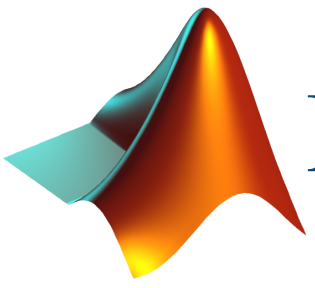
임베디드 신호처리 실습

IIR 실습 결과보고서

전자공학부 임베디드시스템

2014146004 김 민 섭



MATLAB

3.0 MATLAB 명령어 (내부함수)

- [numd, dend] = bilinear(num, den, fs)
  : Bilinear transformation을 이용해 아날로그 시스템을 이산화 하는 함수.

- [z, p, k] = tf2zp(num, den)
  : 전달함수로부터 pole과 zero를 구하는 함수. zp2tf 함수의 반대.

- H = freqz(num, den, w)
  : 디지털 시스템의 전달함수 H(z)로부터 주파수 응답을 구한다.

- y = filter(numd, dend, x)
  : 전달함수 H(z)로 표현된 디지털 시스템의 출력을 구한다.

3.1 시스템 이산화

(1) 다음은 cutoff 주파수가 1rad/sec인 3차 아날로그 Butterworth LPF의 전달함수이다.

$$H(s) = \frac{1}{s^3 + 2s^2 + 2s + 1}$$

이 시스템에 bilinear transformation을 적용하여 디지털 필터로 변환하라. 이때 다음과 같은 표본화 주파수를 사용하라. (bilinear 함수사용)

$$f_s \in \{5,\ 10\}\ \mathrm{Hz}$$

(실습 3.1.1) (1)의 아날로그 필터와 디지털 필터의 pole-zero plot을 표시하라. 각 시스템의 안정성을 판단하라. (반지름 1인 원과 함께 표시하라, tf2zp 함수 사용)

=> 실습(DEMO) Figure(1)

(실습 3.1.2) (1)의 아날로그 필터와 디지털 필터의 주파수 응답의 크기를 그래프에 표시하라. (아날로그 주파수 w에 대해 표시할 것, freqs, freqz 함수 사용)

=> 실습(DEMO) Figure(2)

(실습 3.1.1) (1)의 fs = 10[Hz]일 때 구한 디지털 필터의 임펄스 응답을 구하고 n = 0, 1, ... , L-1에 대해 그래프에 표시하라. L = {50, 100, 150}으로 L값을 달리하며 임펄스 응답을 관찰하라. 이 시스템의 임펄스 응답이 유한한지 무한한지 판단하라. (filter 함수 사용)

=> 실습(DEMO) Figure(3)

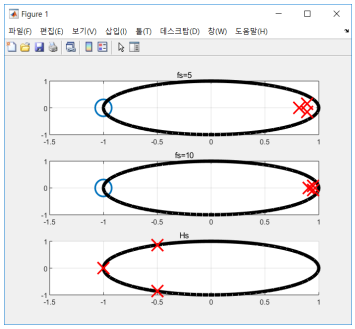
-① 실습 3.1 코딩

```
clear;
clc;
se= linspace(-2*pi,2*pi,360);
f_w= linspace(0,pi,10000);  %주파수 벡터

x=exp(1j*se); % 원 그리기

Hs_bun_ja = 1;                      %분자다항식의 계수벡터
Hs_bun_mo = [1.000,2.000,2.000,1];  %분모다항식의 계수백터

%%%%%%%% 아날로그 필터 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 아날로그 필터의 pole-zero plot을 구하기 위해 필요
% root() : 방정식의 해를 구하는 함수
% 분자 = 0의 해는 zero, 분모 = 0의 해는 pole
Hs_zero = roots(Hs_bun_ja);
Hs_pole = roots(Hs_bun_mo);

% freqs() : 아날로그 시스템의 전달함수 H(s)로 부터 주파수 응답을 구하기
% freqs(분자다항식의 계수벡터, 분모다항식의 계수백터, 주파수 벡터)
lpf_freq_response=freqs(Hs_bun_ja, Hs_bun_mo, f_w);
```

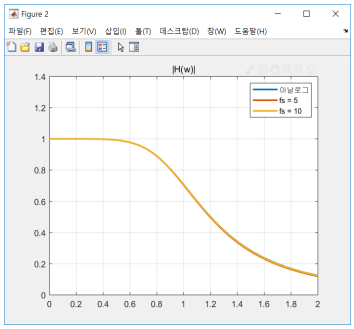
```matlab
%%%%%%%% 디지털 필터 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%% fs = 5 [Hz]%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% []=bilinear() : bilinear transformation을 이용해
%                 아날로그 시스템을 이산화 하는 함수
% 출력 : 이산화된 디지털 시스템의 전달함수 분자, 분모 다항식의 계수벡터
[fs_5_bunja, fs_5_bunmo]=bilinear(Hs_bun_ja, Hs_bun_mo,5.0);
% 위에서 구한 분자, 분모 다항식의 계수벡터(전달함수)로 부터 pole과 zero를 구하기
[fs_5_zero, fs_5_pole, k]=tf2zp(fs_5_bunja, fs_5_bunmo);
% 위에서 구한 분자, 분모 다항식의 계수벡터(전달함수)로 부터 주파수 응답을 구하기
[fs_5_freq_response,ws]=freqz(fs_5_bunja, fs_5_bunmo, f_w);

%%%%%%%%%%%% fs = 10 [Hz] %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[fs_10_bunja, fs_10_bunmo]=bilinear(Hs_bun_ja, Hs_bun_mo,10);
[fs_10_zero, fs_10_pole, k]=tf2zp(fs_10_bunja, fs_10_bunmo);
[fs_10_freq_response,ws]=freqz(fs_10_bunja, fs_10_bunmo, f_w);

%%%%%%%%% 3_1_4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
f_L1= [0:1:49];   % L = 50
f_L2= [0:1:99];   % L = 100
f_L3= [0:1:149];  % L = 150

% residue(분자다항식 계수벡터, 분모다항식 계수벡터)
% fs=10[Hz]일 때 전달함수의 부분분수를 구한다
% 출력:[계수들의 벡터, pole들의 벡터,분자차수>=분모차수일때 분자로 나눠 떨어진 값]
[sys_1_r,sys_1_p,k] = residue(fs_10_bunja,fs_10_bunmo);

% 임펄스 응답을 구하는 함수를 통해 임펄스 응답 구하기
[fs_10_impulse_response_50] = myfun_impulse_response_Des(sys_1_r,sys_1_p,f_L1);
[fs_10_impulse_response_100] = myfun_impulse_response_Des(sys_1_r,sys_1_p,f_L2);
[fs_10_impulse_response_150] = myfun_impulse_response_Des(sys_1_r,sys_1_p,f_L3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1);

subplot(3,1,1);
plot(real(fs_5_zero),imag(fs_5_zero),'o','MarkerSize',20,'LineWidth',2)
hold on;
plot(x,'k:','MarkerSize',20,'LineWidth',4)
plot(real(fs_5_pole),imag(fs_5_pole),'xr','MarkerSize',20,'LineWidth',2)
hold off;
title("fs=5")
grid on;

subplot(3,1,2);
plot(real(fs_10_zero),imag(fs_10_zero),'o','MarkerSize',20,'LineWidth',2)
hold on;
plot(x,'k:','MarkerSize',20,'LineWidth',4)
```

```
plot(real(fs_10_pole),imag(fs_10_pole),'xr','MarkerSize',20,'LineWidth',2)
hold off;
title("fs=10")
grid on;

subplot(3,1,3);
plot(real(Hs_zero),imag(Hs_zero),'o','MarkerSize',20,'LineWidth',2)
hold on;
plot(x,'k:','MarkerSize',20,'LineWidth',4)
plot(real(Hs_pole),imag(Hs_pole),'xr','MarkerSize',20,'LineWidth',2)
hold off;
title("Hs")
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(2);

plot(f_w,abs(lpf_freq_response),'MarkerSize',20,'LineWidth',2)
hold on;
plot(f_w.*5,abs(fs_5_freq_response),'MarkerSize',20,'LineWidth',2)
plot(f_w.*10,abs(fs_10_freq_response),'MarkerSize',20,'LineWidth',2)
hold off;
title("|H(w)|")
axis([0 2 -0 1.4])
legend('아날로그','fs = 5','fs = 10');
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(3);

subplot(3,1,1);
stem(f_L1,fs_10_impulse_response_50,'MarkerSize',3,'LineWidth',2)
title("L=50")
grid on;

subplot(3,1,2);
stem(f_L2,fs_10_impulse_response_100,'MarkerSize',3,'LineWidth',2)
title("L=100")
grid on;
subplot(3,1,3);
stem(f_L3,fs_10_impulse_response_150,'MarkerSize',3,'LineWidth',2)
title("L=150")
grid on;
```

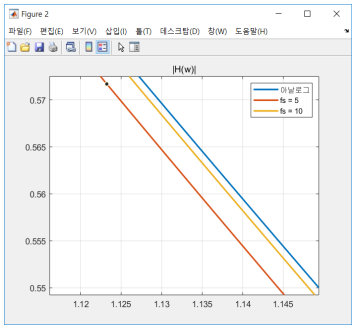
-② 실습 3.1 코딩의 결과_ 그래프



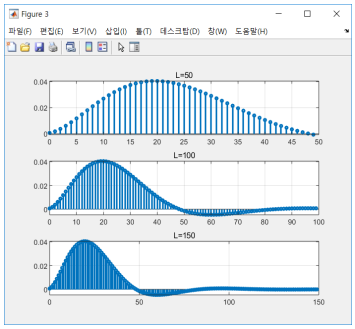
▲ 실습 3.1.1의 결과



▲ 실습 3.1.2의 결과



▲ 실습 3.1.2의 결과(확대)



▲ 실습 3.1.3의 결과

-③ DEMO 결과 해석

Figure (1)

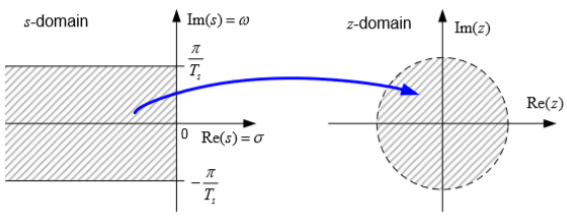
subplot(311) : fs = 5[Hz]를 사용해 주어진 전달함수 H(s)에 bilinear transformation을 적용하여 디지털 필터로 변환했을 때의 pole-zero plot이다.

subplot(312) : fs = 10[Hz]를 사용해 주어진 전달함수 H(s)에 bilinear transformation을 적용하여 디지털 필터로 변환했을 때의 pole-zero plot이다.

subplot(313) : 주어진 전달함수 H(s)의 pole-zero plot이다. 3차 Butterworth LPF이며, zero가 없고 pole이 원주상에 위치하고 있는 형태의 결과를 확인 할 수 있다.

⇒ 시스템의 차수는 pole이 결정하며, 필터를 조정할 수 있는 파라미터가 pole의 개수, 즉 시스템의 차수이다.

⇒ 주어진 전달함수 H(s)의 pole-zero plot을 살펴보면 모든 pole이 좌방평면에 위치하므로 시스템이 수렴하여 안정하다고 예측해 볼 수 있다. 그리고 bilinear transformation을 적용하여 디지털 필터로 변환했을 때의 pole-zero plot을 살펴보면 모든 pole이 단위원 내부에 위치하므로 시스템이 수렴하여 안정하다고 예측해 볼 수 있다.



s-영역의 좌평면이 z-영역의 단위원 내부로 변환되는 것을 확인할 수 있으며, 이를 통해 주어진 시스템이 안정하다는 것을 확인 할 수 있다. 또한 디지털 시스템은 아날로그 시스템의 제한적인 부분의 정보만을 가지고 있으며, 그 한계는 표본화 주

기 Ts에 의해 결정된다는 점이다. Ts가 작아질수록, 즉 표본화 주파수 fs가 높아질수록 디지털 시스템이 포괄 할 수 있는 s의 영역이 넓어지는 것을 확인 할 수 있다.

Figure (2)

아날로그 필터와 디지털 필터의 주파수 응답의 크기를 나타낸 것이다. LPF의 형태로 나타나는 것을 확인 할 수 있다.

fs를 5[HZ]와 10[Hz]일 때, 그리고 아날로그 필터의 주파수 응답의 크기는 미세하게 차이가 존재한다. 샘플링 주파수가 커질수록 아날로그 필터의 주파수 응답과 비슷하게 나타나는데 신호를 디지털 신호로 변환할 때, 잃어버리는 정보의 양이 샘플링 주파수가 커질수록 적어지기 때문이다.

(**Bilinear Transformation**은 전달함수에 log함수가 포함된다는 문제점을 해결하기 위해 근사화 기법을 이용한 것이므로, 약간의 오차가 발생한다.)

Figure (3)

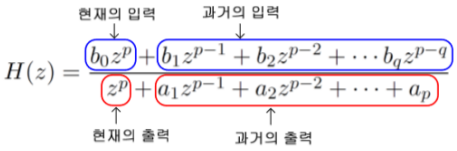
subplot(311) : L = 50일 때의 fs = 10[Hz]로 구한 디지털 필터의 임펄스 응답이다.

subplot(312) : L = 100일 때의 fs = 10[Hz]로 구한 디지털 필터의 임펄스 응답이다.

subplot(313) : L = 150일 때의 fs = 10[Hz]로 구한 디지털 필터의 임펄스 응답이다.

⇒ L을 달리 해가며 디지털 필터의 임펄스 응답을 관찰 했을 때, L을 키워도 계속해서 신호가 나오는 것을 확인 할 수 있다. 이 시스템은 임펄스 응답이 무한히 나타나므로 IIR시스템이다. 그리고 임펄스 응답이 수렴하므로 시스템은 안정한 시스템이라는 것도 확인 할 수 있다.

⇒ 임펄스 응답의 길이는 과거의 출력이 현재의 출력에 영향을 미치면 임펄스 응답은 무한해지고 그렇지 않다면 임펄스 응답은 유한한 길이를 갖는다. 따라서 현재의 출력에 영향을 미치는 세 가지 요소 중 과거의 출력에 의해 임펄스 응답의 길이를 결정한다. 즉, 과거의 출력이 현재의 출력에 영향을 미치면 임펄스 응답의 길이가 무한하고 과거의 출력이 현재의 출력에 영향을 미치지 않으면 임펄스 응답의 길이는 유한하다. 임펄스 응답의 길이가 무한한 시스템을 IIR(Infinite Impulse Response) 시스템 이라고 하고, 임펄스 응답의 길이가 유한한 시스템을 FIR(Finite Impulse Response) 시스템 이라고 한다.



3.2 디지털 BPF, HPF 설계

(1) 7차 아날로그 Butterworth 프로토타입 필터를 이용해 다음과 같은 아날로그 필터를 설계하라. (buttap, lp2bp, lp2hp 함수 사용)

- BPF, passband = [2000, 4000] Hz
- HPF, cutoff 주파수 fc = 4000 Hz

(실습 3.1.1) 표본화 주파수 fs = 50,000[Hz]일 때 Bilinear transformation을 이용해 (1)의 BPF와 HPF를 이산화 하여 IIR필터를 설계한 뒤 주파수 응답을 아날로그 필터와 비교하라.

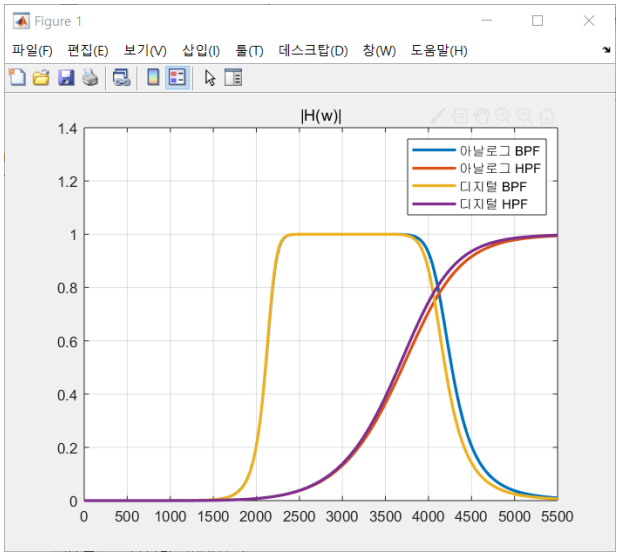
-① 실습 3.2 코딩

```
clear;
clc;
f_w_High= linspace(0,5500,800000);
fd = linspace(0,2*pi,50000);

%%%%%% Analog %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 아날로그 Butterworth 필터의 pole, zero 구하기
[buttap_z, buttap_p, buttap_k] = buttap(7);
% pole과 zero로 부터 전달함수 구하기
[buttap_zero_vector, buttap_pole_vector]=zp2tf(buttap_z,buttap_p,buttap_k);

%%%%% LPF -> BPF %%%%%
% cutoff주파수 1인 LPF -> 중심 주파수가 Wo(3000), 대역폭이 Bw(2000)
% 출력 : [변환된 BPF의 전달함수의 분모 다항식 계수벡터, 분자다항식 계수벡터]
[bunja_bpf, bunmo_bpf]=lp2bp(buttap_zero_vector, buttap_pole_vector, 3000,2000);
% 전달함수H(s)로부터 주파수 응답 구하기
bpf_analog_freq_response=freqs(bunja_bpf, bunmo_bpf, f_w_High);

%%%%% LPF -> HPF %%%%%
% cutoff주파수 1인 LPF -> cutoff 주파수가 Wo인 HPF
% 출력 : [변환된 HPF의 전달함수의 분모 다항식 계수벡터, 분자다항식 계수벡터]
[bumja_hpf, bunmo_hpf]=lp2hp(buttap_zero_vector, buttap_pole_vector, 4000);
% 전달함수H(s)로부터 주파수 응답 구하기
hpf_analog_freq_response=freqs(bumja_hpf, bunmo_hpf, f_w_High);

%%%%%% Digital%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```

%%%%% LPF -> BPF %%%%%
% cutoff주파수 1인 LPF -> 중심 주파수가 Wo(3000), 대역폭이 Bw(2000)
% 출력 : [변환된 BPF의 전달함수의 분모 다항식 계수벡터, 분자다항식 계수벡터]
% 우리에게 주어진 중심주파수와 대역폭은 각주파수가 아닌 주파수 영역이라
% 2*pi를 곱해주어 각주파수를 입력해주어야 한다.
[bunja_bpf, bunmo_bpf]=lp2bp(buttap_zero_vector, buttap_pole_vector, 3000*2*pi,2000*2*pi);
% []=bilinear() : bilinear transformation을 이용해
%                 아날로그 시스템을 이산화 하는 함수
% 출력 : 이산화된 디지털 시스템의 전달함수 분자, 분모 다항식의 계수벡터
```

```
37 —    [bpf_des_bunja, bpf_des_bunmo]=bilinear(bunja_bpf, bunmo_bpf,50000);
38      % 전달함수H(z)로부터 주파수 응답 구하기
39 —    [bpf_dig_freq_response,ws1]=freqz(bpf_des_bunja, bpf_des_bunmo,fd);
40
41      %%%%% LPF -> HPF %%%%%
42      % cutoff주파수 1인 LPF -> cutoff 주파수가 Wo인 HPF
43      % 출력 : [변환된 HPF의 전달함수의 분모 다항식 계수벡터, 분자다항식 계수벡터]
44      % 우리에게 주어진 cutoff주파수는 각주파수가 아닌 주파수 영역이라
45      % 2*pi를 곱해주어 각주파수를 입력해주어야 한다.
46 —    [bumja_hpf, bunmo_hpf]=lp2hp(buttap_zero_vector, buttap_pole_vector, 4000*2*pi);
47      % []=bilinear()
48 —    [hpf_des_bunja, hpf_des_bunmo]=bilinear(bumja_hpf, bunmo_hpf,50000);
49      % 전달함수H(z)로부터 주파수 응답 구하기
50 —    [hpf_dig_freq_response,ws2]=freqz(hpf_des_bunja, hpf_des_bunmo,fd);
51
52      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53 —    figure(1);
54
55 —    plot(f_w_High,abs(bpf_analog_freq_response),'MarkerSize',20,'LineWidth',2)
56 —    hold on;
57 —    plot(f_w_High,abs(hpf_analog_freq_response),'MarkerSize',20,'LineWidth',2)
58 —    plot(ws1.*8000,abs(bpf_dig_freq_response),'MarkerSize',20,'LineWidth',2)
59 —    plot(ws2.*8000,abs(hpf_dig_freq_response),'MarkerSize',20,'LineWidth',2)
60
61 —    hold off;
62
63 —    title("|H(w)|");
64 —    legend('아날로그 BPF','아날로그 HPF','디지털 BPF','디지털 HPF');
65 —    grid on;
66 —    axis([0 5500 0 1.4])
```

-② 실습 3.1 코딩의 결과_ 그래프



-③ DEMO 결과 해석

아날로그와 디지털 BPF, HPF의 형태를 확인 할 수 있다. BPF는 원하는 구간을 정해 그 구간의 신호만 통과 시키고 HPF는 높은 주파수 영역 구간의 신호만 통과시키는 특성을 가진다. 이 그래프 역시 실습 3.1.2의 그래프와 마찬가지로 아날로그와 디지털 필터의 주파수 응답의 크기에서 미세한 차이를 보인다. 아날로그를 디지털로 변경하면서 정보를 잃어버리게 되는데 그 잃어버린 정보 때문에 차이가 생기게 되는 것이다.
(**Bilinear Transformation**은 전달함수에 log함수가 포함된다는 문제점을 해결하기 위해 근사화 기법을 이용한 것이므로, 약간의 오차가 발생한다.)

※Bilinear Transformation

식 $s = \frac{1}{T_s}\ln z$ 를 이용해 아날로그 필터를 디지털 필터로 변환 할 수 있다. 따라서 이 관계를 이용해 아날로그 필터 H(s)로부터 디지털 필터의 전달함수 $H_d(z) = H(s)|_{s=\frac{1}{T_s}\ln z}$ 를 얻을 수 있다. 그러나 여기에서 초월함수인 log 함수가 사용된다는 문제점이 발생한다. $s = \frac{1}{T_s}\ln z$ 을 그대로 이용하면 자연스럽게 Hd(z)에 log함수가 포함될 것이다. 그러나 전달함수는 분모와 분자가 z의 다항식으로 이루어진 유리식으로 표현되어야 하는데 log함수가 나타나면 유리식의 형태가 되지 않는다. 이를 해결하기 위해 다음과 같은 근사화를 이용한다.

$\ln z \approx 2\frac{z-1}{z+1}$ 이 관계를 이용해 s를 $s = \frac{2}{T_s}\frac{z-1}{z+1}$ 으로 표현 할 수 있다. 이 관계를 이용하면 s → z, z → s중 어떤 방향으로 변환 하더라도 선형성을 유지한다는 의미에서 **bilinear transformation**이라 한다. 이를 이용하면 식 Hd(z)도 유리식이 되어 디지털 시스템을 이용해 손쉽게 구현 할 수 있다.

Bilinear Transformation은 전달함수에 log함수가 포함된다는 문제점을 해결하기 위해 근사화 기법을 이용한 것이므로, 약간의 오차가 발생한다. 이 오차가 발생할 것을 미리 감안하여 아날로그 필터의 사양을 조정하는 등의 다양한 기법을 통해 오차를 해결 할 수 있다.

3.3 신호의 필터링

(1). 7차 안날로그 Butterworth 프로토타입 필터를 이용해 다음과 같은 아날로그필터를 설계하라.

- LPF, cutoff 주파수 $f_c = 300Hz$

아날로그 프로토타입 실습에서와 같이 buttap 와 lp2lp 를 사용하면 다음과 같이 구현이 가능하다.

```
[buttap_z, buttap_p, buttap_k] = buttap(7);
[buttap_zero_vector, buttap_pole_vector]=zp2tf(buttap_z,buttap_p,buttap_k);
[bunja_lpf, bunmo_lpf]=lp2lp(buttap_zero_vector, buttap_pole_vector, 300);
```

Buttap 내장 함수를 사용할 경우 pole 과 zero 값으로 출력이 되기 때문에 lp2lp에서 사용할 수 있도록 전달함수로 만들어 준 다음에 cutoff 주파수가 300Hz인 lowpass filter로 변활 할 수 있었다.

(2). 표본화 주파수 $f_s = 3000Hz$ 일때 bilinear transformaion을 이용해(1)의 LPF를 이산화하여 IIR 필터를 설계하라.

문제를 진행하기에 앞서 lp2lp의 내장함수를 살펴보자면 [bt,at] = lp2lp(b,a,Wo) 형식이며 b와 a는 각각 전달함수 분자항,분모항의 계수백터이다. w0는 각주파수를 입력한다. 우리에게 주어진 cutoff 주파수는 각주파수가 아닌 주파수 영역이라 2π를 곱해주어 각주파수를 입력해 주어야 한다. 구현은 다음과 같다.

```
[bunja_lpf_D,bunmo_lpf_D]=lp2lp(buttap_zero_vector,buttap_pole_vector, 300*2*pi);
[lpf_des_bunja, lpf_des_bunmo]=bilinear(bunja_lpf_D, bunmo_lpf_D,3000);
```

(3). 다음과 같은 신호 x[n]을 발생하라.

$$x[n] = cos(2\pi \frac{f_1}{f_s} n) + cos(2\pi \frac{f_2}{f_s} n)$$

- $f_1 = 100Hz, f_2 = 500Hz, f_s = 3000Hz$
- $n = 0,1,2,...N, N = 500$

위 신호를 구하는 것에는 큰 어려움이 없다. 다음과 같이 구현 할 수 있었다.

```
f_N_2= linspace(0,499,500);
x_n = cos(2*pi*(100/3000).*f_N_2) + cos(2*pi*(500/3000).*f_N_2);
```

(4). (3)의 신호 x[n]을 (2)에서 설계한 디지털 필터에 입력하여 출력 y[n]을 구하라.

MATLAB 내장 함수인 filter의 사용법만 알고 있다면 큰 어려움 없이 그대로 적용하면 된다. 구현은 다음과 같이 진행하였으며 필터를 거친 y[n]을 구할 수 있었다.

```
y = filter(lpf_des_bunja,lpf_des_bunmo,(x_n));
```

(5). x[n]과 y[n]의 크기 스펙트럼을 구하라.

크기 스펙트럼을 구하는 것 자체는 Sampling 실습에서 사용하였던 myfun_SA 함수를 사용해서 쉽게 구할 수 있다. 하지만 이 값의 결과는 -0.5~0.5만 보여주고 있다. 우리는 주파수 영역에서 본 신호를 관찰하고자 하기 때문에 다음과 같은 수식을 사용하였다.

- $t = T_s = \frac{n}{f_s}$

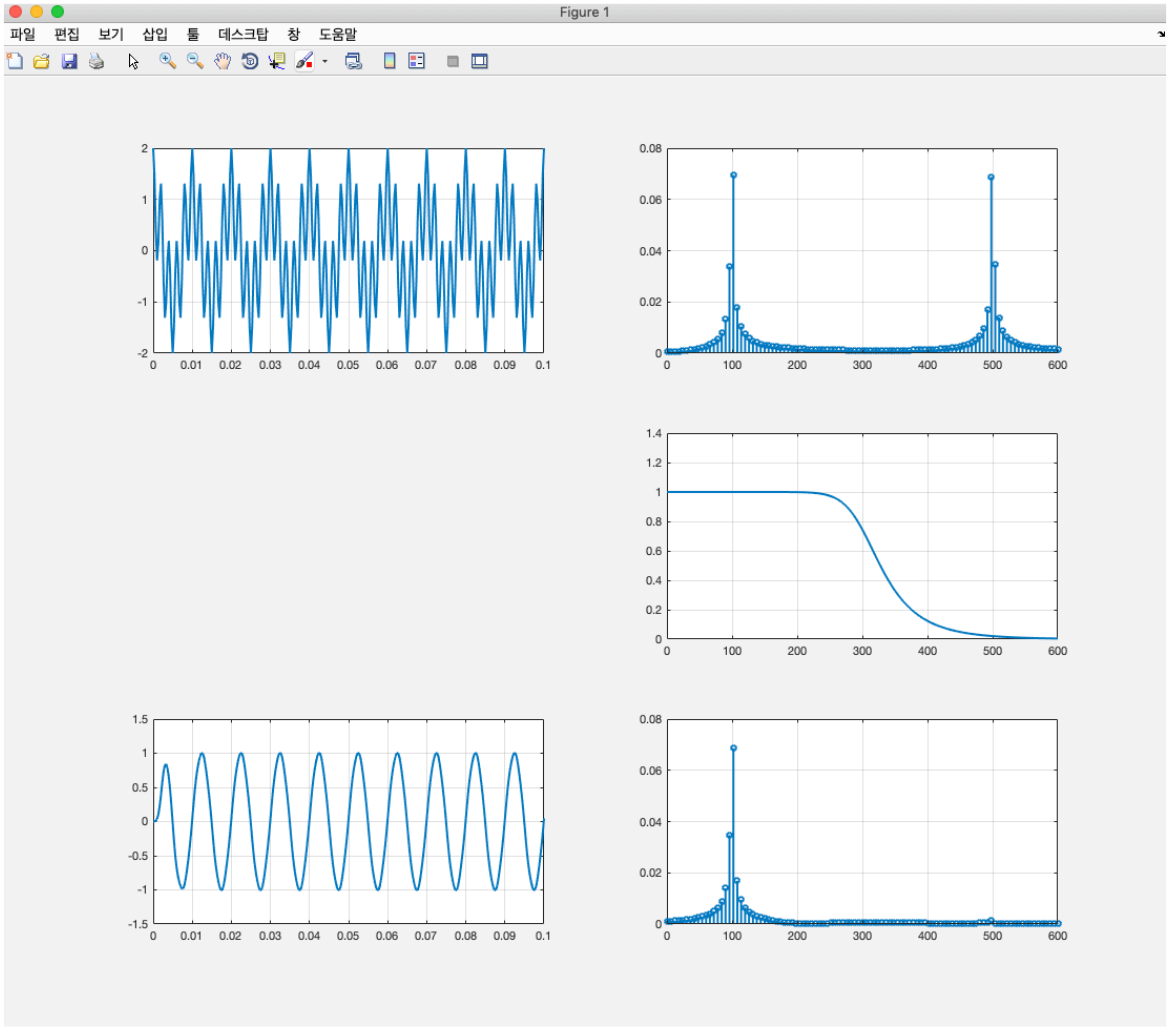
위 식을 이용하고 myfun_SA을 사용한 크기스펙트럼 구현은 다음과 같이 구할 수 있다.

```
[f0 , X] = myfun_SA(f_N_2/3000,x_n);
[f3 , Y] = myfun_SA(f_N_2/3000,y);
```

시간축의 입력이 들어가는 것에 Sampling 주파수를 나누어주면 된다. 이에 대한 결과는 다음 실습문제에서 확인해 보도록 하자.

(6). 입력신호 x[n]과 크기스펙트럼, 출력신호 y[n]과 크기 스펙트럼, 디지털 필터 $H_d(z)$의 주파수 응답의 크기를 그래프에 표시하라.

위 실습을 통해 얻어진 값들로 그려진 그래프는 다음과 같이 나타내어진다.



먼저 입력신호에 대해서 살펴보도록 하자. 입력신호 는 두 정현파의 합으로 이루어진 신호이다. 그려진 그래프를 보아도 하나의 정형파의 파형이 아닌 두가지 종류의 정형파가 합쳐진 형태로 대략적인 판별이 가능하다. 주파수 영역으로 와서 크기 스펙트럼을 바라보면 이는 더욱 명확해진다. 100hz 와 500hz에서 값을 가지는 모습을 보여준다. 이로서 우리가 입력 신호 x[n]을 잘 표핸했다는 사실을 알 수 있다. (1)에서 구한 필터의 주파수 응답을 살펴보자. Cutoff 주파수가 300정도인 LPF 라는 사실을 알 수 있다. 이 필터를 입력신호가 통과한다면 어떻게 될까? 이상적인 LPF라면 300보다 낮은 주파수 영역은 그대로 통과할 것이고 300을 초과한다면 통과하지 못해 걸러질 것이다. 실제로 그런지 그래프를 통하여 확인해 보자. 그래프를 보아하니 500hz대에 있는 주파수 성분이 상쇄된 것을 볼 수 있다. 하지만 이는 이상적인 필터가 아니므로 완전한 상쇄가 아닌 찌꺼기 성분이 존재함을 알 수 있다. 이를 다시 시간축에서 바라보아도 마찬가지의 결과를 낳게 된다. 이상적인 필터라면 주파수가 정확히 하나만 존재하는 정형파가 그려져야 한다. 하지만 초반에 약간의 찌거기 성분이 존재하는 정현파가 그려지게 된다. 최종적인 코드는 다음과 같다.

```
f_w_High= linspace(-2*pi,2*pi,3000);
f_N_2= linspace(0,499,500);
fs = 3000;
f1 = 100;
f2 = 500;
[buttap_z, buttap_p, buttap_k] = buttap(7);
[buttap_zero_vector, buttap_pole_vector]=zp2tf(buttap_z,buttap_p,buttap_k);
[bunja_lpf, bunmo_lpf]=lp2lp(buttap_zero_vector, buttap_pole_vector, 300);

[bunja_lpf_D, bunmo_lpf_D]=lp2lp(buttap_zero_vector, buttap_pole_vector,
300*2*pi);
[lpf_des_bunja, lpf_des_bunmo]=bilinear(bunja_lpf_D, bunmo_lpf_D,3000);
[fs_10_freq_response,ws]=freqz(lpf_des_bunja, lpf_des_bunmo, f_w_High);

x_n = cos(2*pi*(100/3000).*f_N_2) + cos(2*pi*(500/3000).*f_N_2);
y = filter(lpf_des_bunja,lpf_des_bunmo,(x_n));
[f0 , X] = myfun_SA(f_N_2/3000,x_n);
[f3 , Y] = myfun_SA(f_N_2/3000,y);
figure(1);
subplot(3,2,1);
plot(f_N_2/3000,x_n,'MarkerSize',5,'LineWidth',2)
axis([0 0.1 -2 2])
grid on;
subplot(3,2,2);
stem(f0,abs(X),'MarkerSize',5,'LineWidth',2)
grid on;
axis([0 600 0 0.08])

subplot(3,2,4);
plot(ws*3000/(2*pi),abs(fs_10_freq_response),'MarkerSize',5,'LineWidth',2)
grid on;
axis([0 600 -0 1.4])

subplot(3,2,5);
plot(f_N_2/3000,y,'MarkerSize',5,'LineWidth',2)
grid on;
axis([0 0.1 -1.5 1.5])

subplot(3,2,6);
stem(f0,(abs(Y)),'MarkerSize',5,'LineWidth',2)
axis([0 600 0 0.08])
grid on;
```

위 실험을 통해 디지털 필터를 설계하고 사용해 보았다. Sampling 을 많이 하면 할 수록 Analog filter 와 비슷해 진다는 사실을 알 수 있었다. 또한 필터 사용시에 완전히 신호를 상쇄해 주지는 못한다는 것을 다시한번 느낄 수 알아 볼 수 있었다. 신호를 원하는 대역대에서만 정확하게 추출해내려면 한가지 필터만을 사용한다면 오차가 나올 수 있기 때문에 실제로 사용하기 위해서는 좀 더 많은 연구와 공부가 필요함을 느낄 수 있는 실습이었다.