ECE310 – Project 3

Jonathan Lam

November 8, 2020

1 Filter specs

Desired IIR digital filter specs

```
%% filter specs
Wp = 2500 / (fs/2);
Ws = 4000 / (fs/2);
Rp = 3;
Rs = 95;
```

Desired FIR filter specs

```
% ripple needs to be in linear units for the FIR spec
% for passband, it's the difference between 1 and the linear value of the
% attentuation factor
Rp_linear_hat = 1-10^(-Rp/20);
% for stopband, it's the linear value of the attenuation factor
Rs_linear_hat = 10^(-Rs/20);

% using the equations from exercis 7.3 in the textbook to convert to the
% FIR spec (these are all in linear units):
% Rp = Rp_hat/(2-Rp_hat)
% Rs = 2Rs_hat/(2-Rp_hat)
Rp_linear = Rp_linear_hat/(2-Rp_linear_hat);
Rs_linear = 2*Rs_linear_hat/(2-Rp_linear_hat);

f = [Wp Ws];
a = [1 0];
```

2 Filter design and implementation

Butterworth filter. The multiplication by a factor of 100 is used to produce the desired 40dB gain the passband. The filter is implemented using second-order-sections and a direct-form II transposed implementation.

```
%% butterworth
[n, Wn] = buttord(Wp, Ws, Rp, Rs);
[z, p, k] = butter(n, Wn);
k = k * 100;
```

```
%% implementation as df2t sos cascaded
butter_filter = dfilt.df2tsos(zp2sos(z, p, k));
butter_filtered = filter(butter_filter, noisy);
```

Chebyshev types I and II filters

```
%% cheby1
[n, cheby1_Wp] = cheb1ord(Wp, Ws, Rp, Rs);
[z, p, k] = cheby1(n, Rp, cheby1_Wp);
k = k * 100;

cheby1_filter = dfilt.df2sos(zp2sos(z, p, k));
cheby1_filtered = filter(cheby1_filter, noisy);

%% cheby2
[n, cheby2_Ws] = cheb2ord(Wp, Ws, Rp, Rs);
[z, p, k] = cheby2(n, Rs, cheby2_Ws);
k = k * 100;

cheby2_filter = dfilt.df2sos(zp2sos(z, p, k));
cheby2_filtered = filter(cheby2_filter, noisy);
```

Elliptic filter

```
%% elliptic
[n, ellip_Wp] = ellipord(Wp, Ws, Rp, Rs);
[z, p, k] = ellip(n, Rp, Rs, ellip_Wp);
k = k * 100;

ellip_filter = dfilt.df2sos(zp2sos(z, p, k));
ellip_filtered = filter(ellip_filter, noisy);
```

Parks-McClellan filter. Note that the +6 is a fudge factor; documentation says if the order calculated from firpmord doesn't match the spec, try increasing the order. (See: https://www.mathworks.com/help/signal/ref/firpm.html) The division by max(abs(H)) and subsequent multiplication by 100 is used to scale the maximum gain in the passband down to 0dB and then up to the desired 40dB.

```
%% parks mclellan
pm_filter = firpm(n+6, fo, ao, w);
H = freqz(pm_filter);
pm_filter = pm_filter / max(abs(H)) * 100;

pm_filtered = conv(pm_filter, noisy);
```

Kaiser filter

```
%% kaiser
[n, Wn, beta, ftype] = kaiserord(f, a, [Rp_linear Rs_linear]);
kaiser_filter = fir1(n, Wn, ftype, kaiser(n+1, beta), 'noscale');
H = freqz(kaiser_filter);
kaiser_filter = kaiser_filter / max(abs(H)) * 100;

kaiser_filtered = conv(kaiser_filter, noisy);
```

3 Plotting code

Plot filter given its zpk (for IIR filters). Note that the zpk form is used for zplane, since it is most stable. However, the impz, freqz, and grpdelay don't support the zpk form for its input arguments, so the second-order-systems matrix is used instead. (The SOS is also used to implement the IIR filters.)

```
function sos = plot_filter_zpk(z, p, k, name)
        sos = zp2sos(z, p, k);

        fig = figure('Visible', 'Off');
        tiledlayout(2, 2, 'TileSpacing', 'compact');

        nexttile();
        [h, t] = impz(sos, 100);
        stem(t, h);
        title('Impulse Response');
        ylabel('Amplitude');
        xlabel('n (samples)');

        nexttile();
        [H, w] = freqz(sos);
        plot(w, 20*log10(abs(H)));
        title('Frequency Respose');
        ylabel('Magnitude (dB)');
        xlabel('Digital Frequency');

        nexttile();
        zplane(z, p, k);
        title('Pole-Zero Plot');

        nexttile();
        grpdelay(sos);
        title('Group Delay');

        set(fig, 'PaperUnits', 'centimeters');
        set(fig, 'PaperPosition', [0 0 30 20]);
        saveas(fig, sprintf('fig_%s.eps', name));
end
```

Plot filter given its impulse response (for FIR filters).

```
function plot_h(h, name)
        fig = figure('Visible', 'Off');
        tiledlayout(2, 2, 'TileSpacing', 'compact');

        nexttile();
        stem(h);
        title('Impulse Response');
        ylabel('Amplitude');
        xlabel('n (samples)');

        nexttile();
        [H, w] = freqz(h);
        plot(w, 20*log10(abs(H)));
        title('Frequency Respose');
        ylabel('Magnitude (dB)');
```

```
        xlabel('Digital Frequency');

        nexttile();
        zplane(h);
        title('Pole-Zero Plot');

        nexttile();
        grpdelay(h);
        title('Group Delay');

        set(fig, 'PaperUnits', 'centimeters');
        set(fig, 'PaperPosition', [0 0 30 20]);
        saveas(fig, sprintf('fig_%s.eps', name));
end
```

4 Plots

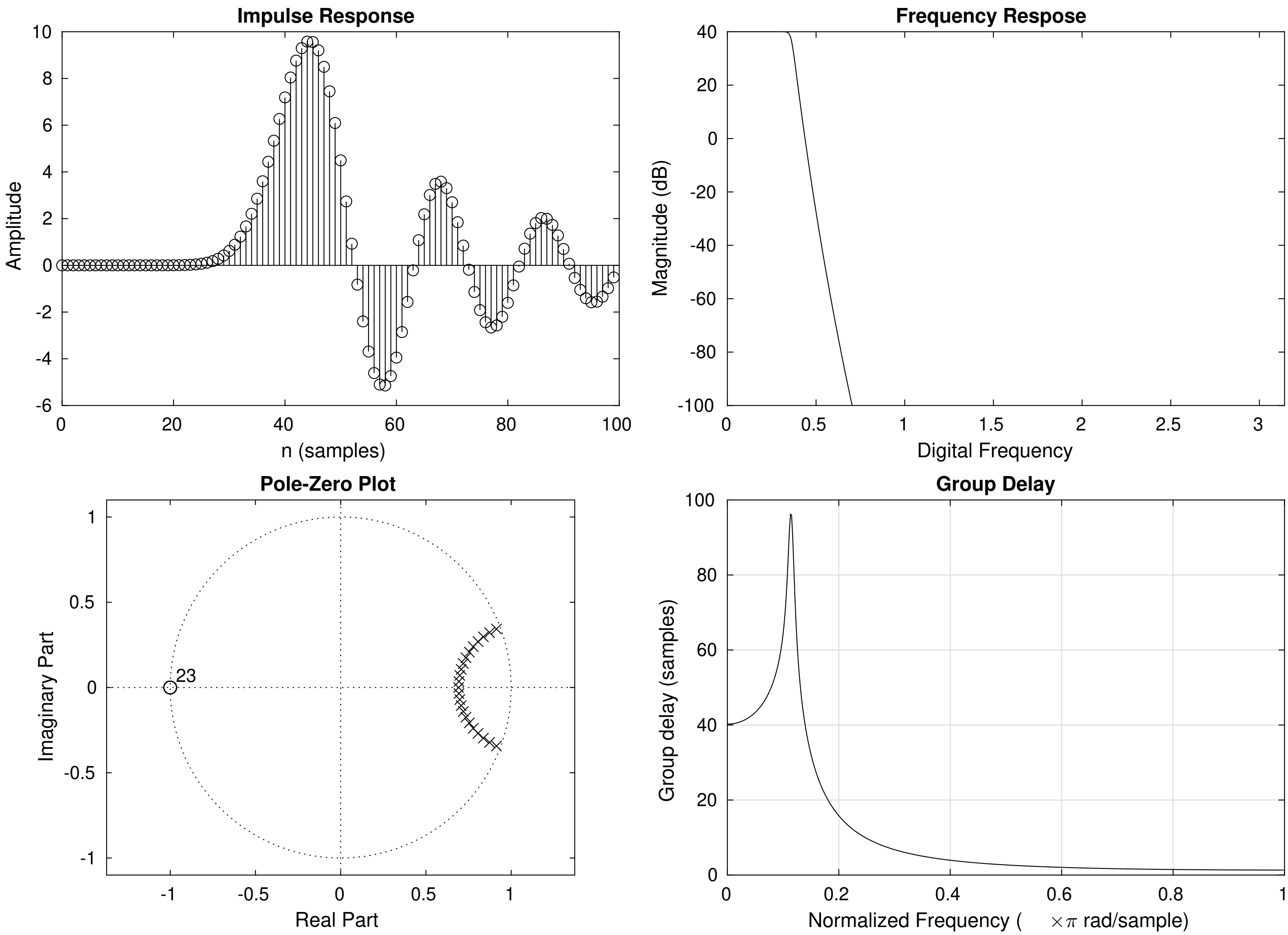

Figure 1: Butterworth filter

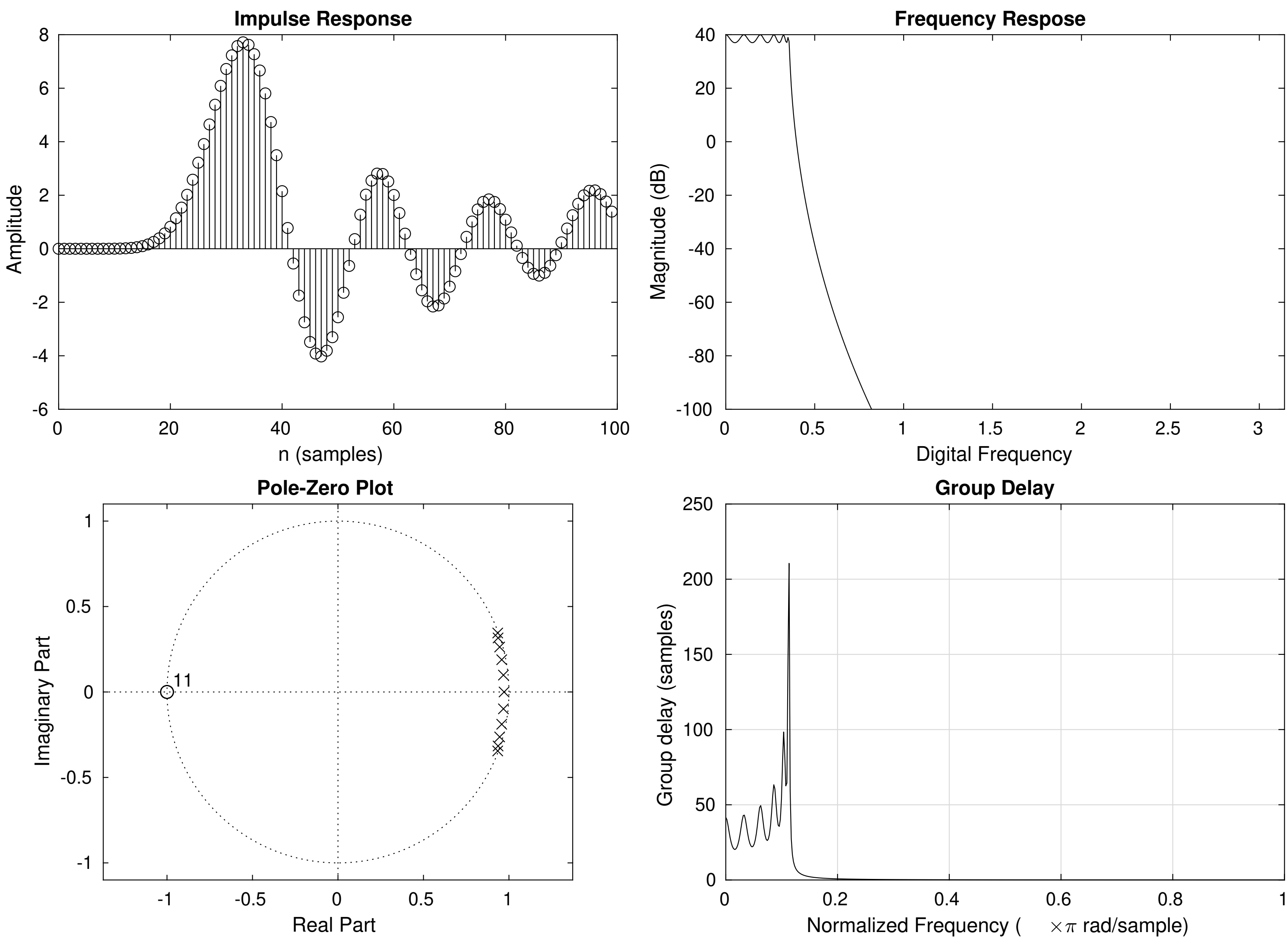

Figure 2: Chebyshev Type I Filter

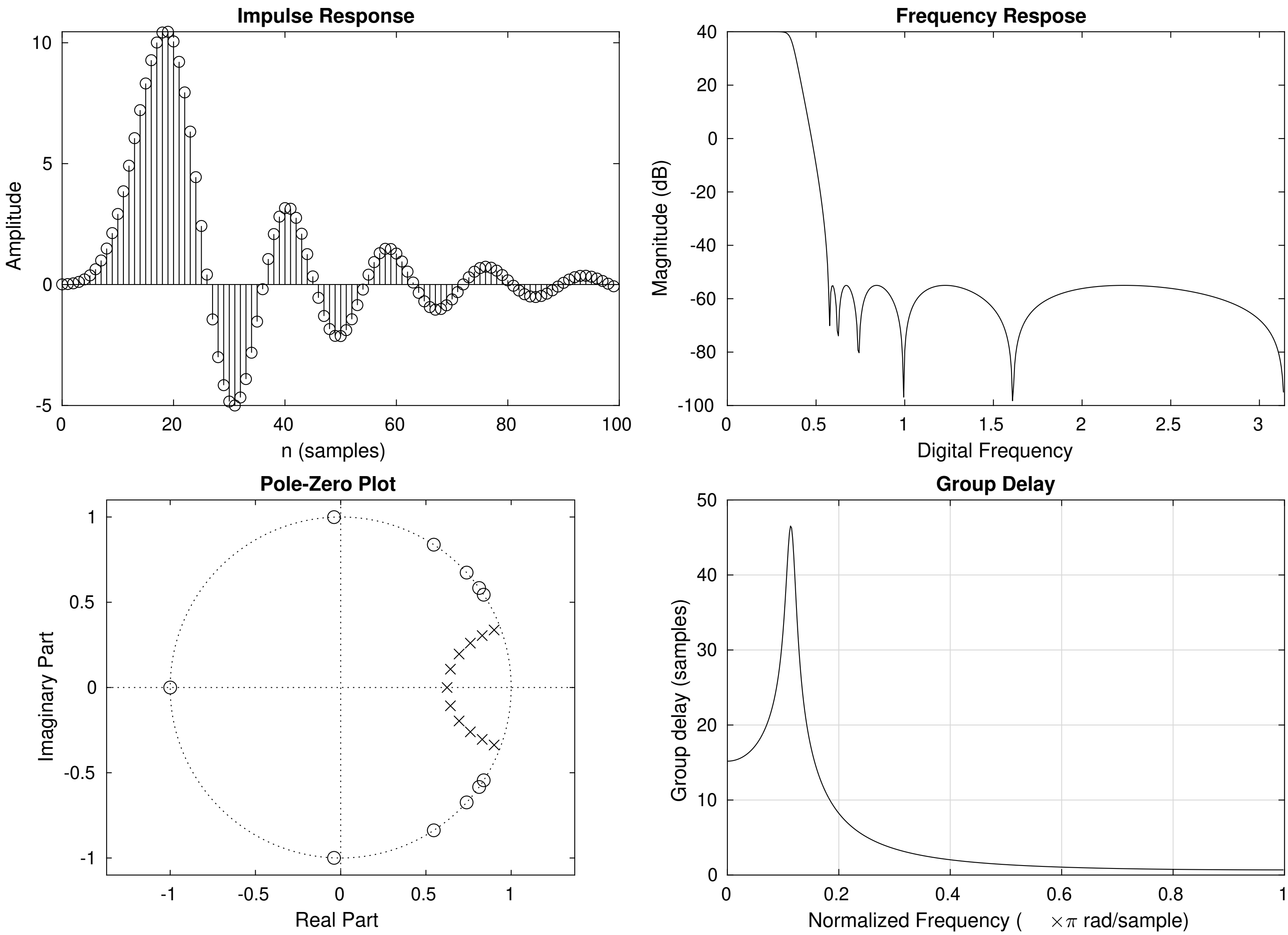

Figure 3: Chebyshev Type II Filter

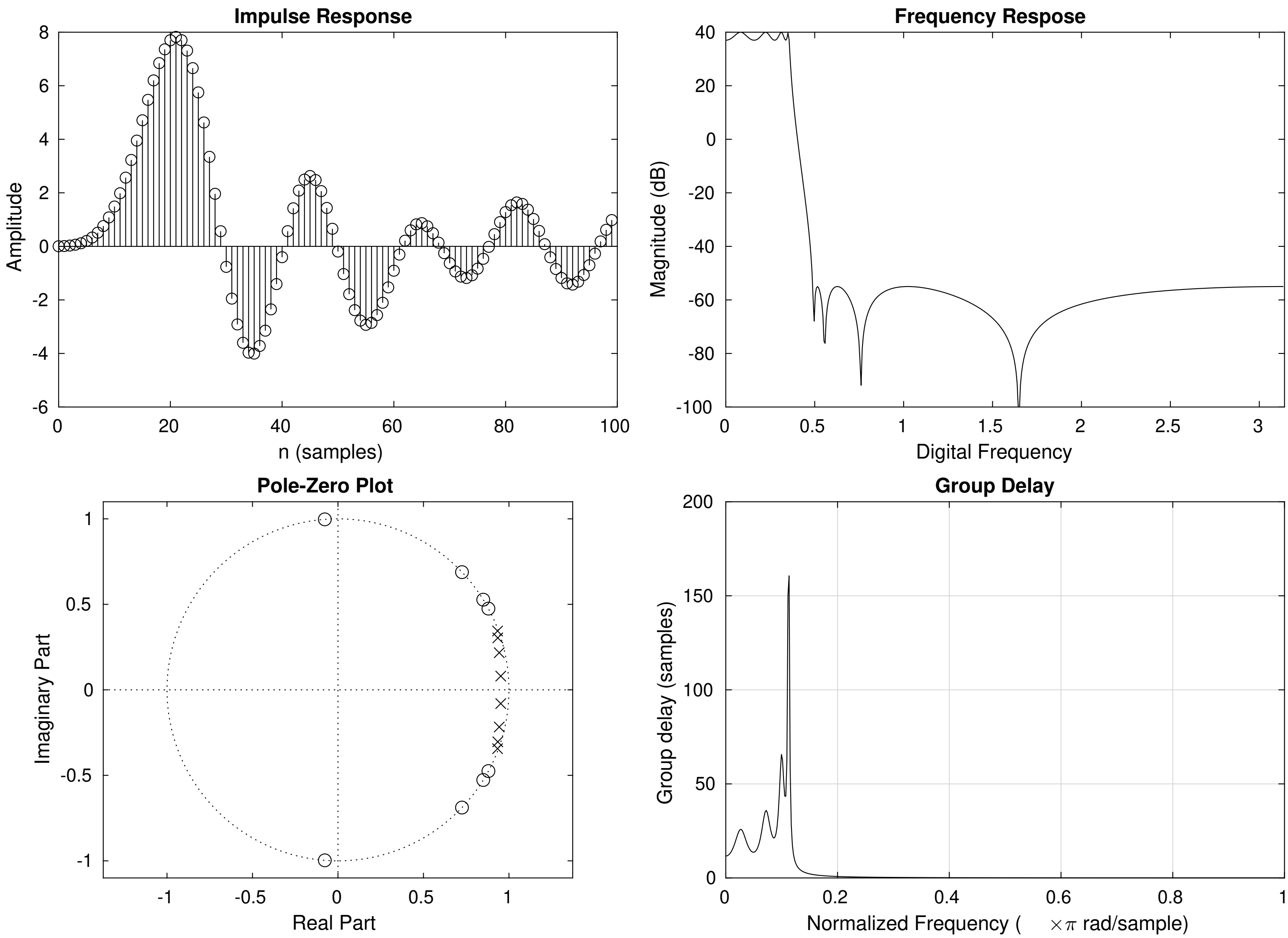

Figure 4: Elliptical Filter

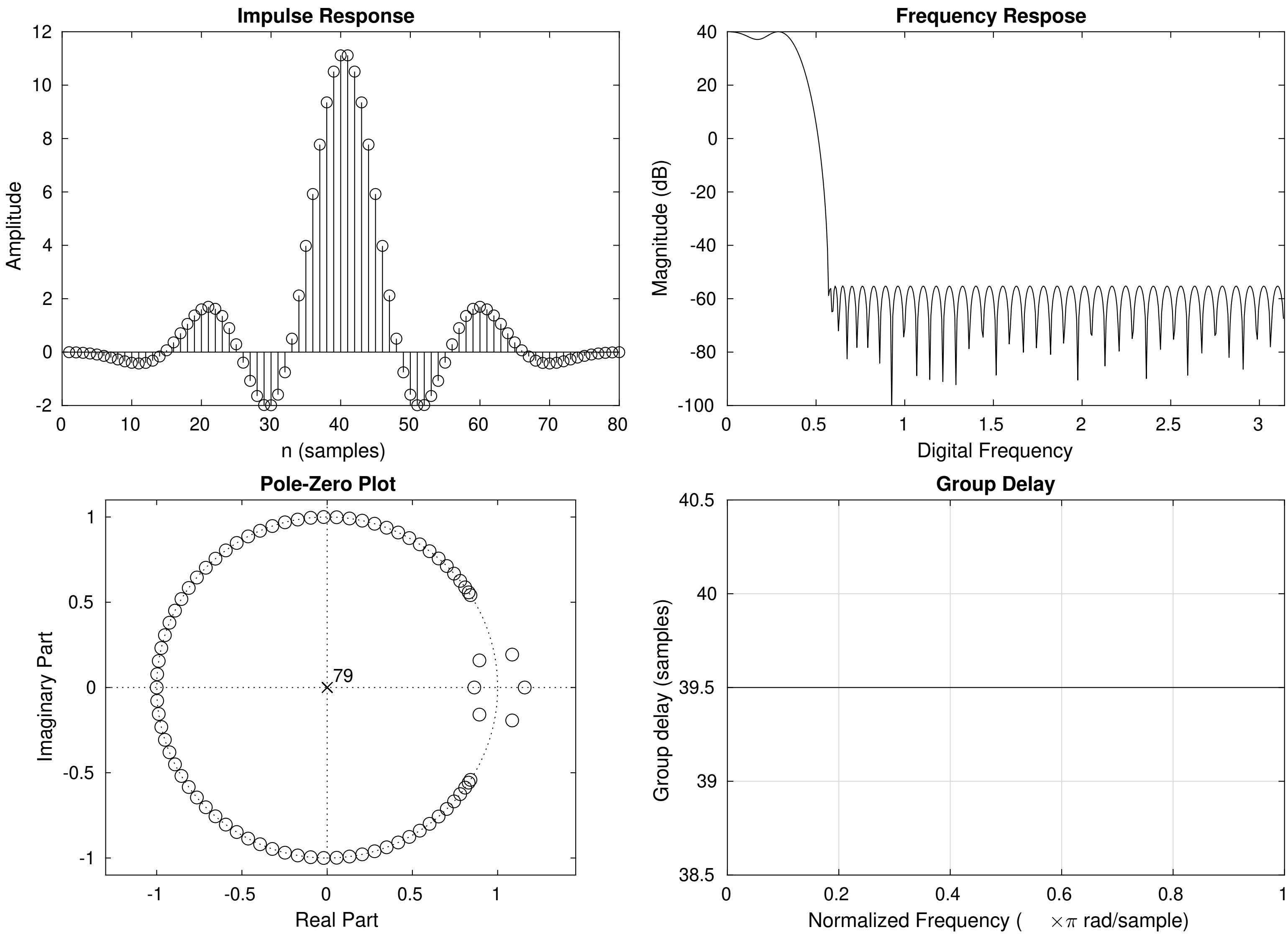

Figure 5: Parks-McClellan Filter

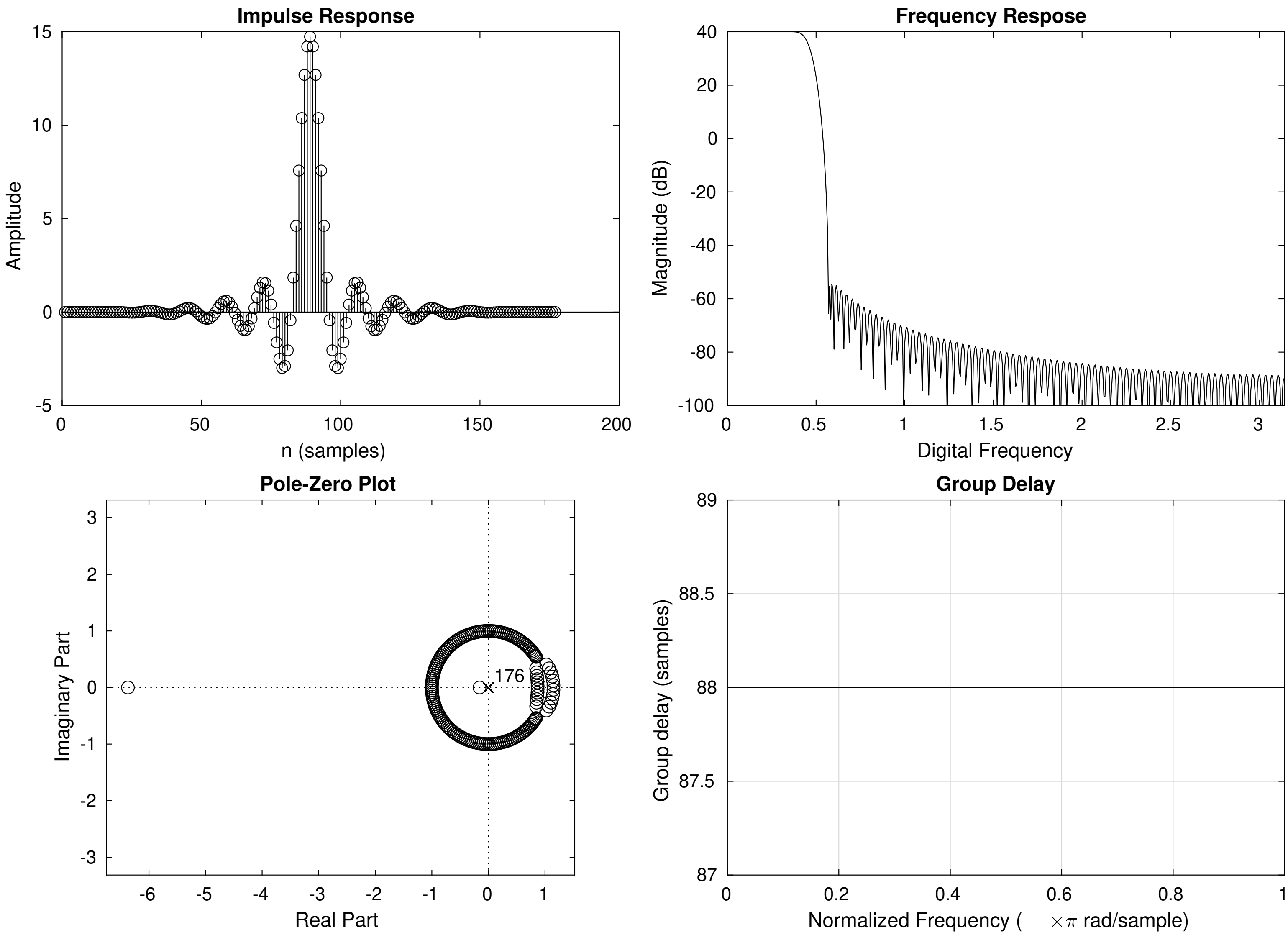

Figure 6: Kaiser Window Filter