임베디드 신호처리 실습
결과리포트
LAB-7

전자공학부 임베디드시스템
2016146048 한 대성
2014146012 박동훈
김수민 교수님

1. 아날로그 프로토타입 필터

```
================================================================================
n = linspace(-pi,pi,360);
t = linspace(0, 2, 1000);

[zero, pole, k] = buttap(5); % Butterworth filter
[zero2, pole2, k2] = cheb1ap(5, 10); % Chebyshev Type_1 filter
[zero3, pole3, k3] = cheb2ap(5, 30); % Chebyshev Type_2 filter
[zero4, pole4, k4] = ellipap(5, 10, 30); % Ellipitic filter
Zero = [zero3 zero4]; % zero,zero2는 빈 영점이 없기 때문에 제외했다.
Pole = [pole pole2 pole3 pole4]; % 각 filter별 극점 배열

figure(1)
for i = 1 : 4
    subplot(2,2,i);
    plot(cos(n), sin(n), '--k'); % 반지름 1인 단위원
    hold on; grid on; % 점선 표기 및 그래프 중첩을 위한 구현
    % 각 pole값별 실수 및 허수 값
    plot(real(Pole(:,i)), imag(Pole(:,i)) , 'rx', 'Markersize',20);

    if(i == 1) % Butterworth filter 일때 zero 값은 없기 때문에 따로 구현
        plot(real(zero), imag(zero) , 'o', 'Markersize',20);
    end

    if(i == 2) % Chebyshev Type_1 filter 일때 zero 값은 없기 때문에 따로 구현
        plot(real(zero2), imag(zero2) , 'o', 'Markersize',20);
    end

    if(i>2)    % 각 Zero값별 실수 및 허수 값
        plot(real(Zero(:,i-2)), imag(Zero(:,i-2)) , 'o', 'Markersize',20);
    end
    xlabel('Real(s)'); ylabel('Image(s)'); % X, Y축 이름 설정
end

[zero_h, pole_h] = zp2tf(zero, pole, k); % freqs()사용하기 위해 전달함수를 구현
H1 = freqs(zero_h, pole_h, t); % 주파수 응답을 구현
[zero_h2, pole_h2] = zp2tf(zero2, pole2, k2);
H2 = freqs(zero_h2, pole_h2, t);
[zero_h3, pole_h3] = zp2tf(zero3, pole3, k3);
H3 = freqs(zero_h3, pole_h3, t);
```

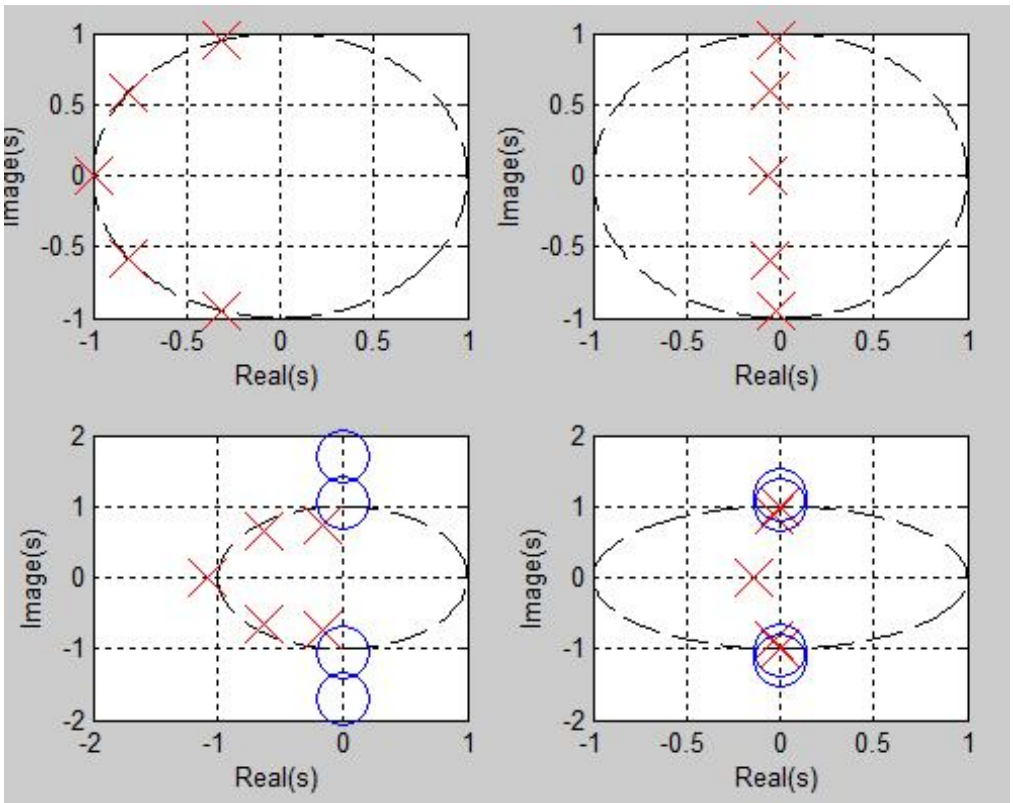
```
[zero_h4, pole_h4] = zp2tf(zero4, pole4, k4);
H4 = freqs(zero_h4, pole_h4, t);

H = [H1; H2; H3; H4]; % 각 주파수 응답에 대한 배열

figure(2) % 주파수 응답 plot
for i = 1 : 4
    subplot(2,2,i);
    plot(t, 10 * log10(abs(H(i,:)).^2) ); % 10log10|H(w)|^2 구현
    xlabel('Frequency w [rad/sec]'); ylabel('|H(w)|^2 [db]');
    grid on; ylim([-70 5]); % y축 범위 설정
end
```

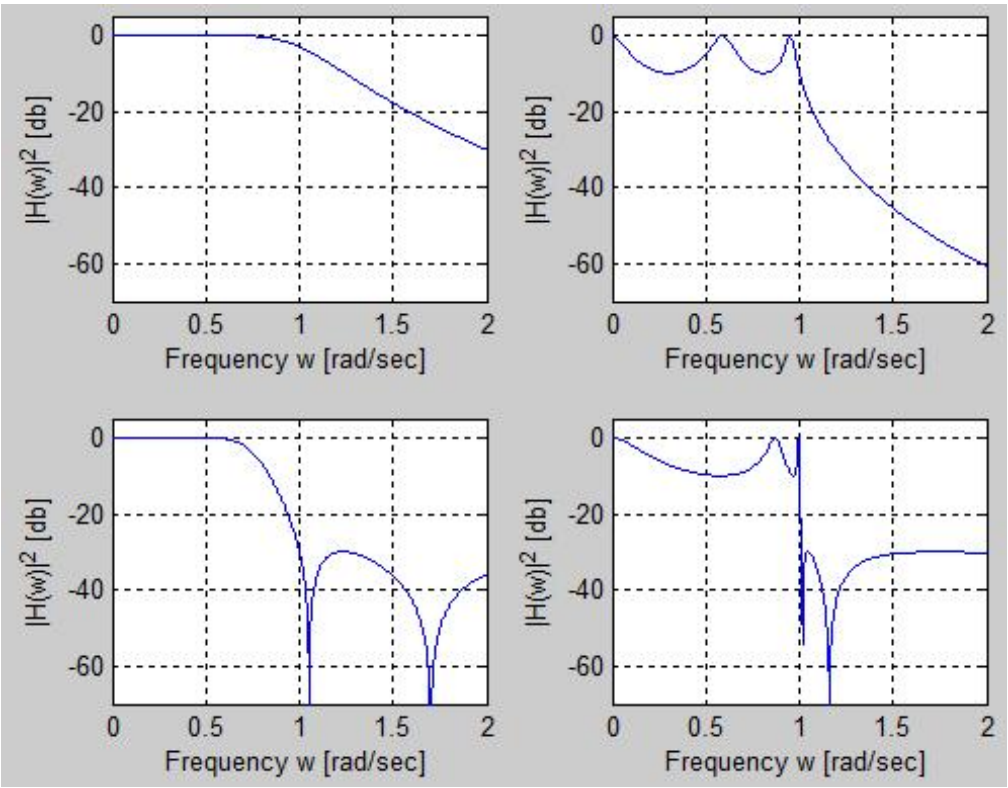
==========

- Pole-Zero plot 그래프



1, 2 ,4번째 그래프를 보면 pole값들이 단위원 내부 혹은 좌측 반원에 등간격 위치함으로 안정성을 확인할 수 있다.
반면, 3번째 그래프는 pole값 하나가 단위원 외부에 위치함으로써 불안정성을 확인했다.

- 주파수 응답 그래프



1번째 그래프는 Butterwotrh필터로써 passband, stopband에서 평평하며 이를 유지하기 위해 천이구간이 가장 길다.
2번째 그래프는 Chebyshev Type_1 필터로써 천이구간을 줄이고자 passband에 리플을 허용한 필터이다. 하지만 리플을 허용함으로써 passband에 통과된 신호로 인해 신호왜곡이 발생한다.
3번째 그래프는 Chebyshev Type_2 필터로써 천이구간을 줄이고자 stopband에 리플을 허용한 필터이다. 하지만 리플을 허용함으로써 stopband에 통과된 신호로 인해 신호왜곡이 발생한다.
4번째 그래프는 Elliptic 필터로써 2, 3번째 특징을 합친 그래프이다.
즉, stopband, passband에 동시에 리플을 허용함으로써 천이구간이 가장 짧다.

위의 그래프를 통해 Butterwotrh필터가 평평한 반면 천이구간이 가장 긴 것을 알 수 있으며 Chebyshev Type_1, Chebyshev Type_2 필터는 저지 및 통과대역에 리플을 허용함으로써 이전보다 천이구간이 줄어들었으며 Elliptic 필터는 저지, 통과대역 동시에 리플을 허용함으로써 천이구간이 가장 짧은 것을 확인했다.

위의 실습을 통해 리플이 많을수록 천이구간이 짧아져서 좋은 것이지만 신호왜곡이 발생한다는 사실을 확인했다.

3.2 Butterworth 필터

(1) 실습 그림 11과 같은 사양의 LPF를 Butterworth 필터로 구현하고자 할때 필요한 최소 차수는 얼마인가?

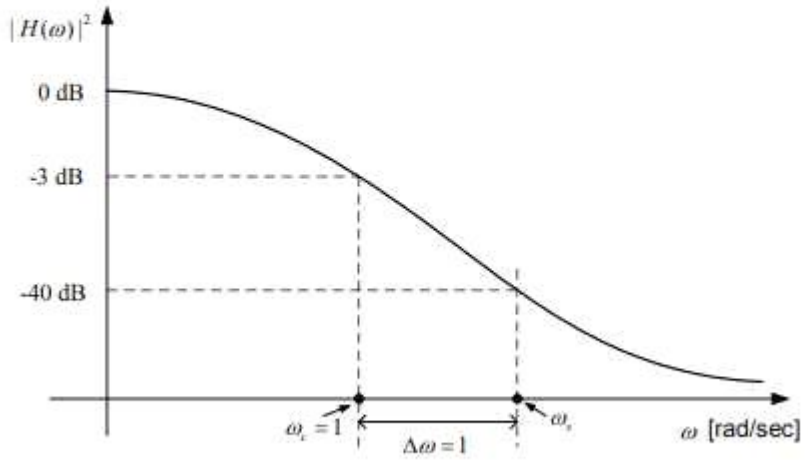


그림 11: 구현하고자 하는 필터사양.

저지대역의 기준을 -40dB로 잡고 △W를 1로 해주는 차수를 구하기 위해 강의자료에 주어진 수식에 대입하여 풀어보았다.

$$N = \left\lceil \frac{\log\left(\frac{1}{P_s} - 1\right)}{2\log\left(1 + \Delta\omega\right)} \right\rceil$$

여기서 주의할 점은 Ps에 dB값이 아닌 dB스케일로 변환전의 값을 넣어야한다. -40dB는 $-\frac{1}{10000}$ 이고 이를 수식에 대입

```
>> (log10((10000-1)) / (2*log10(1+1)))

ans =

    6.6438
```

6.6538이 나오고 올 성능을 만족하는 최소차수는 7이 된다.

2. Butterworth 필터

```
================================================================================
n = linspace(-pi,pi,360);
t = linspace(0, 2.5, 1000);
t2 = linspace(0, 5000, 10000);

[zero, pole, k] = buttap(7); % Butterworth filter
[zero_h, pole_h] = zp2tf(zero, pole, k); % freqs()사용하기 위해 전달함수를 구현

figure(1) % pole-zero plot
plot(cos(n), sin(n), '--k'); % 반지름 1인 단위원
hold on; grid on; % 점선 표기 및 그래프 중첩을 위한 구현
    % 각 pole값별 실수 및 허수 값
plot(real(pole), imag(pole) , 'rx', 'Markersize',20);
plot(real(zero), imag(zero) , 'o', 'Markersize',20);
xlabel('Real(s)'); ylabel('Image(s)');

H = freqs(zero_h, pole_h, t);

figure(2) % 주파수 응답 plot
plot(t, 10 * log10(abs(H).^2) );
grid on;
ylim([-50 5]);
xlabel('Frequency w [rad/sec]'); ylabel('|H(w)|^2 [db]');

[LPF, LPF2] = lp2lp(zero_h, pole_h, 300); % cutoff 주파수
H_db = freqs(LPF, LPF2, t2);

[BPF, BPF2] = lp2bp(zero_h, pole_h, 650, 700); % 중심 주파수, 대역폭
H_db2 = freqs(BPF, BPF2, t2);

[BPF3, BPF4] = lp2bp(zero_h, pole_h, 1500, 1000); % 중심 주파수, 대역폭
H_db3 = freqs(BPF3, BPF4, t2);

[BPF5, BPF6] = lp2bp(zero_h, pole_h, 3000, 2000); % 중심 주파수, 대역폭
H_db4 = freqs(BPF5, BPF6, t2);

[HPF, HPF2] = lp2hp(zero_h, pole_h, 4000); % cutoff 주파수
H_db5 = freqs(HPF, HPF2, t2);
```

```
H_DB = [H_db; H_db2; H_db3; H_db4; H_db5];

figure(3) %5가지의 필터 표시
hold on;
plot(t2, 10 * log10(abs(H_db).^2), 'b' ); %LPF
plot(t2, 10 * log10(abs(H_db2).^2), 'g' ); %BPF
plot(t2, 10 * log10(abs(H_db3).^2), 'r' ); %BPF
plot(t2, 10 * log10(abs(H_db4).^2), 'c' ); %BPF
plot(t2, 10 * log10(abs(H_db5).^2), 'm' ); %HPF
xlabel('Frequency [Hz]'); ylabel('|H(w)|^2 [db]');
grid on; ylim([-30 10]);
legend('Filter1, LPF', 'Filter2, BPF', 'Filter3, BPF', 'Filter4, BPF', 'Filter5, HPF');
```

==========

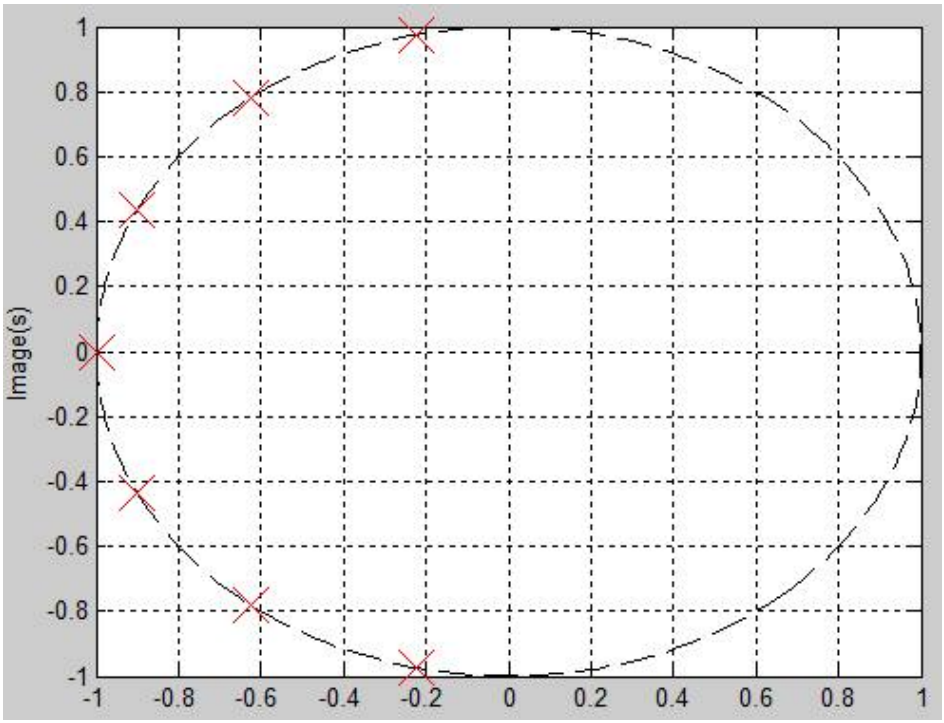
(2) 실습 **DEMO** 위에서 구한 차수의 Butterworth 필터 전달함수를 구하고 pole-zero plot을 반지름 1인 원과 함께 그래프에 표시하라. (buttap 함수 이용) (그림 12 참고)

(3) 실습 **DEMO** (2)에서 구한 필터의 주파수 응답의 크기를 그래프에 표시하고 설계 목표에 맞는지 확인하라. (주파수 응답은 전력에 대해 dB-scale로 표시하라. 즉, $10\log_{10}|H(\omega)|^2$으로 환산하여 표시할 것.) (그림 13 참고)

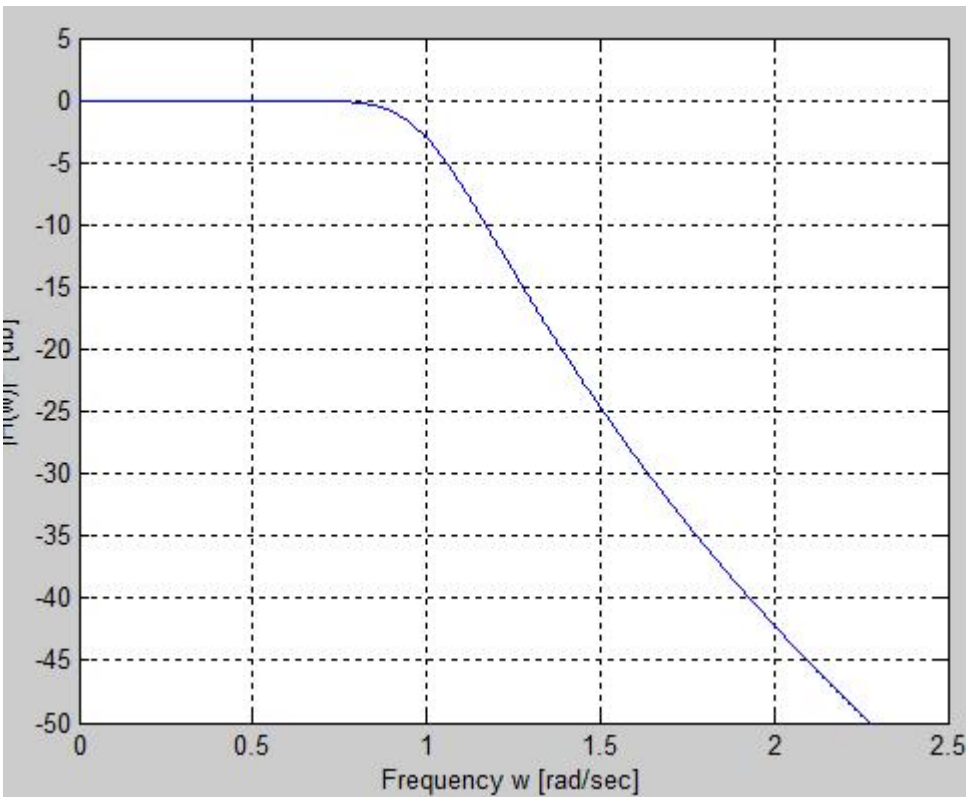
(4) 실습 **DEMO** (2)에서 설계한 LPF를 다음과 같은 다섯 종류의 필터로 변환하고 각 필터의 주파수 응답을 전력의 dB-scale로 그래프에 표시하라. (lp2lp, lp2bp, lp2hp, freqs함수 사용) (그림 14 참고)

1. LPF, cutoff 주파수 = 300 Hz
2. BPF, passband = [300, 1000] Hz
3. BPF, passband = [1000, 2000] Hz
4. BPF, passband = [2000, 4000] Hz
5. HPF, cutoff 주파수 = 4000 Hz

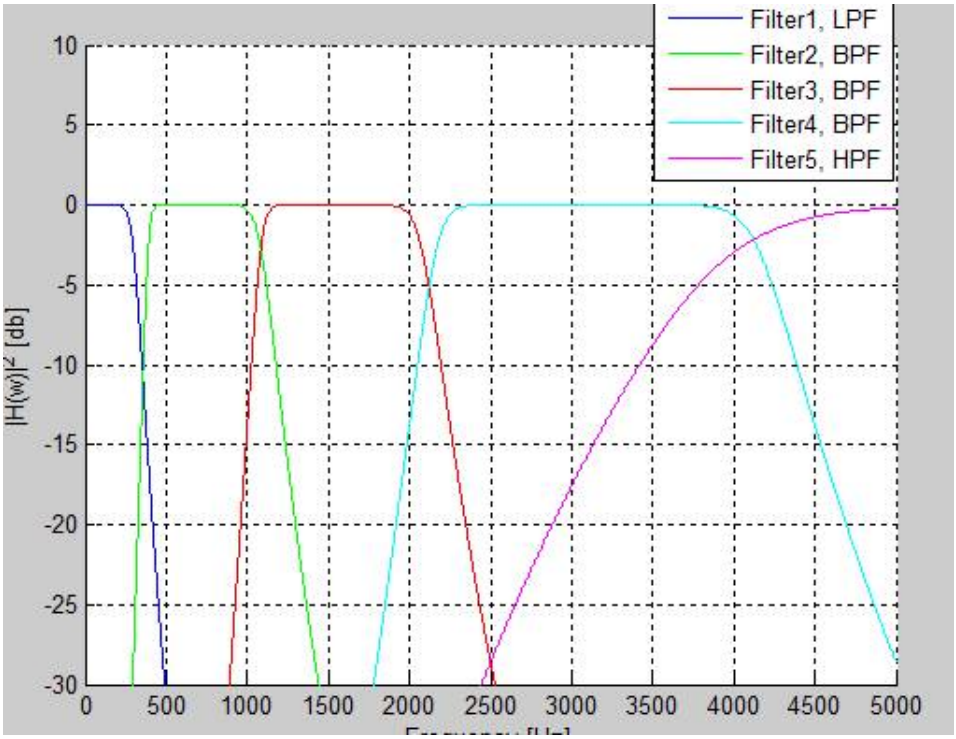
(1)


Image(s)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

(2)


5
0
-5
-10
-15
-20
-25
-30
-35
-40
-45
-50
0
0.5
1
1.5
2
2.5
Frequency w [rad/sec]

(3)



(1), (2) 7차 Butterworth 필터를 사용했고, 단위원을 그린 뒤 단위원에 극점과 영점을 찍어 주었다. 실습 1.1처럼 같은 형식의 그래프가 그려짐을 알 수 있다.

1번에서 계산한 값과 공식을 그대로 가져왔다. ws는 절대값 H(w)의 제곱이 -40db으로 나타날 때의 저지대역이 시작되는 주파수이다. 그리고 cutoff 주파수가 wc고 이것이 보통 -3db에서 시작한다. 그래프를 보면 이 구간의 차이가 1과 비슷하다(1보다 조금 작다). 그래서 제대로 7차 butterworth 필터를 설계했다고 볼 수 있을 것 같다.

(3) Butterwortch LPF를 lp2lp, lp2bp, lp2hp를 사용하여 각각의 필터를 설계했다. BPF는 각각의 주파수 대역의 중심주파수와 대역폭을 함수에 대입하여 구간을 정했고, 각각의 필터를 freqs 함수를 사용한여 주파수 응답을 구해준 후 dbscale로 환산하고 그래프에 색깔별로 표시해 주었다. BPF은 원하는 특정 주파수대역 내의 세력만 감쇠 없이 통과시키고, 나머지 주파수 세력은 감쇠시키는 특성을 가지고 있어서 이 사이 부분은 감쇠 없는 깔끔한 응답을 보일 것이다.

+ 만약 500, 3000의 주파수를 합성했는데 500Hz의 주파수 대역을 저지하고 3000Hz의 주파수 대역만 보고 싶을 때 BPF를 사용하여(여기서는 Filter 4를 이용) 특정 주파수 대역의 응답만 확인할 수 있다.