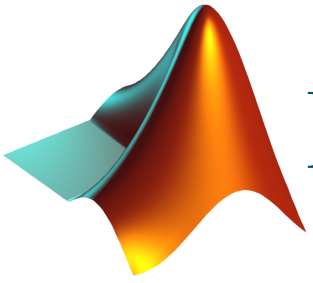
임베디드 신호처리 실습

TF 실습 결과보고서

전자공학부 임베디드시스템

2014146004 김 민 섭



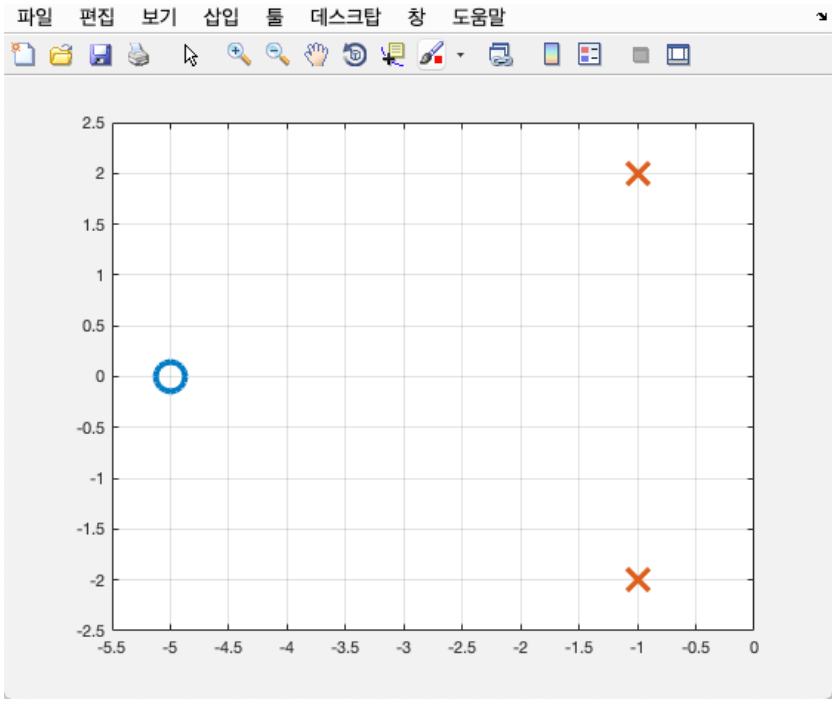
MATLAB

1. 연속 시스템의 전달함수

1.1 H(s)의 pole-zero plot을 그리고 시스템의 안정성을 판단하라.

$$H(s) = \frac{s+5}{s^2+2s+5}$$

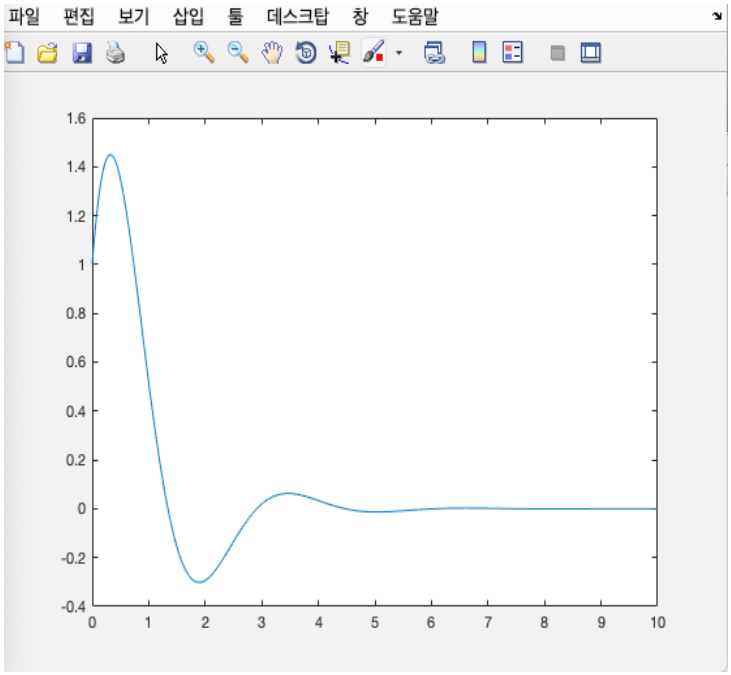
<pole-zero plot>



시스템이 '안정하다'라는 것은 발산하지 않고 시간이 지날수록 전달함수 H(s)를 pole-zero plot으로 그려보면 위의 그림과 같이 2개의 극점과 한 개의 영점이 나오게 된다. 여기서 영점 (zero)란 전달함수 H(s)의 분자를 0으로 만드는 인자를 나타내고 극점 (pole)이랑 분모를 0으로 만드는 인자를 말한다. H(s)의 영점은 -5이고 극점은 -1+2j, -1-2j이다. pole-zero plot은 복소평면 상에 영점은 'O'로, 극점은 'X'로 표현하는 것이다. pole-zero plot을 보고 주어진 시스템의 안정성을 판단할 수 있는데 보통 연속 함수인 경우 Y축을 기준으로 극점인 pole이 모두 좌측에 있으면 안정, 우측에 하나라도 있을 때 불안정, Y축에 걸칠 때 제한적 안정이라고 판단한다. 왜 Y축 기준 우측에 극점이 하나라도 있다면 시스템이 불안정하다고 하는 걸까? 이는 전달함수를 라플라스 역변환한 충격응답을 보면 알 수 있다. 예를 들어 극점 한 개가 Y축의 우측값인 3이라고 가정해 보자. 부분분수로 나누었을 때 극점 3을 가진 항은 $\frac{a}{s-3}$형태로 나타나게 된다. 여기서 a는 임의의 상수라고 가정하자. 라플라스 역변환을 하게 되면 ae^{3t}이 된다. 이는 상수 a가 양수인지 음수인지에 따라서 -∞와 ∞로 발산한다. 결과적으로 0으로 수렴하지 않기 때문에 안정하지 않다고 하는 것이다. 따라서 주어진 전달함수 H(s)는 안정이다.

1.2. 위 시스템 H(s)로부터 충격응답 h(t)를 그래프에 표시하고 안정성을 판단하라.

앞서 말했듯이 주어진 전달함수 H(s)는 안정이다. 하지만 이를 충격응답으로 변환하여 그래프로 확인해보고자 한다. 우선 수식으로 본다면 H(s)는 부분분수로 바뀌어서 $\frac{a}{s-z_1}+\frac{b}{s-z_2}$ 이러한 형태가 된다. 여기서 라플라스 역변환을 한다면 $a \cdot e^{z_1}+b \cdot e^{z_2}$ 이 된다. Z1과 Z2는 극점으로 양수가 되면 시스템은 발산하게 된다. 다음은 MATLAB을 사용하여 나타낸 그래프이다.



기존 전달함수 H(s)를 h(t)로 변환하여 나타낸 그래프로 우측으로 갈수록 h(t)의 값이 발산하지 않고 점점 0으로 수렴하는 것을 볼 수 있다. 1번에 나온 pole-zero plot 에서 극점이 Y축 기준 왼쪽에 있어 안정하다는 결과와 일치 한다는 것을 알 수 있다. 전달함수 H(s)에서 충격응답 h(t)로 변환은 myfun_impulse_response 함수를 사용했다.

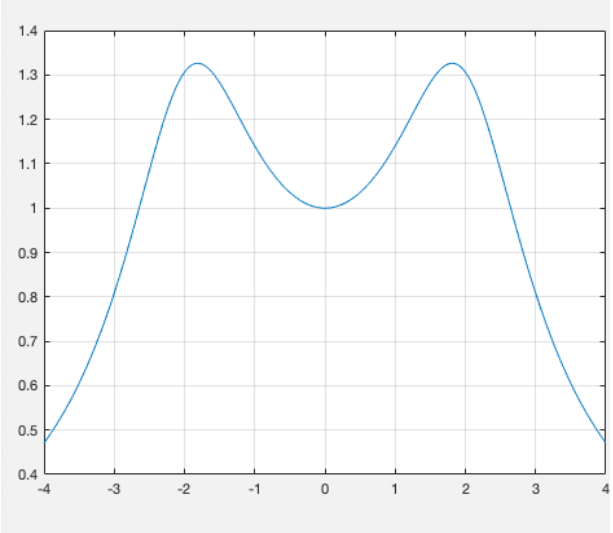
```
1   function [impulse_response] = myfun_impulse_response(r,p,t)
2 -   impulse_mid = [;];
3 -   impulse_mid = r.*exp(t.*p);
4 -   impulse_response = sum(impulse_mid);
```

1.3 주어진 H(s)로부터 주파수응답 H(ω) 를 구하고 크기를 그래프에 표시하여라.

H(s)를 주파수응답인 H(ω)로 바꾸는 것은 s대신 jω를 쓰면 된다. MATLAB에서 사용한 코드를 살펴보면 myfun_freq_response 함수를 사용하였다.

```
[f_w,freq] = myfun_freq_response(bun_ja,bun_mo);
```

```
function [se,freq] = myfun_freq_response(r,p)
se= linspace(-4,4,1000);
x=(1j*se);
bun_mo = [];
bun_ja = [];
    bun_mo = abs(x-p);
    bun_ja = abs(x-r);
if size(bun_mo) >1
    bun_mo_mult = prod(bun_mo);
else
    bun_mo_mult = bun_mo;
end
if size(bun_ja) >1
    bun_ja_mult = prod(bun_ja);
else
    bun_ja_mult = bun_ja;
end
freq = (bun_ja_mult./bun_mo_mult);
```



처음에 X라는 변수에 $j\omega$를 할당시켜 주었다. 이후 주어진 식에 S자리에 $j\omega$를 넣어줌으로써 주파수응답 H(ω)를 만들었다. 이때 입력받는 행렬은 가로의 행 형태가 아닌 열의 형태라 prod함수를 사용하여 서로 곱해주었다.
결과에 대한 그래프를 보게 되면 -4부터 4까지 m자 모양을 띄고 있다. 1번의 pole-zero plot을 함께 본다면 H(ω)는 s 대신 $j\omega$를 사용하였다. pole-zero plot을 통해 본다면 Y축은 $j\omega$이다. 그래프대로 해석하게 된다면 분모의 항은 Y축의 임의의 $j\omega$로부터 극점(pole)까지의 거리의 곱이다. 분자 또한 마찬가지로 $j\omega$로부터 영점(zero)까지의 거리의 곱인 셈이다. 이는 H(ω)값이 극점과 $j\omega$의 거리가 가까울수록, 영점과 $j\omega$의 거리가 멀어질수록 커진다는 얘기다. ω가 $-\infty$ 부터 ∞ 까지 이동함에 따라 Y축에서 $j\omega$은 아래부터 위로 이동하게 된다. 이동하면서 극점과 영점과의 거리를 측정하여 곱해보면 H(ω)은 0부터 극점에 접근하면서 점차 증가하게 된다. 하지만 중간에 영점과의 거리가 최소가 되어 약간 줄어들게 되고 다시 다른 극점과 가까워지며 증가한 후에 점차 멀어지면서 0으로 수렴하게 된다.

2. 연속 시스템의 안전성

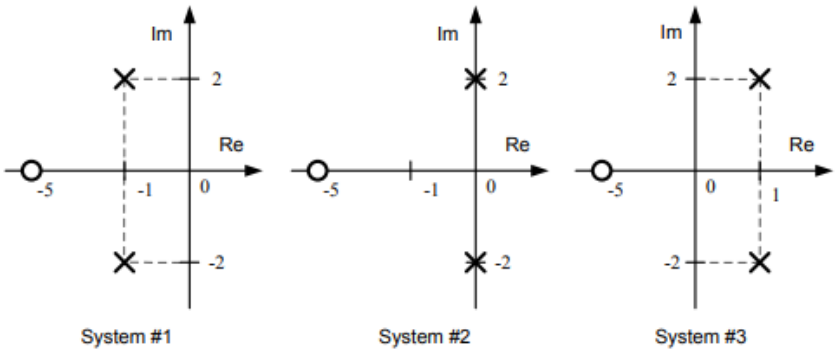


그림 17: 실습 3.2의 시스템.

위 그림의 세 시스템에 대해 전달함수를 구하고 충격응답을 그래프에 표시.
첫 번째 그림 SYS#1의 영점은 -5이다. 극점은 -1+2j, -1-2j이다. SYS#2의 영점 또한 -5이고, 극점은 2j와 -2j이다. SYS#3의 영점도 -5이고, 극점은 1+2j, 1-2j이다.

각각의 전달함수는 다음과 같다. SYS#1 = $\frac{s-5}{(s-(-1+2j))(s-(-1-2j))}$

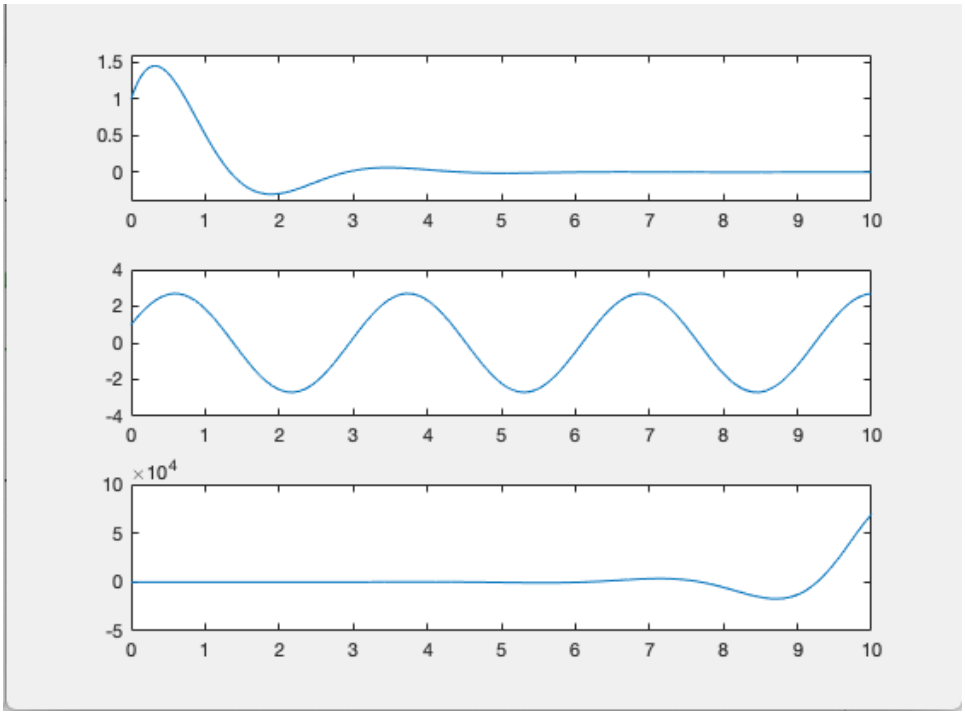
SYS#2 = $\frac{s-5}{(s-2j)(s+2j)}$, SYS#3 = $\frac{s-5}{(s-(1+2j))(s-(1-2j))}$

이를 내부함수 zp2tf()를 사용하여 전달함수의 분자와 분모 다항식의 계수에 대한 벡터를 구한 후 residue() 함수를 사용하여 부분분수로 전개한다. 이후 우리는 my_fun_impulse_response() 함수를 사용하여 충격응답을 구한다. 다음은 내장함수 zp2tf(), residue(), my_fun_impulse_response()를 사용한 예이다.

```
13 -    [sys_1_ja, sys_1_mo]=zp2tf(sys_1_zero,sys_1_pole,1);
14 -    [sys_1_r,sys_1_p,k] = residue(sys_1_ja,sys_1_mo);
15 -    impulse_response_1 = myfun_impulse_response(sys_1_r,sys_1_p,t);
```

```
1     function [impulse_response] = myfun_impulse_response(r,p,t)
2 -   impulse_mid = [];
3 -   impulse_mid = r.*exp(t.*p);
4 -   impulse_response = sum(impulse_mid);
5
```

my_fun_impulse_response() 함수는 직접 라플라스 역변환을 해주는 역할이다. 결과적으로 충격응답을 직접 구해주는 함수이다. 이를 이용하여 다음과 같은 충격응답 그래프를 그렸다.



위 그래프는 주어진 전달함수에 대한 시간 영역에서의 충격응답이다. 첫 번째부터 SYS#1, SYS#2, SYS#3에 대한 충격응답 그래프이다. 우선 SYS#1의 그래프는 시간이 지날수록 0으로 수렴하는 모습을 볼 수 있다. 두 번째 그림 SYS#2는 무한대로 발산하는 것은 아니지만 0으로 수렴하지 않고 진동하는 모습을 볼 수 있다. 마지막 SYS#3의 그림은 시간이 지날수록 완전히 무한대로 발산하는 모습을 보인다. 수식으로 보게 되면 부분분수로 전개한 전달함수의 한 항을 라플라스 역변환을 해 보면 $r \bullet e^{pt}$ 이런 식의 형태를 갖는다. 여기서 r은 부분분수 항의 분자 부분, 즉 계수 부분을 뜻하고 p는 분모 부분의 극점들의 벡터이다. 만약 극점 p가 #1의 그림과 같이 음수이게 되면 주어진 항은 라플라스 역변환하게 되면 0으로 수렴하게 될 것이다. 반대로 #3의 그림과 같이 양수로 주어지게 되면 무한대로 발산하게 될 것이다. 그림17 pole-zero plot을 보게 되면 #1의 극점은 전부 Y축 기준 좌측에 위치한 것을 볼 수 있다. #2의 극점은 Y축에, 그리고 #3의 극점은 Y축 기준 우측에 위치하고 있다. 문제 1번에 언급 한 대로 Y축 기준 좌측에 극점이 있을 경우 시스템은 안정하다, 그 외의 경우는 불안정하다 라는 결과를 충격응답을 직접 구함으로써 다시 한번 확인할 수 있었다. 진동하는 경우는 극점이 Y축에 있는데 이는 정수 부분이 없다는 것이다. 즉 진동 부분인 허수부분만 있어서 0으로 수렴하거나 무한대로 발산하지 못하고 계속 진동하게 되는 것이다.

3. 이산시스템의 전달함수

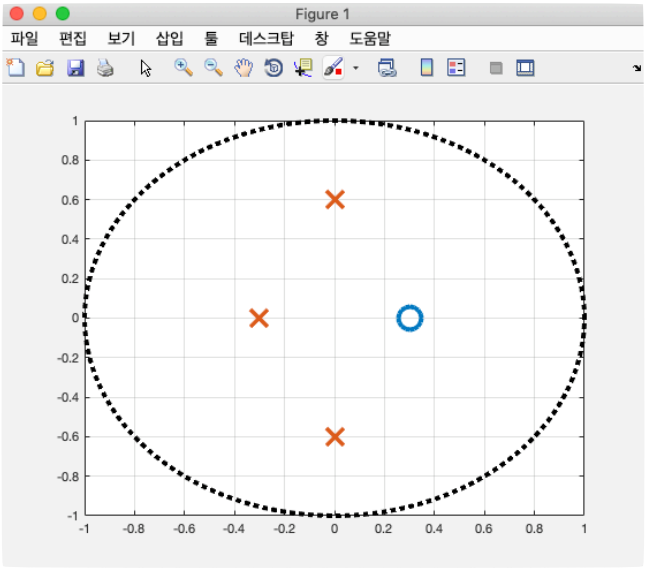
3.1 다음 시스템의 Pole-Zero plot 을 그리고 안정성을 판단하라. 반지름 1인 원과 함께 표시하라.

$$H(z) = \frac{z - 0.3}{Z^3 + 0.3z^2 + 0.36z + 0.108}$$

위 시스템의 pole- zero plot 을 그리려면 먼저 pole, zero 값을 알아야 한다. Pole 값과 Zero 값은 전달함수를 통해서 알 수 있다. 분모가 0 이 되게 만드는 것이 pole 갑이고 분자가 0이 되게 만드는 값이 zero 값이다. 바로 바라보더라도 zero 값은 0.3 이라는 것을 알 수 있다. 하지만 pole 값은 우리가 손으로 계산하기 어렵기 때문에 MATLAB을 통해서 구해보겠다. 다음은 MATLAB을 통해서 얻어낸 Pole, Zero 값이다.

- Zero : 0.3000
- Pole : 0.0000 + 0.6000i, 0.0000 - 0.6000i, -0.3000 + 0.0000i

Pole, Zero 값을 알아내었으니 우리는 Pole,-Zero plot 를 그릴 수 있다. 다음과 같이 말이다.



반지름이 1인 원도 함께 표시하였다. X 표시가 된것은 Pole값이며 O 표시는 Zero 값이다. Pole-Zero plot 을 살펴보니 위 시스템은 안정하는것을 알 수 있다. 모든 Pole의 위치가 단위원 내부에 있기 때문에 위 시스템의 임펄스 응답은 최종적으로는 수렴하는 값을 가지게 될 것이라 예상된다. 우리가 예상한 값이 맞는지에 대해서는 다음 문제로 부터 알아보자 한다.

3.2 위 시스템 H(z) 로부터 충격응답 h[n]을 그래프에 표시하고 시스템의 안정성을 판단하라.

충격응답을 구하려면 Z-transform 의 역변환 과정을 알아야 한다. 이에 대해 간략하게만 살펴보면 다음과 같이 표현할 수 있다.

$$H(Z) = C_1 + \frac{C_2 Z}{(Z - p_1)} + \frac{C_3 Z}{(Z - p_2)} \ldots + \frac{C_n Z}{(Z - p_{n-1})}$$

Z-Transform 역변환~~

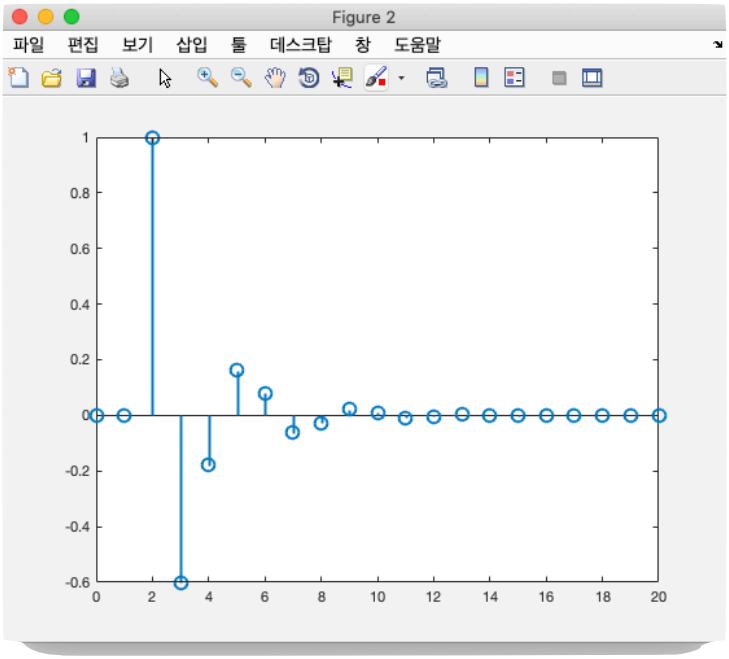
$$h[n] = C_1\delta[n] + C_2 P_1^n + C_3 P_2^n + \ldots C_n P_{n-1}^n$$

위의 식처럼 전달함수를 부분분수 형태로 만든 다음에 이를 이용해서 충격응답 h[n]을 구할 수 있다. 위 과정에 대한 구현 코드는 다음과 같다.

```
[r,p,k] = residue([1,-0.3],[1,0.3,0.36,0.108,0]);

function [impulse_response] = myfun_impulse_response_Des(r,p,t)
impulse_mid= r.*(p.^t);
impulse_response = sum(impulse_mid);
```

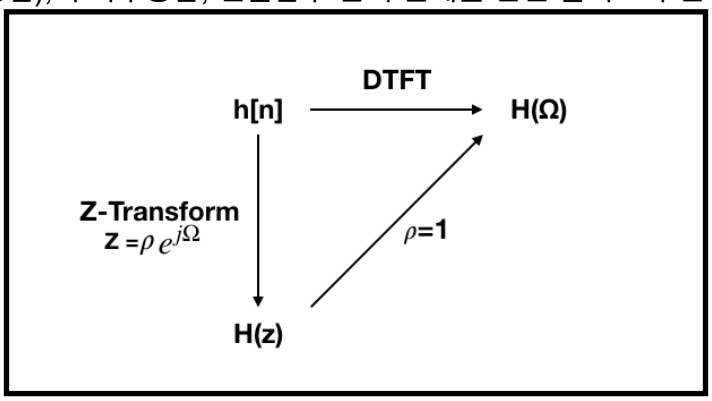
위 코드중 residue 부분의 분모항의 계수를 입력할때 마지막에 0 을 하나 더 입력해준다.
역변환 과정중 Z 를 곱하는 것에 의해서 생겨나는 과정이다. 함수부분을 살펴보면 위에있는 식대로 진행한 모습을 살펴 볼 수 있다. 이를 통해서 나오는 결과값은 다음과 같다.



충격응답을 살펴보니 수렴한다. 즉 시스템이 안정하다고 판단할 수 있다. 3.1에서 예상 한 것과 같이 전달함수의 Pole 값들이 단위원 내부에 위치하게 되면 안정하다 라는 사실을 다시한번 확인 할 수 있었다.

3.3 위 시스템 H(z) 로부터 주파수응답 H(Ω)을 구하고 크기 |H(Ω)| 를 그래프에 표시하여라.

주파수응답을 구할때 MATLAB 내부함수를 이용해도 좋지만 우리는 배우는 단계인 만큼 차례대로 접근해 보았다. 이산 시스템의 주파수 응답은 Pole 값과 단위원 사이의 거리 분의 Zero값과 단위원 사이의 거리이다. 왜 이렇게 되는지에 대해서 간략하게 살펴보고자 한다.
우선 충격응답(임펄스응답), 주파수응답, 전달함수 간의 관계를 한번 살펴보자 한다.



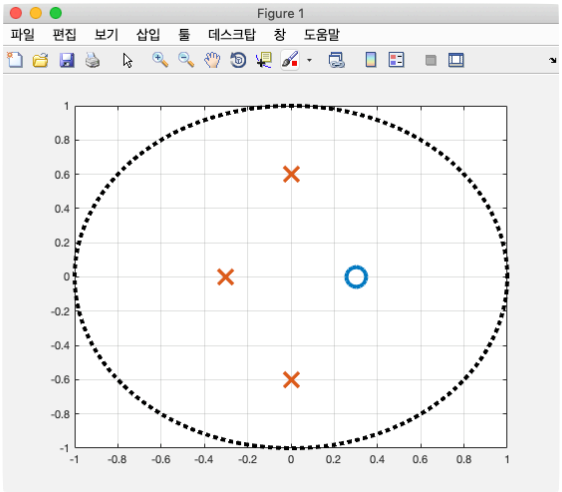
임펄스 응답에 대해서 Z 변환을 실시하면 전달함수가 되며 임펄스 응답에 대해 DTFT를 하면 주파수 응답을 구할 수 있다. 하지만 이 셋간의 성질을 잘 이용하면 전달함수에서 바로 주파수 응답을 구할 수 있다. $Z = \rho e^{j\Omega}$ 여기서 ρ 값이 1이라면 $Z = e^{j\Omega}$가 되게 된다. 이를 통해서 전달함수에서 바로 주파수 응답을 구할 수 있는데 식으로 과정을 나타내면 다음과 같다.

$$H(Z) = \frac{(Z - z_1)(Z - z_2)\dots(Z - z_n)}{(Z - p_1)(Z - p_2)\dots(Z - p_n)}$$

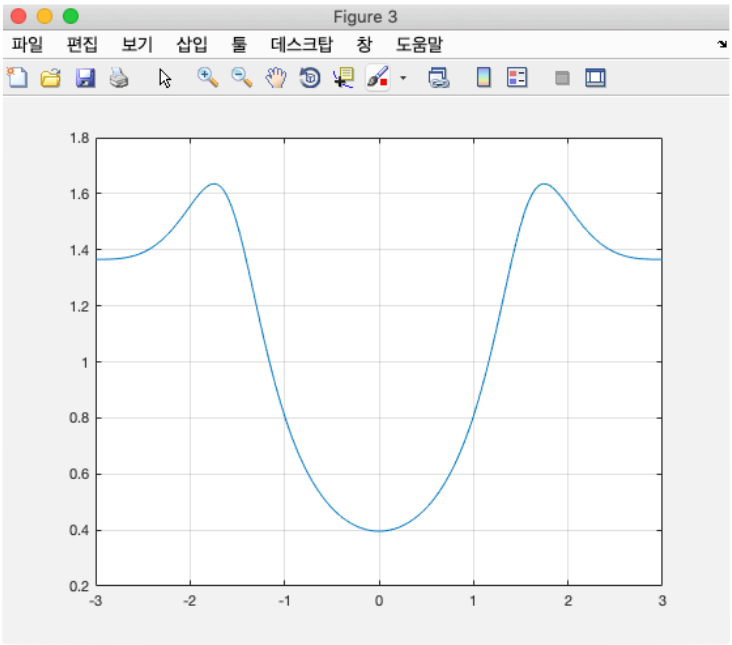
우선 전달함수를 다음과 같은 형태로 만들어 준다. 그 이후 $Z = e^{j\Omega}$ 값을 집어 넣어 주면 다음과 같다.

$$H(\Omega) = \frac{(e^{j\Omega} - z_1)(e^{j\Omega} - z_2)\dots(e^{j\Omega} - z_n)}{(e^{j\Omega} - p_1)(e^{j\Omega} - p_2)\dots(e^{j\Omega} - p_n)}$$

위 식에 절대값을 취해주면 각각의 분자식, 분모식은 단위원과 각 p,z 까지의 거리를 구하는 식이랑 동일하다고 볼 수 있다. 이를 가지고도 주파수 응답의 모습을 대략적으로 예측할 수도 있다.
단위원을 도는 점과 pole, zero 사이의 거리이기 때문이다. 원점과 pole,zero 까지를 일직선 상으로 그었을 때 작은 값을 가지게 된다고 볼 수 있다. 하지만 위는 분수식이기 때문에 분모가 작아질때 전체적인 값이 커지게 되고 분자가 작아질때 전체적인 값이 커지게 된다. 즉 원점과 pole을 일직선으로 그었을때의 단위원의 지점에서 큰 값을 가지고 원점과 zero를 일직선으로 그었을때의 단위원의 지점에서 작은 값을 가지게 된다고 볼 수 있다.



Pole-zero plot 을 다시한번 살펴보면 영점인 0에서 값이 가장 작을 것이고 $\pi/2$, $-\pi/2$ 에서 큰 값을 가지며 π에서는 앞선 두 값보다는 다소 작은 값을 가지게 될것이라 예상할 수 있다.
그렇다면 실제 주파수 응답은 어떻게 나오는지 살펴보도록 하자.

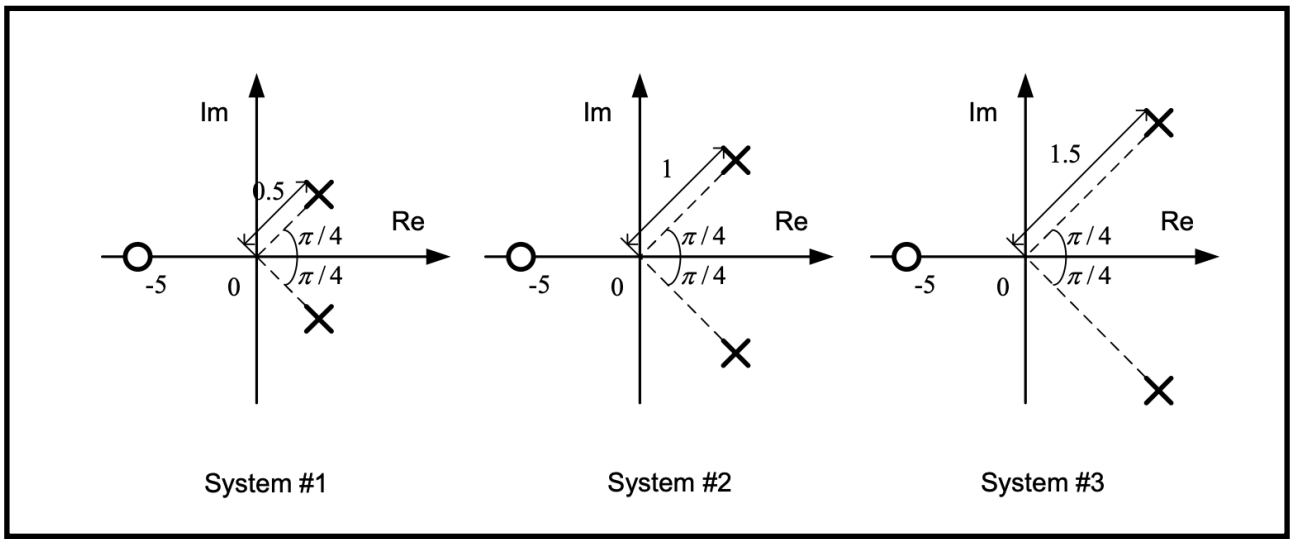


우리의 예상대로 분모값이 가장 작아지는 지점에서 주파수 응답의 크기는 커졌으며 분자값이 작아지는 지점에서는 주파수 응답의 크기는 작아지는 것을 확인 할 수 있었다. 이를 구현하는 코드는 다음과 같다.

```
function [se,freq] = myfun_freq_response_Des(r,p)
se= linspace(-pi,pi,360);
x=exp(1j*se);
   bun_mo = abs(x-p);
    bun_ja = abs(x-r);
if size(bun_mo) >1
    bun_mo_mult = prod(bun_mo);
else
    bun_mo_mult = bun_mo;
end
if size(bun_ja) >1
    bun_ja_mult = prod(bun_ja);
else
    bun_ja_mult = bun_ja;
end
freq = (bun_ja_mult./bun_mo_mult);
```

4. 이산시스템의 안정성

4.1 다음 세 시스템에 대해 전달함수를 구하고 ,충격응답을 그래프에 표시하라.



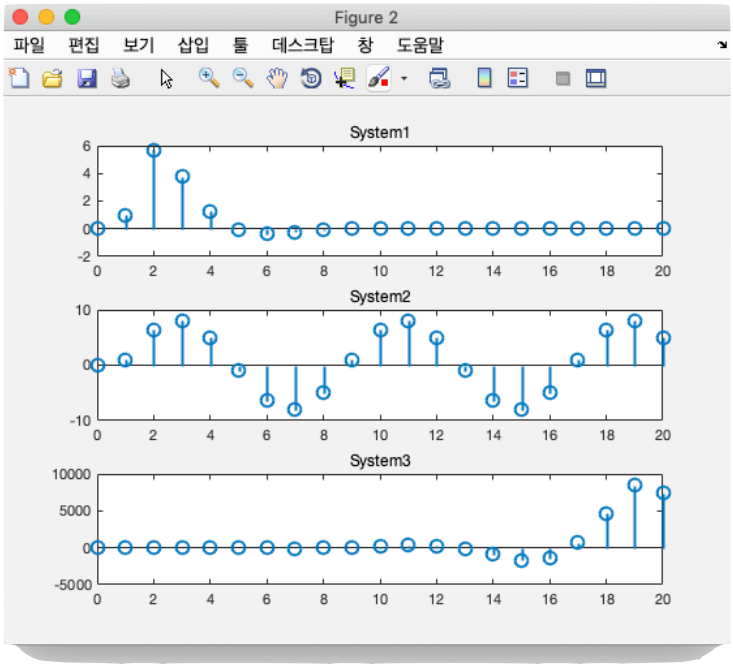
위 세 시스템의 Pole-zero plot이 주어졌으니 우리는 세 시스템의 pole 값과 zero 값을 구할 수 있다. 공통적으로 zero 값은 -5 임으로 pole 에대해서만 구해보면 다음과 같다.

$$System1_{pole} = 0.5e^{j\frac{\pi}{4}}, 0.5e^{j\frac{-\pi}{4}}$$
$$System2_{pole} = e^{j\frac{\pi}{4}}, e^{j\frac{-\pi}{4}}$$
$$System3_{pole} = 1.5e^{j\frac{\pi}{4}}, 1.5e^{j\frac{-\pi}{4}}$$

이제 Pole 값과 Zero 값을 알아내었으니 이를 전달함수 식으로 만든 다음에 다시 부분분수 형태로 변환 후 임펄스 응답을 구하면 된다. 임펄스 응답을 구하는 과정은 3.2에서도 설명하였으니 생략하도록 하겠다. Pole 과 Zero 값으로 부터 전달함수를 구하는 과정은 MATLAB에서 zp2tf 내장함수를 사용하면 쉽게 구할 수 있다. 구체적인 코드는 다음과 같다.

```
sys_1_zero = [-5];
sys_1_pole = [0.5*exp(1j*(pi/4)) , 0.5*exp(1j*(-pi/4))];
[sys_1_ja, sys_1_mo]=zp2tf(sys_1_zero,sys_1_pole,1)
[sys_1_r,sys_1_p,k] = residue(sys_1_ja,[sys_1_mo,0])
impulse_response_1 = myfun_impulse_response_Des(sys_1_r,sys_1_p,t);
```

이러한 방식으로 System 1,2,3 에대서 임펄스 응답을 구할 수 있으면 임펄스 응답의 그래프는 다음과 같이 그려진다.



구체적으로 안정성에 관한 이야기는 다음 질문에서 계속 이어나가도록 하겠다.

4.1 위 세시스템의 안정성을 판단하고 결과를 시간영역에서 확인하라.

위 그래프를 보아하니 System 1은 수렴하며, System 2 는 진동하고 System 3 는 발산하는 모습을 볼 수 있다. 이러한 차이에 대한 설명은 3.2 에서 이미 설명이 되어 있다. 다음 식을 한번 더 바라봐 보자.

$$H(Z) = C_1 + \frac{C_2 Z}{(Z - p_1)} + \frac{C_3 Z}{(Z - p_2)} \ldots + \frac{C_n Z}{(Z - p_{n-1})}$$

Z-Transform 역변환~~

$$h[n] = C_1\delta[n] + C_2 P_1^n + C_3 P_2^n + \ldots C_n P_{n-1}^n$$

전달함수를 Z 역변환 하면 다음과 같은 임펄스 응답을 구할 수 있다. 세 시스템의 차이가 나는 이유는 Pole 값에 있다. Pole 에 n 승이 되어 있는 꼴인데 Pole 값이 1보다 크면 이는 무한대로 커질 것이다. 반면 Pole이 1보다 작으면 수렴하게 된다. 그리고 Pole 값이 1이면 계속 같은 값을 반복하니 진동하게 되는 것이다. 결과적으로 Pole의 크기가 1보다 작은 System 1 은 수렴하게 되고, Pole 의 크기가 1인 System 2는 진동하게 되며, Pole 의 크기가 1보다 큰 System 2 는 발산하게 된다.