CHAPTER

04

데이터를 한눈에 보는 그래프

CONTENTS

LESSON
01 2차원 그래프 그리기

그래프를 이용하여 결과를 시각적으로 표현하면 실험이나 수식을 이용해서 얻은 복잡한 수치와 자료를 훨씬 명쾌하게 해석할 수 있다. MATLAB에서 선형 $x-y$ 그래프(2차원 그래프)를 그리기 위해 사용하는 명령어를 알아보자.

Keyword | plot | hold | semilogx | semilogy | 2차원 그래프의 모양과 종류 |

plot

plot 명령어는 x 축(가로축)에 지정되는 입력 자료 벡터 **x**에 대응하여 y 축(세로축)에 지정되는 출력 자료 벡터 **y**의 그래프를 생성한다. 여러 개의 입력 값(x1, x2, …)에 대한 출력 값(y1, y2, …)의 그래프를 생성하는 plot 명령어의 기본 형태는 다음과 같다.

```
plot(x1,y1,x2,y2,...)
```

입력 값이 지정된 벡터 **x**와 출력 값이 지정된 벡터 **y**를 이용하여 그래프를 그릴 때 plot 명령어의 기본 형태는 다음과 같다.

```
plot(x,y,'s')
```

- **x** : 입력 값이 지정된 벡터
- **y** : 출력 값이 지정된 벡터
- **'s'** : 자료 기호, 선 형태, 색상 지정(생략 가능하며 순서는 상관없다.)

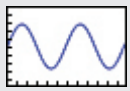


표 4-1 다양한 자료 기호 및 선과 색상

자료 기호		선 형태		색상	
동그란 점 (.)	.	실선	–	검은색	k
별표 (*)	*	일점쇄선	—·	파란색	b
x 표 (x)	x	쇄선 · 점선 혼합	–.	청록색	c
원형 (o)	o	점선	:	초록색	g
플러스 부호 (+)	+			자홍색	m
정사각형 (□)	s			빨간색	r
다이아몬드 (◇)	d			흰색	w
별 모양 (★)	p			노란색	y

hold

두 개 이상의 선을 그림 창 하나에 나타내는 경우 그래프 하나를 먼저 그린 다음에 hold 명령어를 사용하면 그려진 그림을 확인한 후 그 다음에 그릴 그래프를 추가할 수 있다. hold 명령어를 번갈아 입력하면 hold 명령어 기능의 켜기/끄기를 반복할 수 있다.

예제 4-1

1부터 7까지 증가하는 입력 자료에 대한 세 개의 출력 자료 **y1**, **y2**, **y3**가 있다. 이때 입력변수 **x**에 대한 출력 변수 **y1**과 **y2**를 그래프로 나타내라. 또한 hold 명령어를 이용하여 입력변수 **x**에 대한 출력변수 **y3**을 그래프로 나타내라.

$$\mathbf{y1} = [0\ \ 0.48\ \ 0.84\ \ 1\ \ 0.91\ \ 0.6\ \ 0.14]$$
$$\mathbf{y2} = [0.2\ \ 0.5\ \ 1\ \ 0.84\ \ 0.6\ \ 0.32\ \ 0.09]$$
$$\mathbf{y3} = [0.5\ \ 0.24\ \ 0.31\ \ 0.7\ \ 1\ \ 0.8\ \ 0.44]$$

풀이

```
Command Window
>> x=[1 2 3 4 5 6 7];
>> y1=[0 .48 .84 1 .91 .6 .14];
>> y2=[.2 .5 1 .84 .6 .32 .09];
>> y3=[.5 .24 .31 .7 1 .8 .44];
>> plot(x,y1,'-*g',x,y2,'-.ob') ❶
>> hold ❷
Current plot held
>> plot(x,y3,':r') ❸
>> hold ❹
Current plot released
```

그림 4-1 2차원 그래프 그리기 코딩 결과

❶ 한 번의 입력으로 두 개의 그래프를 그려 서로 비교할 수 있다. 이때 **x1**, **x2**의 값이 **x**로 동일하기 때문에 **y1**과 **y2**의 입력 순서는 바뀌어도 된다. 두 선을 구분하기 위해 **y1**은 초록색 실선으로 나타내고, 각 자료점에 별표를 표시한다. **y2**은 파란색 쇄선 · 점선 혼합으로 나타내고, 각 자료점에 원형을 넣어 표시한다.

❷ hold 명령어가 활성화되어 현재 그림 창의 그래프를 계속 나타낸다. 따라서 현재 그림 창에 다른 그래프를 추가하여 나타낼 수 있다.

❸ 벡터 **x**, **y3**을 이용하여 그래프를 그린다.

❹ hold 명령어가 비활성화되어 더 이상 현재 그림 창에 다른 그래프를 추가할 수 없다.

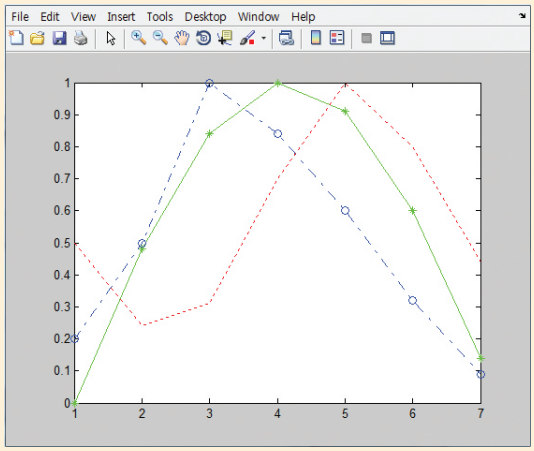


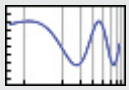
그림 4-2 2차원 그래프 그리기 실행 결과

semilogx

semilogx 명령어는 x 축만을 로그 눈금으로 그린다. semilogx 명령어의 기본 형태는 다음과 같다.

semilogx(x, y)

- **x** : 입력 값이 지정된 벡터
- **y** : 출력 값이 지정된 벡터

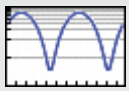


semilogy

semilogy 명령어는 y 축만을 로그 눈금으로 그린다. semilogy 명령어의 기본 형태는 다음과 같다.

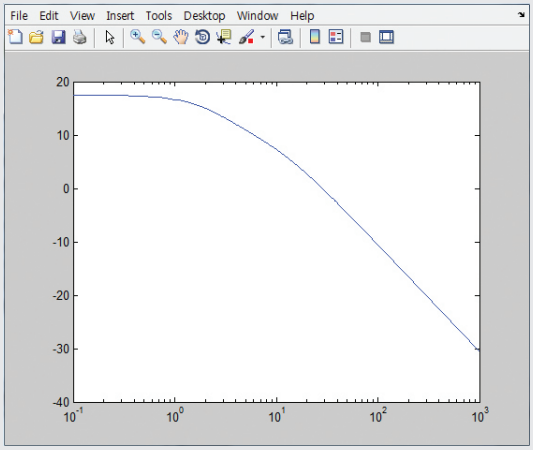
semilogy(x, y)

- **x** : 입력 값이 지정된 벡터
- **y** : 출력 값이 지정된 벡터

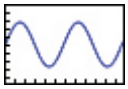
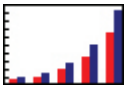
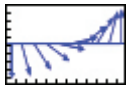
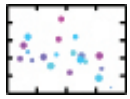
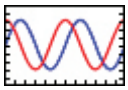
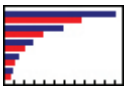
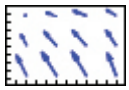
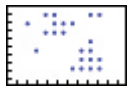
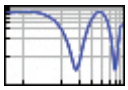
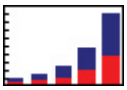
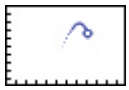
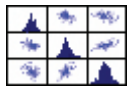
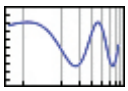
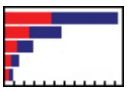
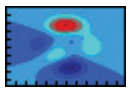
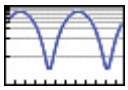
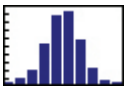
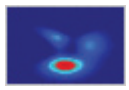
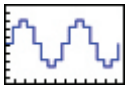
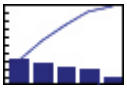
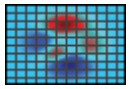
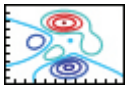
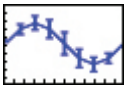
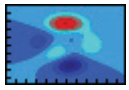
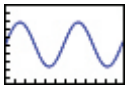
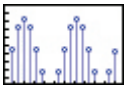
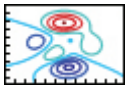


예를 들어 제어공학에서는 x 축이 로그 눈금인 보드선도를 그려서 시스템을 분석한다. MATLAB을 이용하여 보드선도를 그릴 때는 x 축을 로그 눈금으로 설정하기 위해 semilogx 명령어를 사용한다. 다음은 CHAPTER 09의 [예제 9-19]에서 다루는 BodeDiag.m의 일부와 그 결과이다.

```
num=30*[0 1 5];
den=conv([1 2], [1 10]);
omega=0.1:0.1:1e3;
G2=freqs(num,den,omega);
mag=abs(G2);
phase=angle(G2);
semilogx(omega, 20*log10(mag))
```



2차원 그래프의 모양과 종류

선 그래프	막대 그래프	면 그래프	방향 그래프	극형식 그래프	선점도 그래프
plot(X,Y) 	bar(x,Y,'grouped') 	area(X,Y) 	feather(U,V) 	polar(theta,rho) 	scatter(X,Y) 
plotyy(x1,y1,x1,y2) 	barh(x,Y,'grouped') 	pie(x,explode) 	quiver(x,y,u,v) 	rose(theta,x) 	spy(S) 
loglog(x,y) 	bar(x,Y,'stacked') 	fill(X,Y,C) 	comet(x,y) 	compass(U,V) 	plotmatrix(X,Y) 
semilogx(x,y) 	barh(x,Y,'stacked') 	contourf(X,Y,Z) 		ezpolar(fun) 	
semilogy(x,y) 	hist(Y,x) 	image(x,y,C) 			
stairs(x,y) 	pareto(Y,x) 	pcolor(X,Y,C) 			
contour(x,y,z) 	errorbar(X,Y,E) 	ezcontourf(fun) 			
ezplot(fun) 	stem(X,Y) 				
ezcontour(fun) 					

LESSON
02 그래프 속성 지정하기

그래프의 정보를 명확히 전달하는 데 도움이 되는 그래프 속성 지정 명령어를 살펴보자.

Keyword | title | xlabel, ylabel | axis | grid | gtext | text | legend | subplot | figure | 그래프 저장하고 불러오기 |

title

그래프 제목을 지정하는 title 명령어의 기본 형태는 다음과 같다.

title('text')

- **'text'** : 그래프 제목

xlabel, ylabel

x 축(가로축)과 y 축(세로축)의 축 이름을 나타내는 xlabel, ylabel 명령어의 기본 형태는 다음과 같다.

xlabel('text')
ylabel('text')

- **'text'** : 축 이름

axis

그림 창에서 각 축의 크기를 조절하는 axis 명령어의 기본 형태는 다음과 같다.

axis([xmin xmax ymin ymax])

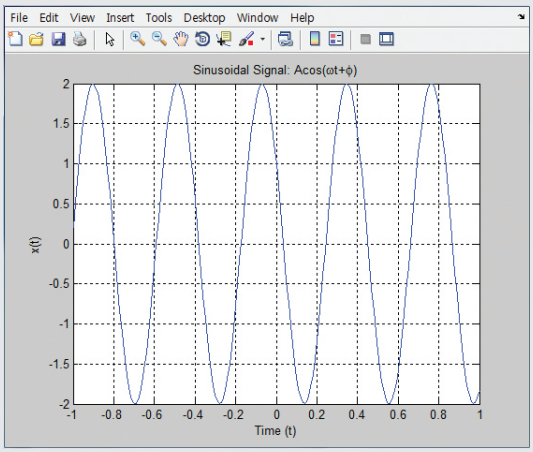
- **xmin, xmax** : x 축의 시작점에 해당하는 최소 크기와 끝점에 해당하는 최대 크기
- **ymin, ymax** : y 축의 시작점에 해당하는 최소 크기와 끝점에 해당하는 최대 크기

grid

그래프에 모눈을 생성하기 위해 grid 명령어를 사용한다. grid 명령어를 번갈아 입력하여 켜기/끄기를 반복하거나, grid on 명령어를 입력하여 모눈을 생성하고 grid off 명령어를 입력하여 모눈을 지울 수 있다.

예를 들어 신호에서는 실제 파형을 그래프로 그려서 확인하는 것이 매우 중요하다. 이때 MATLAB에서 plot, grid, xlabel, ylabel, title 명령어를 사용하여 실제 파형을 그래프로 그리고 속성을 추가할 수 있다. 다음은 CHAPTER 10의[예제 10-1]에서 다루는 ConSinus.m의 일부와 그 결과이다.

```
t=-1:0.01:1;
…(생략)
xt=A*cos(omega*t+phi);
plot(t,xt);
grid;
xlabel('Time (t)');
ylabel('x(t)');
title('Sinusoidal Signal: Acos(\omegat+\
phi)');
```



gtext

마우스를 이용하여 원하는 위치에 문자열을 삽입하는 gtext 명령어의 기본 형태는 다음과 같다.

gtext('s')

- **'s'** : 삽입할 문자열

그림 창에 십자선이 표시되면 클릭하여 문자열의 위치를 지정한다.

text

설정한 좌표에 문자열을 입력하는 text 명령어의 기본 형태는 다음과 같다.

text(x,y,'s')

- **x** : 문자열이 위치할 x 좌표
- **y** : 문자열이 위치할 y 좌표
- **'s'** : 삽입할 문자열

그래프가 그려짐과 동시에 해당 위치에 문자열 's'가 표시된다.

예제 4-2

$0 \le x \le 2\pi$에 위치한 사인 함수 $y_1 = \sin(x)$와 코사인 함수 $y_2 = \cos(x)$에 대한 그래프를 그리려 한다. 주기는 $\pi/180$ 라디안이며, x축 이름은 'Radian Value', y축 이름은 'Magnitude', 그래프 제목은 'Sine and Cosine Function'으로 지정한다.

(a) gtext 명령어를 이용하여 각각 문자열 'sin(x)'와 'cos(x)'를 표시하라.
(b) text 명령어를 이용하여 좌표 위치 (3.0, 0.5)에 'sin(x)'를, 좌표 위치 (1.5, −0.6)에 'cos(x)'를 표시하라.

풀이 (a)

Command Window

```
>> x=0:pi/180:2*pi;
>> y1=sin(x);
>> y2=cos(x);
>> plot(x,y1,x,y2,':');
>> axis([0 2*pi -1 1]); ❶
>> xlabel('Radian Value'); ❷
>> ylabel('Magnitude');
>> title('Sine and Cosine Function'); ❸
>> grid; ❹
>> gtext('sin(x)'); ❺
>> gtext('cos(x)');
```

그림 4-3 gtext 명령어를 이용한 코딩 결과

❶ axis 명령어를 이용하여 x축의 범위를 0 라디안부터 2π 라디안까지, y축의 범위는 −1부터 1까지로 지정한다.

❷ xlabel, ylabel 명령어를 이용하여 x축과 y축의 이름을 지정한다.

❸ title 명령어를 이용하여 그래프의 제목을 지정한다.

❹ grid 명령어를 이용하여 그래프에 모눈을 생성한다.

❺ 사인 함수와 코사인 함수를 나타내는 두 선이 쉽게 구별되도록 gtext 명령어를 사용한다. 필요한 수만큼 gtext 명령어를 입력하면 [그림 4-4]와 같이 문자열의 위치 선정을 위한 십자선이 나타난다. 문자가 위치할 곳을 클릭하면 [그림 4-5]와 같이 문자가 삽입된다.

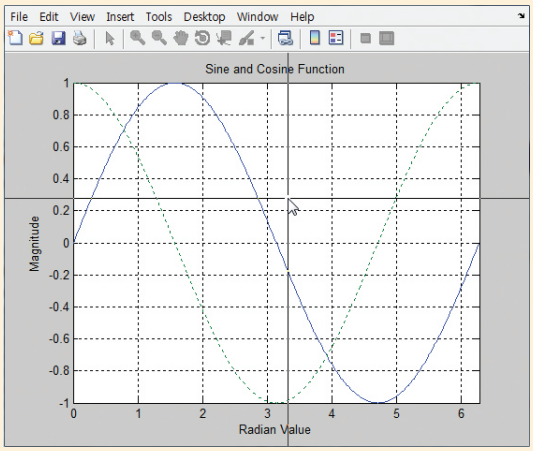


그림 4-4 문자열의 위치 선정을 위한 십자선

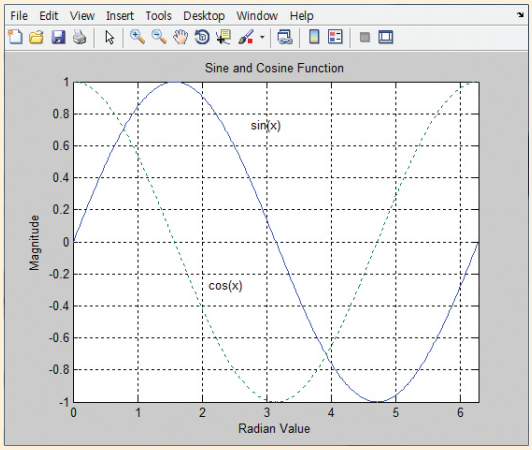


그림 4-5 gtext 명령어를 이용한 실행 결과

(b)

Command Window

```
>> x=0:pi/180:2*pi;
>> y1=sin(x);
>> y2=cos(x);
>> plot(x,y1,x,y2,':');
>> axis([0 2*pi -1 1]);
>> xlabel('Radian Value');
>> ylabel('Magnitude');
>> title('Sine and Cosine Function');
>> text(3.0,0.5,'sin(x)'); ❻
>> text(1.5,-0.6,'cos(x)'); ❼
```

그림 4-6 text 명령어를 이용한 코딩 결과

❻ text 명령어로 x좌표 3.0과 y좌표 0.5를 지정하여 (3.0, 0.5)에 문자열 'sin(x)'를 표시한다.

❼ text 명령어로 x좌표 1.5와 y좌표 −0.6을 지정하여 (1.5, −0.6)에 문자열 'cos(x)'를 표시한다.

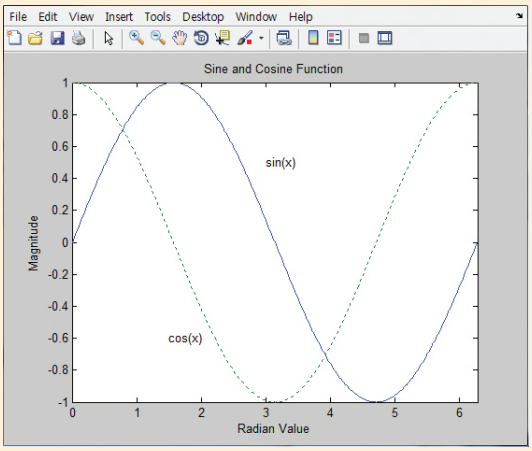


그림 4-7 text 명령어를 이용한 실행 결과

legend

각 그래프에 대한 내용을 's1', 's2', … 등 문자열 형태로 지정하여 범례 형태로 표시하는 legend 명령어의 기본 형태는 다음과 같다.

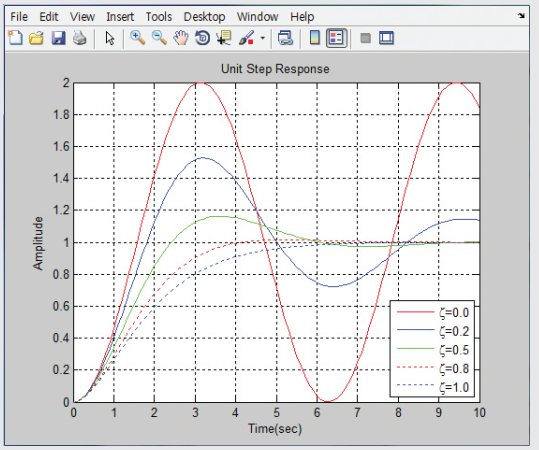
legend('s1','s2',..., location)

- **'s1', 's2'** : 범례로 표시할 그래프의 이름
- **location** : 범례의 위치 지정(오른쪽 위 : 1, 왼쪽 위 : 2, 왼쪽 아래 : 3, 오른쪽 아래 : 4)
  생략 가능하며, 생략할 경우 오른쪽 위로 지정된다.

plot 명령어에 사용한 순서대로 문자열을 지정하며, 반드시 plot 명령어를 사용한 후에 사용해야 한다. 또한 표시된 범례는 마우스 왼쪽 버튼을 누른 채 드래그하면 원하는 위치로 이동할 수 있다. legend 명령어를 사용하지 않고도 [범례] 버튼을 클릭하면 범례가 표시된다.

예를 들어 제어공학에서 감쇠비 값의 변화에 따른 계단 응답의 결과를 비교하고자 할 때, MATLAB에서 legend 명령어를 사용하면 범례를 표시하여 여러 그래프를 비교할 수 있다. 다음은 CHAPTER 09의 [예제 9-13]에서 다루는 CtrlDampRt.m의 일부와 그 결과이다.

```
t=0:0.1:10;
…(생략)
plot(t,y(1:101,1),'r-');
hold;
plot(t,y(1:101,2), 'b-');
…(생략)
legend('\zeta=0.0', '\zeta=0.2',
'\zeta=0.5', '\zeta=0.8', '\zeta=1.0',4);
```



예제 4-3

$0 \le x \le 2\pi$에 위치한 사인 함수 $y_1 = \sin(x)$와 코사인 함수 $y_2 = \cos(x)$에 대한 그래프를 그리려 한다. 주기는 $\pi/180$ 라디안이며, x축 이름은 'Radian Value', y축 이름은 'Magnitude', 그래프 제목은 'Sine and Cosine Function'으로 지정한다. legend 명령어를 이용하여 그래프의 왼쪽 아래에 $\sin(x)$와 $\cos(x)$의 범례를 표시하라.

풀이

```
Command Window
>> x=0:pi/180:2*pi;
>> y1=sin(x);
>> y2=cos(x);
>> plot(x,y1,x,y2,':');  ❶
>> axis([0 2*pi -1 1]);
>> xlabel('Radian Value');
>> ylabel('Magnitude');
>> title('Sine and Cosine Function');
>> legend('sin(x)','cos(x)',3);  ❷
```

그림 4-8 legend 명령어를 이용한 코딩 결과

❶ 벡터 **y1**은 실선으로 그래프를 그리고, 벡터 **y2**는 점선으로 그래프를 그린다.

❷ plot 명령어의 괄호 안과 동일한 순서로 legend 명령어의 괄호 안을 입력하고, 범례의 위치를 왼쪽 아래로 지정한다.

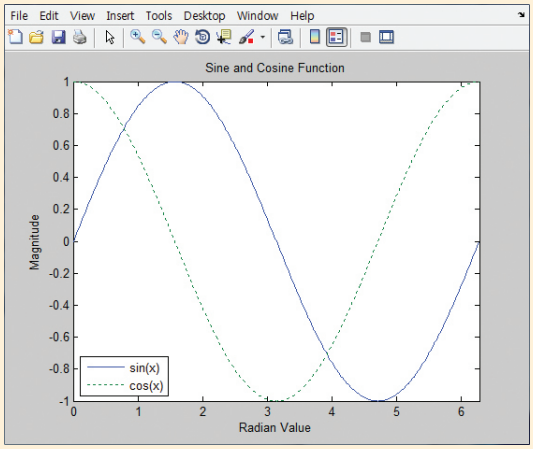


그림 4-9 legend 명령어를 이용한 실행 결과

subplot

그림 창을 분할하여 여러 그래프를 그리는 subplot 명령어의 기본 형태는 다음과 같다.

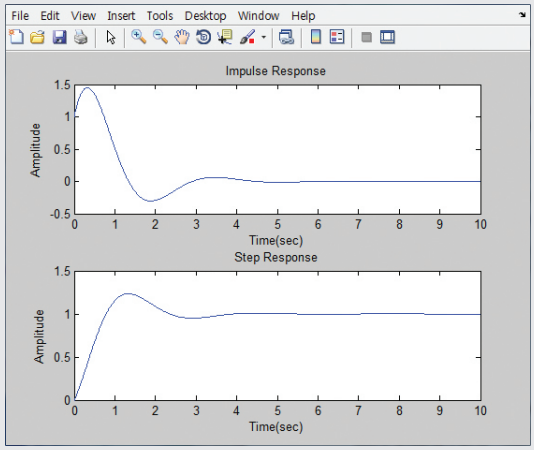
subplot(mnp)

- **m** : 그림 창 안에 분할하려는 행의 개수
- **n** : 그림 창 안에 분할하려는 열의 개수
- **p** : 그래프가 나타날 위치

subplot(mnp) 명령어를 사용할 때 p는 행과 열을 곱한 수보다 큰 수를 지정할 수 없으며, 반드시 plot 명령어 이전에 실행되어야 한다.

예를 들어 제어공학에서 임펄스 응답의 결과와 계단 응답의 결과를 비교하고자 할 때, MATLAB에서 subplot 명령어를 사용하면 한 그림 창에서 두 그래프를 비교할 수 있다. 다음은 CHAPTER 09의 [예제 9-5]에서 다루는 TfResp.m의 일부와 그 결과이다.

```
t=0:0.01:10;
…(생략)
G_s=tf(num,den);
y_imp=impulse(G_s,t);
y_st=step(G_s,t);
subplot(211);
plot(t,y_imp);
…(생략)
subplot(212);
plot(t,y_st);
…(생략)
```



figure

여러 그래프를 각각 다른 그림 창에 그리기 위해 그림 창을 생성하는 figure 명령어의 기본 형태는 다음과 같다.

figure(n)

- **n** : 그림 창의 번호(지정하지 않으면 자동으로 지정된다.)

예제 4-4

$0 \le x \le 2\pi$에 위치한 사인 함수 $y_1 = \sin(x)$와 코사인 함수 $y_2 = \cos(x)$에 대한 그래프를 한 그림 창 안에 각각 두 개의 분할된 그래프로 그리려 한다. 주기는 $\pi/180$ 라디안이며, x축 이름은 'Radian Value', y축 이름은 'Magnitude', 각 그래프의 제목은 'Sine Function'과 'Cosine Function'으로 지정한다.

(a) subplot 명령어를 이용하여 두 그래프를 좌우로 분할하여 그려라.
(b) subplot 명령어를 이용하여 두 그래프를 상하로 분할하여 그려라.
(c) figure 명령어를 이용하여 두 그래프를 두 그림 창에 각각 그려라.

풀이

(a)

```
Command Window
>> x=0:pi/180:2*pi;
>> y1=sin(x);
>> y2=cos(x);
>> subplot(121); ❶
>> plot(x,y1);
>> axis([0 2*pi -1 1]);
>> xlabel('Radian Value');
>> ylabel('Magnitude');
>> title('Sine Function');
>> subplot(122); ❷
>> plot(x,y2);
>> axis([0 2*pi -1 1]);
>> xlabel('Radian Value');
>> ylabel('Magnitude');
>> title('Cosine Function');
```

그림 4-10 subplot 명령어를 이용한 코딩 결과

❶ subplot(121) 명령어를 이용하여 함수 y_1의 그래프를 그림창 왼쪽에 그린다.

❷ subplot(122) 명령어를 이용하여 함수 y_2의 그래프를 그림창 오른쪽에 그린다.

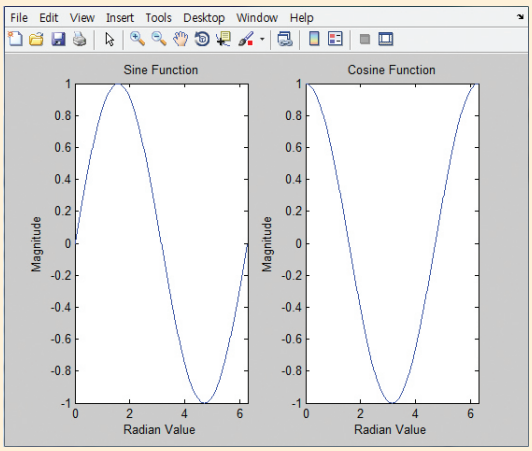


그림 4-11 좌우로 나타난 subplot 명령어 실행 결과

(b)

```
Command Window
>> x=0:pi/180:2*pi;
>> y1=sin(x);
>> y2=cos(x);
>> subplot(211); ❸
>> plot(x,y1);
>> axis([0 2*pi -1 1]);
>> xlabel('Radian Value');
>> ylabel('Magnitude');
>> title('Sine Function');
>> subplot(212); ❹
>> plot(x,y2);
>> axis([0 2*pi -1 1]);
>> xlabel('Radian Value');
>> ylabel('Magnitude');
>> title('Cosine Function');
```

그림 4-12 subplot 명령어를 이용한 코딩 결과

❸ subplot(211) 명령어를 이용하여 함수 y_1의 그래프를 그림창 위쪽에 그린다.

❹ subplot(212) 명령어를 이용하여 함수 y_2의 그래프를 그림창 아래쪽에 그린다.

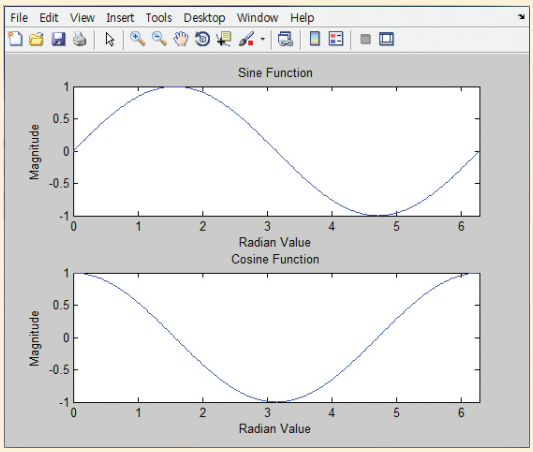


그림 4-13 상하로 나타난 subplot 명령어 실행 결과

(c)

```
Command Window
>> x=0:pi/180:2*pi;
>> y1=sin(x);
>> y2=cos(x);
>> figure(1)  ❺
>> plot(x,y1);
>> axis([0 2*pi -1 1]);
>> xlabel('Radian Value');
>> ylabel('Magnitude');
>> title('Sine Function');
>> figure(2)  ❻
>> plot(x,y2);
>> axis([0 2*pi -1 1]);
>> xlabel('Radian Value');
>> ylabel('Magnitude');
>> title('Cosine Function');
```

그림 4-14 figure 명령어를 이용한 코딩 결과

❺ figure 명령어를 이용하여 그림 창(figure 1)을 생성한다.

❻ figure 명령어를 이용하여 그림 창(figure 2)을 생성한다.

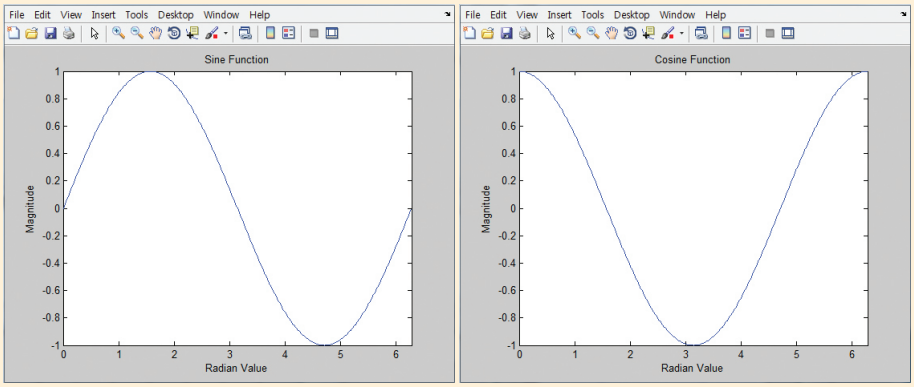


그림 4-15 두 그림 창에 나타난 figure 명령어 실행 결과

그래프 저장하고 불러오기

MATLAB에서는 그림 창에 나타낸 그래프를 저장할 수 있다. 방법 1 그림 창의 메뉴 표시줄에서 [File] – [Save As...] 메뉴를 클릭하고 저장할 파일 이름에 확장자(.fig)를 붙여서 저장하거나 방법 2 명령창에서 saveas 명령어를 사용하여 그래프를 저장할 수 있다. 그래프를 'Filename.fig'로 저장하는 saveas 명령어의 기본 형태는 다음과 같다.

saveas(gcf, 'Filename')

- **gcf** : 저장할 파일이 그래프임을 지정하는 파일 형식
- **'Filename'** : 저장할 파일 이름

또한, 방법 1 그림 창의 메뉴 표시줄에서 [File]–[Open...] 메뉴를 클릭하여 사용하려는 그림 파일을 두 번 클릭하거나 방법 2 명령창에서 openfig 명령어를 사용하여 그래프를 불러올 수 있다. 'Filename.fig' 파일을 불러오는 openfig 명령어의 기본 형태는 다음과 같다.

openfig('Filename')

LESSON
03 3차원 그래프 그리기

그래프를 입체적으로 생성할 수 있는 3차원 그래프 그리기 명령어를 알아보자.

Keyword | plot3 | meshgrid | mesh | surf | contour | meshc | meshz | waterfall | 3차원 그래프의 모양과 종류 |

plot3

x축, y축, z축의 직선들을 서로 연결하여 3차원 공간에 차례로 그리는 plot3 명령어의 기본 형태는 다음과 같다.

plot3(x,y,z)

- **x, y, z** : 동일한 원소의 수를 가진 벡터

예제 4-5

아래 방정식들은 시간 변수 t가 변화함에 따라서 3차원 곡선을 생성한다.

$$x = \cos t, y = \sin t, z = t$$

t가 0에서 15π까지 $\pi/30$씩 증가할 때, plot3 명령어를 이용하여 3차원 그래프를 그려라. 축 이름은 각각 'x–axis', 'y–axis', 'z–axis'로 지정하라.

풀이

```
Command Window
>> t=0:pi/30:15*pi;
>> x=cos(t);
>> y=sin(t);
>> z=t;
>> plot3(x,y,z); ❶
>> xlabel('x-axis'); ❷
>> ylabel('y-axis');
>> zlabel('z-axis');
>> grid; ❸
```

그림 4–16 plot3 명령어를 이용한 코딩 결과

❶ plot3 명령어를 이용하여 3차원 그래프를 그린다.

❷ xlabel, ylabel, zlabel 명령어를 이용하여 x, y, z축 이름을 지정한다.

❸ 그래프에 모눈을 그린다.

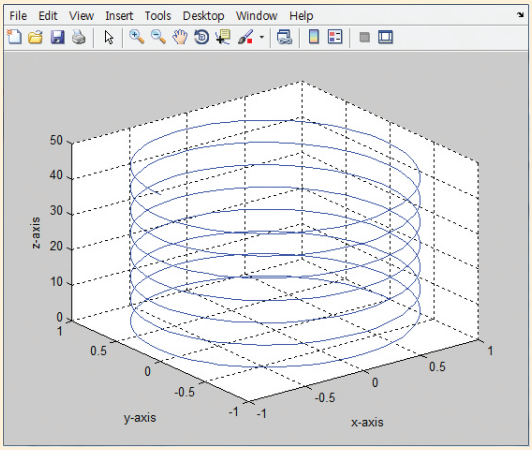


그림 4-17 plot3 명령어를 이용한 실행 결과

meshgrid

벡터 **x**와 **y**로 지정된 영역을 3차원 그래프로 나타내려면 두 변수 x와 y를 배열 **X**와 **Y** 형태로 변환해야 한다. 이때 사용하는 meshgrid 명령어의 기본 형태는 다음과 같다.

[X, Y] = meshgrid(x, y)

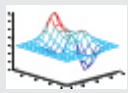
- **X, Y** : 배열로 나타난 결과
- **x, y** : 배열로 나타낼 벡터 또는 변수

mesh

배열 **x**, **y**의 함수로 구성된 **z**의 그래프($z = f(x, y)$)를 그리는 mesh 명령어의 기본 형태는 다음과 같다.

mesh(x,y,z)

- **x, y** : 축 정보가 저장된 배열
- **z** : 배열 x, y의 값으로 이루어진 배열



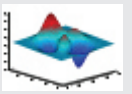
mesh 명령어를 이용하면 3차원 격자무늬를 그릴 수 있다. 이때 z에 비례하여 표면의 높이와 그래프의 색상이 결정된다.

surf

3차원 격자무늬에 색을 채워 그리는 surf 명령어의 기본 형태는 다음과 같다.

surf(x,y,z)

- **x, y** : 표면의 구성 요소를 정의하는 벡터
- **z** : x, y로 표현된 식 또는 벡터



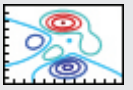
surf 명령어는 함수의 모든 곡면에 색을 채워 그려주므로, mesh 명령어를 사용했을 때보다 색이 더 뚜렷하게 채워진다. 이때 z에 비례하여 표면의 높이와 그래프의 색상이 결정된다.

contour

x와 **y**를 이용하여 **z**의 값을 2차원 등고선 그래프로 나타내는 contour 명령어의 기본 형태는 다음과 같다.

contour(x,y,z)

- **x, y** : x 축과 y 축의 한계를 결정하는 벡터
- **z** : x, y로 표현된 식 또는 벡터



예제 4-6

$-3 \le x \le 3$과 $-3 \le y \le 3$에 0.1 간격으로 놓여 있는 함수 $z = x^2 - y^2$이 있다. 세 개의 축 이름은 각각 'x–axis', 'y–axis', 'z–axis'로 지정하라.

(a) mesh 명령어를 이용하여 3차원 격자무늬를 그려라.
(b) surf 명령어를 이용하여 3차원 격자무늬 사이에 색을 채워 그려라.
(c) contour 명령어를 이용하여 2차원 등고선을 그려라.

풀이 (a)

Command Window

```
>> [x,y]=meshgrid(-3:0.1:3); ❶
>> z=x.^2-y.^2;
>> mesh(x,y,z); ❷
>> xlabel('x-axis'); ❸
>> ylabel('y-axis');
>> zlabel('z-axis');
```

그림 4–18 mesh 명령어를 이용한 코딩 결과

❶ meshgrid 명령어를 이용하여 x, y의 범위를 동시에 벡터로 표현한다.
❷ mesh 명령어를 이용하여 3차원 격자무늬를 그린다.
❸ xlabel, ylabel, zlabel 명령어를 이용하여 x, y, z 축 이름을 지정한다.

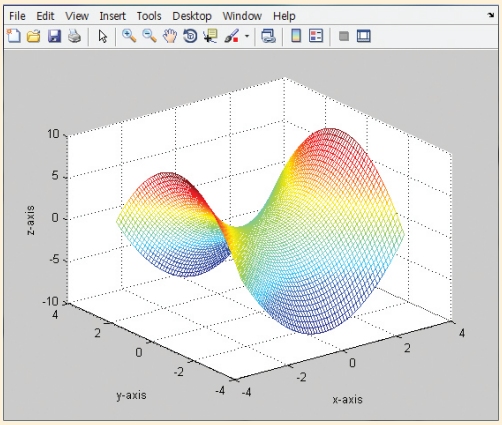


그림 4-19 mesh 명령어를 이용한 실행 결과

(b)

```
Command Window
>> [x,y]=meshgrid(-3:0.1:3); ❹
>> z=x.^2-y.^2;
>> surf(x,y,z); ❺
>> xlabel('x-axis'); ❻
>> ylabel('y-axis');
>> zlabel('z-axis');
```

그림 4-20 surf 명령어를 이용한 코딩 결과

❹ meshgrid 명령어를 이용하여 x, y의 범위를 동시에 벡터로 표현한다.

❺ surf 명령어를 이용하여 3차원 격자무늬에 색을 채워 그린다.

❻ xlabel, ylabel, zlabel 명령어를 이용하여 x, y, z축 이름을 지정한다.

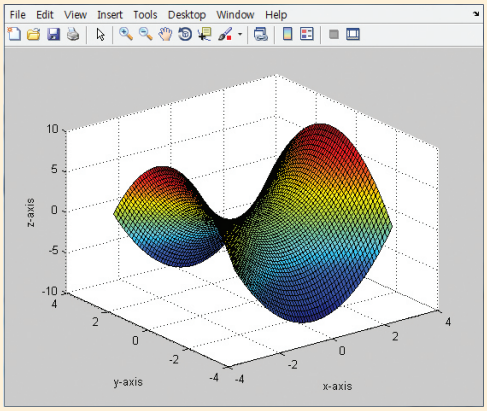


그림 4-21 surf 명령어를 이용한 실행 결과

(c)

```
Command Window
>> [x,y]=meshgrid(-3:0.1:3); ❼
>> z=x.^2-y.^2;
>> contour(x,y,z); ❽
>> xlabel('x-axis'); ❾
>> ylabel('y-axis');
```

그림 4-22 contour 명령어를 이용한 코딩 결과

❼ meshgrid 명령어를 이용하여 x, y의 범위를 동시에 벡터로 표현한다.

❽ contour 명령어를 이용하여 2차원 등고선을 그린다.

❾ xlabel, ylabel 명령어를 이용하여 x, y 축 이름을 지정한다.

TIP contour 명령어가 3차원 그래프를 그리는 명령어이지만 2차원 등고선 그래프를 묘사하기 때문에 zlabel 명령어는 쓸 필요가 없다.

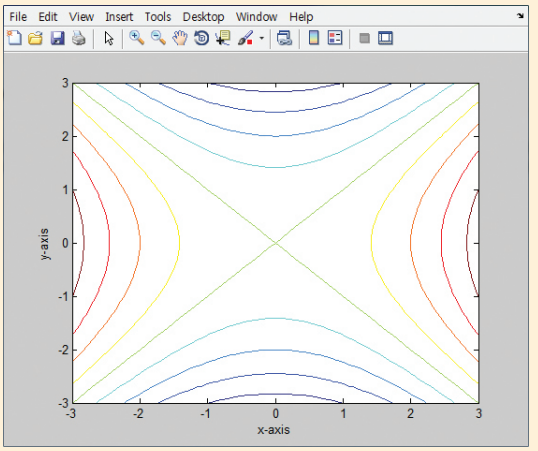


그림 4-23 contour 명령어를 이용한 실행 결과

meshc

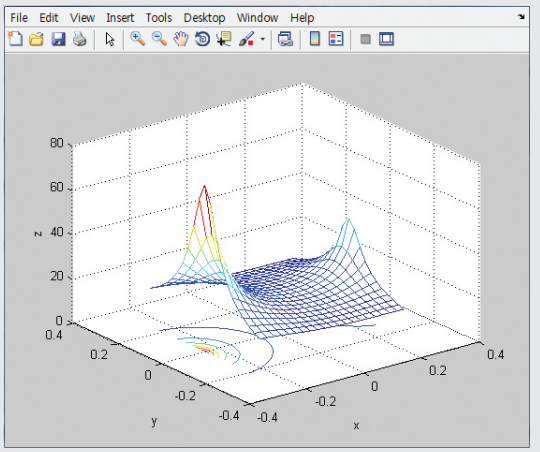
3차원 격자무늬 그래프와 등고선을 함께 그리는 meshc 명령어의 기본 형태는 다음과 같다.

meshc(x,y,z)

- **x, y** : 축에 대한 정보가 저장된 배열
- **z** : 배열 x, y의 값으로 이루어진 배열

예를 들어 전자기학에서는 특정한 점에서의 전기 에너지 분포인 전위의 형태를 알아보기 위해 전위를 그래프로 나타낸다. MATLAB에서는 meshc 명령어를 이용하여 전위의 형태를 3차원 그래프로 나타낼 수 있다. 다음은 CHAPTER 11의 [예제 11-23]에서 다루는 Potential.m의 일부와 그 결과이다.

```
…(생략)
[x y] = meshgrid(-0.25:0.025:0.25);
…(생략)
V = (1/(4*pi*epsilon0))*(q1./r1 +
      q2./r2);
meshc(x,y,V);
xlabel('x');
ylabel('y');
zlabel('z');
```



meshz

x축과 y축 양방향으로 수직선을 그리는 3차원 격자무늬 그래프와 장막을 함께 그리는 meshz 명령어의 기본 형태는 다음과 같다.

meshz(x,y,z)

- **x, y** : 축에 대한 정보가 저장된 배열
- **z** : 배열 x, y의 값으로 이루어진 배열

waterfall

x축 또는 y축 한 방향으로 수직선을 그리며 폭포 형태를 생성하는 waterfall 명령어의 기본 형태는 다음과 같다.

waterfall(x,y,z)

- **x, y** : 축에 대한 정보가 저장된 배열
- **z** : 배열 x, y의 값으로 이루어진 배열

예제 4-7

$-3 \le x \le 3$과 $-3 \le y \le 3$에 0.1 간격으로 놓여 있는 함수 $z = x^2 - y^2$이 있다. 세 개의 축 이름은 각각 'x–axis', 'y–axis', 'z–axis'로 지정하라.

(a) meshc 명령어를 이용하여 3차원 격자무늬 그래프와 등고선을 함께 그려라.
(b) meshz 명령어를 이용하여 3차원 격자무늬 그래프와 장막 모양을 함께 그려라.
(c) waterfall 명령어를 이용하여 3차원 수직선 그래프와 폭포 모양의 그래프를 함께 그려라.

풀이

(a)

Command Window

```
>> [x,y]=meshgrid(-3:0.1:3); ❶
>> z=x.^2-y.^2;
>> meshc(x,y,z); ❷
>> xlabel('x-axis'); ❸
>> ylabel('y-axis');
>> zlabel('z-axis');
```

그림 4–24 meshc 명령어를 이용한 코딩 결과

❶ meshgrid 명령어를 이용하여 x, y의 범위를 동시에 벡터로 표현한다.

❷ meshc 명령어를 이용하여 3차원 격자무늬 그래프와 등고선을 함께 그린다.

❸ xlabel, ylabel, zlabel 명령어를 이용하여 x, y, z축 이름을 지정한다.

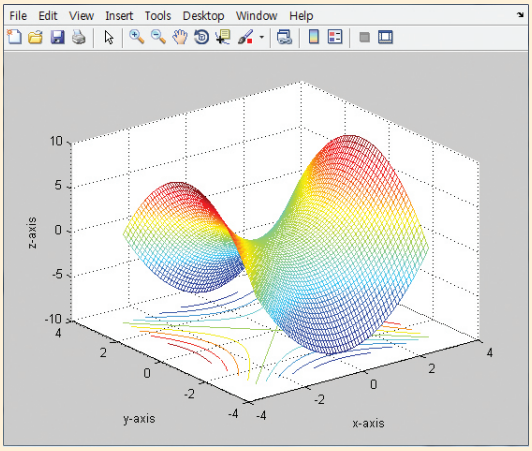


그림 4-25 meshc 명령어를 이용한 실행 결과

(b)

```
Command Window
>> [x,y]=meshgrid(-3:0.1:3); ❹
>> z=x.^2-y.^2;
>> meshz(x,y,z); ❺
>> xlabel('x-axis'); ❻
>> ylabel('y-axis');
>> zlabel('z-axis');
```

그림 4-26 meshz 명령어를 이용한 코딩 화면

❹ meshgrid 명령어를 이용하여 x, y의 범위를 동시에 벡터로 표현한다.

❺ meshz 명령어를 이용하여 3차원 격자무늬 그래프와 장막 모양을 함께 그린다.

❻ xlabel, ylabel, zlabel 명령어를 이용하여 x, y, z축 이름을 지정한다.

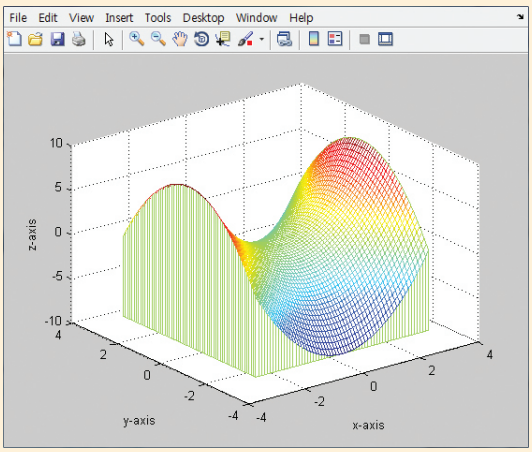


그림 4-27 meshz 명령어를 이용한 실행 결과

(c)

```
Command Window
>> [x,y]=meshgrid(-3:0.1:3); ❼
>> z=x.^2-y.^2;
>> waterfall(x,y,z); ❽
>> xlabel('x-axis'); ❾
>> ylabel('y-axis');
>> zlabel('z-axis');
```

그림 4-28 waterfall 명령어를 이용한 코딩 화면

❼ meshgrid 명령어를 이용하여 x, y의 범위를 동시에 벡터로 표현한다.

❽ waterfall 명령어를 이용하여 3차원 수직선 그래프와 폭포 모양의 그래프를 함께 그린다.

❾ xlabel, ylabel, zlabel 명령어를 이용하여 x, y, z 축 이름을 지정한다.

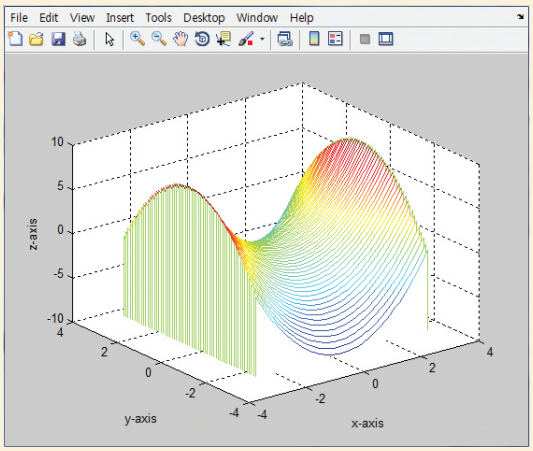
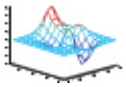
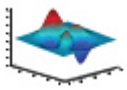
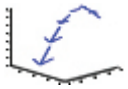
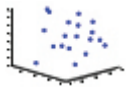
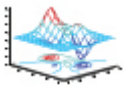
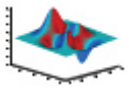
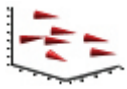
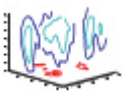
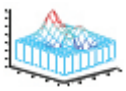
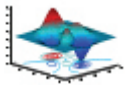
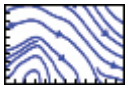
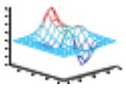
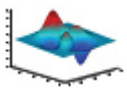
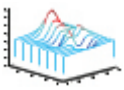
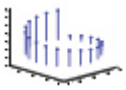
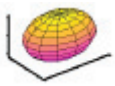
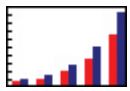
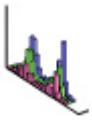
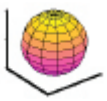


그림 4-29 waterfall 명령어를 이용한 실행 결과

3차원 그래프의 모양과 종류

선 그래프	격자무늬 그래프와 막대 그래프	공간 그래프	면 그래프	방향 그래프	체적 그래프
plot3(X,Y,Z) 	mesh(X,Y,Z) 	pie3(X,explode) 	surf(X,Y,Z) 	quiver3 (x,y,z,u,v,w) 	scatter3(X,Y,Z,S,C) 
contour3(X,Y,Z) 	meshc(X,Y,Z) 	fill3(X,Y,Z,C) 	surfl(X,Y,Z) 	comet3(x,y,z) 	coneplot(X,Y,Z,U,V, W,Cx,Cy,Cz) 
contourslice (X,Y,Z,V,Sx,Sy,Sz) 	meshz(X,Y,Z) 	patch(X,Y,C) 	surfc(X,Y,Z) 	streamslice (X,Y,U,V) 	streamline(X,Y,Z,U,V, W,startx,starty,startz) 
ezplot3 (funx,funy,funz) 	ezmesh(fun) 	cylinder [X,Y,Z] = cylinder(r) 	ezsurf(fun) 		streamribbon(X,Y,Z, U,V,W,startx,starty, startz) 
waterfall(X,Y,Z) 	stem3(X,Y,Z) 	ellipsoid [x,y,z] = ellipsoid (xc,yc, zc,xr,yr,zr) 	ezsurfc(fun) 		streamtube(X,Y,Z,U, V,W,startx,starty, startz) 
	bar3(Y) 	sphere(n) 			
	bar3h(Y)