

#### 4. Control System Toolbox

본 장에서는 제어시스템을 분석하거나 시뮬레이션 하는데 유용하게 쓸 수 있는 control system toolbox의 함수를 소개한다. State-space model은 다루지 않고, transfer function model만 다룬다. 자세한 내용은 MATLAB의 control system toolbox document를 참조하는 것이 좋다.

##### 4.1. System Model

시스템 모델을 분석하거나 시뮬레이션하려면 우선 시스템을 표현해야 한다. Control system toolbox는 해당 toolbox 내에서 시스템을 표현하는데 사용하기 위한 변수 타입을 정의하였으며, 이 변수는 toolbox가 제공하는 함수를 통해 만들 수 있다.

###### ① Transfer Function Model

Transfer function의 분자, 분모의 계수로 변수를 만들어주는 함수이다. 첫 번째 인자에 분자, 두 번째 인자에 분모의 계수를  $s$ 에 대한 차수가 높은 순으로 벡터로 입력한다.

예제4.1. Transfer Function	
<pre>&gt;&gt; sys1 = tf([1 2],[1 0 10])</pre> Transfer function: $\frac{s + 2}{s^2 + 10}$	변수 sys1에 분모 [1 2], 분자 [1 0 10]의 계수를 갖는 transfer function을 저장한다. 출력된 결과를 통해 원하는 형태가 되었는지 확인할 수 있다.

###### ② Zero-Pole-Gain Model

Transfer function에서 zero와 pole, gain을 통해 변수를 만들어준다. 첫 번째 인자는 zero를 나타내는 벡터, 두 번째 인자는 pole을 나타내는 벡터, 세 번째 인자는 gain을 나타내는 스칼라를 입력한다.

예제4.2. Zero-Pole-Gain	
<pre>&gt;&gt; sys2 = zpk([2 3], [0 -1], -5)</pre> Zero/pole/gain: $\frac{-5(s-2)(s-3)}{s(s+1)}$	변수 sys2에 zero [2 3], pole [0 -1], gain -5를 갖는 transfer function을 저장한다.  만약 zero나 pole이 없을 경우, [ ]를 통해 빈 행렬을 인자로 넣어준다.

###### ③ PID Controller (Parallel)

PID controller의 transfer function은 함수 tf를 응용하여 만들 수 있다.

$$K_p + K_i/s + K_d s = (K_d s^2 + K_p s + K_i) / s$$

위와 같이 변형하면 PID를 2차/1차의 transfer function으로 변환한 것이다.

#### 예제4.3. PID Controller

```
>> Kp = 10; Ki = 3; Kd = 0.1;
>> pid_sys = tf([Kd Kp Ki], [1 0])
```

Transfer function:

$$0.1 s^2 + 10 s + 3$$

-----

s

Kp, Ki, Kd를 미리 만든 후, PID controller의 transfer function을 만든다. 분모의 s는 계수 벡터 [1 0]으로 표현한다.

#### ④ Lead-Lag Compensator

Compensator는 함수 zpk를 통해 만드는 것이 편리하다.

Lead Compensator	$K (s+z) / (s+p), \quad z < p$
Lag Compensator	$K (s+z) / (s+p), \quad z > p$
Lead-Lag Compensator	$K (s+z_{lag})(s+z_{lead}) / (s+p_{lag})(s+p_{lead}), \quad z_{lag} > p_{lag}, \quad z_{lead} < p_{lead}$

#### 예제4.4. Lead-Lag Compensator

```
>> K = 205;
>> zld = 3; pld = 9.61; zlg = 0.092; plg = 0.01;
>> comp_sys = zpk([-zld -zlg], [-pld -plg], K)
```

Zero/pole/gain:

$$205 (s+3) (s+0.092)$$

-----

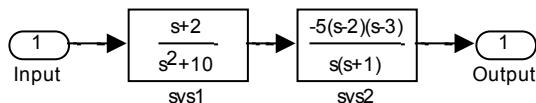
$$(s+9.61) (s+0.01)$$

K와 각 zero, pole을 미리 만든 후, compensator의 transfer function을 만든다. 변수 zld, pld, zlg, plg는 bandwidth를 나타내므로 zero와 pole로 입력할 때는 - 부호를 붙여주어야 한다.

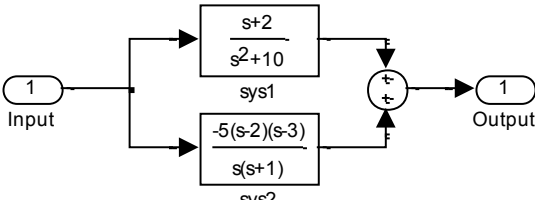
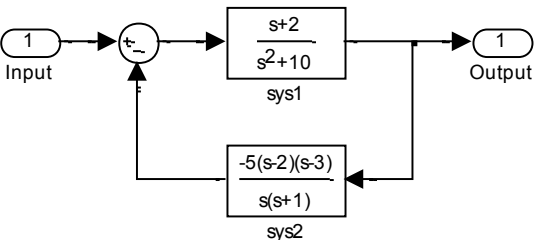
Transfer function은 시스템에서 입력과 출력의 관계를 나타내는 frequency-domain의 식으로, 서로 다른 시스템의 입력과 출력을 연결할 수 있다. 다르게 말하면, 각 부분 시스템을 연결하여 전체 시스템을 표현할 수 있는 것이다. 크게 세 가지의 연결방식을 조합하여 전체 시스템이 완성되는데, 먼저 직렬 연결의 경우 한 시스템의 출력이 다른 시스템의 입력으로 들어가는 형태로, 이는 transfer function 사이의 곱으로 표현할 수 있다. 병렬 연결의 경우, 두 시스템에 같은 입력이 들어가서 나온 출력을 서로 더하게 되므로, transfer function에서도 덧셈으로 표현할 수 있다. 마지막으로, 전체 시스템의 최종 출력이 다시 입력으로 들어가는 형태를 피드백 연결이라고 하는데, 이는  $G/(1+GH)$ 로 나타난다(G는 open-loop, H는 feedback-loop).

#### 예제4.5. System Interconnection

Series



$$\text{Sys} = \text{sys1} \times \text{sys2}$$

Parallel		Sys = sys1 + sys2
Feedback		Sys = sys1 / ( 1 + sys1 X sys2 )
<div><div><pre>&gt;&gt; sys = series(sys1, sys2)</pre><p>Zero/pole/gain:</p><p>-5 (s-2) (s-3) (s+2)</p><p>-----</p><p>s (s+1) (s^2 + 10)</p></div><div><pre>&gt;&gt; sys = parallel(sys1, sys2)</pre><p>Zero/pole/gain:</p><p>-5 (s-3.517) (s-1.771) (s^2 + 0.08743s + 9.635)</p><p>-----</p><p>s (s+1) (s^2 + 10)</p></div><div><pre>&gt;&gt; sys = feedback(sys1, sys2)</pre><p>Zero/pole/gain:</p><p>s (s+2) (s+1)</p><p>-----</p><p>(s+1.793) (s-1.108) (s^2 - 4.685s + 30.2)</p></div><div><p>함수 series, parallel, feedback을 통해 세 가지 연결방식을 사용한다.</p><p>Series에서 각 시스템의 zero와 pole이 연결 후에도 유지되고 있는 것을 확인할 수 있다.</p><p>Parallel에서 시스템 pole은 동일하나, zero가 바뀌는 것을 확인할 수 있다.</p><p>Feedback의 pole은 연결 전과의 관계를 알기 힘들지만, zero는 open-loop의 zero와 feedback-loop의 pole이 나타나 있다.</p><p>현재 각 변수가 zpk 방식(인수분해)으로 출력되었는데, tf 방식(다항식)으로 바꾸고 싶을 경우, tf(sys)를 입력하면 된다. 그 역으로 tf 방식을 zpk 방식으로 바꿀 때도 zpk(sys)를 입력하면 된다.</p></div></div>		

#### 4.2. Time Response

Transfer function은 frequency-domain의 식으로, time-domain에서 임의의 test input을 주었을 때의 response를 알아내기 위해서는 계산을 통해 시뮬레이션 해보아야 한다. Test input이란 시스템의 특성을 평가하기 위해 보편적으로 사용하는 입력 신호로, impulse와 step, ramp, sin 등이 있다.

## ① Impulse Response

시스템에 impulse 입력을 주었을 때의 결과를 나타내는 것으로, transfer function의 역라플라스 변환 식을 그대로 나타낸다. Toolbox에 동일한 이름의 함수가 있으며, 기본형은 impulse(sys)이다.

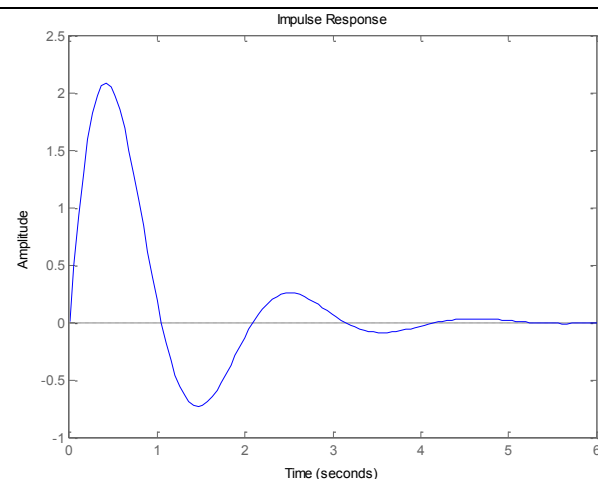
### 예제4.6. Impulse Response

```
>> sys = tf(10, [1 2 10]);
>> impulse(sys)
>> impulse(sys, 3)
>> impulse(sys, [0:0.0001:3])
>> [y, t] = impulse(sys);
>> t = 0:0.0001:3;
>> y = impulse(sys, t);
```

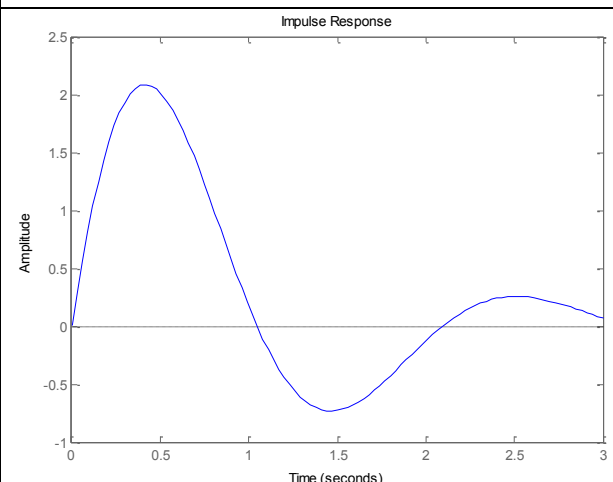
Impulse(sys)는 transfer function sys의 impulse response를 계산한다. 출력 변수를 지정하지 않을 경우 자동으로 plot을 띄워주며, 출력 변수가 있을 경우 계산한 amplitude와 시간 벡터를 출력한다.

두 번째 인자는 시간을 의미하며, 스칼라일 경우 최종 시간, 벡터일 경우 계산에 사용할 시간 벡터를 지정한다. 즉, impulse(sys, [0:0.0001:3])은 impulse response를 0초부터 3초까지, 0.0001초 간격으로 계산하라는 의미이다.

시간 입력이 없으면 입력 transfer function에 따라 적절한 time 벡터를 만들어 쓴다.



Impulse(sys) - 6초까지 시뮬레이션



Impulse(sys, 3) - 3초까지 시뮬레이션

## ② Step Response

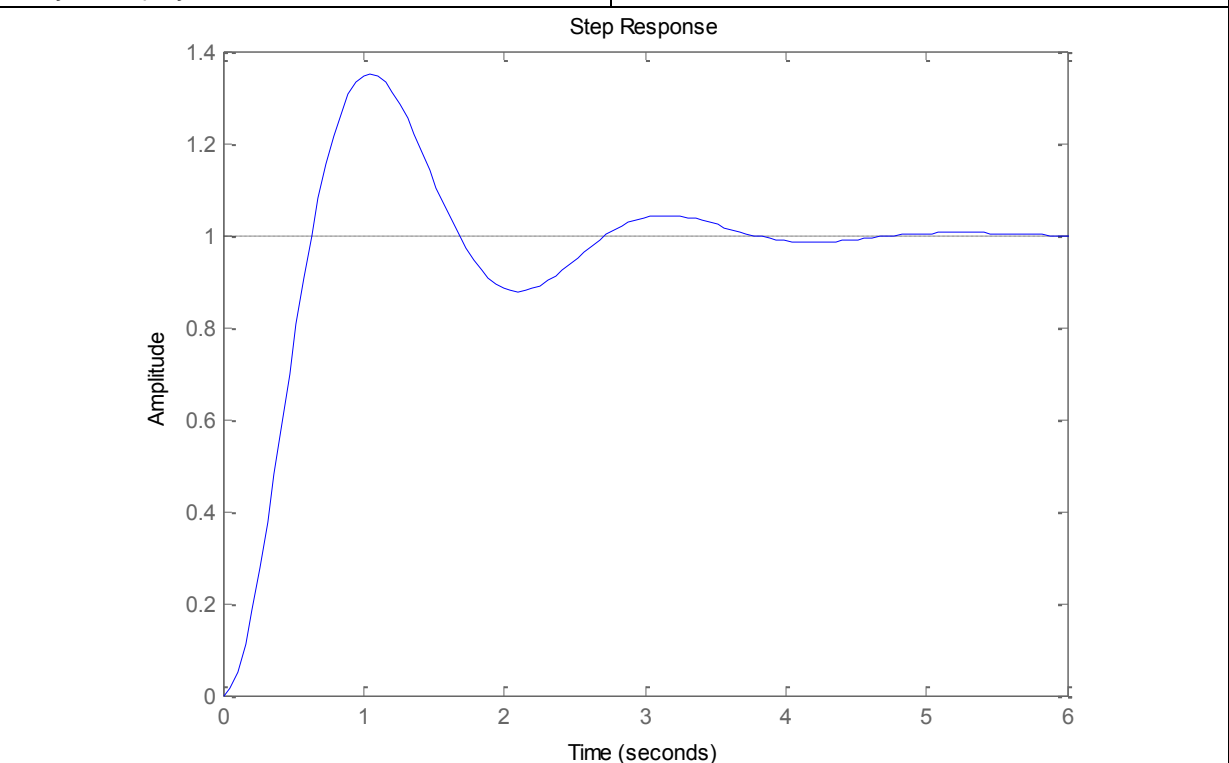
시스템에 step 입력을 주었을 때의 결과이다. 일반적으로 transient response를 평가할 때 사용하는 test input이며, toolbox에 동일한 이름의 함수가 있다. 사용법은 함수 impulse와 동일하다.

### 예제4.7. Step Response

```
>> sys = tf(10, [1 2 10]);
>> step(sys)
>> step(sys, 3)
>> step(sys, [0:0.0001:3])
```

Step(sys)는 transfer function sys의 step response를 계산한다. 함수의 사용법은 impulse와 완전히 동일하다.

```
>> [y, t] = step(sys);
>> t = 0:0.0001:3;
>> y = step(sys, t);
```



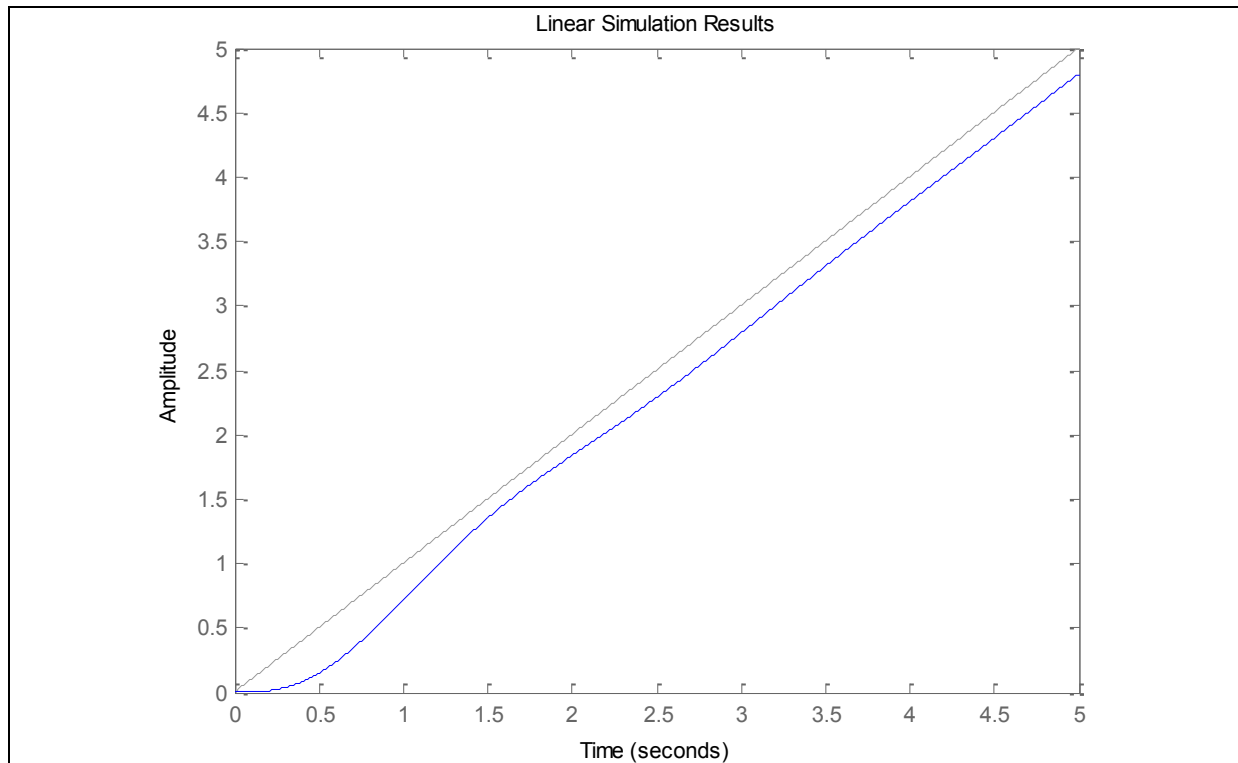
### ③ Ramp Response

Ramp 입력은 일반적으로 tracking performance를 확인할 때 사용하는 test input이다. 이 시뮬레이션은 impulse나 step과 달리 전용 함수가 있는 것이 아니라 lsim이라는 linear simulation 함수를 응용하여 계산한다. 함수 lsim은 입력 벡터를 직접 만들어 넣어서 사용한다. 기본형은 lsim(sys, u, t)로, u는 입력 벡터, t는 시간 벡터이다.

#### 예제4.8. Ramp Response

```
>> sys = tf(10, [1 2 10]);
>> t = 0:0.0001:5;
>> u = t;
>> lsim(sys, u, t)
>> y = lsim(sys, u, t);
```

함수 lsim은 입력 벡터와 시간 벡터를 반드시 지정해야 한다는 점을 제외하면 impulse, step과 사용법이 비슷하다. 먼저 시간 벡터 t를 정의하고, 입력 벡터를 t와 같게 두면 ramp input을 만들 수 있다. 이를 출력 변수 없이 lsim(sys, u, t)로 넣으면 plot을 만들며, 출력 변수를 주면 amplitude를 출력한다.



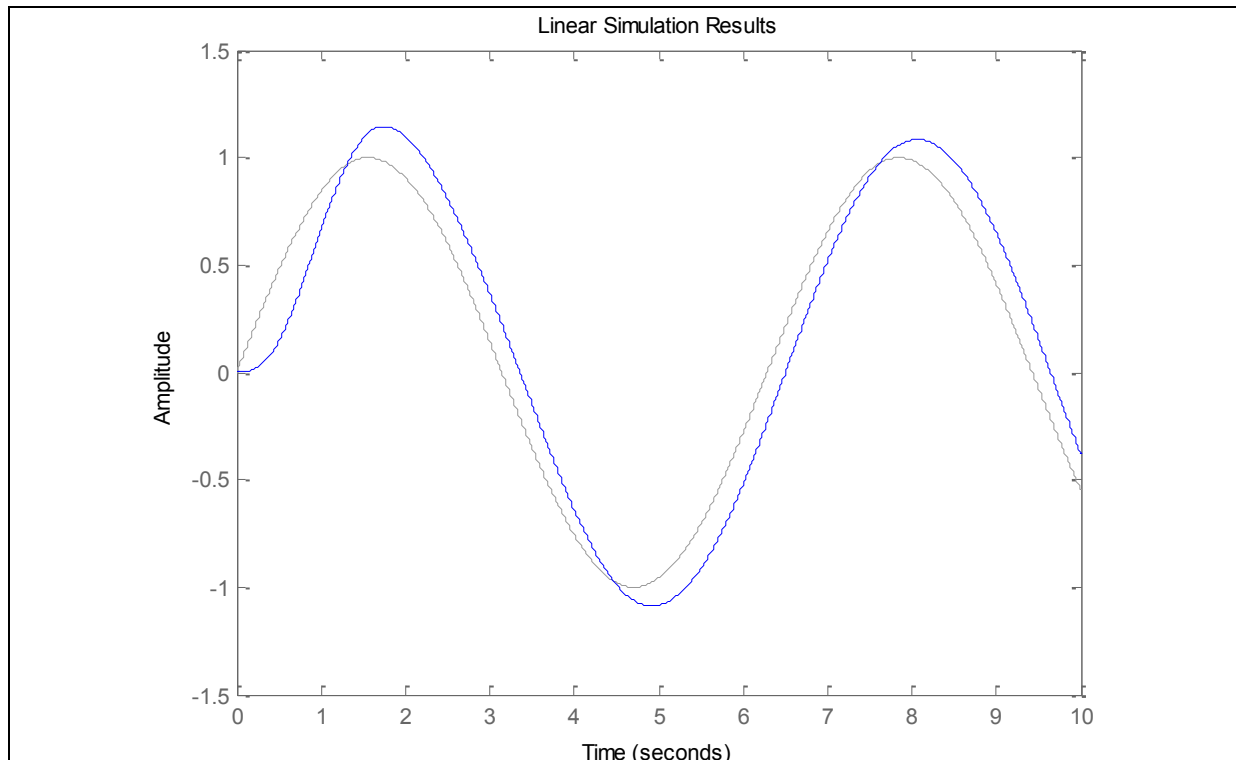
#### ④ Sinusoidal Response

Sinusoidal 입력의 time response 시뮬레이션은 잘 실행하지 않는다. 중요한 특성은 steady state에서의 게인과 위상 지연으로, frequency response에서 쉽게 확인할 수 있다. 그러나 중요한 test input이므로, time response를 확인하는 예제4.9를 첨부하였다. Sinusoidal response도 ramp와 같이 lsim 함수를 응용하여 계산한다.

##### 예제4.9. Sinusoidal Response

```
>> sys = tf(10, [1 2 10]);
>> t = 0:0.001:10;
>> u = sin(t);
>> lsim(sys, u, t)
>> y = lsim(sys, u, t);
```

입력 벡터를 만들 때, sin 함수로 계산하였다. 그 외에는 ramp response와 동일하다. 주파수를 바꾸고 싶을 경우, sin(t)에서 t에 rad/sec 단위의 주파수를 곱해주면 된다.



Time response 시뮬레이션은 보통 transient response를 관찰하기 위해 계산한다. 그 중에서도 특히 step response를 많이 사용하는데, 이 response가 reference transition이 일어날 때와, 고정된 reference로의 steady state를 동시에 잘 보여주기 때문이다. Step response를 평가할 때 보편적으로 rise time, settling time, overshoot, undershoot 등을 기준으로 사용하는데, 함수 step의 출력 변수로부터 계산하려면 번거로운 경우가 많다. 그래서 toolbox에서 stepinfo라는 함수를 제공한다. 이 함수는 시스템의 step response로부터 앞의 특성 값을 계산해준다.

예제4.10. stepinfo	
<pre>&gt;&gt; sys = tf(10, [1 2 10]); &gt;&gt; stepinfo(sys)  ans =      RiseTime: 0.4257   SettlingTime: 3.5359   SettlingMin: 0.8769   SettlingMax: 1.3509     Overshoot: 35.0913     Undershoot: 0         Peak: 1.3509     PeakTime: 1.0492  &gt;&gt; stepinfo(sys, 'SettlingTimeThreshold', 0.01)</pre>	<p>Stepinfo(sys)는 transfer function의 step response 특성을 계산해준다.</p> <p>일반적으로 settling time을 계산할 때 2% 또는</p>

<pre>ans =     RiseTime: 0.4257     SettlingTime: 4.4855     SettlingMin: 0.8769     SettlingMax: 1.3509     Overshoot: 35.0913     Undershoot: 0     Peak: 1.3509     PeakTime: 1.0492  &gt;&gt; stepinfo(sys,'RiseTimeLimits', [0.05 0.95])  ans =     RiseTime: 0.4989     SettlingTime: 3.5359     SettlingMin: 0.8769     SettlingMax: 1.3509     Overshoot: 35.0913     Undershoot: 0     Peak: 1.3509     PeakTime: 1.0492  &gt;&gt; S = stepinfo(sys); &gt;&gt; S.RiseTime  ans =     0.4257</pre>	<p>5% 이내 수렴을 기준으로 하는데, 함수의 기본 값은 2%이다. 이를 변경하여 계산할 경우, stepinfo(sys, 'SettlingTimeThreshold', ST)로 명령하면 ST에 따라 수렴 기준이 변경된다. 1% 수렴을 계산하고 싶을 경우, ST = 0.01이다.</p> <p>Settling time과 마찬가지로 rise time을 계산할 기준도 변경할 수 있다. 함수의 기본 값은 [0.1 0.9]이다.</p> <p>함수 stepinfo의 출력은 구조체 타입으로, C 언어에서 '.'을 이용해 내부 변수를 사용하는 방법과 동일하게 다룰 수 있다.</p>
--	---

#### 4.3. Frequency Response

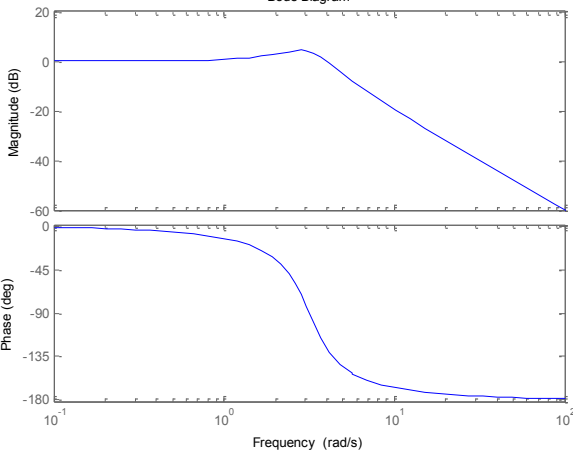
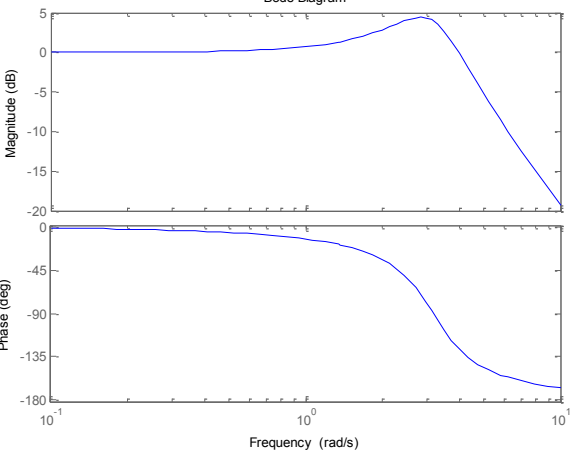
Toolbox에는 LTI 시스템의 frequency response를 보기 위한 함수를 제공하고 있다. 주로 Bode, Nyquist, Nichols plot이 사용된다.

##### ① Bode Plot

Bode plot은 frequency response를 게인과 위상지연으로 나누어 주파수 축에 대하여 그리는 그래프이다. 게인은 dB 단위를 사용하며, 위상 지연은 deg, 주파수는 rad/sec를 사용한다.

예제4.11. Bode Plot	
>> sys = tf(10, [1 2 10]);	Time response 계열 함수와 마찬가지로, bode도

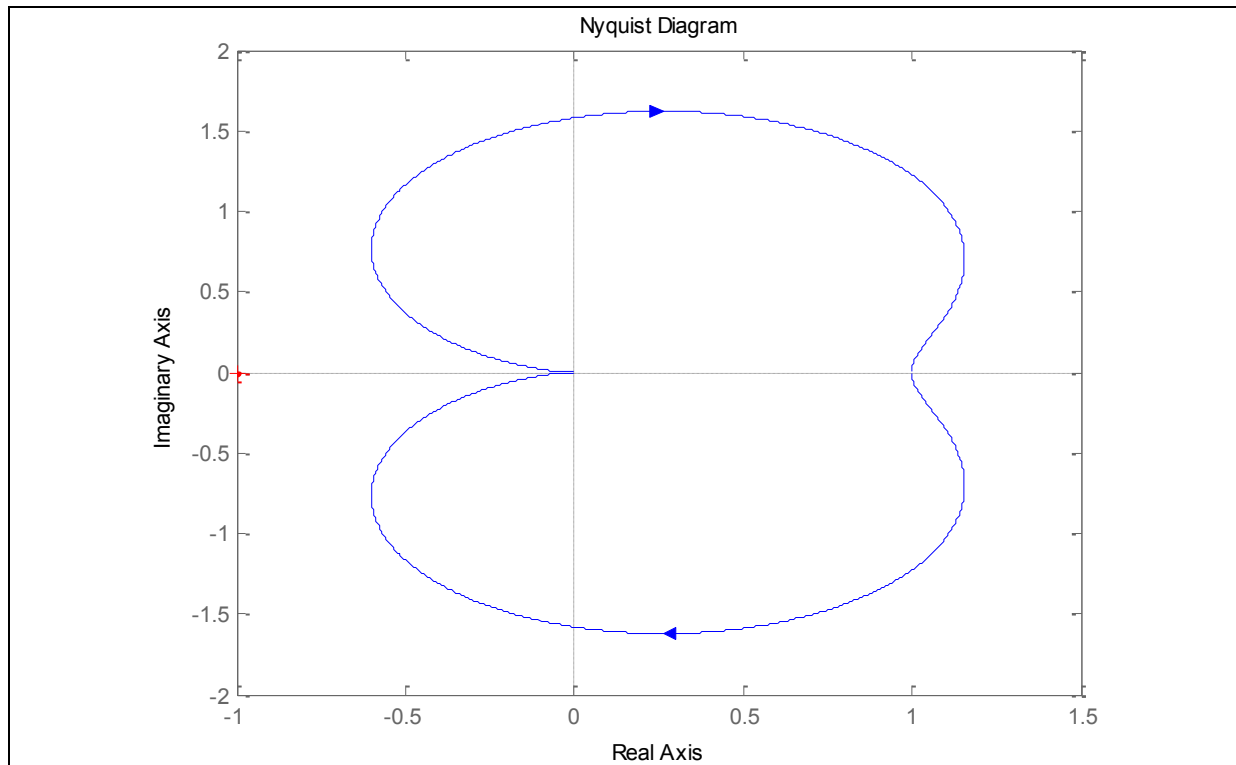


<pre>&gt;&gt; bode(sys) &gt;&gt; w = logspace(-1, 1, 1000); &gt;&gt; bode(sys, w) &gt;&gt; [mag, phase] = bode(sys, w);</pre>	<p>출력 변수가 없을 경우 자동으로 plot을 그리며, 있을 경우 주파수 특성인 게인과 위상 지연을 출력한다.</p> <p>원하는 주파수 벡터를 입력하여 범위와 간격을 조절할 수 있는데, bode plot의 주파수 표현이 로그 스케일이므로 주파수 벡터도 로그 스케일로 만드는 것이 좋다.</p>
 <p>Bode(sys) – 0.1 ~ 100 rad/sec 범위</p>	 <p>Bode(sys, w) – 0.1 ~ 10 rad/sec 범위</p>

## ② Nyquist Plot

Nyquist Plot은 복소평면에서 주파수에 따른 frequency response의 변화를 복소수로 그린 것이다. Frequency response는 게인과 위상 지연으로 이루어져 있는데, 두 값을 하나의 복소수로 나타낼 수 있기 때문에 nyquist plot이 가능한 것이다. 주파수에 따라 서로 다른 복소수 값을 지니며, 차례대로 이은 그래프이기 때문에 방향성이 존재하며, nyquist 함수로 그리면 그래프 상에 삼각형으로 방향이 나타난다.

<p>예제4.12. Nyquist Plot</p> <pre>&gt;&gt; sys = tf(10, [1 2 10]); &gt;&gt; nyquist(sys) &gt;&gt; w = logspace(-2, 4, 10000); &gt;&gt; nyquist(sys, w) &gt;&gt; [re, im] = nyquist(sys, w);</pre>	<p>사용법은 함수 bode와 동일하며, 출력 변수가 게인과 위상 지연이 아니라 복소수의 실수부와 허수부라는 점이 다르다.</p>
--	--



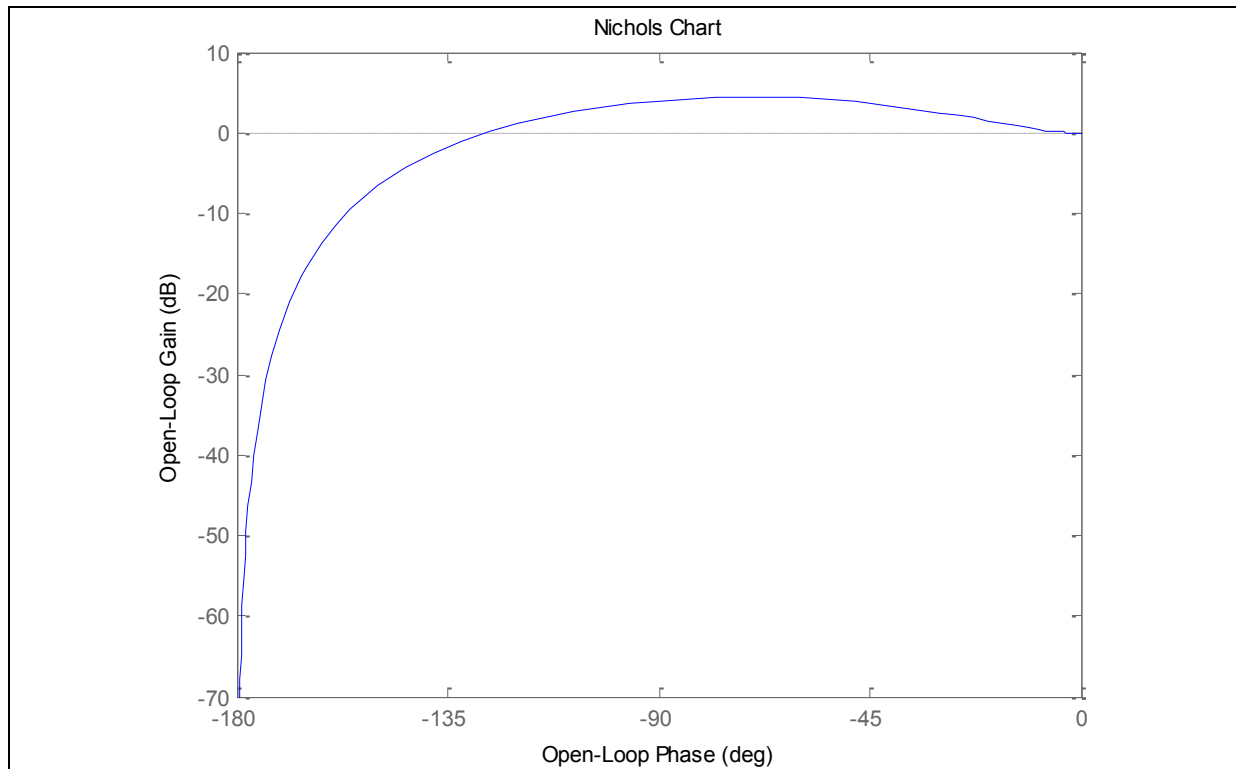
### ③ Nichols Plot

Nichols Plot은 Bode plot의 게인과 위상 지연을 y축과 x축으로 두고 그린 그래프이다. 축의 단위는 Bode와 동일하게 dB와 deg이다.

#### 예제4.13. Nichols Plot

```
>> sys = tf(10, [1 2 10]);
>> nichols(sys)
>> w = logspace(-1, 1, 1000);
>> nichols(sys, w)
>> [mag, phase] = nichols(sys, w);
```

사용법은 앞의 두 함수와 동일하며, 출력 변수는 게인과 위상 지연이다.



Frequency response의 목적은 일반적으로 안정도를 판별하는데 있으며, 특히 gain margin, phase margin, gain crossing frequency, phase crossing frequency가 중요하다. Gain margin은 위상 지연이 180 deg인 주파수에서의 게인을 의미하며, phase margin은 게인이 0 dB인 주파수에서의 위상 지연과 -180 deg 사이의 위상 차를 의미한다. Crossing frequency는 각각 게인이 0 dB, 위상 지연이 -180 deg인 주파수를 말한다. 이 값으로 현재 시스템의 입력과 출력을 연결하여 피드백 해줄 때, 안정도가 어느 정도인지를 판별할 수 있다. Toolbox에서는 margin이라는 함수를 제공하여 이와 같은 값을 쉽게 계산할 수 있도록 한다.

#### 예제4.14. Margin

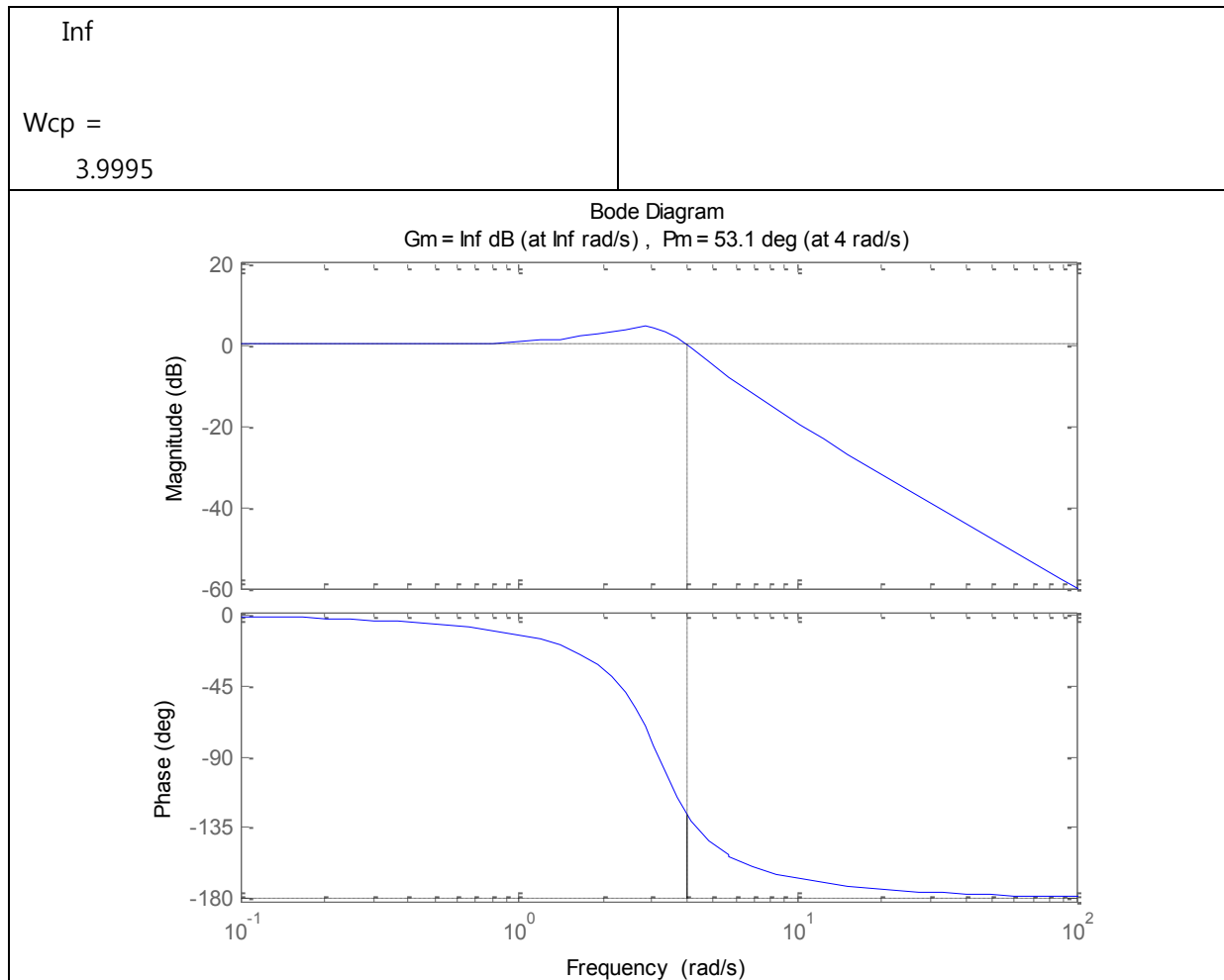
```
>> sys = tf(10, [1 2 10]);
>> margin(sys)
>> [Gm, Pm, Wcg, Wcp] = margin(sys)
```

```
Gm =
    Inf
```

```
Pm =
    53.1445
```

```
Wcg =
```

함수 margin에 transfer function을 입력하면 gain margin, phase margin, crossing frequency를 출력한다. 출력 변수를 지정하지 않았을 때에는 Bode plot에 각 요소를 그려준다.



#### 4.4 Controller Design

Control system toolbox는 feedback controller 설계를 위한 함수를 제공한다. 단일 계인에 대하여 pole의 위치 변화를 알아볼 수 있는 root locus를 구현한 함수와, 자체 알고리즘을 통해 PID 계인을 구해주는 함수가 대표적이다.

##### ① Root Locus

시스템  $G(s)$ 에  $K$ 배의 feedback loop가 주어져 있다고 가정할 때, 전체 시스템의 transfer function은  $G(s)/[1+KG(s)]$ 로 나타난다. Transfer function의 분모의 근(pole)이 시스템의 stability와 response time을 결정하므로, feedback loop의 characteristic equation인  $1+KG(s)=0$ 의 근을 파악하면 시스템의 stability와 response time을 알 수 있는 것이다. Root locus란, 이 characteristic equation에서  $K$ 의 값의 변화에 따라 근이 바뀌는 경로를 그린 것으로, 단일 계인을 정할 때 큰 도움을 준다.

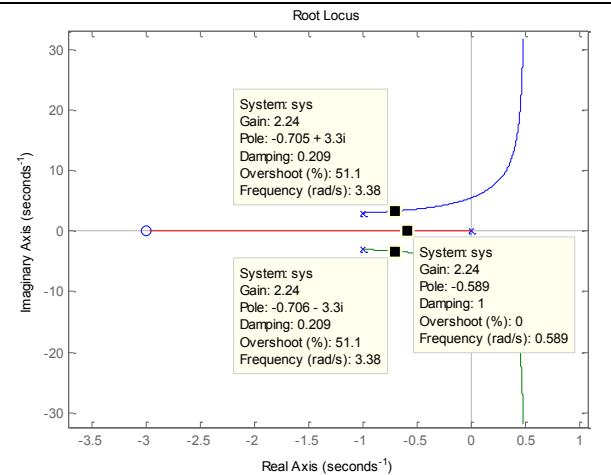
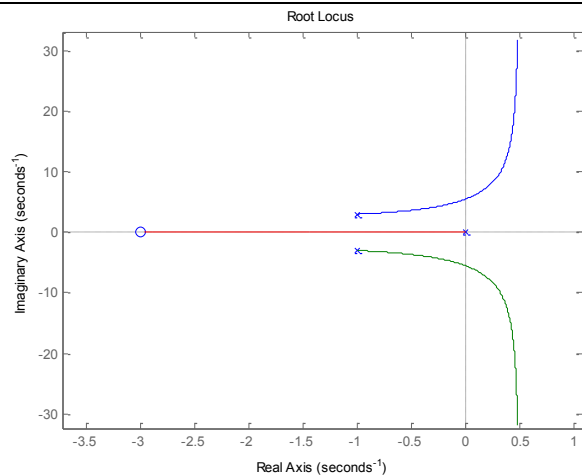
##### 예제4.15. Root Locus

```
>> sys = tf([1 3], [1 2 10 0]);
>> rlocus(sys)
```

함수 rlocus에 transfer function과 feedback 계인  $K$ 를 입력하면 각  $K$ 값에 대한 pole을

```
>> K = logspace(-2, 3, 10000);
>> rlocus(sys, K)
>> R = rlocus(sys, K);
```

출력한다. 출력 변수를 지정하지 않았을 때에는 복소평면에 pole의 경로를 그려주며, 이 때 data cursor를 통해 원하는 pole을 확인할 수 있다.



## ② PID tuner

PID controller는 대부분의 control system에서 사용하는 대표적인 controller이다. 구현이 간단하며, 대부분의 시스템에서 만족할만한 성능을 구현할 수 있다. 그러나 절대적인 설계법이 정해져 있지 않아 설계자의 경험에 의존하는 경우가 많으며, 대부분의 설계 기법이 가이드 라인을 제시할 뿐 여러 번의 실험을 통해 튜닝을 거쳐야 한다. Toolbox는 알려진 시스템에 대하여 튜닝을 시작하기 위한 기준 게인을 마련해주는 역할을 하며, 제한 조건을 다양하게 입력할 수 있다.

함수 pidtune의 기본 형태는  $C = \text{pidtune}(\text{sys}, \text{type})$ 이며, sys는 plant 시스템의 transfer function, type은 controller의 type을 의미한다. Type은 문자열로 입력하며, 다음 표를 참조하자.

Controller Type	설명
'P'	P 제어기
'I'	I 제어기
'PI'	PI 제어기
'PD'	PD 제어기
'PDF'	PD 제어기 + 1차 derivative filter
'PID'	PID 제어기
'PIDF'	PID 제어기 + 1차 derivative filter

Controller type에서 'F'는 D 제어기에 1차 derivative filter를 적용하라는 의미로, 이는 미분할 때 bandwidth를 제한해주는 역할을 한다. 미분 제어요소는 미래의 값을 예측하여 시스템의 출력을 안정시켜주는 역할을 하지만, feedback sensor에 high frequency error가 포함되어 있을 경우 error를 증폭시키는 역할을 한다. (unstable) 따라서, 미분의 bandwidth를 제한하여 D 제어기가

안정적으로 작동하도록 하는 기법을 주로 사용한다.

함수 pidtune의 기본 제한조건은 phase margin 60 deg와 unstable pole 0개이다. 여기에 crossover frequency를 조절하여 response time을 바꿀 수 있다. 이와 같은 설정은 함수 pidtuneoptions에서 할 수 있으며, 자세한 사용법은 예제4.16을 참고하자.

#### 예제4.16. PID Tune

```
>> G = tf(1,[1 3 3 1]);
>> Options = pidtuneOptions( 'CrossoverFrequency', 1.2,
'PhaseMargin', 45 );
>> [C info] = pidtune( G, 'pid', Options )
Continuous-time PID controller in parallel form:
```

$$K_p + K_i \cdot \frac{1}{s} + K_d \cdot s$$

With  $K_p = 3.67$ ,  $K_i = 1.42$ ,  $K_d = 1.84$

info =

Stable: 1  
CrossoverFrequency: 1.2000  
PhaseMargin: 45

함수 pidtune에 transfer function과 controller type을 입력하면 적절한 계인을 구해준다. 이 때, 함수 pidtuneoption을 통해 계인 설정 조건을 변경할 수 있는데, 원하는 사항을 인자로 준 후, 출력 변수를 pidtune의 세 번째 인자로 입력한다.

함수 pidtune의 두 번째 출력 변수인 info는 함수를 통해 만든 controller를 적용했을 때의 feedback system의 성능을 나타내는 것이다.