MATLAB 실습 2

-시스템 구성 및 시간/주파수 응답 그래프-

박사과정 서종상

azuresky@snu.ac.kr

Tel:02-880-1942

301-113

Vehicle Dynamics & Control Laboratory

Set Transfer Function and State Space

1) tf

Transfer Function From = tf(Numerator, Denominator)

Numerator (분자), Denominator(분모) 를 통하여 Transfer Function 을 Define 한다.

2) ss

State Space From = ss(A, B, C, D)

State Space A,B,C,D 로부터 State Space Form 을 유도.

Ex)

Transfer Function 을 Define

$$T.F = \frac{1}{s(s+0.5)}$$

```
system_num=[1];
system_den=[1 0.5 0];
system=tf(system_num,system_den);
```

Transfer Function ↔ State Space

3) tf2ss

[A,B,C,D] = tf2ss(분자, 분모)

Numerator (분자), Denominator(분모) 를 통하여 **State Space Matrix A,B,C,D** 유도

4) ss2tf

(Numerator, Denominator)= ss2tf(A, B, C, D)

Numerator (분자), Denominator(분모) 를 통하여 **Transfer Function (Num,Den)** 유도

Transfer Function 활용법

System1=tf(num1,den1);

System2=tf(num2,den2);

System 간 곱셈, 덧셈 등의 계산이 가능.

```
system_num1=[1];
system_den1=[1 0.5 0];
system1=tf(system_num1,system_den1);

system_num2= [1 3.5];
system_den2= [1 5.8];
system2=tf(system_num2,system_den2);

OLS=system1*system2;
```

선형 시불변 모델의 시간 응답

함수	내 용
step	계단입력에 대한 선형 시간 불변 모델의 응답을 구한다.
Impulse	충격량 입력에 대한 선형 시간 불변 모델의 응답을 구한다.
Initial	상태방정식으로 표현된 모델의 초기 조건에 의한 과도응답을 구한다.
lsim	임의의 입력에 대한 선형 시간 불변 모델의 응답을 구한다.

time response란 선형 시간 불변 모델에 시간 t의 함수로 표현되는 입력이 가해질 경우에 응답을 시간 t에 대해 구하는 것을 말한다. 보통 time response를 구하는 목적은 delay time, rise time, peak time, max overshoot, settling time 등과 같은 과도 상태에서 제어 시스템의 성능을 나타내는 값들을 계산하기 위함이다.

Step Response

step : 단위 계단 입력에 대한 응답을 얻고자 할 경우 사용된다.

```
step(num,den)          % num, den으로 정의된 transfer function에 대한 step response를 그린다.
step(num,den,t)        % num, den으로 정의된 transfer function에 대한 step response를 시간 t에 대해 그린다.
step(A,B,C,D)          % 상태공간으로 정의된 시스템에 대한 step response를 그린다.
step(A,B,C,D,ui)       % 상태공간으로 정의된 시스템의 input ui에 대한 step response를 그린다.
step(A,B,C,D,ui,t)     % 상태공간으로 정의된 시스템에 input ui에 대한 step response를 시간 t에 대해 그린다.
```

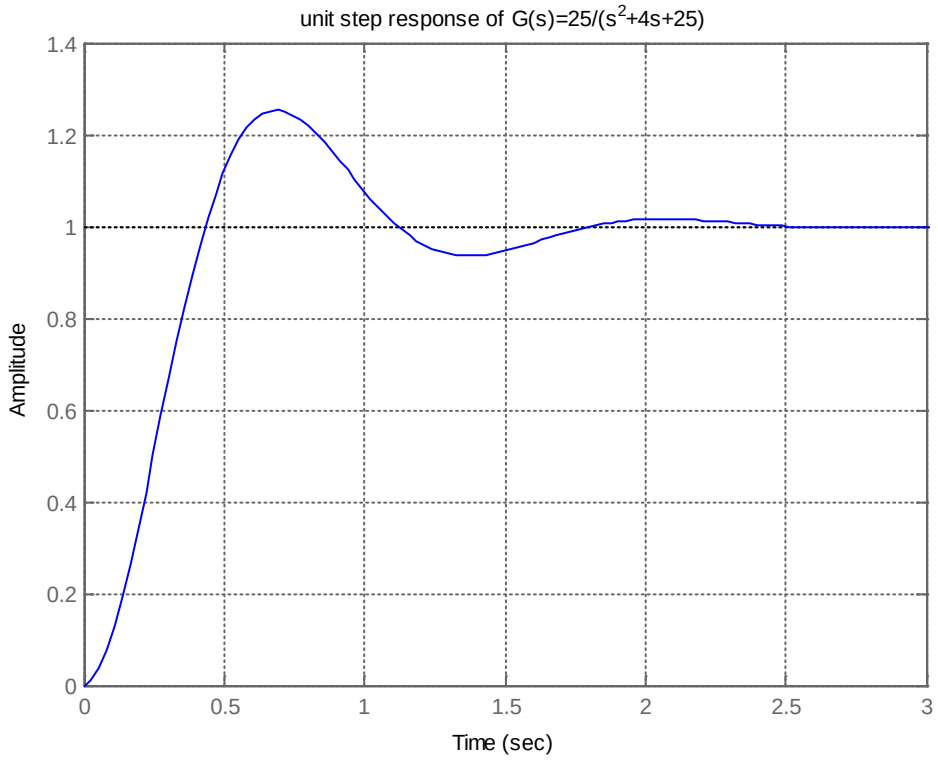
Step Response – Ex. 1

1) m 파일 editor window (명령어 입력)

```
clear; clc;
% t=0:0.01:10;
num=[0 0 25];
den=[1 4 25];
step(num,den);          % 단위계단응답
% step(num,den,t);      시간 t에 대한 단위 계단 응답
grid on;
title('unit step response of G(s)=25/(s^2+4s+25)')
```

Step Response – Ex. 1

2) 결과



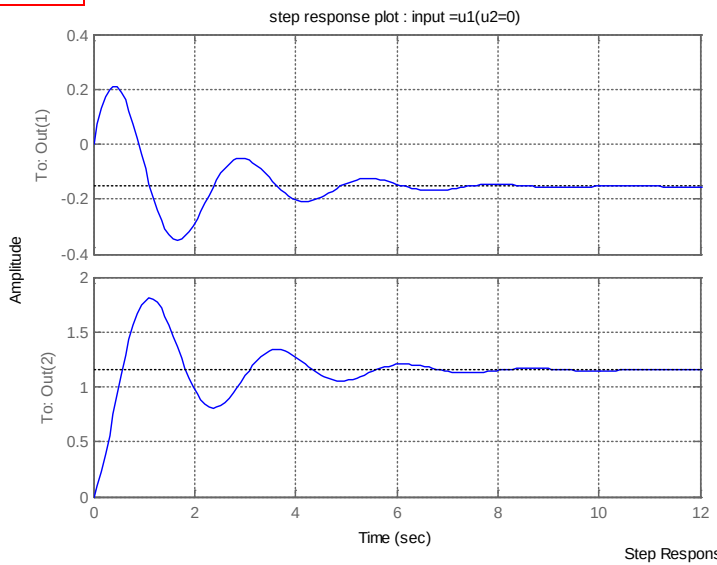
Step Response – Ex. 2

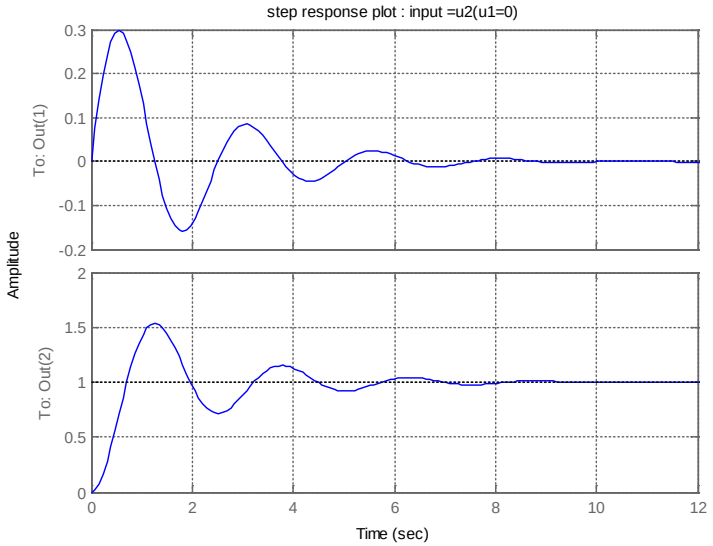
1) m 파일 editor window (명령어 입력)

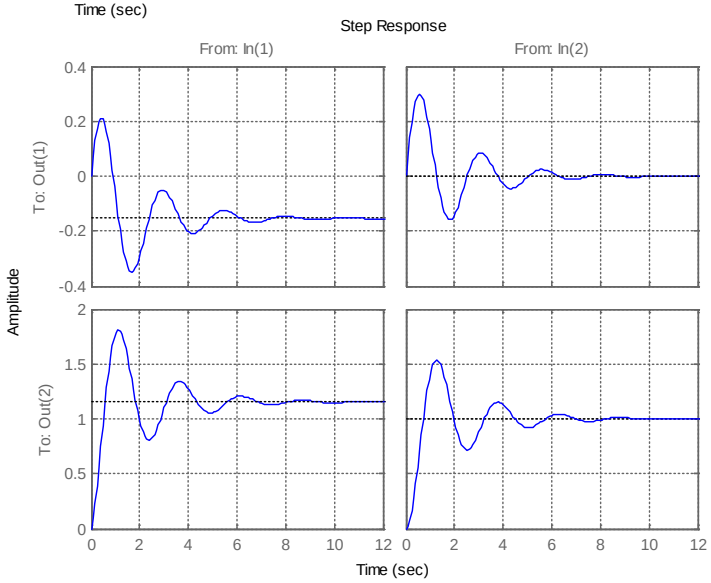
```
clear; clc;

A=[-1 -1; 6.5 0];
B=[1 1; 1 0];
C=[1 0; 0 1];
D=[0 0; 0 0];

figure(1)
step(A,B,C,D,1)                    % input u1에 대한step response
grid on;
title('step response plot : input =u1(u2=0)');

figure(2)
step(A,B,C,D,2)                    % input u2에 대한 step response
grid on;
title('step response plot : input =u2(u1=0)');

figure(3)
step(A,B,C,D)                      % input u1,u2에 대한 step response
grid on;
```

Step Response – Ex. 2

2) 결과







Impulse Response

impulse : 단위 충격량 입력에 대한 응답을 얻고자 할 경우 사용된다.

```
impulse(num,den)          % num, den으로 정의된 transfer function에 대한 impulse response를 그린다.
impulse(num,den,t)        % num, den으로 정의된 transfer function에 대한 impulse response를 시간 t에 대해 그린다.
impulse( A,B,C,D)         % 상태공간으로 정의된 시스템에 대한 impulse response를 그린다.
impulse(A,B,C,D,iu)       % 상태공간으로 정의된 시스템의 input iu에 대한 impulse response를 그린다.
impulse(A,B,C,D,iu,t)     % 상태공간으로 정의된 시스템에 input iu에 대한 impulse response를 시간 t에 대해 그린다.
```

Initial Response

initial : 상태방정식으로 표현된 모델의 초기 조건에 의한 과도응답을 얻고자 할 경우 사용된다.

step 이나 impulse는 외부 입력에 대한 응답을 구한다. 이 때 초기조건은 모두 0으로 되는데, 초기 조건에 대한 응답을 구하고 싶을 때는 initial 을 사용한다. 즉, initial은 외부 입력이 없다고 하고 단지 초기 조건에 대한 응답만을 구한다.

이때에는 모델이 상태방정식으로 표현된 모델이어야만 하는데 그 이유는 전달함수의 경우에는 이미 전달함수를 유도하기 위해 초기조건을 모두 0으로 한다는 가정을 사용했기 때문이다.

Initial Response – Ex. 1

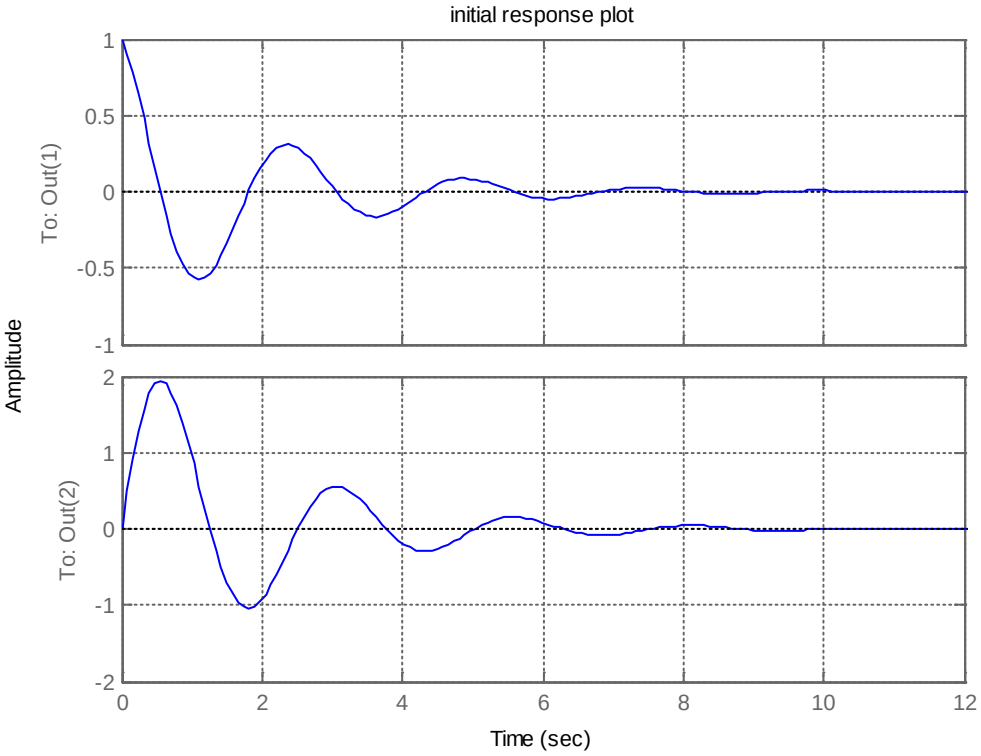
```
Initial( A,B,C,D,x0)          % 상태공간으로 정의된 시스템의 초기값 x0에 대한 response를 그린다.
Initial(A,B,C,D,x0, t)        % 상태공간으로 정의된 시스템에 input iu에 대한 impulse response를 시간 t에 대해 그린다.
```

1) m 파일 editor window (명령어 입력)

```
clear; clc;

A=[-1 -1; 6.5 0];
B=[1 1; 1 0];
C=[1 0; 0 1];
D=[0 0; 0 0];

x0=[1;0];                     % x1의 초기값 1, x2의 초기값 0

figure(1)
initial(A,B,C,D,x0);
grid on;
title('initial response plot');
```

Initial Response – Ex. 1

2) 결과



lsim Response

lsim : 임의의 입력에 대한 선형 시간 불변 모델의 응답을 구한다

step 이나 **impulse**는 미리 정해진 외부 입력에 대한 응답을 구한다.

하지만 **lsim**은 임의의 입력에 대한 응답을 구할 수 있다.

```
lsim(num,den,u,t)        % transfer function으로 정의된 시스템의 input u에 대한 response를 시간 t에 대해그린다.
lsim(num,den,u,t,x0)     % transfer function으로 정의된 시스템의 초기값 x0, input u에 대한 response를 시간 t에 대해그린다.
lsim(A,B,C,D,u,t)        % 상태공간으로 정의된 시스템의 input u에 대한 response를 시간 t에 대해그린다.
lsim(A,B,C,D,u,t,x0)     % 상태공간으로 정의된 시스템의 초기값 x0, input u에 대한 response를 시간 t에 대해그린다.
```

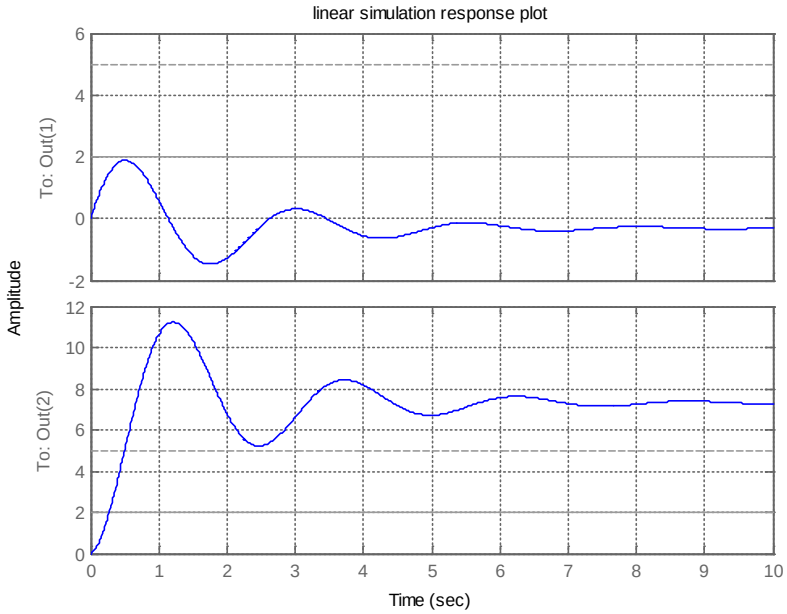
lsim Response – Ex. 1

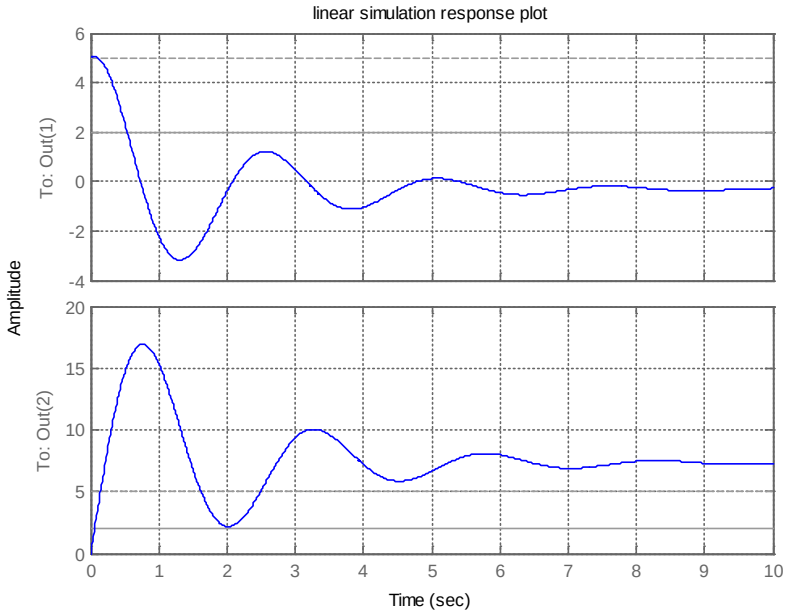
1) m 파일 editor window (명령어 입력)

```
clear; clc;

t=0:0.01:10;
A=[-1 -1; 6.5 0];
B=[1 1; 1 0];
C=[1 0; 0 1];
D=[0 0; 0 0];
u=[0*t+2; 0*t+5];              % input u1=2*1(t), u2=5*1(t)
x0=[5;0];                      % 초기값 x1=5, x2=0

figure(1)
lsim (A,B,C,D,u,t)             % 시간 t에 동안 상태 방정식으로 정의된 시스템의 input u에 대한 response
grid on;
title('linear simulation response plot');

figure(2)
lsim (A,B,C,D,u,t,x0)          % 시간 t에 동안 상태 방정식으로 정의된 시스템의 input u와 초기값 x0에 대한 response
grid on;
grid on;
title('linear simulation response plot');
```

lsim Response – Ex. 1

2) 결과





Plotting bode diagrams with MATLAB

1) From Transfer Function

```
[Magnitude, Phase, Frequency values]=bode(Numerator, Denominator, W);
```

Numerator (분자), Denominator(분모) 를 통하여 **Bode**의 **Magnitude**와 **Phase**를 구한다.

2) From State Space

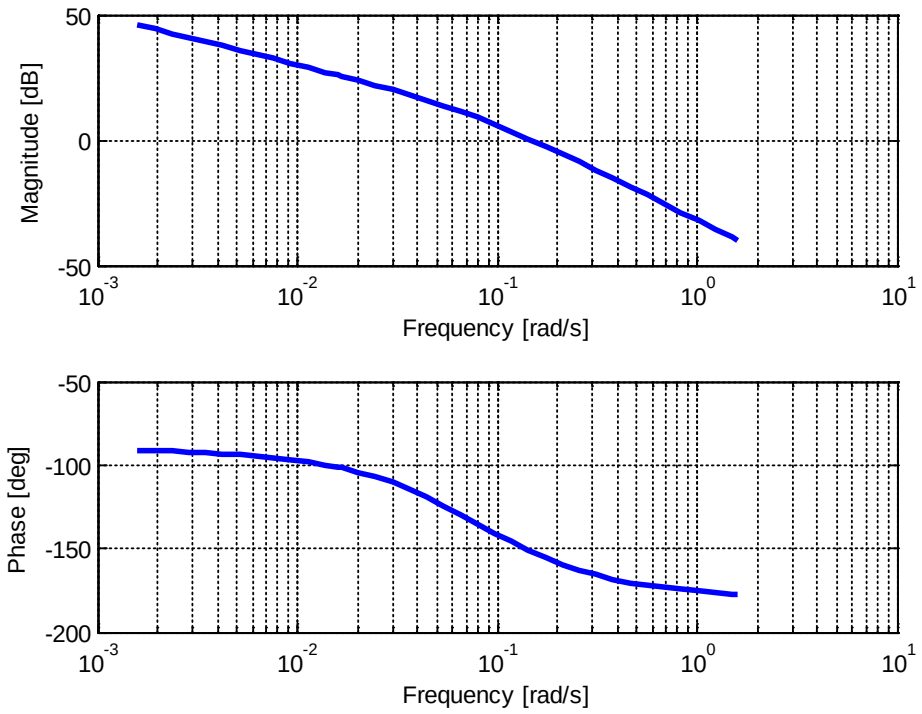
```
[Magnitude, Phase, Frequency values]=bode(A, B, C, D, W);
```

State Space의 각 행렬 **A, B, C, D** 를 통하여 **Bode**의 **Magnitude**와 **Phase**를 구한다.

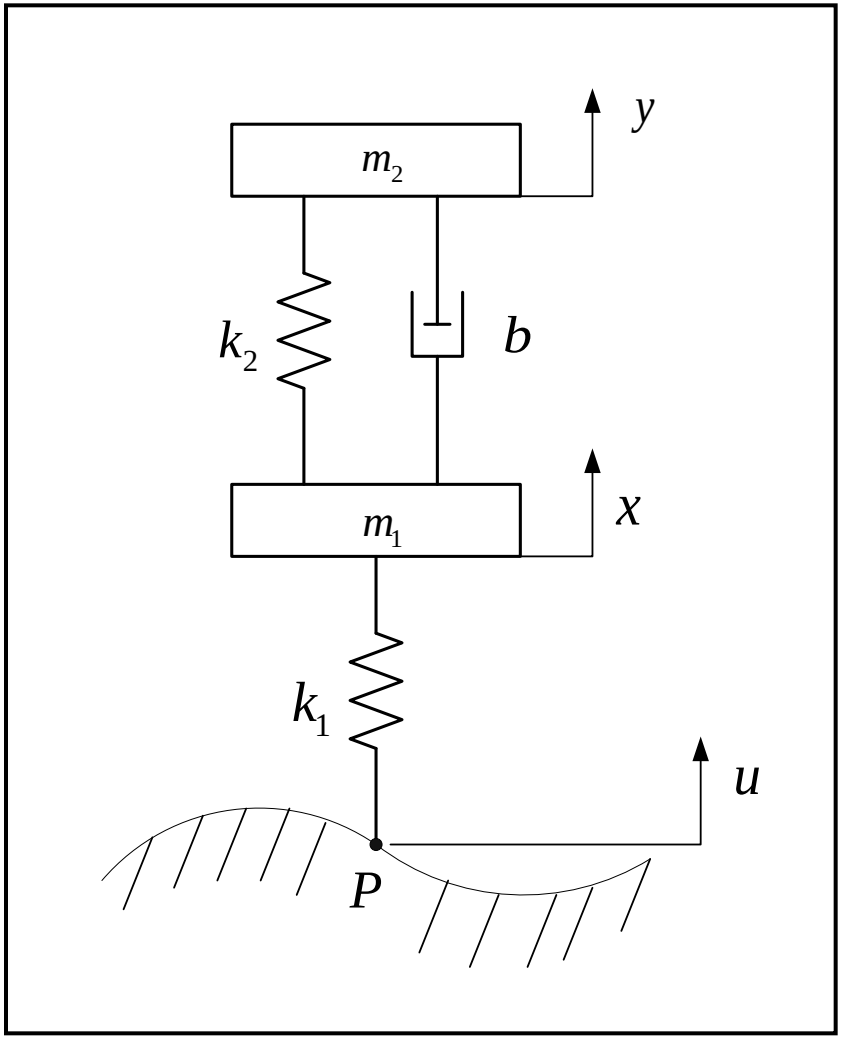
Bode Plot Example

$$T.F = \frac{1}{s(s+0.5)}$$

```
%% System Define
system_num=[1];
system_den=[1 0.5 0];

%% Generates logarithmic points
w=logspace(0,3,100);

%% Bode
[mag,phase,w]=bode(system_num,system_den);
% bode(system_num,system_den);
%% Magnitude
magDB = 20*log10(mag);

%% Plot
figure(1)
subplot(211)
semilogx(w/(2*pi),magDB,'b','linewidth',2.3);hold on; grid on;
xlabel('Frequency [rad/s]')
ylabel('Magnitude [dB]')
subplot(212)
semilogx(w/(2*pi),phase,'b','linewidth',2.3);hold on; grid on;
xlabel('Frequency [rad/s]')
ylabel('Phase [deg]')
```



Suspension Example



▪ Design Considerations

1. Ride Quality

→ *Sprung mass acceleration* : $\ddot{y}$

2. Rattle space

→ *Suspension Deflection* : $y - x$

3. Tire Force Vibration

→ *Tire Deflection* : $x - u$

▪ Suspension Design Parameters

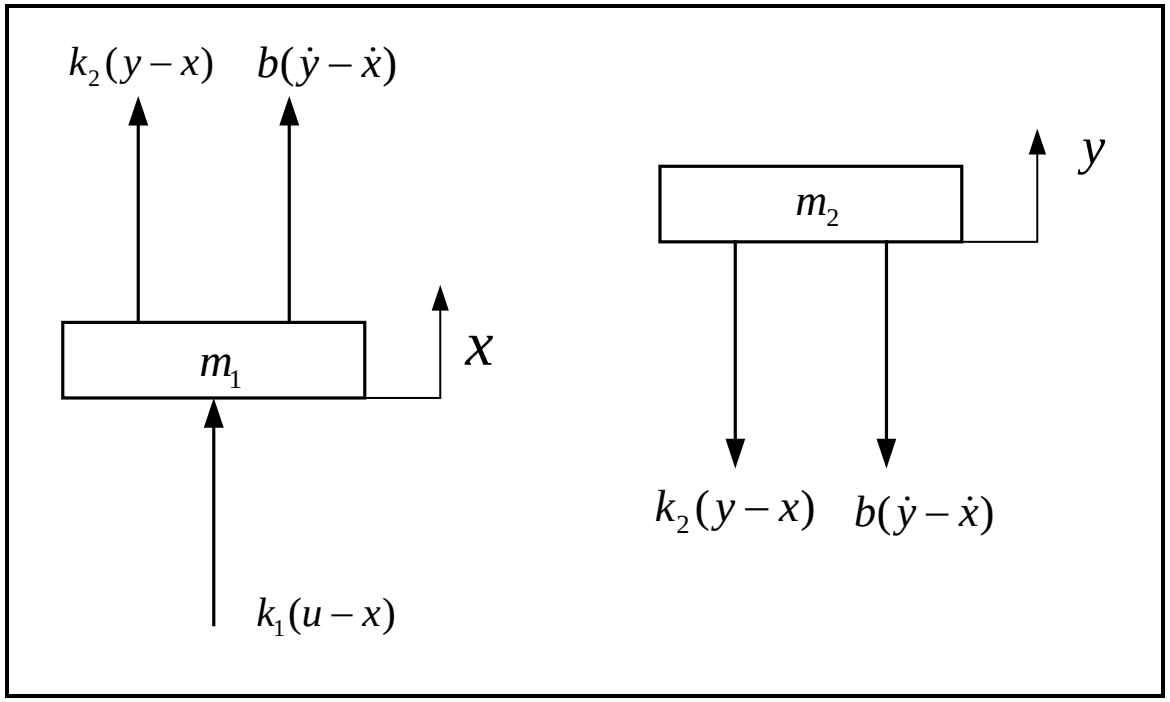
→ *Spring Stiffness* : k_2

→ *Damping Ratio* : b

→ *Tire Stiffness* : k_1

Dynamic Equations

▪ Free Body Diagram



▪ Dynamic Equations

$$m_1\ddot{x} = k_2(y-x) + b(\dot{y}-\dot{x}) + k_1(u-x)$$

$$m_2\ddot{y} = -k_2(y-x) - b(\dot{y}-\dot{x})$$

Laplace Transform

- Laplace Transform

$$[m_1 s^2 + bs + (k_1 + k_2)]X(s) = (bs + k_2)Y(s) + k_1 U(s)$$

$$[m_2 s^2 + bs + k_2]Y(s) = (bs + k_2)X(s)$$

- Displacement of Mass

$$\frac{Y(s)}{U(s)} = \frac{k_1(bs + k_2)}{m_1 m_2 s^4 + (m_1 + m_2)bs^3 + [(k_2 m_1 + (k_1 + k_2)m_2]s^2 + k_1 bs + k_1 k_2}$$

$$\frac{X(s)}{U(s)} = \frac{k_1(m_2 s^2 + bs + k_2)}{m_1 m_2 s^4 + (m_1 + m_2)bs^3 + [(k_2 m_1 + (k_1 + k_2)m_2]s^2 + k_1 bs + k_1 k_2}$$

- Design Considerations

$$G_1(s) = \frac{s^2 Y(s)}{U(s)} = \frac{s^2 k_1(bs + k_2)}{m_1 m_2 s^4 + (m_1 + m_2)bs^3 + [(k_2 m_1 + (k_1 + k_2)m_2]s^2 + k_1 bs + k_1 k_2}$$ → *Sprung mass acceleration* : $\ddot{y}$

$$G_2(s) = \frac{Y(s) - X(s)}{U(s)} = \frac{-k_1 m_2 s^2}{m_1 m_2 s^4 + (m_1 + m_2)bs^3 + [(k_2 m_1 + (k_1 + k_2)m_2]s^2 + k_1 bs + k_1 k_2}$$ → *Suspension Deflection* : $y - x$

$$G_3(s) = \frac{X(s) - U(s)}{U(s)} = \frac{-m_1 m_2 s^4 - (m_1 + m_2)bs^3 - k_2(m_1 + m_2)s^2}{m_1 m_2 s^4 + (m_1 + m_2)bs^3 + [(k_2 m_1 + (k_1 + k_2)m_2]s^2 + k_1 bs + k_1 k_2}$$ → *Tire Deflection* : $x - u$

.

State Equation

- General Form of State Equation

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

- The State variables $(x = z_u, \quad y = z_s)$

$$x_1 = z_s - z_u \quad : Suspension\ Deflection$$
$$x_2 = \dot{z}_s \quad : absolute\ velocity\ of\ sprung\ mass$$
$$x_3 = z_u - u \quad : Tire\ Deflection$$
$$x_4 = \dot{z}_u \quad : absolute\ velocity\ of\ unsprung\ mass$$

- Dynamic Equations

$$m_1\ddot{z}_u = k_2(z_s - z_u) + b(\dot{z}_s - \dot{z}_u) + k_1(u - z_u)$$
$$m_2\ddot{z}_s = -k_2(z_s - z_u) - b(\dot{z}_s - \dot{z}_u)$$

- 1st order State equations

$$\dot{x}_1 = \dot{z}_s - \dot{z}_u = x_2 - x_4$$
$$\dot{x}_2 = -\frac{k_2}{m_2}(z_s - z_u) - \frac{b}{m_2}(\dot{z}_s - \dot{z}_u) = -\frac{k_2}{m_2}x_1 - \frac{b}{m_2}x_2 + \frac{b}{m_2}x_4$$
$$\dot{x}_3 = \dot{z}_u - \dot{u} = x_4 - \dot{u}$$
$$\dot{x}_4 = \frac{k_2}{m_1}(z_s - z_u) + \frac{b}{m_1}(\dot{z}_s - \dot{z}_u) - \frac{k_1}{m_1}(z_u - u) = \frac{k_2}{m_1}x_1 + \frac{b}{m_1}x_2 - \frac{k_1}{m_1}x_3 - \frac{b}{m_1}x_4$$

System Matrix

- Matrix Form of State equations (system matrix)

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & -1 \\ -\frac{k_2}{m_2} & -\frac{b}{m_2} & 0 & \frac{b}{m_2} \\ 0 & 0 & 0 & 1 \\ \frac{k_2}{m_1} & \frac{b}{m_1} & -\frac{k_1}{m_1} & -\frac{b}{m_1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} \dot{u}$$

- Matrix Form of State equations (output matrix)

$$y_1 = \ddot{x}_2 = -\frac{k_2}{m_2} x_1 - \frac{b}{m_2} x_2 + \frac{b}{m_2} x_4 \quad : \textit{Sprung mass acceleration}$$

$$y_2 = z_s - z_u = x_1 \quad : \textit{Suspension Deflection}$$

$$y_3 = z_u - u = x_3 \quad : \textit{Tire Deflection}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} -\frac{k_2}{m_2} & -\frac{b}{m_2} & 0 & \frac{b}{m_2} \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

.

MATLAB Simulation using Laplace Transform

- Suspension Parameters

```
m1=55;          % unsprung mass
m2=400;         % sprung mass
b=1000;         % damping ratio
k1=180000;      % stiffness of Tire
k2=18000;       % stiffness of sping
```

- Displacement of Mass (Transfer function)

```
% Transfer Function of sprung mass displacement
num_s=[k1*b k1*k2];
den=[m1*m2 (m1+ m2)*b [k2*(m1+ m2)+ k1*m2] k1*b k1*k2];

% Transfer Function of sprung mass displacement
num_u=[k1*m2 k1*b k1*k2];
```

- Design Considerations (Transfer function)

```
% Transfer Function of sprung mass acceleration
num_1=[k1*b k1*k2 0 0];

% Transfer Function of suspension deflection
num_2=[-k1*m2 0 0];

% Transfer Function of tire deflection
num_3=[-m1*m2 -(m1+m2)*b -k2*(m1+m2)];

printsys(num_1,den)    % print system transfer function
```

MATLAB Simulation using Laplace Transform

▪ Making Input functions

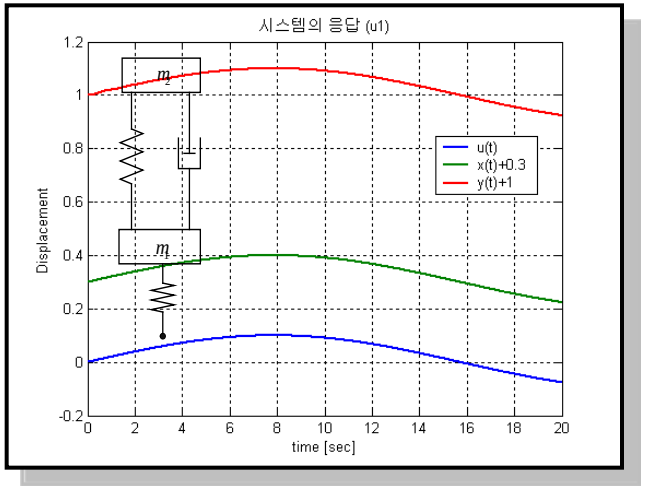
```
t=0:0.01:20;    % 시간을 정의

%sine 함수
u1=0.1*sin(0.2*t);

% sine 함수를 이용한 자갈길
u2=0.02*sin(4*t)+ 0.02*abs(sin(4*t));  % abs() : 절대값 함수

% 과속방지턱
u3=0.05*sin(2*pi/20*(t-5))+ abs(0.05*sin(2*pi/20*(t-5)));
```

▪ Simulation & Plot result

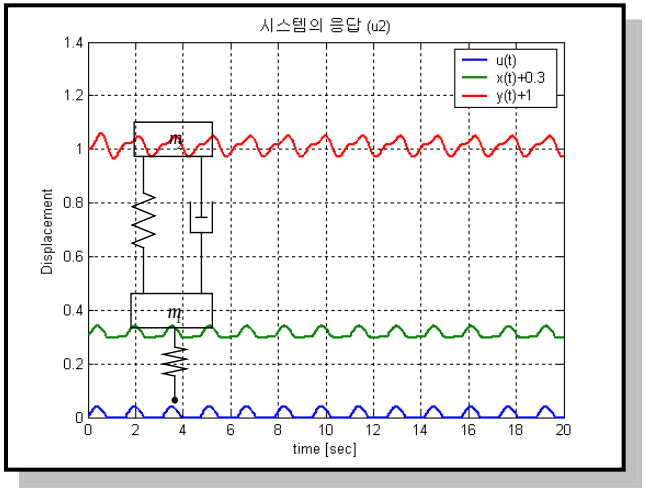
```
% Linear simulation
y=lsim(num_s,den,u1,t);
x=lsim(num_u,den,u1,t);

plot(t,u1,t,x+ 0.3,t,y+ 1);
grid on;

title('시스템의 응답');
xlabel('time [sec]');
ylabel('Displacement');
legend('u(t)','x(t)+ 0.3','y(t)+ 1');
```

Simulation Results

▪ Simulation result(1)

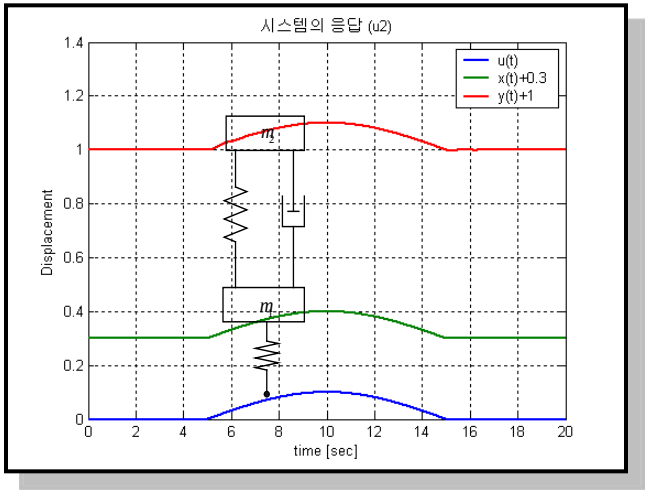


▪ Simulation result(2) : 자갈길



Deflection of Elements

▪ Simulation result(3) : 과속방지턱



MATLAB Simulation using State Equation

▪ Suspension Parameters

```
m1=55;          % unsprung mass
m2=400;         % sprung mass
b=1000;         % damping ratio
k1=180000;      % stiffness of Tire
k2=18000;       % stiffness of sping
```

▪ State Equation

```
% Define State eqations
A=[   0     1      0       -1;
   -k2/m2 -b/m2    0      b/m2;
      0     0      0        1
   k2/m1   b/m1  -k1/m1  -b/m1];

B=[0; 0; -1; 0];

C=[-k2/m2 -b/m2    0      b/m2;
     1     0      0        0;
     0     0      1        0];
D=[0; 0; 0];
```

▪ Making Input functions

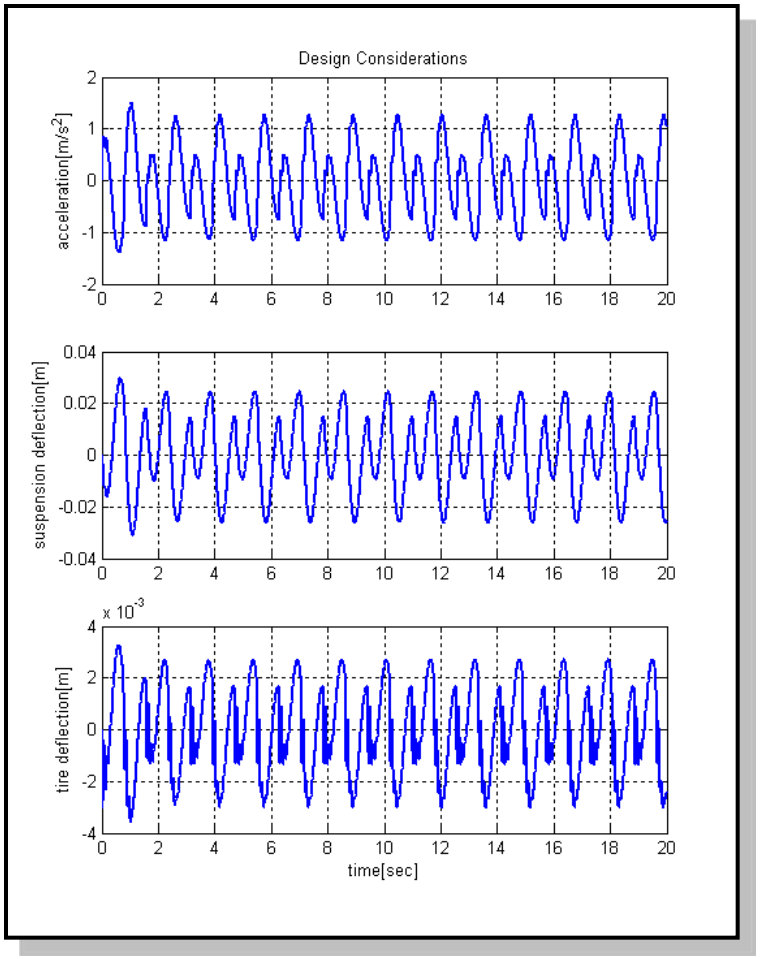
```
t=0:0.01:20;    % 시간을 정의

%sine 함수
u1=0.1*sin(0.2*t);

% sine 함수를 이용한 자갈길
u2=0.02*sin(4*t)+ 0.02*abs(sin(4*t));  % abs() : 절대값 함수

% 과속방지턱
u3=0.05*sin(2*pi/20*(t-5))+ abs(0.05*sin(2*pi/20*(t-5)));

% 입력의 미분함수 구하기 : State equation의 입력
u3_d=diff(u3)./0.01;
u3_d=[u3_d u3_d(length(u3_d))];
```

MATLAB Simulation using State Equation

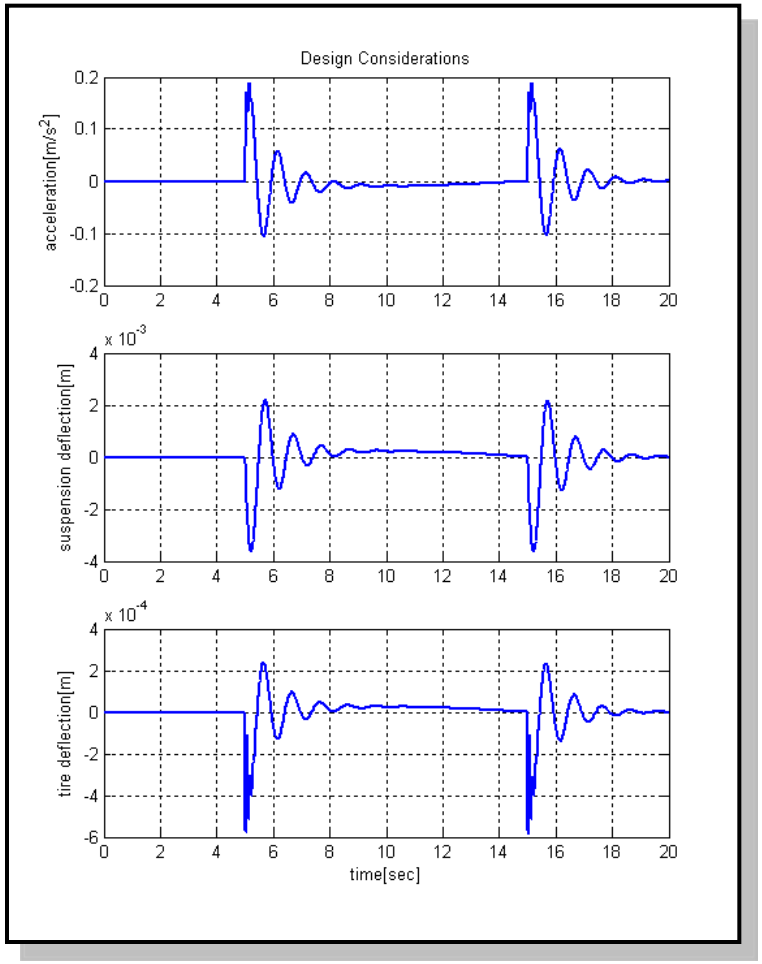
- Simulation & plot result

```
% Linear simulation
y=lsim(A,B,C,D,u3_d,t);  % n-row, 3-column

figure
subplot(311)
plot(t,y(:,1),'linewidth',2);  grid on;
ylabel('acceleration[m/s^2]'); title('Design Considerations');
subplot(312)
plot(t,y(:,2),'linewidth',2);  grid on;
ylabel('suspension deflection[m]');
subplot(313)
plot(t,y(:,3),'linewidth',2);  grid on;
ylabel('tire deflection[m]');  xlabel('time[sec]');
```

Simulation Results



자갈길



과속방지턱

Parametric Study using for loop

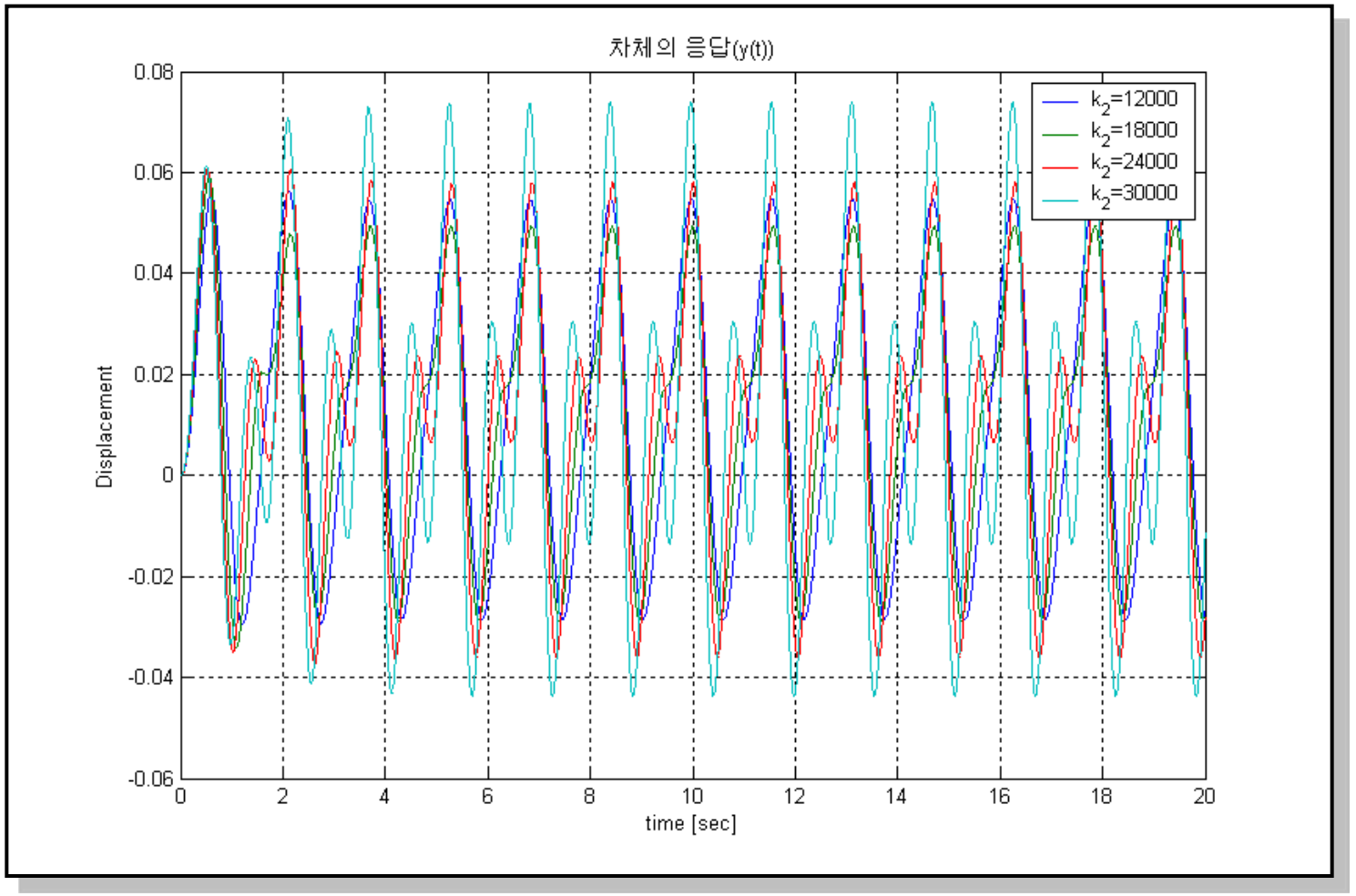
▪ Parameters & Input function

```
m1=55;              % unsprung mass
m2=400;             % sprung mass
b=1000;             % damping ratio
k1=180000;          % stiffness of Tire

t=0:0.01:20;    % 시간을 정의

% sine 함수를 이용한 자갈길
u1=0.02*sin(4*t)+0.02*abs(sin(4*t));  % abs() : 절대값 함수
```

▪ Parametric study

```
k2=[12000 18000 24000 30000];   % 변화시킬 parameter를 배열로 정의

for i=1:1:4       % index의 설정, 시작:간격:끝
    num1=[k1*b k1*k2(i)];    % 전달함수를 정의
    den1=[m1*m2 (m1+m2)*b [m1*k2(i)+(k1+k2(i))*m2] k1*b k1*k2(i)];
    num2=[k1*m2 k1*b k1*k2(i)];
    den2=[m1*m2 (m1+m2)*b [m1*k2(i)+(k1+k2(i))*m2] k1*b k1*k2(i)];

    y(:,i)=lsim(num1,den1,u1,t);     % index에 부여하여 matrix로 저장
    x(:,i)=lsim(num2,den2,u1,t);
end

figure
plot(t,y);          % matrix의 출력
grid on;
title('차체의 응답(y(t)');
xlabel('time [sec]');
ylabel('Displacement'); legend('k_2=12000','k_2=18000','k_2=24000','k_2=30000');
```

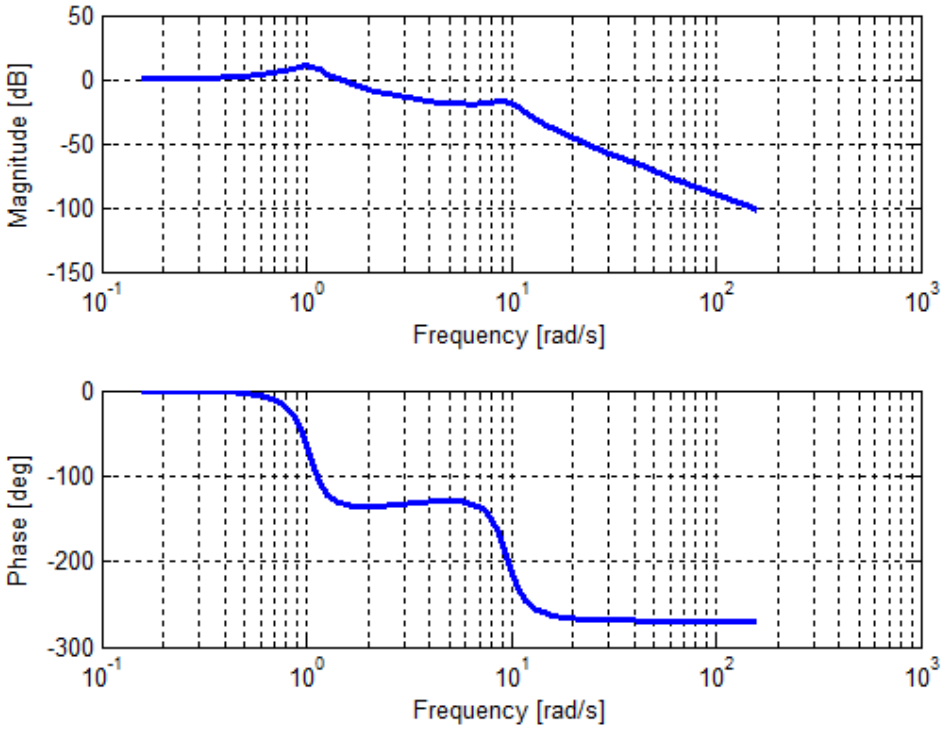
Displacement of Sprungmass



Bode Plot – Transfer Function

- Transfer Function

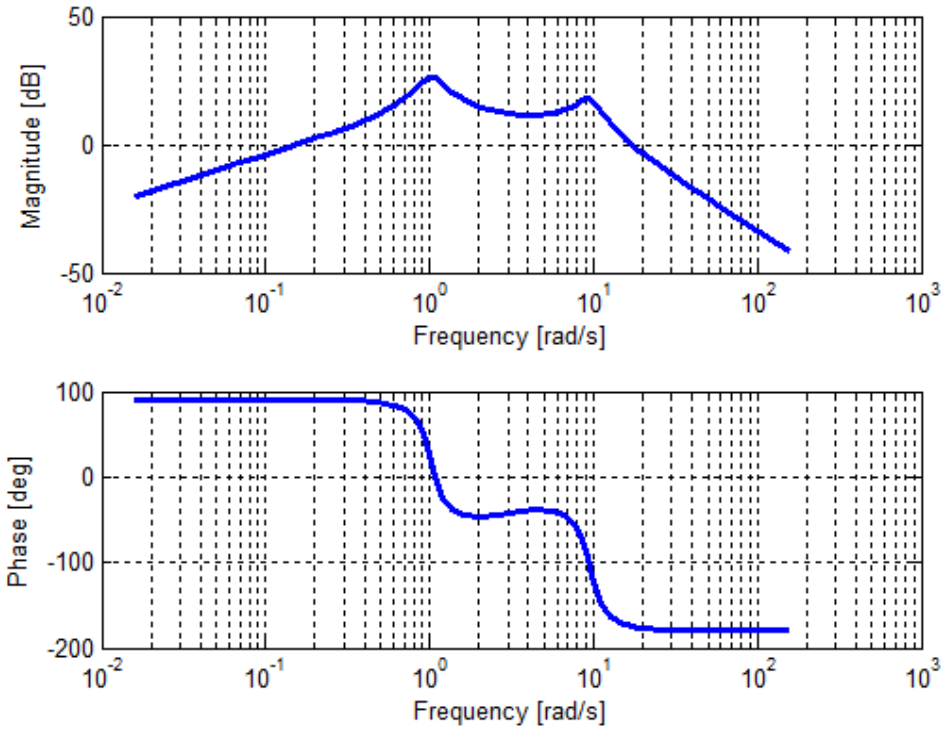
```
% Generates logarithmic points
w=logspace(0,3,100);

% Bode
% [mag,phase,w]=bode(num_s,den,w);
[mag,phase,w]=bode(num_u,den,w);

% Magnitude
magDB = 20*log10(mag);

% Plot
figure(1)
subplot(211)
semilogx(w/(2*pi),magDB,'b','linewidth',2.3);hold on; grid on;
xlabel('Frequency [rad/s]')
ylabel('Magnitude [dB]')
subplot(212)
semilogx(w/(2*pi),phase,'b','linewidth',2.3);hold on; grid on;
xlabel('Frequency [rad/s]')
ylabel('Phase [deg]')
```



Bode Plot – State Space

- State Space

```
% Generates logarithmic points
w=logspace(0,3,100);

% Bode
% [mag,phase,w]=bode(num_s,den);
[mag,phase,w]=bode(num_u,den);
% bode(system_num,system_den);

% Magnitude
magDB = 20*log10(mag);

% Plot
figure(1)
subplot(211)
semilogx(w/(2*pi),magDB,'b','linewidth',2.3);hold on; grid on;
xlabel('Frequency [rad/s]')
ylabel('Magnitude [dB]')
subplot(212)
semilogx(w/(2*pi),phase,'b','linewidth',2.3);hold on; grid on;
xlabel('Frequency [rad/s]')
ylabel('Phase [deg]')
```



Save and Load the Data

1) Save

```
save 파일명 (Workspace 에 있는 변수 명) -ascii
```

Workspace 에 있는 변수를 '**ascii code**' 로 된 '파일명'으로 저장한다.

2) Load

```
load 파일명 -ascii
```

'**ascii code**' 로 된 '파일명'으로 된 파일을 같은 이름의 **Workspace** 변수로 저장한다.
***** 단 이때 문자열이 포함되어서는 안 된다.

```
save result.txt y -ascii  % y 변수를 result.txt 란 파일을 만들어서 저장
clear all;
clc;
load result.txt  % result.txt 를 불러옴.
```

Help function

3) Help

help 명령어

Workspace 에 있는 변수를 '**ascii code**' 로 된 '파일명'으로 저장한다.

```
save result.txt y -ascii  % y 변수를 result.txt 란 파일을 만들어서 저장
clear all;
clc;
load result.txt  % result.txt 를 불러옴.
```