

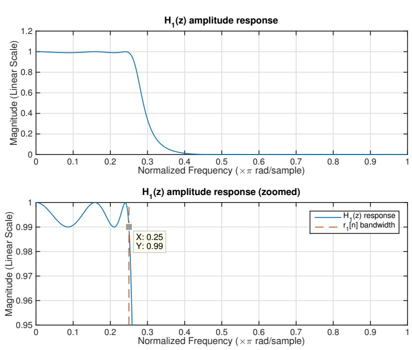

Figure 30: $H_1(z)$ Response

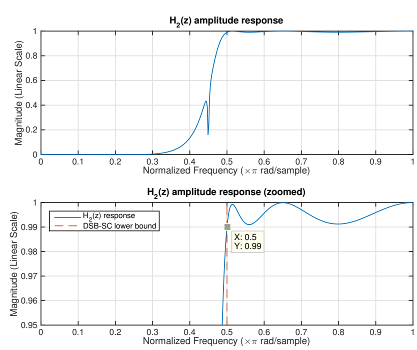

Figure 31: $H_2(z)$ Response

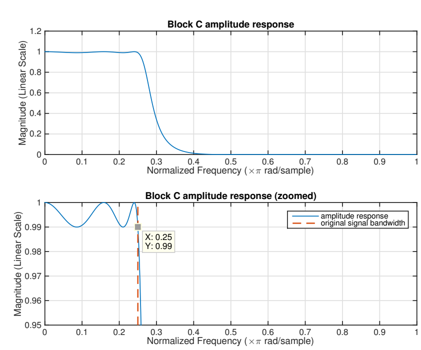

Figure 32: Block C response

In Fig. 30, the maximal ripple of $H_1(z)$ is $1 - 0.99 = 0.01$. In Fig. 31 and Fig. 32, even in the worst situation, the maximal ripple of ABC cascade is $1 - 0.99 \times 0.99 = 0.0199 < 2\%$.

Appendix

find_phi(s2_tilde)

```
function phi = find_phi(s2_tilde)
    F_s = 32.768E3;         % sampling frequency
    F_c = 12.288E3;         % carrier frequency

    N = length(s2_tilde);

    t = 1/F_s * (0:N-1)';
    omegat = 2 * pi * F_c * t;

    phi_vector = (0:0.01:1) * pi;

    tmp = zeros(1, length(phi_vector));

    for k = 1:length(phi_vector)
        v_c = cos(omegat + phi_vector(k));
        y = s2_tilde .* v_c;
        [~, Y] = single_side_FFT(y, F_s);
        tmp(k) = Y(75);     % spike at 96.99Hz
    end

    [~, index] = max(tmp);
    phi = phi_vector(index);
end
```

single_side_FFT(x, F_s)

```
function [frequency_range, X] = single_side_FFT(x, F_s)
        N = length(x);
        X = fft(x);
        X = abs(X);

        X = X(1:ceil(N/2));
        frequency_range = (0:ceil(N/2)-1) / N * F_s;
end
```

analysis.m

```
clear;
close all;

F_s = 32.768E3;
% sampling frequency

load('projsignal1');
N = 25E3;
rs = rs(1:N);
% the first 25000 data points

[frequency_range, Rs] = single_side_FFT(rs, F_s);

stem(frequency_range, Rs, 'marker', 'none');
title('FFT analysis of r_s[n]');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Linear Scale)');

figure;
stem(frequency_range, 20*log10(Rs), 'marker', 'none');
```

```
title('FFT analysis of r_s[n]');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');

fprintf('Sinusoid frequency: F_0 = %.1f Hz\n', frequency_range(Rs==max(Rs)));
```

FIR.m

```
clear;
close all;

notch_F_0 = 7372.8;      % sinusoid noise frequency
F_s = 32.768E3;          % sampling frequency
F_c = 12.288E3;          % carrier frequency

signal_bw = 4096;
r2_lower_bound = F_c - signal_bw;

load('projsignal1');

rs = rs(1:25E3);
% the first 25000 data points

N = length(rs);


%% BlockA notch filter

notch_BW = 1;
% notch filter bandwidth (Hz)

f = [notch_F_0-500 notch_F_0-notch_BW/2 notch_F_0+notch_BW/2 notch_F_0+500];
a = [1 0 1];
dev = [5E-3 1E-3 5E-3];

[nBlockA, fo, ao, w] = firpmord(f, a, dev, F_s);
numBlockA = firpm(nBlockA, fo, ao, w);

[hBlockA, wBlockA] = freqz(numBlockA, 1, 2^10);

figure;
subplot(2, 1, 1);
plot(wBlockA/pi, abs(hBlockA));
title('Block A amplitude response');
ylabel('Magnitude (Linear Scale)');
grid on;

subplot(2, 1, 2);
plot(wBlockA/pi, rad2deg(phase(hBlockA)));
title('Block A phase response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (Degrees)');
grid on;

clear wBlockA hBlockA;


%% BlockA output

r = filter(numBlockA, 1, rs);

[frequency_range, R] = single_side_FFT(r, F_s);
```

```

figure;
stem(frequency_range, R, 'marker', 'none');
title('r[n] in frequency domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Linear Scale)');
clear R;


%% BlockD Lowpass filter

f = [signal_bw signal_bw+500];
a = [1 0];
dev = [5E-3 0.01];

[nBlockD, fo, ao, w] = firpmord(f, a, dev, F_s);
numBlockD = firpm(nBlockD, fo, ao, w);

[hBlockD, wBlockD] = freqz(numBlockD, 1);

figure;
subplot(2, 1, 1);
plot(wBlockD/pi, abs(hBlockD));
title('Block D amplitude response');
ylabel('Magnitude (Linear Scale)');
grid on;

subplot(2, 1, 2);
plot(wBlockD/pi, rad2deg(phase(hBlockD)));
title('Block D phase response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (Degrees)');
grid on;
clear wBlockD hBlockD;


%% H1(z)

[hH1, wH1] = freqz(conv(numBlockA, numBlockD), 1);

figure;
subplot(3, 1, 1);
plot(wH1/pi, abs(hH1));
title('H_1(z) amplitude response');
ylabel('Magnitude (Linear Scale)');
grid on;

subplot(3, 1, 2);
plot(wH1/pi, abs(hH1));
title('H_1(z) amplitude response (zoomed)');
ylabel('Magnitude (Linear Scale)');
grid on;
axis([0 1 0.98 1.02]);
hold on;
plot([0.25 0.25], [0.95 1.05], '--');
legend('H_1(z) response', 'r_1[n] bandwidth');

subplot(3, 1, 3);
plot(wH1/pi, rad2deg(phase(hH1)));
title('H_1(z) phase response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (Degrees)');
```

```
grid on;

clear wH1 hH1;


%% BlockD output

s1 = filter(numBlockD, 1, r);
clear numBlockD;

[frequency_range, S1] = single_side_FFT(s1, F_s);

figure;
stem(frequency_range, S1, 'marker', 'none');
title('s_1[n] in frequency domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Linear Scale)');
clear S1;


%% BlockB Highpass filter

f = [r2_lower_bound-500 r2_lower_bound];
a = [0 1];
dev = [0.01 5E-3];

[nBlockB, fo, ao, w] = firpmord(f, a, dev, F_s);
numBlockB = firpm(nBlockB, fo, ao, w);

[hBlockB, wBlockB] = freqz(numBlockB, 1);

figure;
subplot(2, 1, 1);
plot(wBlockB/pi, abs(hBlockB));
title('Block B amplitude response');
ylabel('Magnitude (Linear Scale)');
grid on;

subplot(2, 1, 2);
plot(wBlockB/pi, rad2deg(phase(hBlockB)));
title('Block B phase response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (Degrees)');
grid on;
clear wBlockB hBlockB;


%% H2(z) cascading BlockA and BlockB

[hH2, wH2] = freqz(conv(numBlockA, numBlockB), 1);
clear numBlockA;

figure;
subplot(3, 1, 1);
plot(wH2/pi, abs(hH2));
title('H_2(z) amplitude response');
ylabel('Magnitude (Linear Scale)');
grid on;

subplot(3, 1, 2);
plot(wH2/pi, abs(hH2));
title('H_2(z) amplitude response (zoomed)');
```

```
ylabel('Magnitude (Linear Scale)');
grid on;
axis([0 1 0.98 1.02]);
hold on;
plot([0.5 0.5], [0.95 1.05], '--');
legend('H_2(z) response', 'DSB-SC lower bound', 'Location', 'northwest');

subplot(3, 1, 3);
plot(wH2/pi, rad2deg(phase(hH2)));
title('H_2(z) phase response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (Degrees)');
grid on;

clear wH2 hH2;


%% BlockB output

s2_tilde = filter(numBlockB, 1, r);
clear numBlockB;

[frequency_range, S2_tilde] = single_side_FFT(s2_tilde, F_s);

figure;
stem(frequency_range, S2_tilde, 'marker', 'none');
title('$\tilde{s}_2[n]$ in frequency domain', 'Interpreter', 'latex');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Linear Scale)');

% [~, index] = max(S2_tilde);
index = 9450;
text(frequency_range(index) + 0.02 * max(frequency_range), S2_tilde(index), ['$f$ = ' num2str(
    frequency_range(index)) ' Hz'], 'Interpreter', 'latex');
text(frequency_range(index) + 0.02 * max(frequency_range), S2_tilde(index) * 0.95, ['$\tilde{s}_2[n]$ = '
    num2str(S2_tilde(index))], 'Interpreter', 'latex');
hold on;
plot(frequency_range(index), S2_tilde(index), 'x');

plot([r2_lower_bound r2_lower_bound], [min(S2_tilde) max(S2_tilde)], '--');
legend('amplitude', 'indicator', 'DSB-SC lower bound', 'Location', 'northwest');

clear S2_tilde;


%% BlockC Demodulator

phi = find_phi(s2_tilde);
fprintf('Demodulation carrier signal: phi = %f * pi\n', phi/pi);

t = 1/F_s * (0:N-1)';
v_c = cos(2 * pi * F_c * t + phi);
y = s2_tilde .* v_c;
clear t v_c;

[frequency_range, Y] = single_side_FFT(y, F_s);

figure;
stem(frequency_range, Y, 'marker', 'none');
title('$\tilde{s}_2[n] \cdot v_c[n]$ in frequency domain', 'Interpreter', 'latex');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Linear Scale)');
```

```
hold on;
plot([signal_bw signal_bw], [min(Y) max(Y)], '--');
legend('amplitude', 'original signal bandwidth');

clear Y;


%% BlockC Lowpass filter

f = [signal_bw signal_bw+500];
a = [1 0];
dev = [5E-3 5E-3];

[nBlockC, fo, ao, w] = firpmord(f, a, dev, F_s);
numBlockC = firpm(nBlockC, fo, ao, w);
clear fo ao w;

[hBlockC, wBlockC] = freqz(numBlockC, 1);

figure;
subplot(3, 1, 1);
plot(wBlockC/pi, abs(hBlockC));
title('Block C amplitude response');
ylabel('Magnitude (Linear Scale)');
grid on;

subplot(3, 1, 2);
plot(wBlockC/pi, abs(hBlockC));
title('Block C amplitude response (zoomed)');
ylabel('Magnitude (Linear Scale)');
grid on;
axis([0 1 0.98 1.02]);
hold on;
plot([0.25 0.25], [0.95 1.05], '--', 'linewidth', 1.5);
legend('amplitude response', 'original signal bandwidth');

subplot(3, 1, 3);
plot(wBlockC/pi, rad2deg(phase(hBlockC)));
title('Block C phase response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (Degrees)');
grid on;

clear wBlockC hBlockC;


%% BlockC output

s2 = filter(numBlockC, 1, y);
clear numBlockC;

[frequency_range, S2] = single_side_FFT(s2, F_s);

figure;
stem(frequency_range, S2, 'marker', 'none');
title('s_2[n] in frequency domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Linear Scale)');

hold on;
plot([signal_bw signal_bw], [min(S2) max(S2)], '--');
```

```
legend('amplitude', 'original signal bandwidth');

clear S2;
```

IIR.m

```
clear;
close all;

notch_F_0 = 7372.8;      % sinusoid noise frequency
F_s = 32.768E3;          % sampling frequency
F_c = 12.288E3;          % carrier frequency

signal_bw = 4096;
r2_lower_bound = F_c - signal_bw;

load('projsignal1');

rs = rs(1:25E3);
% the first 25000 data points

N = length(rs);


%% BlockA 2nd order notch filter

notch_BW = 5E-3 * pi;
% notch filter bandwidth

notch_omega_0 = 2 * pi * notch_F_0 / F_s;

cosine = cos(notch_BW);
notch_alpha = 1/cosine - sqrt(1/cosine^2 - 1);
clear cosine;

notch_beta = cos(notch_omega_0);

fprintf('Sinusoid frequency: F_0 = %.1f Hz\n', notch_F_0);
fprintf('Sinusoid frequency: omega_0 = %f * pi rad/sample\n', notch_omega_0/pi);
fprintf('Notch filter: alpha = %f\tbeta = %f\n', notch_alpha, notch_beta);

numBlockA = ((1+notch_alpha) / 2) * [1 -2*notch_beta 1];
denBlockA = [1 -notch_beta*(1+notch_alpha) notch_alpha];

[hBlockA, wBlockA] = freqz(numBlockA, denBlockA, 1E2);

figure;
subplot(2, 1, 1);
plot(wBlockA/pi, abs(hBlockA));
title('Block A amplitude response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (Linear Scale)');
grid on;

subplot(2, 1, 2);
plot(wBlockA/pi, rad2deg(phase(hBlockA)));
title('Block A phase response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (Degrees)');
grid on;

[~, index] = min(abs(wBlockA(:) - signal_bw * 2 * pi / F_s));
```

```
ripple1 = abs(hBlockA(index));

[~, index] = min(abs(wBlockA(:) - r2_lower_bound * 2 * pi / F_s));
ripple2 = abs(hBlockA(index));

clear wBlockA hBlockA;


%% BlockA output

r = filter(numBlockA, denBlockA, rs);

[frequency_range, R] = single_side_FFT(r, F_s);

figure;
stem(frequency_range, R, 'marker', 'none');
title('r[n] in frequency domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Linear Scale)');
clear R;


%% BlockD Lowpass filter

Rp = 0.99 / ripple1;        % Passband ripple
Rp = -20 * log10(Rp);       % Passband ripple in decibels
Rs = 0.001;                 % stopband attenuation
Rs = -20 * log10(Rs);       % stopband attenuation in decibels

Wp = signal_bw;             % passband edge frequency (Hz)
Wp = 2 * Wp / F_s;          % normalized passband edge frequency (*pi)

Ws = Wp + 0.05;             % normalized stopband edge frequency (*pi)

[nBlockD, ~] = cheb1ord(Wp, Ws, Rp, Rs);
[numBlockD, denBlockD] = cheby1(nBlockD, Rp, Wp, 'low');

[hBlockD, wBlockD] = freqz(numBlockD, denBlockD);

figure;
subplot(2, 1, 1);
plot(wBlockD/pi, abs(hBlockD));
title('Block D amplitude response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (Linear Scale)');
grid on;

subplot(2, 1, 2);
plot(wBlockD/pi, rad2deg(phase(hBlockD)));
title('Block D phase response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (Degrees)');
grid on;
clear wBlockD hBlockD;


%% H1(z)

[hH1, wH1] = freqz(conv(numBlockA, numBlockD), conv(denBlockA, denBlockD));

figure;
subplot(2, 1, 1);
```

```
plot(wH1/pi, abs(hH1));
title('H_1(z) amplitude response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (Linear Scale)');
grid on;

subplot(2, 1, 2);
plot(wH1/pi, abs(hH1));
title('H_1(z) amplitude response (zoomed)');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (Linear Scale)');
grid on;
axis([0 1 0.95 1]);
hold on;
plot([0.25 0.25], [0.95 1], '--');
legend('H_1(z) response', 'r_1[n] bandwidth');
clear wH1 hH1;


%% BlockD output

s1 = filter(numBlockD, denBlockD, r);
clear numBlockD denBlockD;

[frequency_range, S1] = single_side_FFT(s1, F_s);

figure;
stem(frequency_range, S1, 'marker', 'none');
title('s_1[n] in frequency domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Linear Scale)');
clear S1;


%% BlockB Highpass filter

Wp = r2_lower_bound * 2 / F_s;  % Passband corner frequency (* pi)
Ws = Wp - 0.05;                 % Stopband corner frequency (* pi)
Rp = 0.99 / ripple2;            % Passband ripple
Rp = -20 * log10(Rp);           % Passband ripple in decibels
Rs = 0.001;                     % Stopband attenuation
Rs = -20 * log10(Rs);           % Stopband attenuation in decibels

[nBlockB, Wn] = buttord(Wp, Ws, Rp, Rs);
[numBlockB, denBlockB] = butter(nBlockB, Wn, 'high');
clear Wn;

[hBlockB, wBlockB] = freqz(numBlockB, denBlockB);

figure;
subplot(2, 1, 1);
plot(wBlockB/pi, abs(hBlockB));
title('Block B amplitude response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (Linear Scale)');
grid on;

subplot(2, 1, 2);
plot(wBlockB/pi, rad2deg(phase(hBlockB)));
title('Block B phase response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Phase (Degrees)');
```

```
grid on;
clear wBlockB hBlockB;


%% H2(z) cascading BlockA and BlockB

[hH2, wH2] = freqz(conv(numBlockA, numBlockB), conv(denBlockA, denBlockB));
clear numBlockA denBlockA;

figure;
subplot(2, 1, 1);
plot(wH2/pi, abs(hH2));
title('H_2(z) amplitude response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (Linear Scale)');
grid on;

subplot(2, 1, 2);
plot(wH2/pi, abs(hH2));
title('H_2(z) amplitude response (zoomed)');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (Linear Scale)');
grid on;
axis([0 1 0.95 1]);
hold on;
plot([0.5 0.5], [0.95 1], '--');
legend('H_2(z) response', 'DSB-SC lower bound');
clear wH2 hH2;


%% BlockB output

s2_tilde = filter(numBlockB, denBlockB, r);
clear numBlockB denBlockB;

[frequency_range, S2_tilde] = single_side_FFT(s2_tilde, F_s);

figure;
stem(frequency_range, S2_tilde, 'marker', 'none');
title('$\tilde{s}_2[n]$ in frequency domain', 'Interpreter', 'latex');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Linear Scale)');

index = 9450;
text(frequency_range(index) + 0.02 * max(frequency_range), S2_tilde(index), ['$f$ = ' num2str(
    frequency_range(index)) ' Hz'], 'Interpreter', 'latex');
text(frequency_range(index) + 0.02 * max(frequency_range), S2_tilde(index) * 0.95, ['$\tilde{s}_2[n]$ = '
    num2str(S2_tilde(index))], 'Interpreter', 'latex');
hold on;
plot(frequency_range(index), S2_tilde(index), 'x');

plot([r2_lower_bound r2_lower_bound], [min(S2_tilde) max(S2_tilde)], '--');
legend('amplitude', 'indicator', 'DSB-SC lower bound', 'Location', 'northwest');

clear S2_tilde;


%% BlockC Demodulator

phi = find_phi(s2_tilde);
fprintf('Demodulation carrier signal: phi = %f * pi\n', phi/pi);

```

```
t = 1/F_s * (0:N-1)';
v_c = cos(2 * pi * F_c * t + phi);
y = s2_tilde .* v_c;
clear t v_c;

[frequency_range, Y] = single_side_FFT(y, F_s);

figure;
stem(frequency_range, Y, 'marker', 'none');
title('$\tilde{s}_2[n] \cdot v_c[n]$ in frequency domain', 'Interpreter', 'latex');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Linear Scale)');

hold on;
plot([signal_bw signal_bw], [min(Y) max(Y)], '--');
legend('amplitude', 'original signal bandwidth');

clear Y;


%% BlockC Lowpass filter

Rp = 0.99;              % peak-to-peak passband ripple
Rp = -20 * log10(Rp);   % decibels of peak-to-peak passband ripple
Rs = 0.001;             % stopband attenuation
Rs = -20 * log10(Rs);   % decibels of stopband attenuation down from the peak passband value

Wp = signal_bw;         % passband edge frequency (Hz)
Wp = 2 * Wp / F_s;      % normalized passband edge frequency (*pi)

Ws = Wp + 0.01;         % normalized stopband edge frequency (*pi)

[nBlockC, ~] = ellipord(Wp, Ws, Rp, Rs);
[numBlockC, denBlockC] = ellip(nBlockC, Rp, Rs, Wp, 'low');
clear Rp Rs Wp Ws;

[hBlockC, wBlockC] = freqz(numBlockC, denBlockC);

figure;
subplot(2, 1, 1);
plot(wBlockC/pi, abs(hBlockC));
title('Block C amplitude response');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (Linear Scale)');
grid on;

subplot(2, 1, 2);
plot(wBlockC/pi, abs(hBlockC));
title('Block C amplitude response (zoomed)');
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (Linear Scale)');
grid on;
axis([0 1 0.95 1]);
hold on;
plot([0.25 0.25], [0.95 1], '--', 'linewidth', 1.5);
legend('amplitude response', 'original signal bandwidth');
clear wBlockC hBlockC;


%% BlockC output

s2 = filter(numBlockC, denBlockC, y);
```

```
clear numBlockC denBlockC;

[frequency_range, S2] = single_side_FFT(s2, F_s);

figure;
stem(frequency_range, S2, 'marker', 'none');
title('s_2[n] in frequency domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude (Linear Scale)');

hold on;
plot([signal_bw signal_bw], [min(S2) max(S2)], '--');
legend('amplitude', 'original signal bandwidth');

clear S2;
```