목 차

제 1 장 MATLAB 소개

제 2 장 기본적인 MATLAB 사용법

제 3 장 파일 관리 기능

제 4 장 MATLAB 프로그래밍

제 5 장 2차원 및 3차원 그래프

제 6 장 Simulink

제 1 장 MATLAB 소개

1. 개요

MATLAB이란 'Matrix Laboratory'를 뜻하는 말로써, 수치 해석, 행렬 연산, 신호 처리 및 간편한 그래픽 기능 등을 통합하여 고성능의 수치계산 및 결과의 가시화 기능을 제공하는 프로그램이다. MATLAB은 그 이름이 말하듯이 행렬 또는 벡터를 기본 자료로 사용하여 기능을 수행하는 계산 환경을 제공한다.

MATLAB은 본래 Cleve Moler가 Fortran을 사용하여 작성한 프로그램으로서 Linpack과 Eispack 프로젝트에 참여한 사람들에 의하여 기본적인 행렬 알고리즘이 만들어졌다. 현재의 MATLAB은 Mathworks사에서 C로 작성하였으며, 초판은 Steve Bangert, Steve Kleiman, John Little과 Cleve Moler에 의해서 작성되었다. 그 이후에는 MATLAB 개발팀 및 여러 사람들에 의하여 실질적인 개발이 이루어졌다.

MATLAB은 기본적으로 행렬 자료를 다루기 때문에 차원화(dimensioning)가 필요하지 않으며 통상적인 프로그래밍 언어들(Fortran, C, Pascal 및 Basic 등)을 사용하여 프로그램을 작성하지 않고도 쉽게 수치 계산을 수행할 수 있다.

MATLAB의 가장 큰 특징은 M-file을 사용함으로써 특정한 해를 구하는데 필요한 응용 프로그램들을 손쉽게 작성할 수 있다는 점이다. M-파일이란 매크로 파일로서 해석기(interpreter) 방식으로 수행되며 사용자가 직접 작성할 수 있는 프로그램이다. 기본적인 내부 명령들 뿐만 아니라 다른 M-파일들도 불러서 사용할 수 있으며, 특정한 문제를 풀기 위하여 사용자가 직접 손쉽게 M-파일을 작성하여 사용할 수 있다는 점이 커다란 특징이다.

2. MATLAB의 실행

2.1 MATLAB의 명령창과 그림창

MATLAB프로그램을 실행하기 위해서는 Windows가 먼저 활성화되어 있어야 한다. 다음으로, MATLAB을 표시하고 있는 아이콘(icon)을 더블 클릭(double click)함으로써(그림 1.1) MATLAB프로그램을 실행할 수 있는 그림 1.2와 같은 명령창(command window)이 열리게 된다. 이때

≫

모양의 프롬프트(한글 Window일 경우는 □로 표시)를 화면에 표시한다. 이러한 프롬프트가 화면에 나타나면 사용자는 MATLAB 명령어를 입력하여 실행시킬 수 있다.
MATLAB을 끝내기 위해서는

≫ quit

혹은

≫ exit 를 사용한다.

그림 1.1 MATLAB 폴더의 아이콘(Icon)

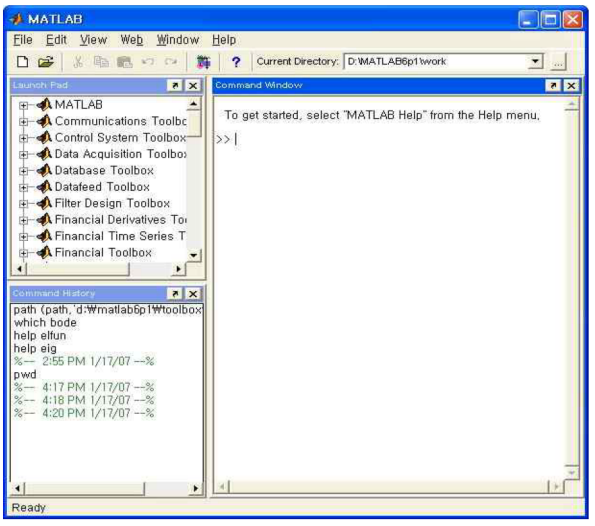


그림 1.2 MATLAB 실행화면

2.2 경로(Path)

MATLAB 프로그램에서도 다른 프로그램들과 마찬가지로 경로가 정확하게 설정되어야만 프로그램을 착오 없이 실행할 수 있다.

MATLAB 프롬프트에서 'path' 라고 입력하면 MATLAB에서 검색하는 모든 경로를 보여준다.

```
≫path

            MATLABPATH
D:\TOOL\MATLAB\toolbox\MATLAB\general
D:\TOOL\MATLAB\toolbox\MATLAB\ops
D:\TOOL\MATLAB\toolbox\MATLAB\lang
D:\TOOL\MATLAB\toolbox\symbolic
                    :
                    :
D:\TOOL\MATLAB\toolbox\map\map
D:\TOOL\MATLAB\toolbox\map\mapdisp
D:\TOOL\MATLAB\toolbox\map\mapproj
D:\TOOL\MATLAB\toolbox\pde
D:\TOOL\MATLAB\toolbox\fixpoint
```

MATLAB 5.x 버전에서는 그림 1.3과 같이 MATLAB 명령창(command window)의 풀-다운 메뉴(File→Set Path)를 이용한 Path Browser에서 쉽게 path를 추가시키거나 제거할 수 있다.

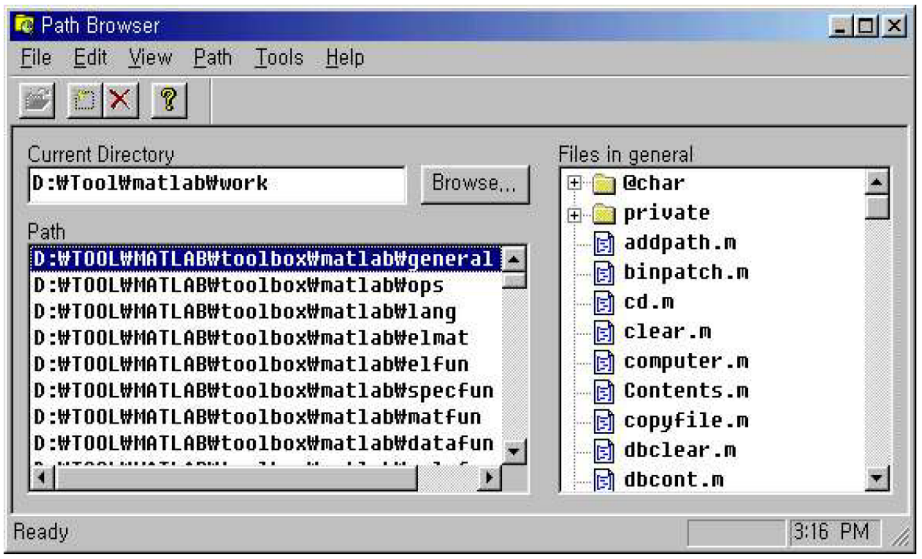


그림 1.3 MATLAB의 Path Brower

MATLAB 6.x 버전에서는 위와 거의 유사한 방법으로 MATLAB 명령창(command window)의 풀-다운 메뉴(File→Set Path)를 이용한 Set Path에서 쉽게 path를 추가시키거나 제거할 수 있다.

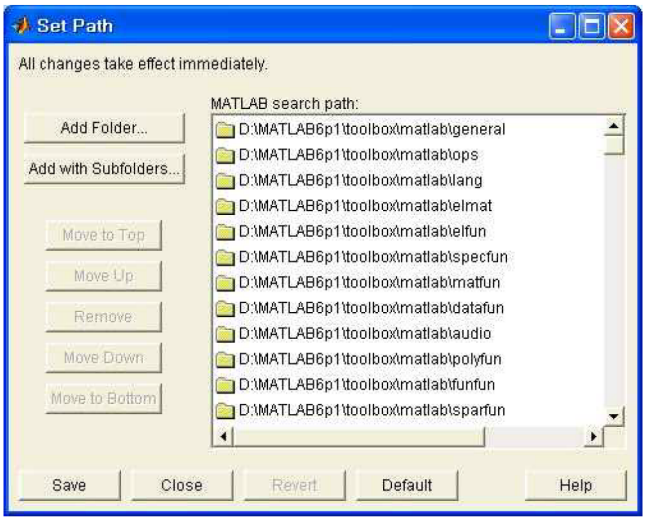


그림 1.4 MATLAB의 Set Path

2.3 도움말 기능

MATLAB과 관련된 내용에 대하여 도움말을 제공받고 싶으면

》help

를 사용한다. 이 경우 MATLAB과 관련된 파일들이 들어 있는 디렉토리의 목록과 디렉토리의 내용을 보여준다. 특정한 디렉토리의 함수들의 목록에 관하여 알아보고자 할 때에는

help 다음에 원하는 디렉토리의 이름을 입력한다. 예를 들면

》 help elfun

》 help eig

등 이다. help 다음 항목이 MATLAB 내부 명령어가 아니거나 MATLAB 경로상의 M-파일이 아닐 경우, 그 항목을 찾을 수 없다.

help보다 일반적인 검색 명령어로서 lookfor가 사용된다. 다음과 같이 입력하면 M-파일 도움말 주석의 첫번째 줄에 'random'이라는 단어를 포함하는 모든 함수들을 나열해 준다.

```
》 lookfor random
  RAND          Uniformly distributed random numbers.
  RANDN         Normally distributed random numbers.
  RANDPERM      Random permutation.
  RJR           Random Jacobi rotation.
  SPRAND        Sparse uniformly distributed random matrix.
  SPRANDN       Sparse normally distributed random matrix.
  SPRANDSYM     Sparse random symmetric matrix.
                        :
                        :
```

또한 함수가 어느 디렉토리에 있는지 그 위치를 알고자 할 때에는 다음과 같이 which 다음에 찾고자 하는 함수를 입력하면 된다.

```
》 which bode
   D:\TOOL\MATLAB\toolbox\control\bode.m
```

위와 같은 방법 외에도 MATLAB 메뉴에서 'Help'를 이용하여 MATLAB에 관련된 내용을 도움받을 수 있다.

2.4 작업의 종료와 저장

quit나 exit를 사용하여 MATLAB을 종료하면 작업 공간에 존재하는 모든 변수들을 삭제한다. 만일 종료직전까지 사용한 변수나 결과를 저장하였다가 다시 사용하고 싶으면 다음 두 가지 방법을 사용할 수 있다.

첫 번째 방법으로 사용자가 작업한 모든 과정을 텍스트 파일로 저장하고 싶을 때 다음 명령어를 사용한다.

》 diary file_name : 그래픽을 제외한 모든 입력, 출력을 파일 *file_name*에 저장

》 diary off : 저장을 중지

》 diary on : 파일 *file_name*에 저장을 다시 시작

두 번째 방법으로 작업 중 사용한 변수를 저장하였다가 다시 사용하고 싶을 때 다음 명령어를 사용한다.

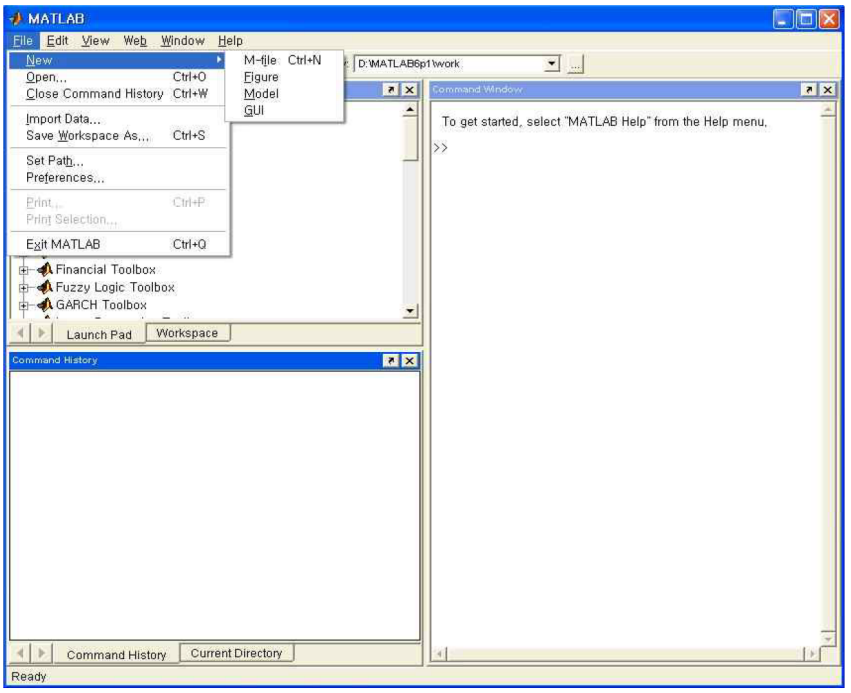
》 save : default 파일인 MATLAB.mat에 모든 변수 저장
》 save file_name : 사용한 모든 변수를 파일 *file_name.mat*에 저장
》 save file_name var_list : 변수 var_list를 파일 *file_name.mat*에 저장
》 load file_name : *file_name.mat*에 저장된 모든 변수를 불러 들임.

diary로 저장된 파일은 텍스트 파일이므로 편집이 가능하나 load 명령어를 사용하여 불러들일 수 없으며, save 명령어로 저장된 파일은 binary형태이므로 편집할 수 없으나 다음과 같이 save 옵션에 -ascii를 사용해서 저장하면 저장된 파일의 편집이 가능하다.

》 save file_name.mat -ascii : 사용한 모든 변수를 파일 *file_name.mat*에 ascii형태로 저장

2.5 MATLAB 풀-다운 메뉴의 사용법

(1) 명령창 메뉴



■ File

① New

M-file
Figure
Model
GUI

▶ M-file : 새로운 M-파일 작성
▶ Figure : 새로운 그림창을 생성
▶ Model : SIMULINK 설치시 새로운 SIMULINK의 모델 작성
▶ GUI : 새로운 GUI창을 생성

② Open ... : 어떤 M-파일을 선택할 것인지를 묻는 파일 선택용 대화상자
③ Close Command History : Command History 창을 닫게 한다.
④ Import Data ... : 다른 응용프로그램의 데이터를 MATLAB으로 불러온다.
⑤ Save WorkSpace As ... : 현재 작업 공간 내의 변수들을 저장
⑥ Set Path... : Set Path 창을 활성화
⑦ Preferences... : 숫자 표현 방식, 에디터 프로그램 설정, Font 설정, Copying Option 설정
⑧ Print... : MATLAB에서 작업한 내용을 출력
⑨ Print Selection... : 프린터 선택
⑩ Exit MATLAB : MATLAB 작업 종료

■ Edit

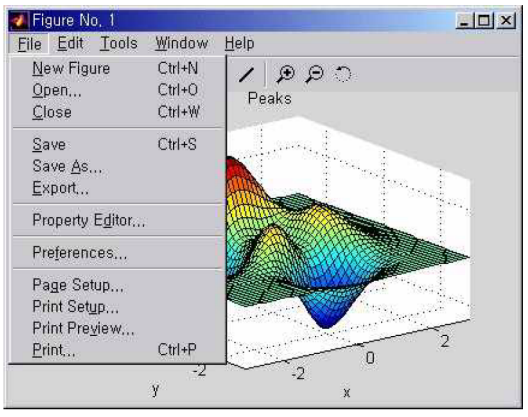
Undo	Ctrl + Z
Redo	Ctrl + Y
Cut	Ctrl + X
Copy	Ctrl + C
Paste	Ctrl + V
Paste Special ...	
Select All	
Delete	
Clear Command Windows	
Clear Command History	
Clear Workspace	

① Undo : 실행취소
② Redo : 재실행
③ Cut : 명령창에서 선택된 부분을 오려내어 그 내용을 클립보드에 저장
④ Copy : 선택된 부분을 그대로 보존하면서 그 내용을 클립보드에 복사
⑤ Paste : 편재 클립보드에 저장되어 있는 내용을 명령창에 복사

⑥ Paste Special ... : 편재 클립보드에 저장되어 있는 내용을 일부분만 명령창에 복사
⑦ Select All : 명령창의 모든 내용을 선택
⑧ Delete : 삭제
⑨ Clear Command Windows : Command Windows창 초기화
⑩ Clear Command History : Command History창 초기화
⑪ Clear Workspace : Workspace창 초기화

■ **View** : Toolbar를 on-off
■ **Web** : 관련 Website로 이동
■ **Window** : 열려져 있는 window 창을 선택
■ **Help** : 각종 도움말 창을 활성화

(2) 그림창 메뉴



■ **File**
① New Figure : 새로운 그림창을 생성 (>>*figure*)
② Open... : 그림파일(*.fig)을 연다.
③ Close : 그림창을 닫는다.
④ Save : 그림창을 저장
⑤ Save As... : 그림창을 다른 이름으로 저장
⑥ Export... : 그림창을 다른 형식의 파일로 저장
⑦ Property Editor : Graphics Property Editor 창을 활성화
⑧ Preferences... : 숫자 표현 방식, 에디터 프로그램 설정, Font 설정, Copying Option 설정
⑨ Page Setup... : 페이지 설정
⑩ Print Setup : 프린터 설정
⑪ Print Preview... : 미리보기
⑫ Print : 인쇄

■ **Edit** : 자르기, 복사, 붙이기
■ **View** : Figure 도구상자와 Camera 도구상자의 표시
■ **Insert** : 축과 선, 글자 등 삽입
■ **Tools**

Edit Plot
Zoom In
Zoom Out
Rotate 3D
Move Camera
Camera Motion
Camera Axis
Camera Reset
Basic Fitting
Data Statistics

① Edit Plot : Plot 을 수정
② Zoom In : 확대
③ Zoom Out : 축소
④ Rotate 3D : 객체를 보는 관점을 변화(방위각, 고도 변경)
⑤ Move Camera : 객체를 보는 관점을 변화
⑥ Camera Motion : Zoom, Roll, Walk 등의 기능 수행
⑦ Camera Axis : 축에따른 이동
⑧ Camera Reset : 객체를 보는 관점을 Reset한다.
⑨ Basic Fitting : 기본 정보 맞춤
⑩ Data Statistics : 데이터 통계

■ **Window** : 열려져 있는 window 창을 선택
■ **Help** : 각종 도움말 창을 활성화

3. MATLAB 도구상자(Toolbox)

MATLAB의 구조는 상당히 유연성이 있고, 특정한 목적을 요구하는 도구상자(toolbox)들을 추가함으로써 보다 다양하고 기본적인 기능을 쉽게 추가할 수 있도록 되어 있다. 이러한 toolbox들은 여러 학문 분야의 개념을 빠른 수학적 알고리즘으로 함수화하여 모아 놓은 MATLAB의 함수들로서 다양한 toolbox가 제공되고 있는데 이들을 간단히 소개하면 다음과 같다.

- *신호 처리(Signal Processing) Toolbox* : 디지털 신호 처리 기술을 위한 다양한 종류의 함수들이 모인 집단으로써 신호 및 선형 시스템의 모델, 디지털 및 아날로그 필터의 설계와 분석 및 Power Spectrum 평가를 위한 기능(FFT 분석)을 포함하고 있다.
- *영상 처리(Image Processing) Toolbox* : 2차원적인 영상 신호의 처리 및 해석을 위한 각종 함수들을 제공한다.
- *제어 시스템(Control System) Toolbox* : 시스템의 분석 및 제어기 설계를 위한 MATLAB프로그램들의 함수들이 있다. 여기서 다루어지는 함수들의 주요한 특징은 전달함수와 상태 방정식을 대상으로 하여 시스템의 해석 및 설계가 가능하며, 수치해석의 알고리즘은 상당한 정확성을 갖고 있다. 그리고 이러한 결과들을 쉽게 다룰 수 있는 그래프를 통해 이해의 폭을 넓게 한다.
- *퍼지(Fuzzy Logic) Toolbox* : 복잡하고 비선형적인 시스템에 대한 정확한 모델링이 없어도 제어를 가능하게 할 수 있는 방법이다. 이러한 퍼지 기술을 응용 시스템에 적용하여 제어기 디자인과 시뮬레이션 및 실시간 제어 등을 가능하게 한다.
- *통신(Communication) Toolbox* : 통신 시스템의 해석과 디자인 및 시뮬레이션을 위한 해석 함수들을 포함하고 있다.
- *신경 회로망(Neural Network) Toolbox* : 패턴 인식이나 비선형 시스템의 모델링과 제어와 같은 해석이 극히 어려운 경우에 신경회로망을 적용하여 종종 만족스런 결과를 얻을 수 있다. 여러 가지의 신경망을 제공하여 시스템의 디자인과 수행 및 시뮬레이션이 가능하도록 하고 있다.
- *최적화(Optimization) Toolbox* : 비선형 함수의 최적화 문제, 선형 2차 형식의 최적화, Non-negative least square, 비선형 방정식의 해 등을 위해 다양한 함수 프로그램들을 제공한다.
- *통계(Statistics) Toolbox* : 확률 및 통계의 자료들을 해석, 모델링 및 시뮬레이션할 수 있는 다양한 함수들을 제공한다.
- *시스템 추정(System Identification) Toolbox* : 입출력 데이터로부터 동적 시스템의 선형 모델을 만들 수 있는 방법으로, 제어 설계 및 신호처리에 응용되고 있으며, 최근에 금융, 경제 분야에서의 모델링 방법으로도 각광을 받고 있다.
- *기초수학(Symbolic Math) Toolbox* : Maple V.를 기본으로 하여 각종 수학 계산식 등을 실행한다.

제 2 장 기본적인 MATLAB 사용법

이 장에서는 MATLAB을 사용하는데 있어서 알아두어야 할 가장 기본적인 사항들과 함수들의 사용법에 대하여 간단히 살펴보기로 한다. 행렬 또는 벡터의 입력 및 조작에 대하여 알아보고, 이러한 자료들을 다루는 작업 공간에 대한 정보의 획득 및 저장과 작업의 중지, 수치 자료의 표현, 출력 형식, 도움말, 함수 등에 관하여 살펴보기로 한다.

1. 행렬의 입력 방법

MATLAB에서 사용되는 행렬 입력방식에는 다음과 같이 몇 가지가 있다.

① MATLAB 명령 행(command-line)에서 직접 행렬의 원소들을 하나씩 입력하여 행렬을 구성
② 내부명령 또는 함수들을 사용하여 행렬을 구성
③ M-파일 내에서 행렬을 구성
④ 외부의 자료 파일로부터 행렬을 불러들임

MATLAB 언어에서는 다른 프로그래밍 언어들과는 달리 차원 선언이나 형 선언이 없다. 현재 사용하고 있는 컴퓨터에서 사용 가능한 크기까지 자동적으로 저장 공간을 할당해 준다. 작은 크기의 행렬을 입력하는 가장 간단한 방법은 다음과 같이 직접적을 원소들을 나열하여 입력하는 것이다.

① 원소들은 빈 칸(공백) 또는 쉼표(,)를 사용하여 분리한다.
② 전체 원소들은 대괄호([ ])로 감싼다.
③ 원소의 끝에 세미콜론(;, semicolon)을 붙이면 한 행의 종료를 의미한다.

2. 행렬의 원소들

- MATLAB에서 사용되는 임의의 표현들 : 수치, 함수, 수식, 문자 등

```
》A=[1 2 3 ; 4, 5, 6 ; 7, 8 9]
  A=
     1 2 3
     4 5 6
     7 8 9
》  x=[-2.5 exp(2.5) 2*3/4]
   x=
     -2.5 12.1825 1.5
》y=['abc' ; 'x' 'y' 'z']
   y=
```

abc
xyz

- 행렬의 원소들 중에서 어느 한 값만을 새로 입력하거나 수정

》 x(5)=abs(x(1))
x=
-2.5 12.128 1.5 0 2.5
↓
※ 값이 지정되지 않은 색인의 원소들은 자동적으로 0으로 지정된다.

- 행렬 자신을 부행렬(sub-matrix)로서 불러서 사용 가능

》 z=[1 3 5];
》 A=[A;z]
A=
1 2 3
4 5 6
7 8 9
1 3 5

- 큰 행렬의 일부 원소들을 선택해서 작은 행렬로 구성

》 B=A(1:3, :)
B=
1 2 3
4 5 6
7 8 9
》 C=A(3:4, 2:3)
C=
8 9
3 5

3. 복소수와 복소 행렬

- 숫자 뒤에 'i' 또는 'j'를 붙임으로써 복소수 표현

》 z=1+2*i 또는 z=1+2i

또는

》 z=1+2*j는 z=1+2j

z=

1.0000 + 2.0000i

- 복소 행렬 구성하는 방법

》 A=[1 2; 3 4]+i*[4 3; 2 1] 또는 A=[1+4i 2+3i; 3+2i 4+i]

※ 주의:'+' 또는 '-'부호 전후에 공백을 두어서는 안된다.

A =

2.0000 + 4.0000i 4.0000 + 3.0000i

6.0000 + 2.0000i 8.0000 + 1.0000i

》 x=[1 2; -3 4];

》 i=abs(-x);

》 z=1+2*i

z=

3 5

7 9 ====> z는 복소수 '1+2i'가 아니라 '2|x|+1'의 값이 된다.

4. 전치 행렬(matrix transpose)

- 작은 따음표(', prime 또는 apostrophe)사용

》 x= [1 2 3]'

x=

1

2

3

- 복소 행렬일 경우 주의! (단순한 전치 행렬이 아니라 공액복소 전치 행렬이 구해짐)

》 a=[1+2i 2-3i]

》 b=a' --> 원하는 결과가 아니다.

b =

1.0000 - 2.0000i

2.0000 + 3.0000i

》 c=a.' 또는 d=conj(a') 사용.

c = d =

1.0000 + 2.0000i 1.0000 + 2.0000i

2.0000 - 3.0000i 2.0000 - 3.0000i

5. MATLAB에 사용되는 사칙 연산자들

+	addition	/	오른 나누기(right division)
-	subtraction	\	왼 나누기(left division)
*	multiplication	^	power

* /, \ : 산술 연산(1/2, 2\1)인 경우는 같은 값을 갖는다.
행렬 연산의 경우 a/b, b\a는 서로 다른 결과를 갖는다.

6. 행렬의 사칙 연산

① 덧셈 및 뺄셈

- 각 행렬의 같은 위치, 즉 행렬상의 색인(index)이 같은 원소끼리 이루어짐.

$x=\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$ $y=\begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix}$ 일 때

$x+y=\begin{bmatrix} x_{11}+y_{11} & x_{12}+y_{12} \\ x_{21}+y_{21} & x_{22}+y_{22} \end{bmatrix}$ 가 된다.

```
》 a=[1 2 3; 4 5 6];
》 b=[2 4 6; 1 3 5];
》 c=a+b
   c=
      3 6 9
      5 8 11
》 d=c-5
   d=
     -2     1     4
      0     3     6
```

② 곱셈

- '*'를 사용하여 표시, 행렬의 내부 차원(inner dimension)이 일치할 경우에만 정의.

```
》 x=[1 2 3];
》 y=[3 4 5];
》 x*y'
   ans=
        26
》 z=x'*y
   z=
```

```
     3     4     5
     6     8    10
     9    12    15
》5*z
ans=
    15    20    25
    30    40    50
    45    60    75
```

③ 나눗셈

A*X=B --> X=inv(A)*B ---> X=A\B (left division)
X*A=B --> X=B*inv(A) ---> X=A/B (right division)

- x+2y+3z=5
  4x+5y+6z=8
  7x+8y=7 일 때

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \\ 7 \end{bmatrix} \Rightarrow x,y,z = ?$$

```
》A=[1 2 3; 4 5 6; 7 8 0];
》b=[5 8 7]';
》X=A\b
X =
   -2.5556
    3.1111
    0.4444
```

④ 행렬의 거듭제곱

```
》a=[1 2; 3 4]
a =
     1     2
     3     4
》a^2
ans =
     7    10
    15    22
》a.^2
ans =
     1     4
     9    16
```

7. 일반적인 명령어

① help : MATLAB의 path에서 디렉토리를 설명
help log : log 함수를 설명

② lookfor log : MATLAB path에 있는 모든 M파일의 첫 번째 코멘트라인에서 log를 찾아 설명한다.

③ type filename : 특정한 파일의 내용을 보여줌.

④ what : 현재의 디렉토리에서 파일을 리스트
what dirname : 검색 경로에서 dirname의 디렉토리에 있는 파일을 리스트한다.

⑤ which funname : 전체 검색 경로에 있는 funname 함수를 화면에 보여준다.

⑥ clear : 작업공간의 모든 변수들을 지운다.
clear x y z : 작업공간의 변수 x, y, z를 지운다.

⑦ length : 벡터의 길이
》x=[2 3 4] ;
》length(x)
ans =3
》max (size(x))
ans =3

⑧ save : 모든 변수를 MATLAB.mat 에 binary 형태로 저장
save filename :모든 변수를 filename에 binary 형태로 저장 (확장자는 생략 default --> .mat)
save filename x y z : 변수 x, y, z를 filename에 binary 형태로 저장
save filename x y z -ascii : 변수 x, y, z를 filename에 ascii 형태로 저장

⑨ load : 변수가 저장된 MATLAB.mat 파일을 로딩
load filename : 변수가 저장된 filename을 로딩

⑩ size : m x n 행렬의 차원

》x=[1 2 3 ; 4 5 6];
》d=size(x)
d=2 3
》[m,n] = size(x)
m=2
n=3

⑪ format : 출력의 형식을 제어하는 명령

명 령	설명과 예
format short	소수점 이하 4 자리수(기본) : pi=3.1416
format long	소수점 이하 14 자리수 : pi=3.14159265358979
format short e	소수점 아래 4 자리수 : pi=3.1416e+000
format long e	소수점 이하 15 자리수 : pi=3.141592653589793e+000
format bank	소수점 아래 2 자리수 : pi=3.14
format +	양수: +, 음수: -, 0: (blank)
format rat	소수점 대신에 '/'를 사용하여 분수 형태로 표시

⑫ more on : 화면에 한 페이지씩 출력 (space bar-한페이지씩, enter key-한줄씩)
more off : 화면에 출력되는 페이지를 제어하지 않는다.

⑬ computer : MATLAB을 실행하고 있는 컴퓨터의 종류를 분별하여 알려준다.

```
》computer
   ans = PCWIN
```

⑭ clock : 현재의 날짜와 시간을 십진수 형태로 6개의 열 벡터에 나타낸다.

```
》clock
    ans =
      1.0e+003 *
         2.0000    0.0070    0.0040    0.0100    0.0480    0.0133
》fix(clock)
   ans =
        2000         7         4        10        49        11
```

8. 그 밖의 행렬 함수

- roots : 고차 방정식의 근을 구함

```
》a=[1 3 2];
》b=roots(a)
   b =
        -2
        -1
```

- poly : 특성 다항식

```
》poly(b)
```

ans =

1 3 2

- det : 행렬식(determinant)

- trace : 대각합(trace)

》 b=[1 2 3;4 5 6;4 5 6]

b =

1 2 3

4 5 6

4 5 6

》 c=trace(b)

c=

12

》 d=[b(1:2, :); 3 4 5; 3 2 1]

d =

1 2 3

4 5 6

3 4 5

3 2 1

》 trace(d)

ans=

11

9. 수학 함수

① 》 abs(-2);

》 abs(3+4i);

》 abs('Q') : Ascii문자의 값을 출력

ans = 81

② 》 exp(5)

ans =

148.4132

③ sqrt(x) : x의 각원소에 대한 제곱근을 계산

음수나 복소수에 대해서는 복소수의 결과를 제공

행렬의 제곱근에는 sqrtm(x)를 사용

》 sqrt((-2:2))

ans =

0 + 1.4142i 0 + 1.0000i 0 1.0000 1.4142

④ log(x) : Natural logarithm
log10(x) : Common(base 10) logarithm

⑤ ceil(x) : x보다 큰 가장 가까운 정수 값을 나타낸다.
fix(x) : 분수 부분을 잘라 버린 정수 값을 갖는다.
floor(x) : x보다 작거나 같은 정수 값을 취한다.
round(x) : x에 가장 가까운 정수 값을 얻는다.

```
》 x=[-1.8 -0.3 3.3 5.7 7.0];
》 ceil(x)
      ans =
          -1     0     4     6     7
》 fix(x)
      ans =
          -1     0     3     5     7
》 floor(x)
      ans =
          -2    -1     3     5     7
》 round(x)
      ans =
          -2     0     3     6     7
```

⑥ real, imag, conj : 복소수의 실수부, 허수부, conjugate를 얻는다.

```
》 x=[2+3i ; 4-5i] ;
》 real(x)
》 imag(x)
》 conj(x)
```

angle : 위상각을 계산하는 함수로, 복소수 행렬의 위상각을 radian으로 계산하고 이 각은 -π에서 π사이에 놓여 있다.

```
》 z=3+4i ;
》 angle(z)
    ans = 0.9273 (53.1°)
```

$M\angle\Theta = Me^{j\Theta} = a+bj$

$M = \sqrt{(a^2+b^2)}$, $\Theta = \tan^{-1}(b/a)$, $a = M\cos(\Theta)$, $b = M\sin(\Theta)$

```
》 c=1-2i
》 mag_c=abs(c);
》 angle_c=angle(c);
》 deg_c=angle_c*180/pi;
》 real_c=real(c);
》 imag_c=imag(c);
```

⑦ rem(x,y) : 나머지를 계산하는 함수 rem(x,y)=x-fix(x./y)*y

```
》x=[5 6 ; 7 8] ; y=[2 3 ; 4 5] ;
》rem(x,y)
  ans = 1 0
        3 3
```

sign(x) : x의 각 원소가 0보다 크면 1, 같으면 0, 작으면 -1을 원소로 하는 mxn의 실수 행렬을 만든다.

```
》x=[ 1 -1 0; 0 1 2];
》sign(x)
   ans = 1 -1  0
         0  1 -1
```

⑧ 삼각함수 : 모든 각은 radian단위이다.

sin(x), cos(x), tan(x), sinh(x), cosh(x), tanh(x), asin(x), acos(x), atan(x), asinh(x), scosh(x), atanh(x)

10. 벡터와 행렬의 조작법

① 벡터의 생성

- "시작값:증분값:최종값" 형식으로 지정
  "시작값:최종값" 형태로 입력시 증분값은 1로 지정

- 콜론(:)을 사용하여 벡터를 생성하면 기본적으로 행벡터 생성

```
》x=1:5
   x=1 2 3 4 5
》y=0:0.5:3
    y =
         0    0.5000    1.0000    1.5000    2.0000    2.5000    3.0000
》z=5:-1:-5
    z =
    5    4    3    2    1    0   -1   -2   -3   -4   -5
》x=(0:.2:2)';
》y=sin(x).*exp(-x);
》[x,y]
    ans =
         0         0
    0.2000    0.1627
    0.4000    0.2610
```

```
    0.6000    0.3099
    0.8000    0.3223
    1.0000    0.3096
    1.2000    0.2807
    1.4000    0.2430
    1.6000    0.2018
    1.8000    0.1610
    2.0000    0.1231
```

- 대수적으로 등간격인 벡터 생성

```
》linspace(-5,5,3)
     ans =
         -5     0     5
```

② 첨자 붙이기

- MATLAB에서는 첨자(subscript)를 사용함으로써 행렬의 각 원소들을 표시 할 수 있다.

```
》a=[0 1 2;4 5 6 ;8 9 10]
     a =
          0     1     2
          4     5     6
          8     9    10
》a(1,1)=a(2,2)-a(2,3)
     a =
         -1     1     2
          4     5     6
          8     9    10
》a(1.25*2 , 6/2.5)              * 1.25*2 = 2.5
     ans =                         6/2.5 =2.4
          9
》a=magic(10)
     a =
    92    99     1     8    15    67    74    51    58    40
    98    80     7    14    16    73    55    57    64    41
     4    81    88    20    22    54    56    63    70    47
    85    87    19    21     3    60    62    69    71    28
    86    93    25     2     9    61    68    75    52    34
    17    24    76    83    90    42    49    26    33    65
    23     5    82    89    91    48    30    32    39    66
    79     6    13    95    97    29    31    38    45    72
    10    12    94    96    78    35    37    44    46    53
    11    18   100    77    84    36    43    50    27    59
```

```
》 a(1:5,5)
》 a(1:5,5:7)
》 a(:,5)
》 a(1:2,:)
》 b=[1 2 3; 4 5 6; 7 8 9]
    b =
         1     2     3
         4     5     6
         7     8     9
》 c=[9 8 7;6 5 4; 3 2 1]
    c =
         9     8     7
         6     5     4
         3     2     1
》 c(:,[1 3])=b(:,2:3)  또는 c(:,[1 3])=b(:,[2 3])
    c =
         2     8     3
         5     5     6
         8     2     9
》 c(3:-1:1,:)      ≪=== c의 3부터 1까지의 행들을 역순으로 배열
    ans =
         8     2     9
         5     5     6
         2     8     3
》 a=[1 2; 3 4]
    a =
         1     2
         3     4
》 b=a(:)   <<===  열벡터로 구성(vectorize)
    b =
         1
         2
         3
         4
》 a(:)=10:13 <<=== reshape
    a =
        10    12
        11    13
```

명령어	기 능
fliplr	행렬의 왼쪽과 오른쪽 열을 바꾸어 준다.
flipud	행령의 위쪽과 아래쪽 방향을 바꾸어 준다.
rot90(A)	행렬 A를 반시계 방향을 90°회전시킨다.
rot90(A,k)	행렬 A를 반시계 방향으로 k*90°회전시킨다.
reshape(A,m,n)	행렬 A를 각 열에서 원소를 취하여 m×n 행렬을 만든다.

```
》A=[1 2 3; 4 5 6; 7 8 9]
  A=
      1   2   3
      4   5   6
      7   8   9
》fliplr(A)
   ans =
      3   2   1
      6   5   4
      9   8   7
》flipud(A)
   ans =
      7   8   9
      4   5   6
      1   2   3
》rot90(A)
  ans =
      3   6   9
      2   5   8
      1   4   7
》B=[1 4; 2 5; 3 6]
   B =
      1   4
      2   5
      3   6
》reshape(B,2,3)
    ans =
        1    3    5
        2    4    6
```

11. 기본적인 행렬들

① ones(n) : 모든 원소가 1인 n x n 행렬을 만든다.
ones(m,n) : 모든 원소가 1인 m x n 행렬을 만든다.
ones(size(A)) : 행렬 A와 같은 크기를 가지고 있고, 모든 원소가 1인 행렬을 만든다.

② zeros(n) : 모든 원소가 0인 n x n 행렬을 만든다.
zeros(m,n) : 모든 원소가 0인 m x n 행렬을 만든다.
zeros(size(A)) : 행렬 A와 크기가 같고, 모든 원소가 0인 행렬을 만든다.

③ eye(n) : n x n의 단위행렬을 만든다.
eye(m,n) : m x n의 단위행렬을 만든다.
eye(size(A)) : 행렬 A와 크기가 같은 단위행렬을 만든다.

④ linspace(x1,x2) : x1과 x2 사이에 구간을 선형적으로 똑같이 등분하여 100개의 원소를 가진 벡터를 생성한다.
linspace(x1,x2,n) : x1과 x2 사이에 n개의 원소를 가지도록 벡터를 만든다.

⑤ logspace(a,b) : 10^a와 10^b의 구간 내를 대수함수적으로 똑같이 50 등분하여 열 벡터를 생성한다.

```
》logspace(2,4,3)
   ans = 100 1000 10000
```

⑥ rand : 0과 1사이의 임의의 숫자와 균일하게 분포한 원소를 가지는 행렬을 생성
randn : 정규분포, 즉 평균이 0이고 편차가 1인 Gaussian 분포를 가지는 행렬을 만든다.
》rand(2,3) : 임의의 숫자를 원소로 하는 2x3 행렬을 생성

12. 계산 속도

```
》format
》d=pi;
》d*ones(3,4)    ①
》d+zeros(3,4)   ②
》d(ones(3,4))   ③
》repmat(d,3,4) ④        (①＜②＜③＜④, more fast)
》tic; plot(rand(5)); toc
    elapsed_time =
              0.8220
```

13. 배열연산

배열 연산(array operation)이란 *, \, /, ^ 또는 ‘ 기호에 의하여 이루어지는 일반적인 선형 행렬 연산(linear algerbraic matrix operation)이 아닌 행렬의 원소와 원소 사이에 성립되는 산술 연산(element-by-element operation)을 가리킨다. 연산자 앞에 점(.)을 찍음으로써 배열 연산 또는 원소 대 원소의 연산을 나타낸다.

1) 배열의 사칙 연산

① 배열연산의 덧셈 및 뺄셈

- 일반적인 행렬 연산(matrix operation)과 동일

② 배열연산의 곱셈 및 나눗셈

- 곱셈(.*) : 배열연산의 곱셈을 나타내며, 두 행렬의 차원이 같은 때 각 행렬의 같은 위치에 있는 원소들끼리 곱셈을 하여 새로운 행렬을 구성한다.

```
》a=[1 2 3; 4 5 6];
》b=[1 3 6; 7 8 9];
》a.*b
   ans =
         1     6    18
        28    40    54
```

- 나눗셈 : './', '.\',

```
》a./b
   ans =
    1.0000    0.6667    0.5000
    0.5714    0.6250    0.6667
》a.\b
   ans =
    1.0000    1.5000    2.0000
    1.7500    1.6000    1.5000
```

③ 배열을 사용한 거듭제곱 : '.^'

```
》x=[1 2 3];
》y=[4 5 6];
》x.^y
    ans =
      1    32   729
》y.^2
    ans =
     16    25    36
```

》 2.^[x ; y]

ans =

2 4 8

16 32 64

2) 기타의 연산

① 관계연산

- 관계 연산자

. 두 행렬의 대응 원소들을 비교하여 그 결과를 0과 1로 구성된 행렬로 나타냄

. '0' : 거짓(false)

'1' : 참(true)

MATLAB에서 사용되는 관계연산들

연산자	의 미
<	~ 미만(less than)
<=	~ 이하(less than or equal)
>	~ 초과(greater than)
>=	~ 이상(greater than or equal)
==	~ 와 같음(equal)
~=	~ 와 같지 않음(not equal)

》 a=magic(3)

a=

8 1 6

3 5 7

4 9 2

》 a>=5

ans =

1 0 1

0 1 1

0 1 0

a(1)=a(1,1) a(2)=a(2,1) a(3)=a(3,1)

a(4)=a(1,2) a(5)=a(2,2) a(6)=a(3,2)

a(7)=a(1,3) a(8)=a(2,3) a(9)=a(3,3)

- find : 원하는 조건을 만족하는 원소들을 찾아 주는 함수

》 k=find(a>=3)

k = 1

1

2

3

```
            5
            6
            7
            8
》a(k)=5
      a =
            5     1     5
            5     5     5
            5     5     2
```

② 논리 연산(Logical operation)

'&' - and
'|' - or
'~' - not

```
》x=[7 -2 6 3 5 10 8 7];
》(x>5) & (x<9)
   ans =
      1 0 1 0 0 0 1 1
》~(x>5)
   ans =
      0 1 0 1 1 0 0 0
```

- any(x) : 벡터 x의 원소들 중 하나라도 0이 아닌 원소가 있으면 1을 돌려주고 모두 0일 때에는 0을 돌려준다.

```
》x=[0 1 2 3];
》any(x)
   ans =
         1
```

- all(x) : 벡터 x의 원소들이 모두 0이 아니어야만 1을, 그렇지 않고 하나라도 0인 원소가 있으면 0을 돌려준다.

```
》all(x)
   ans =
         0
```

- xor(x)

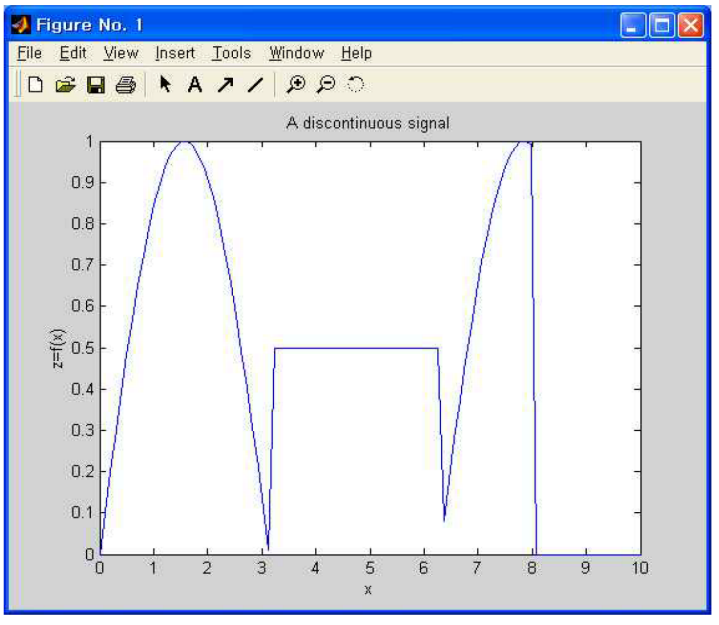
```
》a=[1 2 3]; b=[0 1 5];
》xor(a,b)
   ans = [1 0 0]
```

MATLAB에서 사용되는 관계 및 논리 함수들

함 수	기 능
any	논리적 조건
all	논리적 조건
find	논리값의 배열 지표를 찾아줌
exist	변수의 존재 여부를 점검
isnan	NaN의 존재 검색
isinf	무한대 값의 존재 검색
finite	유한한 값의 존재 검색
isempty	빈 행렬의 검색
isstr	문자열 변수의 검색
isglobal	전역변수(gloval variable)의 검색
issparse	sparse 행렬의 검색

- Discontinuous signal

```
》 x=linspace(0,10,100);
》 y=sin(x);
》 z=(y>0).*y;
》 z=z+0.5*(y<0);
》 z=(x<=8).*z;
》 plot(x,z)
》 xlabel('x'), ylabel('z=f(x)');
》 title('A discontinuous signal');
```



14. 집합과 진수 변수 함수

1) 집합함수

- isequal : 행렬의 차원과 원소가 모두 같으면 '1'을 출력, 그렇지 않으면 '0'을 출력

```
》a=rand(2,5);
》b=rand(2,5);
》isequal(a,b)
   ans =
         0
》isequal(a,a)
   ans =
         1
》a=[2:2:8,4:2:10]
   a=2   4   6   8
     4   6   8   10
```

- unique : 중복되는 원소를 제외한, 유일하게 존재하는 원소들을 오름차순으로 출력

```
》unique(a)
   ans =
     2
     4
     6
     8
    10
```

- ismember(x,y) : x의 원소가 집합 y에 포함되면 '1', 그렇지 않으면 '0'을 출력

```
》a=1:9;
》b=2:2:9;
》ismember(a,b)
   ans =
         0   1   0   1   0   1   0   1   0
》ismember(b,a)
   ans =
         1   1   1   1
》A=eye(3);
》B=ones(3);
》ismember(A,B);
》ismember(B,A);
```

- union(합집합)

```
》a=1:9;
》b=[2 4 6 8 10];
》union(a,b)
  ans =  1 2 3 4 5 6 7 8 9 10
```

- intersect(a,b)

```
》intersect(a,b)
  ans = 2 4 6 8
```

- setdiff(x,y) : x원소 중에서 y에 포함되지 않는 원소

```
》setdiff(a,b)
   ans =
         1     3     5     7     9
》setdiff(b,a)
   ans =
        10
```

2) 진수변환

- dec2bin(x) : 10진수 x를 2진수로 변환, bin2dec(y) : 2진수 y를 10진수로 변환

```
》a=dec2bin(17)
   a = 10001
》class(a)
   ans = char
》bin2dec(a)
   ans = 17
》class(ans)
  ans = double
```

- dec2hex(x) : 2진수를 16진수로 변환, hex2dec(y) : 16진수를 2진수로 변환

```
》a=dec2hex(2047)
   a = 7FF
》class(a)
   ans = char
》hex2dec(a)
   ans = 2047
```

```
》class(ans)
   ans = double
```

- dec2base(x,y) : 십진수 x를 y진수으로 변환, base2dec(x,y) : y진수 x를 10진수로 변환

```
》a=dec2base(26,3)
   a = 222
》class(a)
   ans = char
》base2dec(a,3)
   ans = 26
》class(ans)
   ans = double
```

15. 문자열(character strings)

```
》t='How about this character string?'
   t =
       How about this character string?
》size(t)
   ans = 1   32
》whos
   Name        Size              Bytes  Class
    ans        1x2                  16  double array
    t         1x32                  64  char array
   Grand total is 34 elements using 80 bytes
》u=double(t)
   u =
     Columns 1 through 12
      72   111   119    32    97    98   111   117   116    32   116   104
     Columns 13 through 24
     105   115    32    99   104    97   114    97    99   116   101   114
     Columns 25 through 32
      32   115   116   114   105   110   103    63
》char(u)
   ans =
       How about this character string?
》a=double('a')
   a = 97
》char(a)
   ans = a
```

》 u=t(16:24)

u = character

》 u=t(24:-1:16)

u = retcarahc

- int2str : 정수값을 string으로 변환

》 int2str(eye(3))

```
ans = 1   0   0
      0   1   0
      0   0   1
```

》 size(ans)

ans = 3 7

》 num2str(rand(2,4));

》 size(ans)

ans = 2 43

》 b='Peter';

》 findstr(b,'t')

ans = 3

》 findstr(b,'e')

ans = 2 4

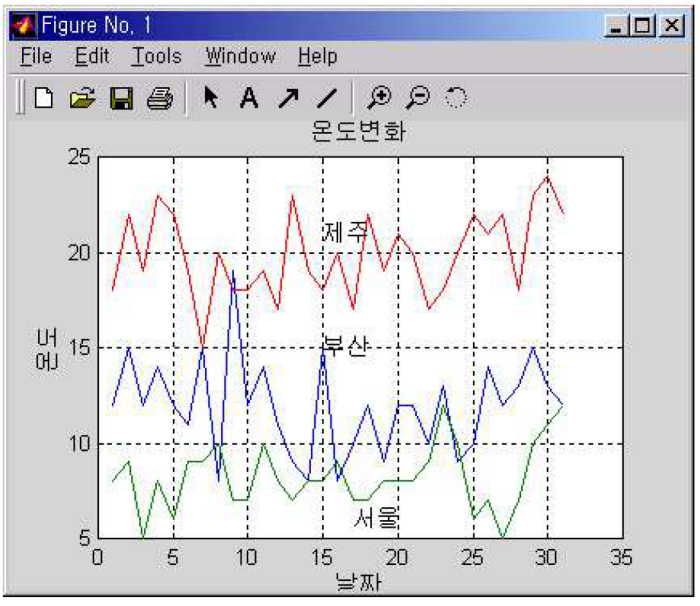
16. Data Analysis

temps.dat

```
12     8    18
15     9    22
12     5    19
14     8    23
12     6    22
11     9    19
15     9    15
 8    10    20
19     7    18
12     7    18
14    10    19
11     8    17
 9     7    23
 8     8    19
15     8    18
 8     9    20
10     7    17
12     7    22
 9     8    19
12     8    21
12     8    20
10     9    17
13    12    18
 9    10    20
10     6    22
14     7    21
12     5    22
13     7    18
```

```
    15     10     23
    13     11     24
    12     12     22
```

```
》load temps.dat
》day=1:31;
》plot(day,temps);
》xlabel('날짜'), ylabel('온도'),
》title('온도변화');
```




```
》avg_temp=mean(temps);          %각 도시의 온도 평균을 구한다.
    avg_temp =
      11.9677    8.2258   19.8710
》avg_avg=mean(avg_temp)         %전체 온도 평균을 구한다.
    avg_avg =
      13.3548
① 》for c=1:3                    %온도차를 구한다.
    tdev(:,c)=temps(:,c)-avg_temp(c);
    end
② 》tdev=temps-avg_temp(ones(31,1),:)
③ 》tdev=temps-repmat(avg_temp,31,1)
》max_temp=max(temps)            %각 도시의 최대 온도를 구한다.
》[max_temp, maxday]=max(temps)
    max_temp =
          19    12    24
    maxday =
           9    23    30
》min_temp=min(temps)            %각 도시의 최소 온도를 구한다.
```

```
》[min_temp,minday]=min(temps)
    min_temp =
          8     5    15
    minday =
          8     3     7
》s_dev=std(temps)              %각 도시의 표준편차를 구한다.
    s_dev =
       2.5098    1.7646    2.2322
》median(temps)                 %중간값을 표시한다.
  ans =
     12    8   20
》dail_change=diff(temps)       %매일매일의 온도차를 구한다.
```

17. 다항식의 계산

1) 근(roots)

- $x^4 - 12x^3 + 25x + 116$의 근을 구하고자 한다.

```
》p=[1 -12 0 25 116];
》r=roots(p)
    r =
      11.7473
       2.7028
      -1.2251 + 1.4672i
      -1.2251 - 1.4672i
》pp=poly(r)
    pp =
       1.0000   -12.0000    -0.0000    25.0000   116.0000
```

2) 다항식의 곱

- $a(x) = x^3 + 2x^2 + 3x + 4$, $b(x) = x^3 + 4x^2 + 9x + 16$

```
》a=[1 2 3 4]; b=[1 4 9 16];
》c=conv(a,b)
    c =
       1     6    20    50    75    84    64
```

따라서 두 다항식의 곱은 $c(x) = x^6 + 5x^5 + 20x^4 + 50x^3 + 75x^2 + 84x + 64$가 된다.

3) 다항식의 합

- $a(x)=x^3+2x^2+3x+4, \quad b(x)=x^3+4x^2+9x+16$

```
》 a=[1 2 3 4]; b=[1 4 9 16];
》 d=a+b
     d =
       2   6   12   20
```

따라서 두 다항식의 합은 $d(x)=2x^3+6x^2+12x+20$이 된다.

- 차수가 다른 다항식의 합

```
》 e=c+[0 0 0 d]
    e =
      1   6   20   52   81   96   84
```

$e(x)=x^6+5x^5+20x^4+52x^3+81x^2+96x+84$가 된다.

4) 다항식의 나눗셈

```
》 [q,r]=deconv(c,b)
     q =      ← quotient polynomial
       1       2       3       4
     r =      ← remainder
       0       0       0       0       0       0       0
```

5) 다항식의 미분

- $g(x)=x^6+6x^5+20x^4+48x^3+69x^2+72x+44$라고 하면

$g'(x)=6x^5+30x^4+80x^3+144x^2+138x+72$가 된다.

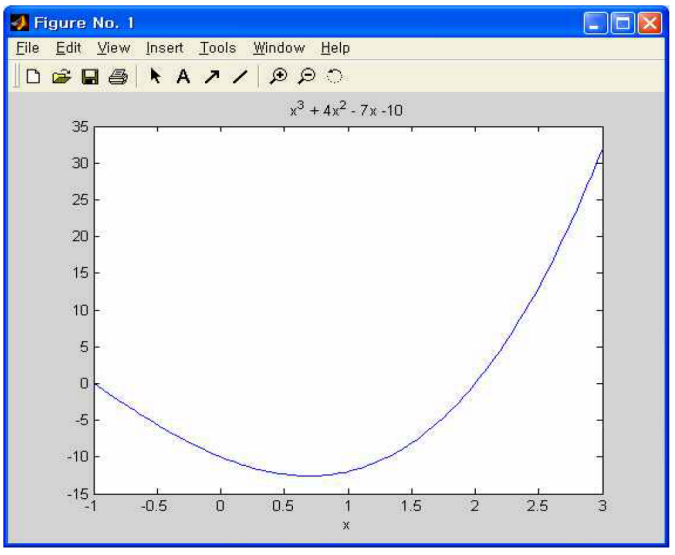
```
》 g=[1 6 20 48 69 72 44];
》 h=polyder(g)
    h =
      6      30      80     144     138      72
```

6) evaluation

```
》 x=linspace(-1,3);
》 p=[1 4 -7 -10]
```

$p(x)=x^3+4x^2-7x-10$

```
》 v=polyval(p,x);
》 plot(x,v),title('x{^3} + 4x{^2} - 7x  - 10'), xlabel('x')
```



7) Rational Polynomials

$$\frac{n(x)}{d(x)} = \frac{N_1 x^m + N_2 x^{m-1} + \ldots + N_{m+1}}{D_1 x^n + D_2 x^{n-1} + \ldots + D_{n+1}}$$

```
》n=[1 -10 100];    % 분자
》d=[1 10 100 0];   % 분모
》z=roots(n)        % Zero of n(x)/d(x)
   z =
     5.0000 + 8.6603i
     5.0000 - 8.6603i
》p=roots(d)        % Pole of n(x)/d(x)
   p =
          0
    -5.0000 + 8.6603i
    -5.0000 - 8.6603i
》[nd,dd]=polyder(n,d)
   nd =
        -1    20   -100   -2000   -10000
   dd =
         1    20   300    2000    10000     0     0
```

- [R,P,K] = residue(B,A)

$$\frac{B(s)}{A(s)} = \frac{R(1)}{s - P(1)} + \frac{R(2)}{s - P(2)} + \ldots. + K(s)$$

```
》[r,p,k]=residue(n,d)
   r =
     0.0000 + 1.1547i
     0.0000 - 1.1547i
```

```
        1.0000
    p =
     -5.0000 + 8.6603i
     -5.0000 - 8.6603i
           0
    k =
        []
```

$$\therefore \quad \frac{n(x)}{d(x)} = \frac{1.1547i}{x+5-8.6603i} + \frac{-1.1547i}{x+5+8.8803i} + \frac{1}{x}$$

```
》[nn,dd]=residue(r,p,k)
    nn =
       1.0000   -10.0000   100.0000
    dd =
       1.0000    10.0000   100.0000         0
```

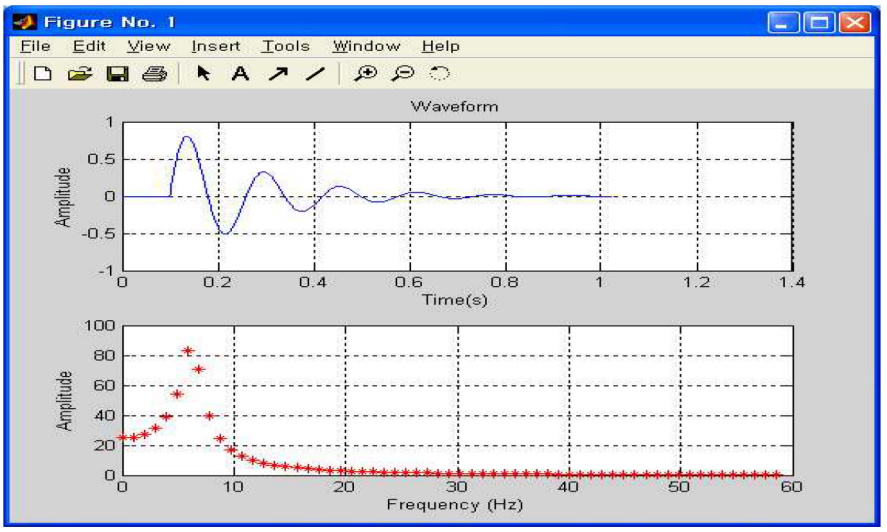
18. 퓨리에 변환(fft)

주어진 x(t)가 다음과 같을 때

$$x(t) = 0 \qquad ; t < 0.1s$$

$$= e^{\frac{-(t-0.1)}{0.176}} \sin\left[\frac{2\pi(t-0.1)}{0.160}\right] \qquad ; 0.1 \le t < 1.024s$$

```
t=0:0.001:1.023;
T=0.001;
k=101:1024;
z=zeros(1,100);
x=[z exp(-(k*T-0.1)/0.176).*sin(2*pi*(k*T-0.1)/0.160)];
subplot(2,1,1)
plot(t,x); grid on;
xlabel('Time(s)')
ylabel('Amplitude');
title('Waveform')
X=fft(x)
amp_spect=sqrt(X.*conj(X))
freq=(0:60)/(1024*0.001)
subplot(2,1,2);
plot(freq,amp_spect(1:61),'r*'); grid on
xlabel('Frequency (Hz)')
ylabel('Amplitude')
```

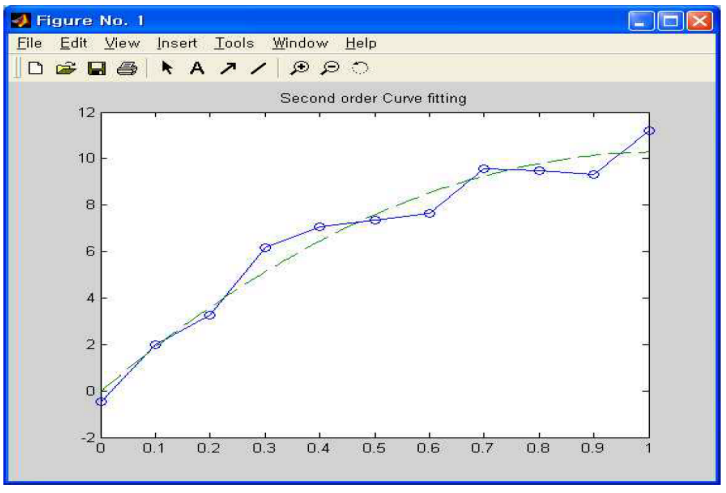


19. Curve fitting

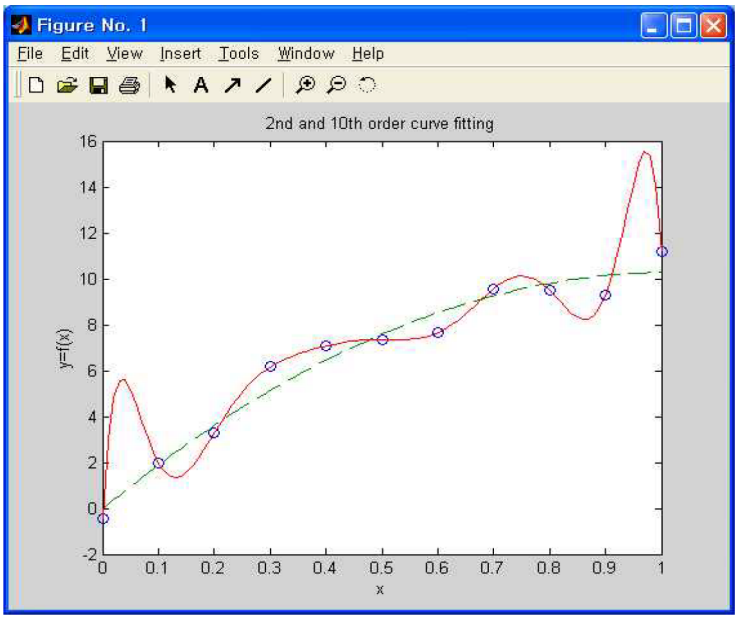
```
》x=[0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1];
》y=[-.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30 11.2];
》n=2;
》p=polyfit(x,y,n)
    p =
     -9.8108    20.1293    -0.0317
```

$\therefore \quad y = -9.8108x^2 + 20.1293x - 0.0317$

```
》xi=linspace(0,1,100);
》yi=polyval(p,xi);
》plot(x,y,'-o',xi,yi,'--');
》title('Second order Curve fitting')
```



```
》 pp=polyfit(x,y,10)
》 format short e
》 pp.'
    ans =
   -4.6436e+005
    2.2965e+006
   -4.8773e+006
    5.8233e+006
   -4.2948e+006
    2.0211e+006
   -6.0322e+005
    1.0896e+005
   -1.0626e+004
    4.3599e+002
   -4.4700e-001
》 y10=polyval(pp,xi);
》 plot(x,y,'o',xi,yi,'--',xi,y10)
》 xlabel('x'),ylabel('y=f(x)')
》 title('2nd and 10th order curve fitting')
```



20. 최소자승법(Least Square Method)

회귀분석(regression analysis)을 위한 MATLAB 함수중에는 polyfit 명령어가 있는데, polyfit는 최소자승법에 근거를 하고 있다. 아래의 데이터에 가장 근접한 직선은 y=mx+b의 형태가 된다.

x	y
0	2
5	6
10	11

최소자승의 이론에 따르면, 보간된 직선과 데이터 점들 사이에서 수직의 차이값을 제곱하여 합한 J값이 최소화되는 가장 근접한 직선을 긋는다. 이들 차이값들은 나머지오차(residues)라고 부른다. 위의 데이터에서는 3개의 데이터 점이 존재하며, 여기서 J값은 다음의 식에 의해 구할 수 있다.

$$J = \sum_{i=1}^{3} (mx_i + b - y_i)^2$$

표에서 주어진 자료값(x_i, y_i)을 대입하여 다음의 식을 얻을 수 있다.

$$J = (0*m + b - 2)^2 + (5*m + b - 6)^2 + (10*m + b - 11)^2$$

여기서, J가 최소값을 갖는 m, b의 값은 $\partial J/\partial m = 0, \partial J/\partial b = 0$의 조건으로부터 계산한다. 이 조건들에 의해서 아래의 두 식으로부터 미지수 m, b를 구할 수 있다.

$$250m+30b=280, \quad 30m+6b=38$$

그 결과는 m=0.9, b=11/6이다. 따라서, 최소자승법에 의한 최적의 직선을 y=0.9x+11/6이 된다. 이 방정식에 x 데이터 0, 5, 10을 입력하면 y의 값은 각각 1.833, 6.333 10.8333이 된다. 따라서 표의 y의 값(y=2, 6, 11)과는 차이가 있는데 이것은 직선의 데이터가 완전하게 근사화(fix)되지 않았기 때문이다. J의 값을 구하면 아래와 같다.

$$J = (1.833 - 2)^2 + (6.333 - 6)^2 + (10.8333 - 11)^2 = 0.16656689$$

다음은 주어진 데이터에 대한 1차에서 4차까지의 다항식을 구하며, 각 다항식의 J값을 계산한다.

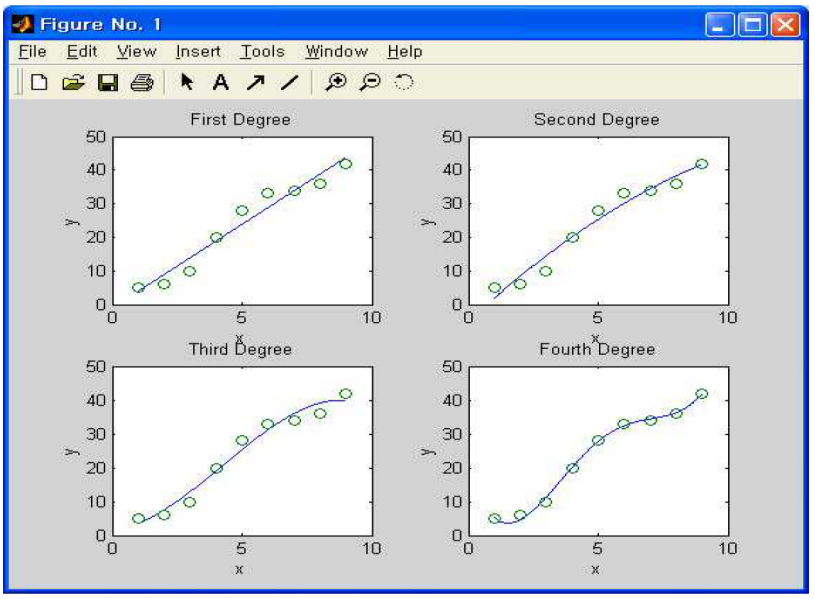
x	1	2	3	4	5	6	7	8	9
y	5	6	10	20	28	33	34	36	42

<curvefit_test.m>

```
close all, clear all
x=[1:9];
y=[5,6,10,20,28,33,34,36,42];
xp=[1:0.01:9];
for k=1:4
    yp(k,:)=polyval(polyfit(x,y,k),xp);
    J(k)=sum((polyval(polyfit(x,y,k),x)-y).^2);
end
subplot(2,2,1)
plot(xp,yp(1,:),x,y,'o'),axis([0 10 0 50])
xlabel('x'),ylabel('y'),title('First Degree')
subplot(2,2,2)
plot(xp,yp(2,:),x,y,'o'),axis([0 10 0 50])
xlabel('x'),ylabel('y'),title('Second Degree')
subplot(2,2,3)
plot(xp,yp(3,:),x,y,'o'),axis([0 10 0 50])
xlabel('x'),ylabel('y'),title('Third Degree')
subplot(2,2,4)
plot(xp,yp(4,:),x,y,'o'),axis([0 10 0 50])
xlabel('x'),ylabel('y'),title('Fourth Degree')
disp(J)
```

》 curvefit_test

```
J =
    71.5389    56.6727    41.8838     4.6566
```



ex) 다음 주어진 데이터에 대하여 1차~4차 다항식을 구하여 도시하고 그때의 계수값과 J값을 확인하시오.

x	0	1	2	3	4	5
y	0	1	60	40	41	47

21. 보간법(interpolation)

지정한 복수의 점을 지나는 곡선과, 이들의 점 가까이를 지나는 매끄러운 곡선을 구하는 것을 보간이라 한다.

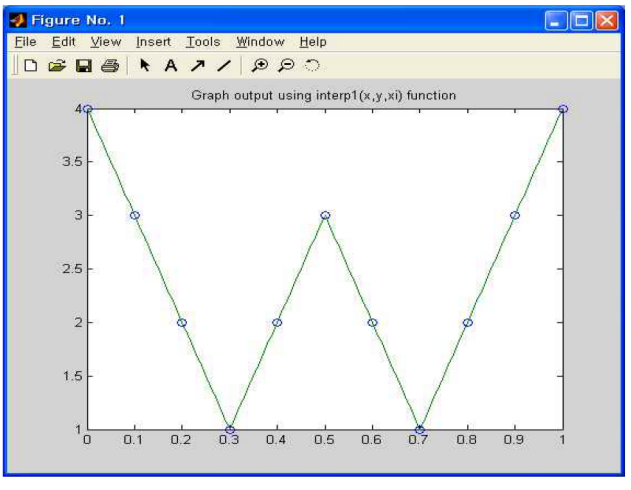
1) 1차원 데이터의 보간

1차원 데이터의 보간으로 사용하는 것으로써 함수 'interp1'명령이 있다. 이 명령어를 사용하는 것으로 다음과 같은 세 가지 방법이 있다.

linear : 선형 보간법
cubic : 3차 곡선 보간법
spline : 3차 곡선 스플라인 보간법

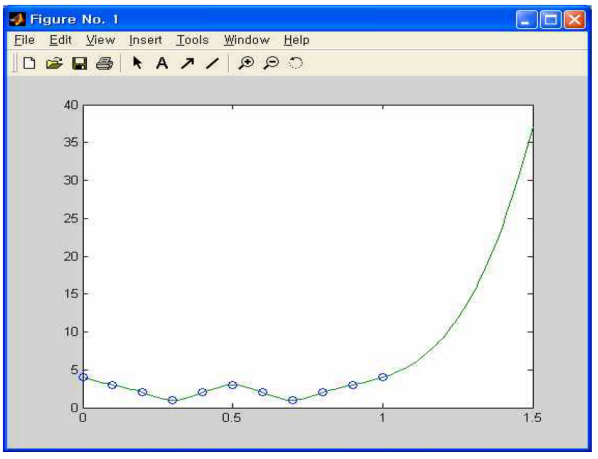
- yi=interp1(x,y,xi) : 벡터 x와 벡터 y에서 보간법에 의해서 결정된 것으로서 벡터 xi에 해당되는 벡터 yi를 되돌린다.

```
》x=0:0.1:1;
》y=[4 3 2 1 2 3 2 1 2 3 4];
》xi=0:0.01:1;
》yi=interp1(x,y,xi);
》plot(x,y,'o',xi,yi)
》title('Graph output using interp1(x,y,xi) function')
```



- yi=interp1(x,y,xi,'method')

```
》 x=0:0.1:1;
》 y=[4 3 2 1 2 3 2 1 2 3 4];
》 xi=0:0.01:1.5;
》 yi=interp1(x,y,xi,'spline');
》 plot(x,y,'o',xi,yi)
```



2) 2차원 데이터의 보간

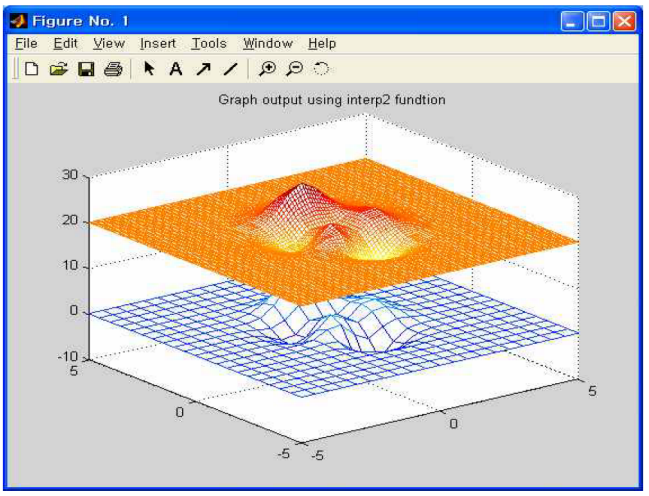
2차원 데이터의 보간으로 사용하는 것으로써 함수 'interp2'명령이 있다. 이 명령어를 사용하는 것으로 다음과 같은 두 가지 방법이 있다.

linear : 선형보간법
cubic : 3차원 곡선 보간법

- zi=interp2(x,y,z,xi,yi)

zi=interp2(x,y,z,xi,yi,'method')

```
》 [x,y]=meshgrid(-5:0.5:5);
》 z=peaks(x,y);
》 [xi,yi]=meshgrid(-5:0.1:5);
》 zi=interp2(x,y,z,xi,yi);
》 mesh(x,y,z)
》 hold on
》 mesh(xi,yi,zi+20)
》 hold off
》 title('Graph output using interp2 fundtion')
```



3) 3차 스플라인 보간

3차 곡선 스플라인의 데이터를 보간한다.

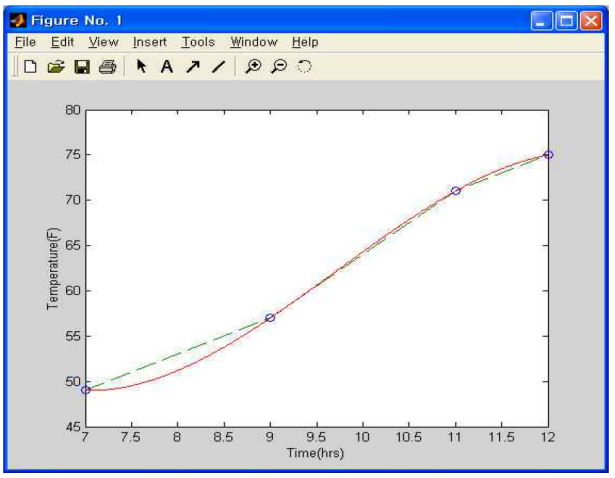
- yi=spline(x,y,xi)

어떤 지역의 시간(t)에 따른 온도(y)를 나타낸 표이다.

x	7	9	11	12
y	49	57	71	75

<spline1.m>

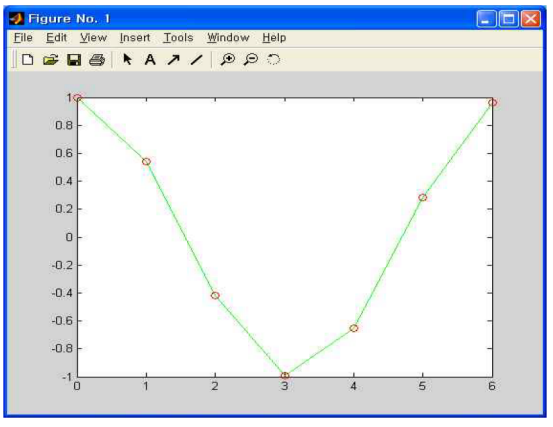
```
x=[7 9 11 12];
y=[49 57 71 75];
x_int=[7:0.01:12];
y_int=spline(x,y,x_int);
plot(x,y,'o',x,y,'--',x_int,y_int);
xlabel('Time(hrs)'), ylabel('Temperature(F)')
axis([7 12 45 80])
```

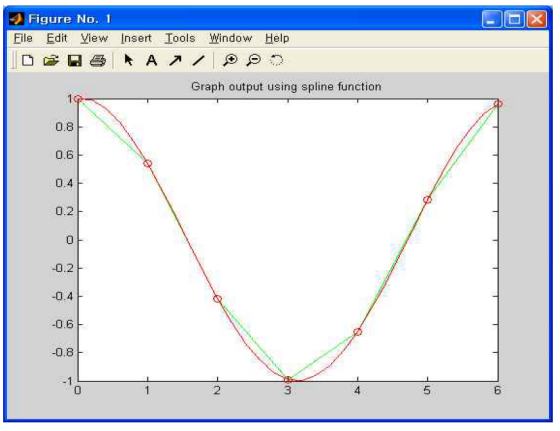


오전 8시의 온도를 스플라인 다항식에 의해서 추정하면 51.2°F이다.(선형보간:53°F)

<spline2.m>

```
x=0:6;
y=[1 0.5403 -0.4161 -0.99 -0.6536 0.2837 0.9602];
xi=0:0.2:6;
yi=spline(x,y,xi);
plot(x,y,'g')
hold on
disp('****** 키보드를 입력하시오 ********')
pause
plot(x,y,'ro')
disp('****** 키보드를 입력하시오 ********')
pause
plot(xi,yi,'r')
hold off
title('Graph output using spline function')
```





제 3 장 파일 관리 기능

MATLAB 명령들 중에서 'save'와 'load'명령은 MATLAB 작업 공간상의 내용을 각각 파일로 저장하고 저장된 파일을 작업공간으로 불러오는 기능을 수행한다. MATLAB은 이 외에 파일을 관리하고 외부 프로그램을 수행할 뿐만 아니라 자료의 입출력 기능을 제공한다.

1. 기본적인 파일 관리 기능

MATLAB 프로그램을 활성화시킨 후에 사용자가 명령어를 이용하여 작업하는 모든 과정을 저장할 수 있다. 즉, 키보드의 입력과 출력에 대한 결과를 전부 저장하고자 한다면 아래와 같이 'diary'명령을 이용하면 된다.

```
》diary file_name
```

위와 같이 하면 이후부터의 모든 키보드의 입력과 출력이 전부 file_name에 저장된다. 또한 더 이상의 저장을 원하지 않을 경우에는 'diary off'를 실행하면 된다.

```
》diary off
```

또한, 'dir', 'type', 'delete' 및 'cd'와 같은 명령들은 일반적인 운영 체제 명령들을 구현한다. 이러한 명령들은 대부분 경로명, 와일드 카드(wildcard; 대표문자) 및 드라이브 지정자(drive designator)를 일반적인 방식으로 사용할 수 있다.

MATLAB	Ms-Dos	Unix	Vax/Vms
dir(또는 ls)	dir	ls	dir
type	type	cat	type
delete	del	rm	delete
cd	chdir(또는 cd)	cd	set default

MATLAB에서는 '!'문자를 이용해서 외부 프로그램을 수행할 수 있다. 느낌표(!)는 '쉘 이탈(shell escape)'을 의미하며, 느낌표 이후에 입력되는 문자들은 운영체제에 대한 명령으로 인식된다. 이러한 기능은 MATLAB을 중지하지 않고 유틸리티를 수행하거나 기타의 다른 작업을 수행하는데 대단히 유용하다.

```
》!copy c:\*.bat d:\
》!dir
》!help
```

2. 내부 파일의 입·출력

MATLAB 프로그램의 계산 결과를 MATLAB의 내부 file로 출력하거나, 이미 file에 들어있는 결과를 입력하기 위한 명령어가 있다. 즉 'save'명령어를 사용하면 현재 메모리 상의 모든 데이터를 저장할 수 있다.

```
》x=[1 3 5 7 9];
》y=1:0.1:2;
》save                          % MATLAB.mat로 저장
  saving to : MATLAB.mat
》save samp1e1                  % sample1.mat로 저장
》save sample2.dat x            % sample2.dat에 변수 x값을 저장
》save sample3.dat y -ascii     % sample3.dat에 변수 y값을 ascii값으로 저장
》save sample4.dat y -ascii -double % sample4.dat에 변수 y값을 double ascii로 저장
》load file_name                % file_name을 작업공간상으로 불러옴
```

'save.m'함수에서 사용되는 옵션은 다음과 같은 것들이 있다.

options	option에 따라 저장되는 형식
-ascii	8-digit Ascii format으로 저장한다.
-ascii -double	16-digit Ascii format으로 저장한다.
-ascii -tabs	tab에 의해서 분리된 데이터를 8-digit Ascii 형식으로 저장한다.
-ascii -double -tabs	tab에 의해서 분리된 데이터를 16-digit Ascii 형식으로 저장한다.

```
》exist('sample1.mat','file')    % 현재 디렉토리에 sample1.mat 파일이 있는가를 검사
  ans =  2
》type sample3.dat;
》whos -file MATLAB.mat          % sample2.dat 파일에 저장된 변수의 정보를 읽어온다.

  Name        Size            Bytes  Class
   x          1x5                40  double array
   y          1x11               88  double array
  Grand total is 16 elements using 128 bytes
```

3. 외부 파일의 입·출력

MATLAB 프로그램의 외부 file 입출력은 다른 형식으로 저장된 자료들을 읽어 들이거나 MATLAB 프로그램에서 생성된 자료들을 다른 프로그램에서 요구하는 형식으로 출력할 수 있도록 해준다.

1) ASCII 데이터 파일의 입·출력

- 외부 파일을 다룰 때는 'fopen'명령어에 의해서 File을 연다.

```
》fid=fopen('data.dat','r')
    fid = 3
```

r : 자료를 읽어 들이는 것을 지정(read)
w : 자료를 쓰는 것을 지정(write)
a : 자료를 추가하는 것을 지정(appending)
r+ : 자료를 읽고 쓰는 것을 지정

file이 열리면 fid는 3이상의 값이 되고, 열리지 않으면 fid는 -1인 값이 된다. 다시 다른 file을 열면 fid의 값은 증가하여 4의 값이 된다. 이와 같이 File을 열 때마다 fid의 값은 증가한다. file을 열어서 작업을 완료한 후에는 'fclose'명령어를 사용하여 fid에 관계한 외부 file을 닫아야 한다. 파일을 닫아 주는데 성공하면 '0'을 돌려주고, 실패하면 '-1'을 반환한다.

```
》status=fclose(fid) : 파일이 하나만 열려 있을 때.
   status = 0
》status=fclose(3)  : 파일이 여러개 열려 있을 때 원하는 fid값을 할당.
   status = 0
》status=fclose('all') : 열려있는 모든 파일을 닫을 때.
   status = 0
```

- 이와 같이 외부 file을 이용하여 데이터를 저장하거나 불러올 경우에 사용하는 명령어는 다음과 같다.

fprintf : ASCII 데이터의 저장에 사용
fscanf : ASCII 데이터를 불러올 경우에 사용
fwrite : Binary 데이터의 저장에 사용
fread : Binary 데이터를 불러올 경우에 사용

① Ascii 데이터 file의 출력

- 아래와 같이 입력하면 지정한 디렉토리에 data.dat 파일이 생성된다.

```
fid=fopen('d:\tool\MATLABr11\work\lecture\data.dat','w')
for i=1:10
```

```
    a(i)=i^2;
    fprintf(fid,'%10d',a(i));
end
fclose(fid);
```

- fid=fopen('file_name','w')

file_name : 사용자가 저장하고자 하는 directory의 명과 file이름을 적어 준다.

- fprintf(fid, 'format', 변수)

fid : 현재 열려있는 파일의 식별자를 되돌린다.
변수 : 저장하고자 하는 변수를 입력
format : %d - 십진수를 저장할 경우
%f - 부동 소수점 값을 저장할 경우
%s - 문자열을 저장할 경우
%e - 지수(exponential)를 저장할 경우
%- - 주어진 field의 왼쪽으로 정렬
%+ - 부호를 표시하게 함.
%0 - 주어진 field의 spaces에 0을 채움

또한 format에 사용되는 escape 문자는 아래와 같다.

escape 문자	escape 문자에 대한 설명
\n	New line
\t	Horizontal tab
\b	Backspace
\r	Carriage return
\f	Form feed
\\	Backslash를 표시
\''	단일 인용 부호를 표시
%%	Percent 문자를 표시

예) %-12.5e
- 주어진 값을 지수형태로 변환하여, 정수부는 12자 이내로 하고, 소수부는 소수점 이하 5자리까지 주어진 field안에서 왼쪽으로 정렬해서 출력하라.

② Ascii 데이터 file의 입력

- 앞에서 저장한 data.dat 파일을 불러오고자 한다.

```
fid=fopen('d:\tool\MATLABr11\work\lecture\data.dat','r')
x=fscanf(fid,'%10d');
fclose(fid);

》x
   x=1
     4
     :
    100
```

- fscanf(fid, 'format')

  format : %d - 십진수를 불러올 경우
  %f - 부동점 소수값을 불러올 경우
  %s - 문자열을 불러올 경우
  %e - 지수(exponential)를 불러올 경우

2) Binary 데이터 파일의 입출력

① Binary 데이터 file의 출력

- Binary 데이터를 저장할 경우의 예로서, 다음과 같이 입력해보자.

```
fid=fopen('d:\tool\MATLABr11\work\lecture\binary.dat','w')
for i=1:10
    a(i)=i^2;
    fwrite(fid,a(i),'int');
end
fclose(fid);
```

- fwrite(fid, 변수, '정밀도')

  char 또는 uchar : 8bit의 부호가 있는 문자와 부호가 없는 문자
  short 또는 ushort : 16bit의 부호가 있는 정수와 부호가 없는 정수
  int 또는 uint : 16bit 또는 32bit의 부호가 있는 정수와 부호가 없는 정수
  long 또는 ulong : 32bit 또는 64bit의 부호가 있는 정수와 부호가 없는 정수
  float : 32bit의 부동 소수값
  double : 64bit의 배정도 부동 소수값

② Binary 데이터 file의 입력

- 위의 프로그램에서 binary 데이터로 저장되어 있는 'binary.dat'를 불러오고자 한다.

```
fid=fopen('d:\tool\MATLABr11\work\lecture\binary.dat','r')
```

```
x=fread(fid,inf,'int');
fclose(fid);
```

- fread(fid, 크기, '정밀도')

  크기 : inf - 데이터의 개수를 알 수 없는 경우에 사용
  5 - 데이터 파일에서 5개의 데이터만 불러온다.

3) 파일 내에서의 위치제어

함수 'fseek' 및 'ftell'은 파일 내에서 다음 번에 나타날 입력 또는 출력의 위치를 설정하고 조회(query)할 수 있도록 해준다.

함수 'ftell'은 변위(offset)를 주는데, 여기서 변위는 지정된 파일에 대한 파일 위치 지시자의 바이트 수로 주어진다. 함수 'fseek'는 자료들을 건너뛰거나 파일의 앞 부분으로 되돌아갈 수 있도록 파일 위치 지시자를 재위치 시킨다(reposition).

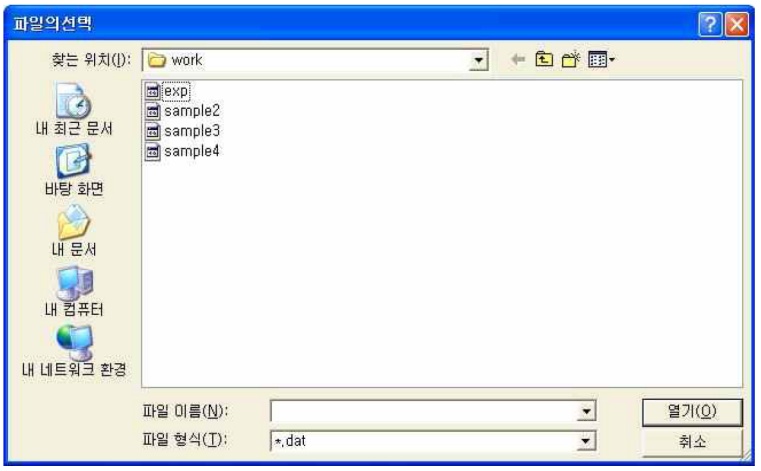
```
》A=[1 2 3 4 5];
》fid=fopen('5.dat','w');
》fwrite(fid,A,'short');
》fclose(fid);
》fid=fopen('5.dat','r');
》fseek(fid,6,'bof');          % cof:현재 위치, bof:시작 위치, eof:끝 위치
》fread(fid,1,'short')
  ans = 4
》ftell(fid)
  ans = 8
》fseek(fid,-4,'cof');
》fread(fid,1,'short')
  ans = 3
》fclose(fid)
```

4) 파일 입·출력에서 dialog box 이용하기

MATLAB에서는 파일 입·출력을 더욱 완성도 있게 보이기 위해서 'uiputfile.m','uigetfile.m'과 같은 함수를 제공한다.

- [fname, pname]=uigetfile('file_name','dialogtitle',x,y)
  : pixel 단위로 x,y 위치에 'dialogtitle'이름의 열기 window를 만들어 준다. 임의의 파일을 선택하면, 'fname'에는 선택한 파일의 이름이 반환되고, 'pname'에는 선택한 파일이 위치하는 경로가 할당된다.

```
》 x=0:0.1:1;
》 y=[x ; exp(x)];
》 fid=fopen('exp.dat','w')
    fid = 3
》 count=fprintf(fid, '%6.2f %12.8f\n',y)
    count = 220
》 fclose('all')
    ans = 0
》 [fname, pname]=uigetfile('*.dat','파일의 선택')
```




```
    fname = exp.dat
    pname = D:\Tool\MATLABR11\work\lecture\
》 fid=fopen(fname, 'r');
》 [a, count]=fscanf(fid,'%f %f\n',[2 inf]);
》 fclose('all')
```

제 4 장 MATLAB 프로그래밍

1. 흐름 제어문

MATLAB에는 명령의 흐름을 제어하는 4가지 방법, 즉, "if문, for문, while문, switch문"들이 있다.

1) IF 문

'if'문에 대한 명령은 주로 조건부 수행을 하고자 할 경우에 사용된다. 일반적인 형태는 다음과 같다.

```
if 논리식
      명령어 문장
elseif 논리식
      명령어 문장
   else
      명령어 문장
end
```

만일, 'if문'의 논리적인 식이 'nonzero(즉, true)'이면, MATLAB은 명령어 문장을 수행한다. 그러나, 논리적인 식의 값이 'zero(즉, false)'이면 명령어 문장을 skip한다. 위에서 'elseif'와 'else' 명령어의 차이점은 논리식을 갖는가 갖지 않는가 하는 점이다.

만일, 논리적인 식의 값이 scalar가 아니라 배열(array)로 주어진다면, 그 배열의 모든 원소가 'nonzero'이어야 '참(true)'이 된다.

```
》x=200;
》if x>100
    a='Large'
  elseif x>50
    a='Middle'
  else
    a='Small'
  end
 a = Large
```

- 정수 n의 부호와 짝·홀수 여부(parity)에 따라서 계산 방법이 세 가지로 분류

```
》if n < 0
        (음수 n에 대한 실행문)
    elseif rem(n,2) == 0
```

```
    (짝수에 대한 실행문)
else
    (홀수에 대한 실행문)
end
```

2) FOR 문

'for문'은 일련의 작업을 정해진 횟수만큼 반복하여 수행한다. for문에 대한 일반적인 형태는 다음과 같다.

```
for 변수 =초기치:증분:최종치
      명령어 문장들
end
```

기본적으로 증분량을 지정하지 않으면, 매번 반복할 때마다 증분하는 양은 '1'이 된다.

- 벡터 x에 1부터 10까지의 숫자를 채우는 방법은 다음과 같다.

```
》for i=1:10 , x(i)=i ; end   또는
》for i=1:10 ; x(i)=i ; end   또는
》for i=1:10
    x(i) = i
  end
》x
  x = 1   2   3   4   5   6   7   8   9   10
```

- 행렬의 인수(행과 열) 모두를 변화 : 중첩된(nested loop) 사용한다.

```
》for i=1:3
    for j=1:4
          A(i,j)=i+j;
    end
  end
》A;

》for i = 1:4
    for j=1:5
      a(i,j)=1/(i+j);
    end
  end
```

```
》a
   a =
        0.5000    0.3333    0.2500    0.2000    0.1667
        0.3333    0.2500    0.2000    0.1667    0.1429
        0.2500    0.2000    0.1667    0.1429    0.1250
        0.2000    0.1667    0.1429    0.1250    0.1111

》x='1/(i+j)';
》for i = 1:3
    for j = 2:4
      a(i,j) = eval(x);
    end
  end
》a
   a =
     0.5000    0.3333    0.2500    0.2000
     0.3333    0.2500    0.2000    0.1667
     0.2500    0.2000    0.1667    0.1429
```

ex) 1. x=[1 0 3 0 5 0 7 0 9]를 for문을 이용해서 만드시오.
    2. y=[9 0 7 0 5 0 3 0 1]를 for문을 이용해서 만드시오.

- vandermonde 행렬
:임의의 열벡터의 각 원소들에 대한 n 거듭제곱을 열의 원소로 갖는 행렬

```
》x=[-3 0 1 2 5]'
```

$$v = \begin{bmatrix} 81 & -27 & 9 & -3 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 16 & 8 & 4 & 2 & 1 \\ 625 & 125 & 25 & 5 & 1 \end{bmatrix}$$

```
》n=length(x)
》for k=1:n
    for i=1:n
      v(i,k)=x(i)^(n-k);
    end
  end
```

또는

```
》n=length(x);
》v( : , n) = ones(n,1);
》for i=n-1:-1:1
    v(:,i) = x.*v( : , i+1) ;
```

```
end
```

- break 문

```
for k=0:9
for m=0:9
    if m==5
        [k m]
    break
    end
end
end
```

3) WHILE 문

for문일 경우는 반복 횟수가 지정된다. 그렇지만 반복처리를 할 경우, 횟수를 지정하는 것이 아니고 종료 조건이 일치할 때 반복을 끝내고 싶은 경우가 있다. 즉, 조건이 만족될 때까지 계속해서 반복한다. while문의 일반적인 형태는 다음과 같다.

```
while 논리식
        명령어 문장들
end
```

- 2의 거듭제곱을 계산하여 그 값이 10을 넘는 n의 값을 구하고자 한다.

```
》n=0;
》while 2^n <= 10
      n=n+1;
  end
》n
   n = 4
```

- n!의 값이 처음으로 십진수 10000보다 커지는 n의 값을 구할 때

```
》n=1;
》while prod(1:n) <= 10000      % <= : 관계 연산자
     n=n+1 ;
  end
》n
   n = 8
```

- 임의의 양의 정수를 취하여, 만약 그 수가 짝수이면 2로 나누고 홀수이면 3을 곱한 후에 1을 더한다. 정수가 1이 될 때까지 이 과정을 계속 반복 → 이 계산 과정이 중단되지 않고 계속되는 정수가 존재할 것인가?

```
》while 1
  n=input('Enter positive integer, n! ');
  if n <= 0, break, end
  while n > 1
        if rem(n,2) == 0
           n=n/2;
        else
          n=3*n+1;
       end
  end
```

4) SWITCH 문

C언어에서의 switch 문과는 달리, MATLAB에서의 switch문은 변수 또는 식의 값과 맞는(equal) 'case 문'의 명령어만 수행하므로 C언어에서 처럼 'break'명령어가 따로 필요없다. 또한, C언어에서의 'switch 문'에 있는 'default' 명령어는 MATLAB에서 'otherwise'에 해당한다. switch 문의 일반적인 형태는 다음과 같다.

```
switch 변수 또는 식
   case 값 1
       명령어 문장
   case 값 2
       명령어 문장

    . . . 임의의 수의 case 문들 . . .

   otherwise
       명령어 문장
end
```

- 다음은 짝수와 홀수를 판별하는 문장이다.

```
》if x ~= abs(x)
     error ('입력 값이 실수가 아니다.');
      elseif x ~= fix(x)
          error('정수가 아니다.');
         elseif x < 0
              error('양수여야 한다.');
```

```
end
y=rem(x,2)
switch y
    case 0
        disp('입력 값은 짝수!');
    otherwise
        disp('입력 값은 홀수!');
end
```

2. M-file 작성

일반적으로 MATLAB은 명령구동방식(command-driven mode)으로 사용된다. 즉, 사용자가 한 줄짜리 명령을 입력하면 MATLAB은 그 명령을 즉시 처리하여 결과를 모니터 화면에 표시한다. MATLAB에서는 또한 일련의 명령들을 파일 내에 저장했다가 한꺼번에 실행할 수도 있다. 아울러 이러한 두 가지 방식은 해석기 방식의 환경(interpertive environment)을 구성한다.

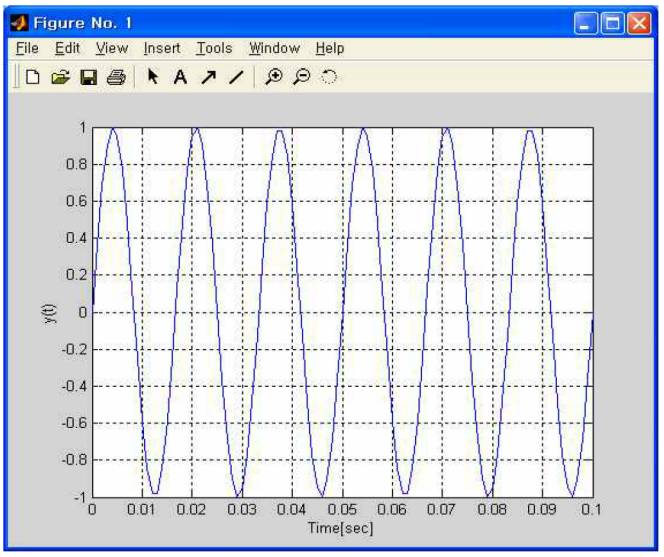
MATLAB의 실행 문장들을 포함하고 있는 파일들을 m-file이라고 부르는데, 그 이유는 파일이름의 확장자가 'm'이기 때문이며 확장자 'm'은 매크로 파일(macro file)을 의미하기도 한다. m-file은 일련의 정상적인 MATLAB 실행 문장들로 구성되어 있으며, 실행 문장들 중에는 다른 m-file을 참조하는 문장도 있을 수 있다. 또한 m-file은 자체 m-file 내에서 그 자신을 부를 수도 있다(recursive call).

MATLAB의 m-file은 '스크립트(script)'와 '함수(function)'의 두 가지 유형으로 분류할 수 있다. 먼저 '스크립트(script)'파일이란 일련의 긴 명령들을 한꺼번에 자동적으로 수행해 주는 파일을 가리키며, '함수(function)' 파일이란 다른 전문적인 프로그래밍 언어에 버금가도록 MATLAB의 기능성을 확장시켜 주는 파일이다.

1) 스크립트(Script) 화일

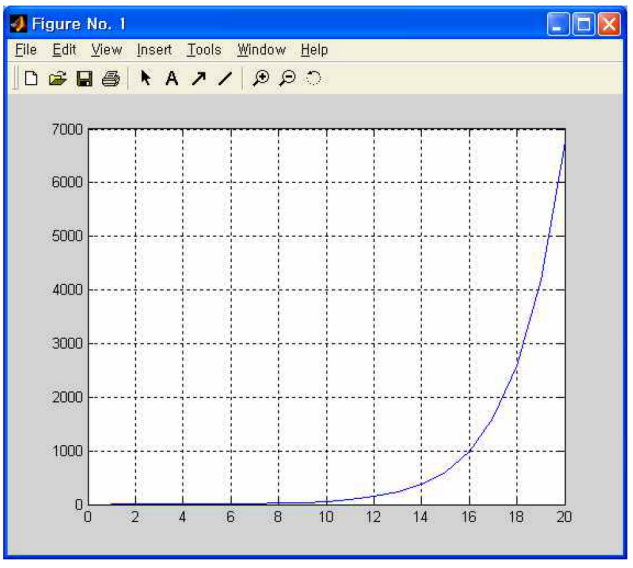
- 단순히 파일 내에 들어 있는 명령들을 차례대로 실행
- 스크립트 내의 변수들은 전역변수(global variable)로 취급
- 정현파를 출력하는 프로그램을 스크립트 파일로 작성

```
%sine 그래프 출력 스크립트 파일
%M-file Program sin_plot.m
t=[0:0.001:0.1];
freq=60;
y=sin(2*pi*freq*t);
plot(t,y),grid
xlabel('Time[sec]'); ylabel('y(t)')
```



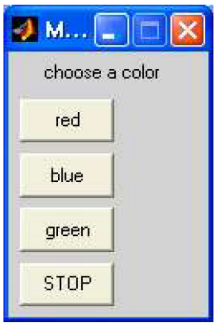
- Fibonacci수를 계산하는 fibon2.m 파일 작성

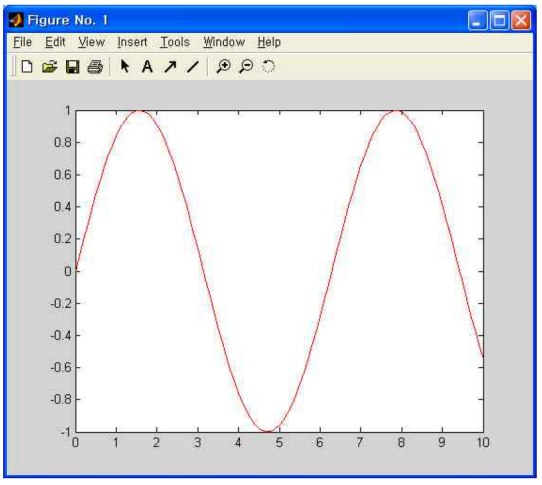
```
% Script file to calculate fibonacci numbers
i=1 ; f=[1 1]; % 색인과 함수값의 초기화
while f(i)+f(i+1) < 10000
    f(i+2) = f(i) + f(i+1);
    i = i + 1 ;
end
plot(f),grid
```



함수 'menu'를 사용하면 화면에 메뉴를 출력하고, 메뉴의 원하는 항목을 선택하는 것에 의해서 프로그램 중에 이 번호에 대응한 처리를 할 수 있다.

s=menu('Choose a Color','Red','Blue','Green');





'if_test.m'

```
close all
t=0:0.1:10;
while 1
s=menu('      choose a color      ','red','blue','green','STOP');
if s==1
    plot(t,sin(t),'r')
elseif s==2
    plot(t,sin(t),'b')
elseif s==3
    plot(t,sin(t),'g')
else
    close all;break
end
end
```

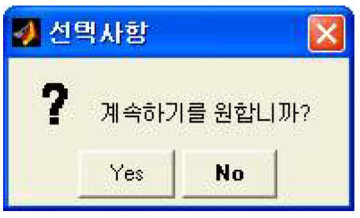
- dialog형(type) 함수들

① R=questdlg('qstring','title','str1','str2','default')

: 질문형 dialog box를 만들고, 두 개의 'str1','str2' label를 가진 push 버튼을 만든다.
'default' 버튼은 default 상태를 지정하게 해 준다.
선택된 str이 출력변수 R로 반환된다.

```
close all
line_color=['r','g','b'];
str='Yes';

t=0:0.1:10;
while 1
```

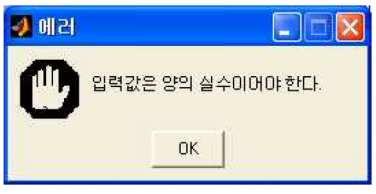
```
    if strcmp(str,'Yes')
        s=menu('      선의 색을 결정          ','red','blue','green');
        plot(t,sin(t),line_color(s));
        str=questdlg('계속하기를 원합니까?','선택사항','Yes','No','No');
    else
        close all, break
    end
end
```



② msgbox('message', 'title', 'icon')

: 'title'을 box이름으로 하고, 'message'를 표시하는 box를 만든다.
'icon'은 messgae box안에 표현할 icon모양을 결정한다.('error', 'help','warn')

```
function PM_cb(x)
if x~=abs(x)
    msgbox('입력값은 양의 실수이어야 한다.','에러','error');
    return;
elseif x~=fix(x)
    msgbox('정수가 아니다','에러','error');
    return;
end

y=rem(x,2);

switch y
case 0
    disp('입력값은 짝수!')
otherwise
    disp('입력값은 홀수!')
end
```



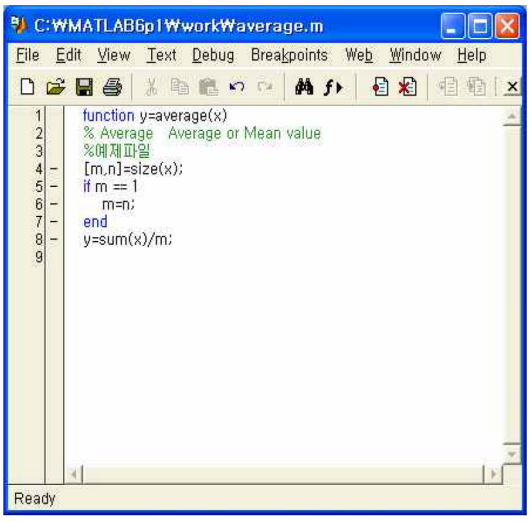
2) 함수(Function) 파일

스크립트 file 외에 함수를 정의해서 m-file로 작성할 수 있다. MATLAB 프로그램에서 사용되는 명령어들은 주로 여기서 설명하는 함수로써 구성되어 있다. 이와 같은 성질을 이용하면 사용자의 필요에 의해서 자신만의 고유의 함수 file을 만들 수 있다. 일적적인 함수 file 형태는 아래와 같다.

```
function[output1, output2, ... ] = filename(input1, input2, ...)
% 주석문(함수의 일반적인 설명)
MATLAB 명령어
```

함수를 정의하기 위해서는 MATLAB Desktop Windows toolbar에서 맨 왼쪽에 있는 "New M-File" 을 클릭한다.

"New M-File" 을 클릭하면 다음그림과 같은 창이 뜰 것이다.



위 그림은 평균값을 계산 함수 'average.m'을 작성한 것이다. 위의 내용을 'average.m'으로 저장한 후 MATLAB Command Window에서 다음과 같은 명령을 실행한다.

```
》t=1:100;
》average(t)
   ans = 50.5000
```

① 파일의 첫 번째 줄에 함수의 이름, 입력 인수 및 출력 인수가 선언되며, 이 부분이 '함수 파일'과 '스크립트 파일'의 가장 큰 차이점이다.
② '%' 기호의 뒤에 나오는 문장들은 주석(comment)으로 인식되어 무시된다.

③ '%' 기호와 함께 적어 놓은 설명 부분은 MATLAB 프롬프트에서 'help average'이라고 입력했을 때 모니터 화면상에 표시된다.

④ MATLAB 프롬프트에서 'lookfor average'이라고 입력하면 average가 포함된 항목들을 모두 찾아 준다.(주석의 첫 번째 줄만 해당됨)

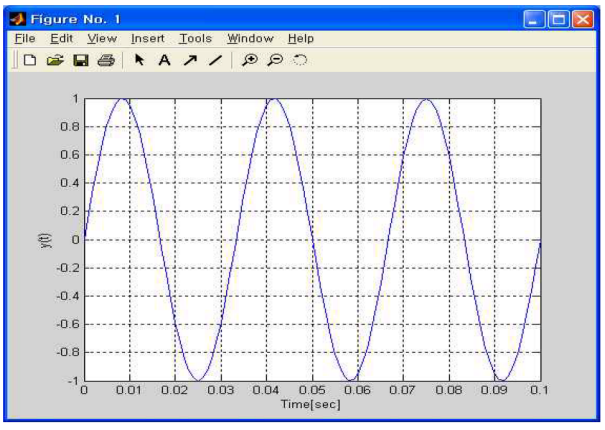
⑤ 파일의 첫 번째 줄에 나타나는 변수들 y, m 및 n은 함수 'average'에 국한된 변수들로써 'average'이 실행되고 난 후에는 작업 공간상에 존재하지 않게 된다.

⑥ 함수 파일의 변수들은 지역 변수들이기 때문에, 함수 'average'에 사용되는 입력 인수의 이름이 반드시 m-file 내의 변수인 'x'와 같을 필요는 없다.

정현파를 출력하는 프로그램을 함수 파일로 작성하면 아래와 같다.

```
function a=sin_plot2(freq)
%Sine 그래프를 출력하는 함수 파일
%M-file program sin_plot2
t=[0:0.001:0.1];
y=sin(2*pi*freq*t);
plot(t,y),grid
xlabel('Time[sec]'),ylabel('y(t)')
```

```
》sin_plot(30)
```



다음의 예는 표준편차를 계산하는 'stat.m' 파일을 작성한 것이다.

```
function [mean, stdev] = stat(x)
[m,n] = size(x);
if m == 1
m=n;
end
mean = sum(x)/m
stdev=sqrt((sum((x-mean).^2))/(m-1))
```

- 전역변수

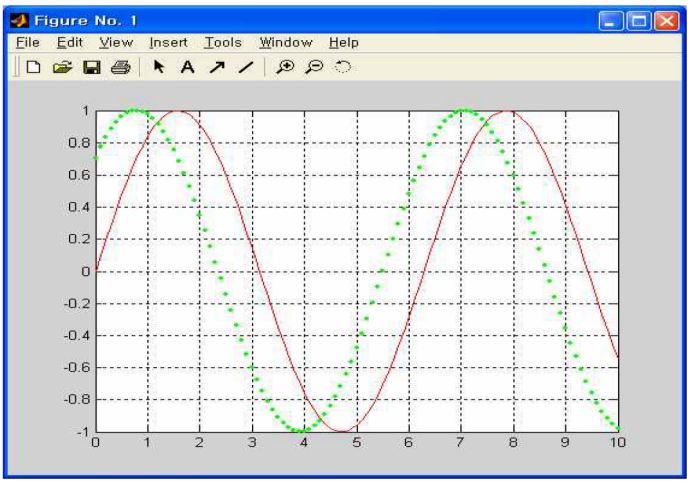
대개의 m-file로 정의된 MATLAB 함수들은 자체적으로 지역변수들을 사용한다. 이러한 지역 변수들은 다른 함수의 변수들, 기본 작업 공간의 변수들 및 스크립트 파일의 변수들과 분리하여 취급한다. 그러나, 만약 몇 개의 함수들에서 특정한 이름의 변수를 전역변수로 선언했다면, 그 변수를 전역 변수로 선언한 함수들과 MATLAB 작업 공간은 그 변수를 공유하게 된다.

① function workspace와 base workspace간의 데이터 교환 시

<plot_sin.m>

```
function y=plot_sin(t)
global a_deg
y1=sin(t)
y2=sin(t+a_deg*pi/180);
plot(t,y1,'r-',t,y2,'g.')
grid on
```

```
》 x=0:0.1:10;
》 global a_deg;
》 a_deg=45;
》 plot_sin(x)
```




```
》 whos global
  Name          Size              Bytes  Class
   a_deg        1x1                   8  double array (global)
  Grand total is 1 elements using 8 bytes      Your variables are:
```

② function workspace간의 데이터 교환 시

<newstats1.m>

```
function [avg,med]=newstats1(in)
%newstats 서브함수를 이용하여 평균과 중간값을 구하는 함수

global n;

n=length(in);
avg=newmean(in);
med=newmedian(in);

%서브함수1
function a=newmean(in)
global n;
a=sum(in)/n;

%서브함수2
function m=newmedian(v)
global n;
w=sort(v);
if rem(n,2)==1
    m=w((n+1)/2);
else
    m=(w(n/2)+w(n/2+1))/2;
end
```

- 서브 함수(subfunction)

MATLAB 5.0이후 버전부터는 다른 High_level 언어에서처럼 서브함수(subfunctions)를 정의 할 수 있게 되었다.

function mode m-file의 첫 번째 함수를 'primary function'이라 하고, 또한, primary function의 이름이 해당 m-file의 파일 이름이 된다. 그리고, primary function 안에 새롭게 정의된 함수들을 'subfunction'라고 한다.

subfunction은 해당 primary function안에서, 또는 같은 파일 안에 있는 subfunction에서만 참조가 가능하다. 각각의 subfunction 역시, 자신만의 함수 정의부를 소유해야 한다.

다음은 벡터의 평균(average)와 중간값(median value)를 구하는 함수이다.

```
function [avg,med]=newstats(in)
%newstats 서브함수를 이용하여 평균과 중간값을 구하는 함수
%
n=length(in);
avg=newmean(in,n);
med=newmedian(in,n);
```

```
%서브함수1
function a=newmean(in,n)
a=sum(in)/n;

%서브함수2
function m=newmedian(v,n)
w=sort(v);
if rem(n,2)==1
    m=w((n+1)/2);
else
    m=(w(n/2)+w(n/2+1))/2;
end
```

```
》x=[1 4 2 10 3 2 7];
》[avg,med]=newstats(x)
 avg = 4.1429
 med = 3.0000
```

- 입·출력 매개변수의 개수를 제어하는 'nargin, nargout'함수

function mode m-file에서 'nargin(number of function input argument), nargout(number of function output argument)' 함수들은 얼마나 많은 입·출력 매개변수가 사용되고 있는지를 알려준다.

앞에서 살펴본 'newstats1.m'파일을 출력변수가 하나 일 때는 평균을 출력하고, 둘 일 때는 평균과 중간값을 반환하는 함수로 바꾸면 다음과 같다.

<messtats2.m>

```
function [avg,med]=newstats2(in)
%newstats 서브함수를 이용하여 평균과 중간값을 구하는 함수
%
global n;
n=length(in);
if nargout ==1
    avg=newmean(in);
else
    avg=newmean(in);
    med=newmedian(in);
end

%서브함수1
function a=newmean(in)
global n;
a=sum(in)/n;

%서브함수2
function m=newmedian(v)
global n;
```

```
w=sort(v);
if rem(n,2)==1
    m=w((n+1)/2);
else
    m=(w(n/2)+w(n/2+1))/2;
end
```

3. 기타 사항

1) MATLAB 명령어 입력시 검색 단계

MATLAB에서 어떤 이름, 예를 들어 'test'와 같은 이름이 입력되면, 일반적으로 MATLAB 해석기(interpreter)는 다음과 같은 단계를 거쳐 그 이름을 처리한다.

① 'test'를 변수로 취급하여 검색한다.
② 'test'를 내장함수(built-in function)로 취급하여 점검한다.
③ 현재 사용중인 디렉토리에 'test.m'이라는 이름의 파일이 존재하는지 찾아본다.
④ MATLAB의 검색 경로에 지정된 디렉토리들 중에 'test.m'이라는 이름의 파일이 존재하는지 검색한다.

위의 검색 단계들로부터 알 수 있듯이 MATLAB에서는 어떤 이름이 입력되면 그것을 함수로 취급하기 전에 먼저 변수로써 취급한다.

2) echo, input, keyboard, pause

① echo

정상적으로 MATLAB의 m-파일이 수행되는 동안에는 그 파일 내의 명령들이 모니터 화면에 표시되지 않는다. 그러나 'echo' 명령을 사용하면 m-파일이 실행될 때 각각의 명령들이 모니터 화면에 나타난다.

② input

m-file이 실행될 때 사용자가 자료를 입력할 수 있도록 해준다. 'input'를 실행시키면, 자료가 입력될 때까지 더 이상 프로그램을 실행하지 않고 대기 상태로 머무르게 된다.

```
》input('how old are you?')
    how old are you? 25
    ans = 25
```

③ keyboard

컴퓨터 자판을 마치 스크립트 파일처럼 사용하도록 해준다. m-file 내부에 이 함수를 삽입해 놓으면, 파일의 잘못된 곳을 고치거나 파일이 수행되는 동안에 변수를 수정하는데 유용하게 사용할 수 있다.

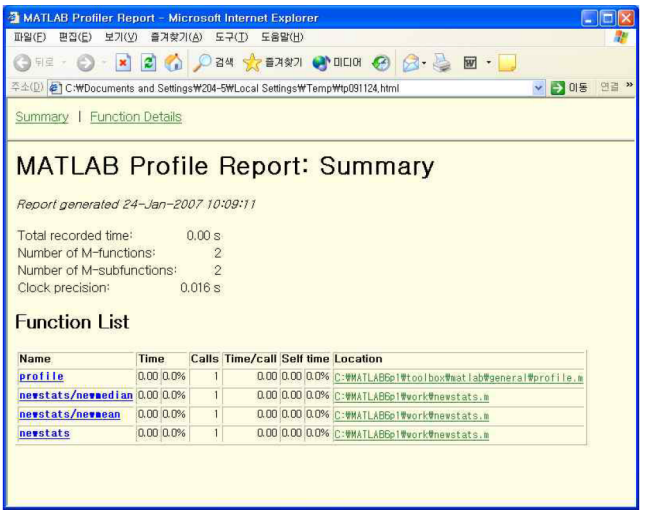
④ pause

프로그램의 실행을 일시적으로 중지 시키는 기능을 수행한다.

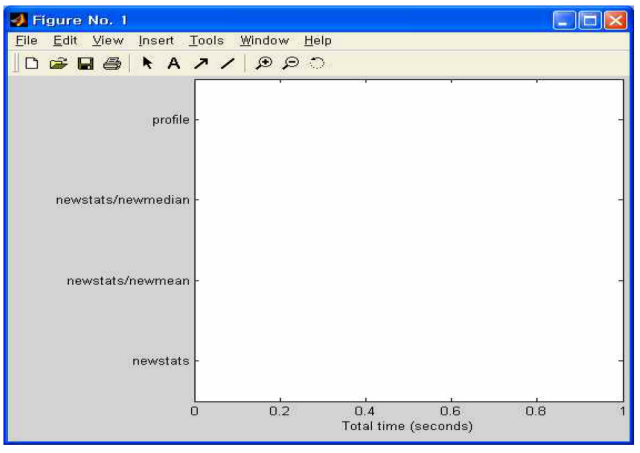
3) profile

profile 명령어를 실행한 후, M-file을 실행하면 각각의 함수들의 처리속도를 계산한 후 web browser를 이용해서 결과를 보여준다.

```
》profile on
》x=[1 4 2 10 3 2 7];
[avg,med]=newstats(x)
avg =      4.1429
med =       3
》profile report
```




```
》profile plot
```



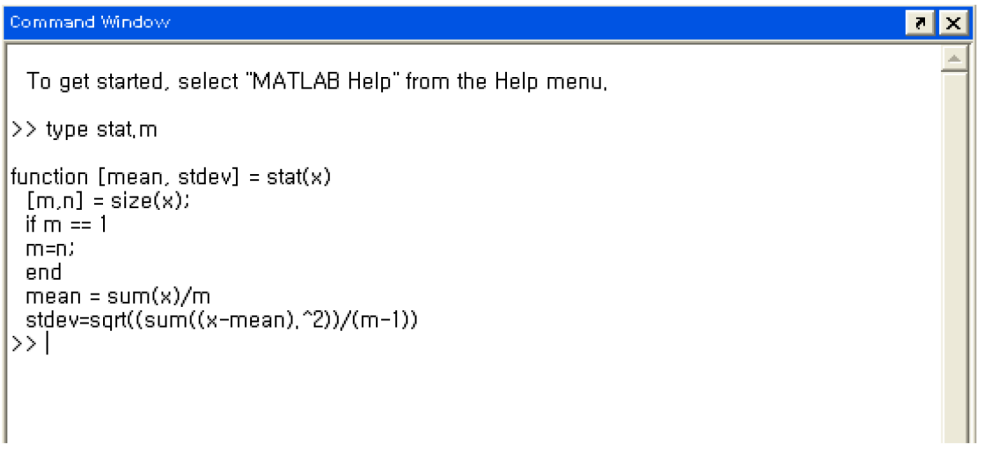

```
》profile off
```

4) P-code 파일

M-file은 모든 소스를 다른 사람들이 쉽게 볼 수가 있어서 프로그램을 보호 할 수가 없었다. 'pcode.m'은 MATLAB의 모든 m-file을 parsing하여 pseudocode로 변환한 후 확장자가 '.p'인 파일로 저장하므로 에디터 프로그램으로 소스 코드(source code)를 볼 수 없게 한다.

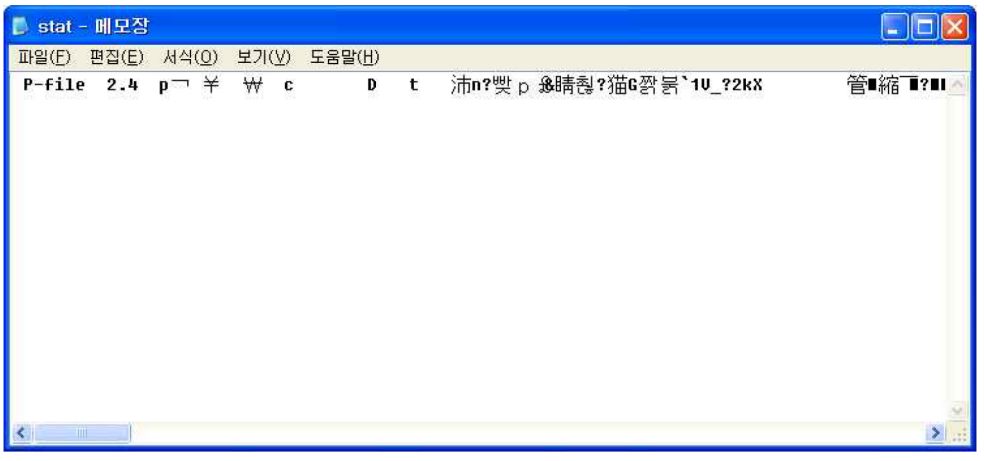
또한, 파일이름이 같은 m-file과 p-파일이 있다면, MATLAB은 우선 p-파일을 실행하게 된다.

》 type stat.m



》 pcode stat.m

》 !notepad stat.p



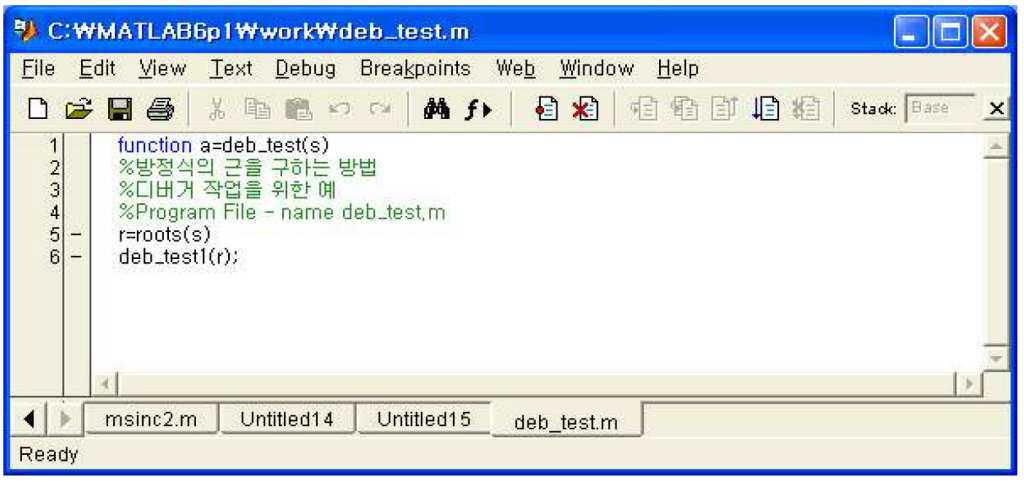
4. MATLAB 프로그램의 디버거(debugger)

1) 디버거 명령어

MATLAB 5 버전의 디버거에서는 MATLAB 4 버전에서 사용한 디버깅의 명령어들을 모두 수용하고 있지만, MATLAB 5 이후 버전에는 자체 내장되어 있는 m-file editor/debugger version을 이용하여 매우 편리하게 디버거 작업을 수행할 수 있다. m-file editor/debugger에 포함되어 있는 새로운 디버깅 명령어들은 다음과 같다.

Step : 현재의 줄을 실행
Step In : 현재의 줄에 있는 명령어로 실행을 옮김
Step Out : 현재의 줄에 있는 명령어에서 다른 명령어로 이동
Continue : 현재의 중지점에서부터 계속하여 실행
Go Until Cursor : 다음 지정된 곳으로 이동
Exit Debug Mode : 디버거 중료

위의 명령을 선택할 수 있는 MATLAB 5.3 이후 버전에 자체 내장되어 있는 m-file editor/debugger version은 다음과 같다.



디버거 작업을 시작하기 전에 먼저 디버깅할 두 개의 m-file을 'deb_test.m'과 'deb_test1.m' 이름으로 다음과 같이 작성한다. 먼저, 방정식의 근을 구하는 것으로서 아래와 같이 입력하여 'deb_test.m'이름으로 m-file을 작성한다.

```
function a=deb_test(s)
% 방정식의 근을 구하는 방법
% 디버거 작업을 위한 예
% Program File - name deb_test.m
r=roots(s)
deb_test1(r);
```

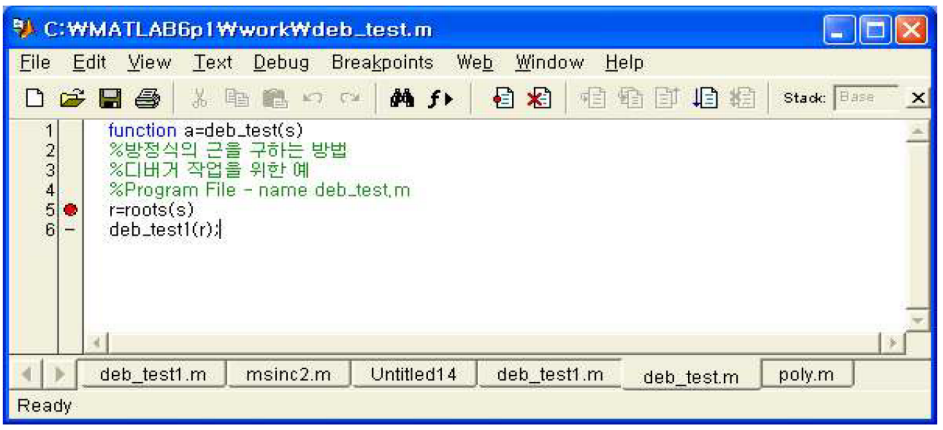
계속해서, 위에서 구한 근을 이용하여 반대로 방정식을 구하는 것으로 아래와 같이 입력하여 'deb_test1.m' 이름으로 m-file을 만든다.

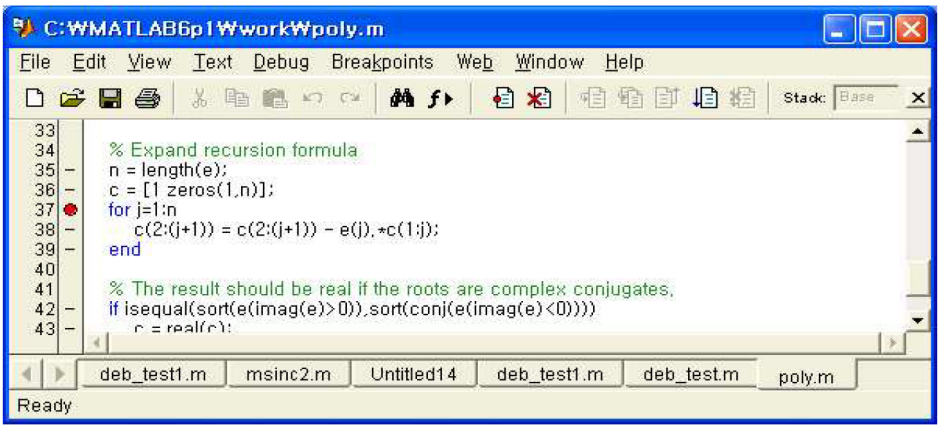
```
function p=deb_test1(r)
% 근을 이용해서 방정식을 구하는 방법
% 디버거 작업을 위한 예
% Program File - name deb_test1.m
p=poly(r)
```

우선, 방정식 $y = x^2 - 3x + 2$에 대해서 위의 예제 프로그램인 'deb_test.m'을 실행하면 방정식의 근과 함께 근을 이용한 방정식이 구해진다.

```
》s=[1 -3 2];
》deb_test(s)
   r = 2
       1
   p =
       1   -3   2
```

2) 중지점 설정



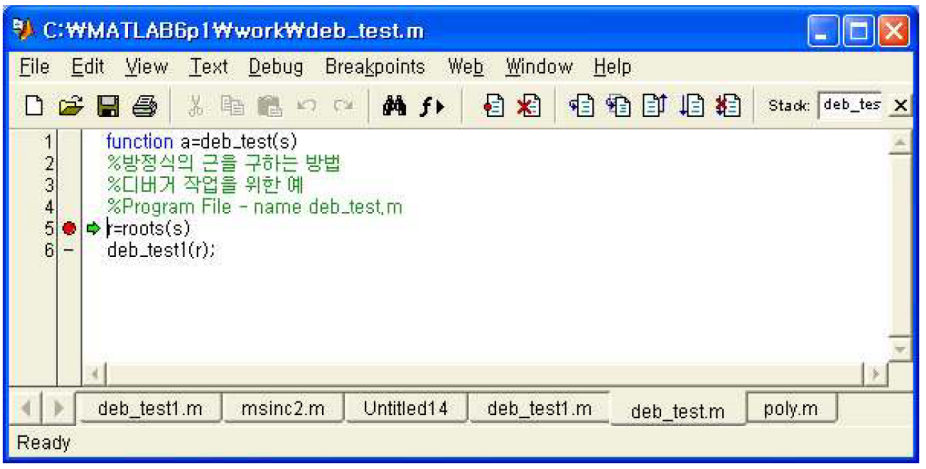


3) M-file의 실행과 스택의 표시

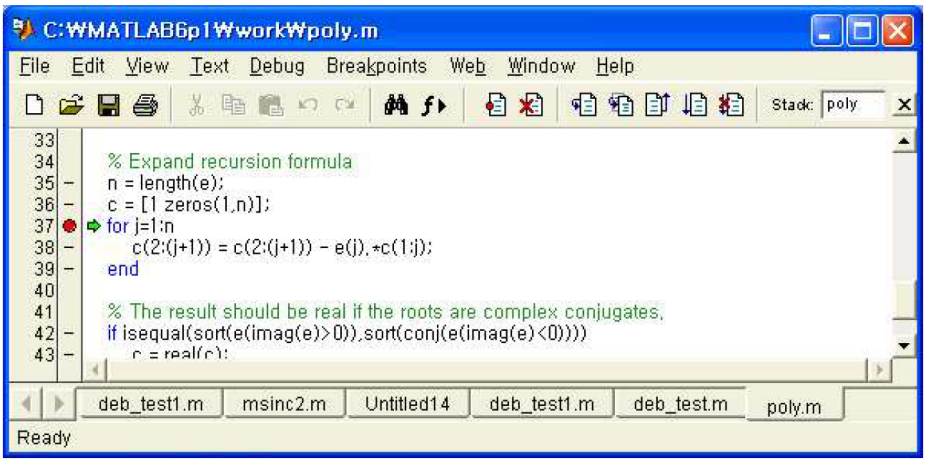
중지점을 설정한 후에 다음과 같이 명령('deb_test([1 -3 2])'을 실행하면

```
》deb_test([1 -3 2])
K》
```

프로그램 실행 다음 줄에 'K》'의 모양으로 현재 디버거 작업 중임을 나타내는 프롬프트가 나타남과 동시에 m-file editor/debugger가 활성화되면서 'deb_test.m' 이름의 m-file에 설정해 놓은 중지점인 5번째 줄에 녹색의 화살표가 오른쪽으로 나타나고 실행을 정지하게 된다.



계속하여 함수를 실행하기 위해서 'continue'명령을 클릭하면 두 번째로 설정해 놓은 중지점인 'poly'함수의 37번째 줄에서 실행을 일시 정지하게 되며 녹색의 화살표로 현재의 위치를 나타낸다.



지금까지 실행한 함수 호출에 대한 내용을 확인하게 위해서 stack의 옆에 있는 선택 화살표를 클릭하면 실행한 함수에 대한 리스트가 나타나게 된다.

4) 작업공간 변경과 변수의 내용

호출된 각각의 함수에 대한 작업 공간에서의 변수를 확인하고자 할 경우가 있다. 이러한 경우에 m-file editor/debugger의 스택을 이용하여 각각의 함수에 대한 작업 공간으로 이동을 하여 현재 사용되고 있는 변수를 확인하게 된다.

스택을 이용하여 'deb_test.m' 함수를 선택하여 작업 공간을 옮긴 후에 MATLAB 명령창에서 다음과 같이 실행하면 'deb_test.m'함수에서 현재 사용하고 있는 변수명들을 확인 할 수 있다.

```
K》whos
  Name        Size            Bytes  Class
   r          2x1                16  double array
   s          1x3                24  double array
Grand total is 5 elements using 40 bytes
```

또한, 'deb_test1.m' 함수를 선택하여 변수를 확인하면 다음과 같다.

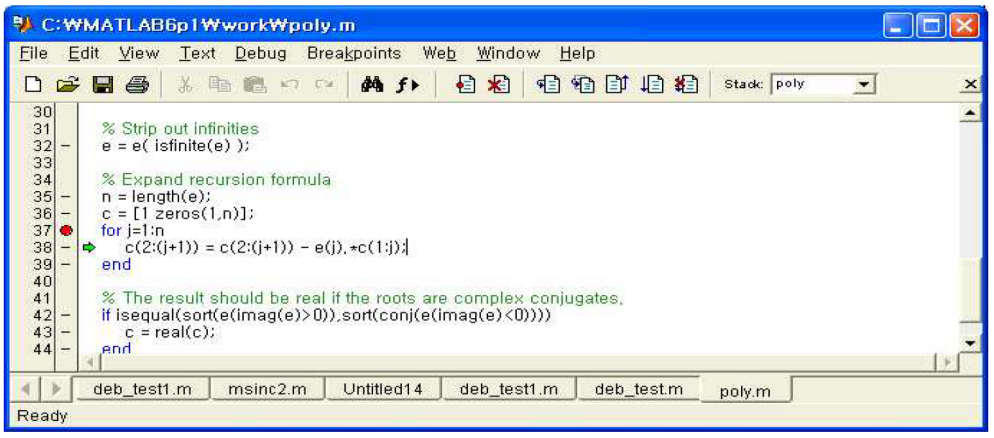
```
K》whos
  Name        Size            Bytes  Class
   r          2x1                16  double array
Grand total is 2 elements using 16 bytes
```

마지막으로 사용된 'poly.m'함수로 작업 공간을 이용하여 변수를 확인하면 다음과 같다.

```
K》whos
  Name        Size            Bytes  Class
   c          1x3                24  double array
   e          2x1                16  double array
   m          1x1                 8  double array
   n          1x1                 8  double array
   x          2x1                16  double array
Grand total is 9 elements using 72 bytes
```

5) 다음 줄 실행과 변수

지금까지 실행된 것은 중지점에 의해서 'poly.m' 함수의 37번째 줄에서 정지된 상태이다. 따라서, 계속해서 프로그램을 실행할 필요성이 있다. 이러한 경우에 'step'을 클릭함으로써 다음 줄을 실행할 수 있게 된다.



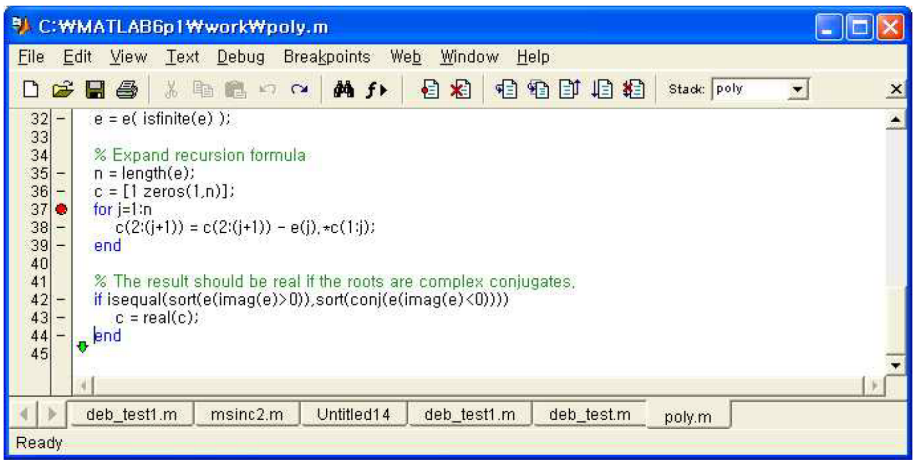
K》whos

Name	Size	Bytes	Class
c	1x3	24	double array
e	2x1	16	double array
j	1x1	8	double array
m	1x1	8	double array
n	1x1	8	double array
x	2x1	16	double array

Grand total is 10 elements using 80 bytes

6) 디버깅의 계속 실행

'poly.m' 함수가 종료될 때까지 계속하여 'step' 명령을 실행한다. 이와 같이, 계속해서 실행하다 보면 함수가 종료되는 시점에서는 녹색의 화살표가 아래쪽으로 향하게 된다.

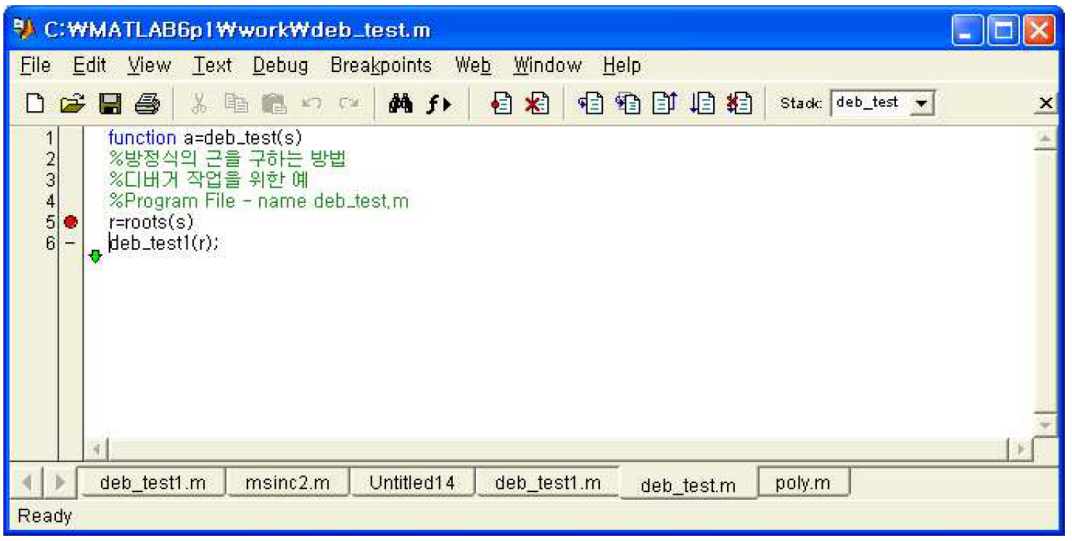


계속해서 'single step'명령을 수행하면, 다음과 같은 결과가 명령창에 나타남과 동시에 'deb_test1.m'함수가 활성화 된다.

K》

p=

1 -3 2



7) 디버깅의 종료

디버깅을 'single step' 명령을 이용하여 프로그램을 끝까지 계속해서 실행하여 결과를 확인하거나, 그렇지 않고 중지점이나 프로그램의 중간에서 디버깅의 종료시까지 함수를 계속해서 실행하기 위해서는 'continue' 명령을 사용하면 된다. 'continue' 명령을 실행하면 디버거 프롬프트 ('K》')에서 MATLAB 프롬프트('》')가 나타나게 된다. 또는 디버깅을 벗어나기 위해서는 'quit debugging'명령을 사용하여 디버깅을 종료할 수 있다.

'quit debugging' 명령에 의한 디버깅 종료는 중지점을 제거하지 않기 때문에 'set/clear breakpoint'와 'clear all breakpoint'를 이용해서 중지점을 제거 할 수 있다.

제 5 장 2차원 및 3차원 그래프

1. 2차원 그래프

1.1 기본적인 그래프 함수들

MATLAB에서 그래프를 그리는데 있어서 기본적으로 사용되는 함수들은 아래와 같다. 각각의 경우에 그래프의 축에 대한 표시 방법만 다를 뿐이며, 이 함수들은 입력을 행렬 또는 벡터의 형태로 받아들이며, 입력 자료에 따라 자동적으로 축의 배율을 설정해서 그래프를 그려 준다.

plot - 벡터 또는 행렬의 열(column)에 대하여 선형 배율(linear scale)의 그래프를 그려 준다.
loglog - x와 y축 모두를 대수 배율(log scale)에 맞추어 그래프를 그려 준다.
semilogx - x축은 대수 배율에, y축은 선형 배율에 맞추어 그래프를 그려 준다.
semilogy - y축은 대수 배율에, x축은 선형 배율에 맞추어 그래프를 그려 준다.

위의 함수 명령들로 축의 형태를 설정한 후에, 다음의 함수들을 사용하여 그래프의 제목, 축의 이름, 눈금 격자 및 그래프에 대한 설명 등을 추가할 수 있다.

title - 그래프에 제목을 붙인다.
xlabel - x축에 축의 이름을 붙여 준다.
ylabel - y축에 축의 이름을 붙여 준다.
text - 지정한 위치에 원하는 문구를 표시해 준다.
gtext - 마우스를 사용하여 그래프 상에 나타내고자 하는 문구의 위치를 지정한다.
grid - 눈금 격자(grid line)들을 그래프에 나타낼 것인지를 결정한다.

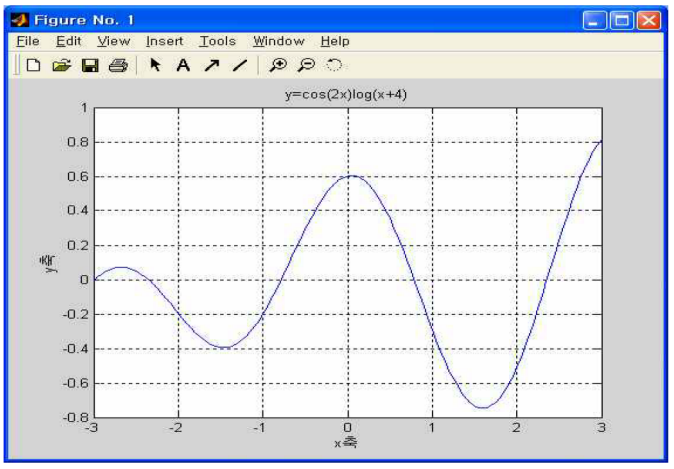
1.2 그래프 그리기

'plot(y)' 명령어는 y의 요소들을 y축의 값으로 선정하고, x축은 y의 요소들의 색인(index)을 x축으로 선정하여 그래프를 출력한다. 또한, 'plot(x,y)' 명령어는 x의 요소들을 x축의 값으로, y의 요소들을 y축의 값으로 선정하여 그래프를 출력한다.

다음은 'y=cos(2x)log$_{10}$(x+4)'를 그래프로 출력하는 과정이다.

<plot_test1.m>

```
x=-03:0.05:3;
y=cos(2*x).*log10(x+4);
plot(x,y)
xlabel('x축'),ylabel('y축');
title('y=cos(2x)log(x+4)');
grid
```



1.3 선의 형태와 기호 및 색상

'plot' 명령에서 일련의 문자들을 입력함으로써 선의 종류와 색상을 사용자가 지정할 수 있다. 다음과 같은 'plot' 명령에서

```
》 plot(x,y,s)
```

x와 y는 각각 그래프의 x축 및 y축 자료를 지정하며, s는 하나, 둘 또는 세 개의 문자로 구성된 인수로서 작은따옴표(')를 사용하여 구분되는데, 각각의 기호들이 의미하는 바를 아래의 표에서 나타내었다.

MATLAB 그래프에 사용되는 색상 및 기호들

기 호	색 상	기 호	선의 형태	기 호	선의 형태
b	blue(파란색)	.	point(점)	>	triangle(right)
g	green(녹색)	o	circle(원)	p	pentagram
r	red(빨간색)	x	x-mark(x 표시)	h	hexagram
c	cyan(하늘색)	+	plus(덧셈 기호)	-	solid line
m	magenta(자홍색)	*	star(별표)	:	dotted(점선)
y	yellow(노란색)	s	square solid(실선)	-.	dashdot(일점 쇄선)
k	black(검정색)	v	triangle(down)	--	dashed(단선)
w	white(흰색)	^	triangle(up)	d	diamond
		<	triangle(left)		

'plot'은 또한 아래와 같이 x값, y값, 옵션을 여러개 반복해서 지정할 수 있다. 이러한 경우 옵션에서 색상을 지정하지 않으면, 위의 색상표에 열거된 파란색부터 검정색까지의 순서로 그래프를 도시한다.

```
》plot(x1,y1,s1,x2,y2,s2,x3,y3,s3,...)
```

1.4 기존의 그래프에 새로운 선을 추가. - hold 사용

'hold' 명령을 사용하면 이미 작성되어 있는 그래프에 새로운 선들을 추가할 수 있다. 'hold' 명령을 'on' 상태로 설정하면, MATLAB에서는 기존의 그래프를 제거하지 않은 상태에서 같은 그래프 상에 새로운 선을 추가하여 그려준다. 새로 그려지는 선의 배율이 다를 경우에는 자동적으로 배율을 재설정해서 그려 준다.

```
》plot(x)
》hold on
》plot(y1)
》plot(y2)
》hold off
```

1.5 복소수 자료의 도시

Z가 하나의 복소 벡터 또는 복소 행렬일 때 다음의 명령은 실수부와 허수부를 분리해서 그래프로 도시하게 된다.

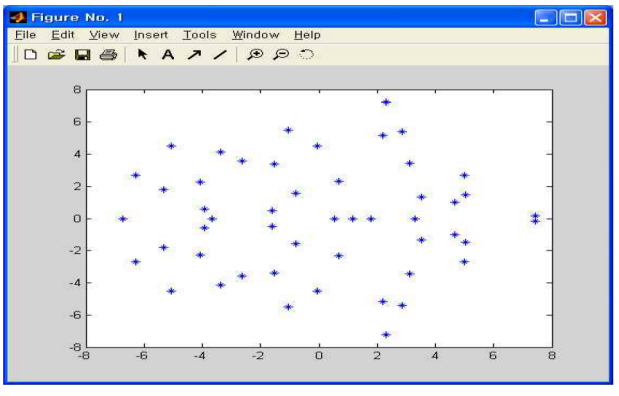
```
》plot(Z)
```

이것은 다음과 같은 동작을 하게 된다.

```
》plot(real(Z), imag(Z))
```

다음의 예는 50×50인 난수 행렬의 고유치(eigenvalue)들의 분포를 그래프로 나타낸 것이다.

```
》Z=eig(randn(50,50));
》plot(Z,'*')
```



하지만 여러 가지의 복소수들(Z1, Z2, Z3)을 한꺼번에 도시하고자 할 경우에는 plot(Z1,Z2,Z3)와 같이 쓸 수 없으며 다음과 같이 입력하여야 한다.

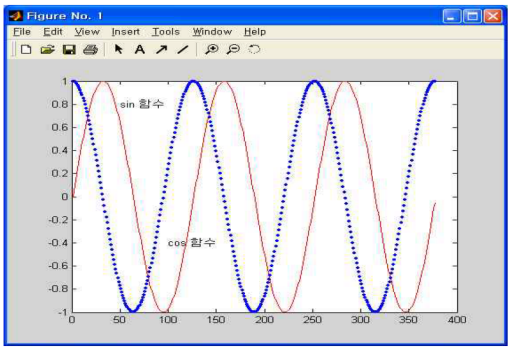
```
》plot(real(Z1), imag(Z1),'*', real(Z2), imag(Z2), 'd', real(Z3), imag(Z3),'^')
```

1.6 텍스트(text)의 삽입

텍스트 삽입을 위한 명령에는 'text'와 'gtext' 명령어가 있다. 'text' 명령어는 x축과 y축으로 지정된 출력 위치마다 텍스트를 삽입한다. 'gtext' 명령어는 마우스를 이용하여 임의의 위치를 선정하면 그 위치에 텍스트를 삽입한다.

<plot_test1.m>

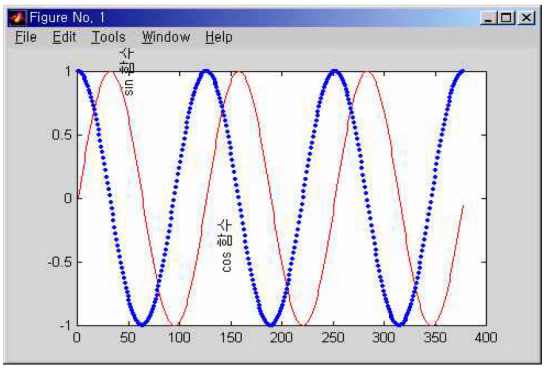
```
t=0:0.05:6*pi;
y1=sin(t);
y2=cos(t);
plot(y1,'r')
hold on
plot(y2,'b.')
text(50,0.8 ,'sin 함수');
text(100,-0.4, 'cos 함수');
hold off
```



수직방향으로 텍스트를 삽입하기 위해서는 위의 m-파일에서 plot 명령 윗줄에

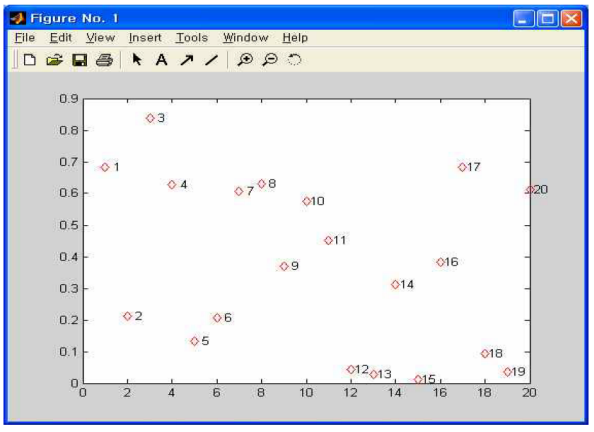
```
set(gcf, 'defaulttextrotation,90')
```

을 추가하면 된다.



각각의 데이터 그래프로 그린 후, 데이터 점의 index를 표시하는 방법은 아래와 같다.

```
》 y=rand(1,20);
》 x=1:length(y);
》 n=num2str(x');
》 plot(x,y,'dr')
》 text(x+0.2,y,n)
```

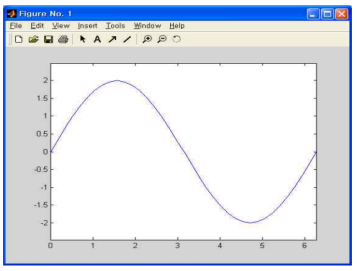


1.7 axis 함수

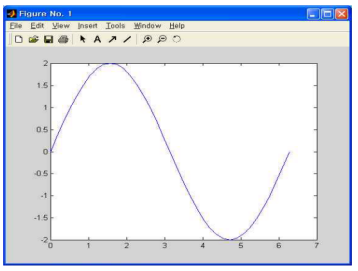
함수 'axis'는 사용자가 원하는 대로 그래프의 종횡비, 방향 및 배율 따위를 조정할 수 있도록 여러 가지 선택 사항들을 갖추고 있다. 대개의 경우에 MATLAB은 도시할 자료의 최대 및 최소값을 찾아 줄뿐만 아니라 적당한 그래프 상자와 축의 이름을 선택해 주지만, 사용자가 원할 때에는 다음과 같이 축의 한계 값을 새로이 설정할 수 있다.

2차원 : axis([xmin xmax ymin ymax])
3차원 : axis([xmin xmax ymin ymax zmin zmax])

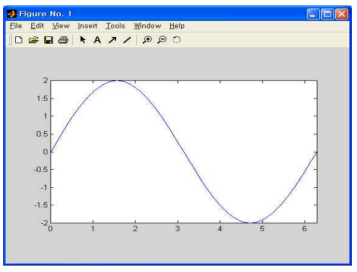
```
》 t=[0:10:360]*(pi/180);
》 x=2*sin(t);
》 plot(t,x)
》 axis equal
》 axis image
》 axis square
```



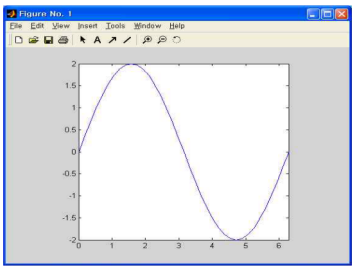
equal



default



image



square

1.8 Peaks M-파일

MATLAB에서는 다음과 같이 3개의 극대점과 2개의 극소점을 가지면서 두 개의 변수로 구성된 함수를 이용하여, 원하는 크기의 행렬을 구성하여 주는 'peaks'라는 m-파일을 제공하고 있다.

$$f(x,y)=3(1-x)^2e^{-x^2-(y+1)^2}-10\left(\frac{x}{5}-x^3-y^5\right)e^{-x^2-y^2}-\frac{1}{3}e^{-(x+1)^2-y^2}$$

이 'peaks' m-파일은 -3부터 3 사이의 x 및 y 값을 사용하여 위의 함수로부터 구해지는 값들을 원소로 하는, 지정한 차원만큼의 정방행렬을 구성해 준다.

```
》peaks(10);
```

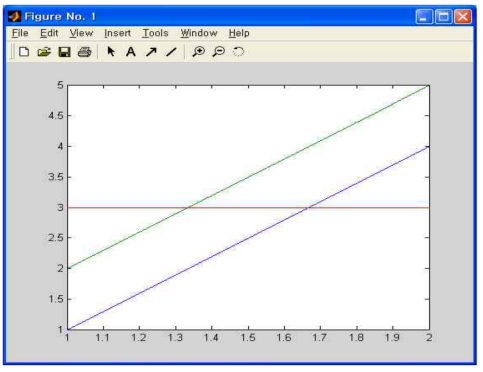
이라고 입력하면 10×10의 정방행렬을 구성하여 표시해 준다. 만약 입력 인수를 지정하지 않으면 기본값인 49를 사용하여 49×49의 정방행렬을 생성한다.

1.9 행렬의 도시법

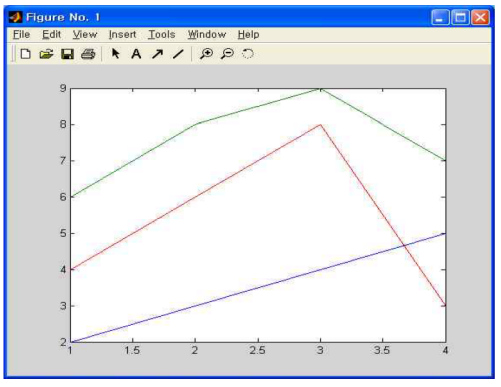
차원이 m×n인 임의의 행렬 X를 plot 명령으로 도시하면, X의 각 열의 원소들을 자료로 하는 선을 그리는데, 이때 x축의 좌표는 1부터 n까지 각 열의 번호로 주어지며 각 선의 자료점은 m개가 된다.

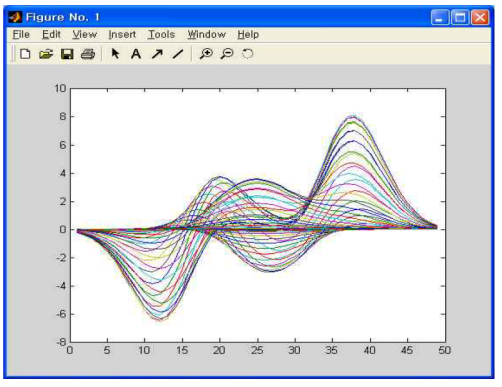
```
》X=[1 2 3; 4 5 3]
    X= 1   2   3
```

```
     4     5     3
》plot(X)
```




```
》Y=[2 6 4 ;3 8 6;4 9 8; 5 7 3]
   Y =
      2     6     4
      3     8     6
      4     9     8
      5     7     3
》plot(X)
》plot(peaks)
```



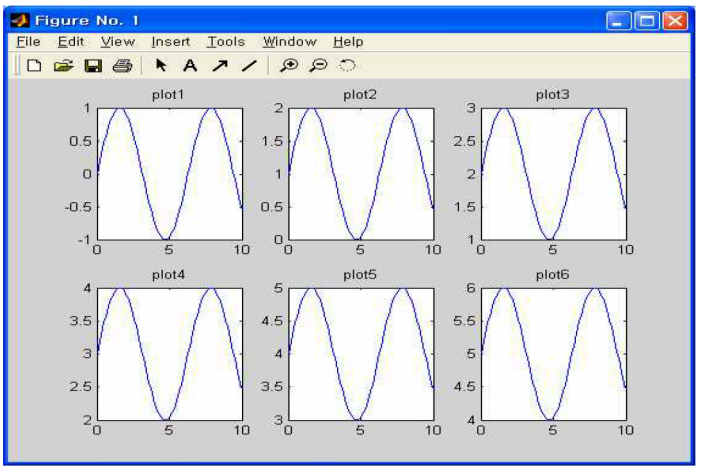


1.10 그래프의 분할

화면에 몇 개의 그래프를 출력하고자 할 경우에는 'subplot(m,n,k)'명령어를 사용한다. m은 그려질 그래프의 행의 개수를 나타내고, n은 열의 개수를 나타낸다. 또한 k는 subplot할 인덱스를 지정하는데 1부터 m*n까지의 크기를 같게 된다.

```
 <plot_test3>
t=0:0.1:10;
y=sin(t);
subplot(2,3,1), plot(t,y),title('plot1')
```

```
subplot(2,3,2), plot(t,y+1),title('plot2')
subplot(2,3,3), plot(t,y+2),title('plot3')
subplot(2,3,4), plot(t,y+3),title('plot4')
subplot(2,3,5), plot(t,y+4),title('plot5')
subplot(2,3,6), plot(t,y+5),title('plot6')
```



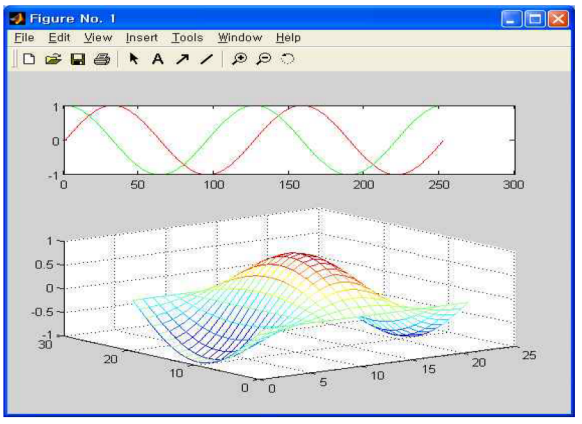
크기가 서로 다른 그래프를 한 화면에 출력하고 할 경우에는 'axes'명령어를 사용한다. 'axes'명령어는 다음과 같이 좌표를 설정하게 된다.

```
axes('position',[left bottom width height])
```

여기서 그림창의 기본값은 왼쪽 아래의 좌표값은(0,0)이고, 오른쪽 위의 좌표값은 (1,1)로 설정되어 있다.

<plot_test4.m>

```
clf
t=0:0.05:4*pi;
y=sin(t);
c=cos(t);
axes('position',[0.1 0.7 0.8 0.2]);
plot(y,'r-');
hold on
plot(c,'g');
axes('position',[0.1 0.1 0.8 0.5])
mesh(sphere)
```

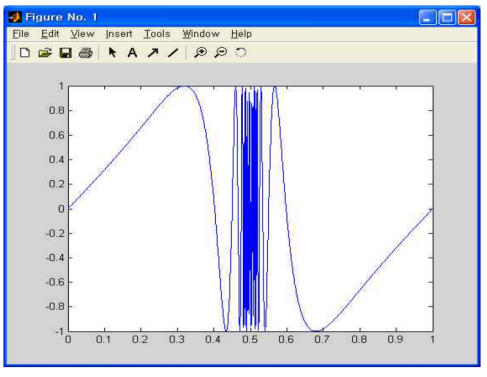


1.11 수학적 함수의 도시

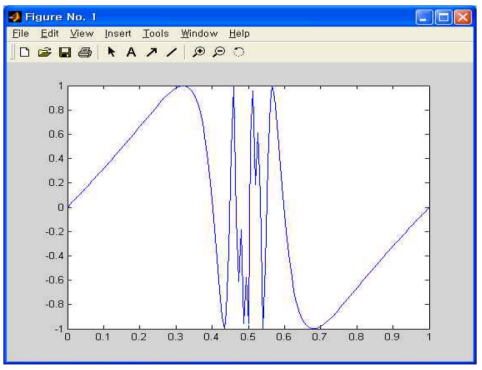
'y = f(x)'와 같은 함수를 도시하는 방법에는 여러 가지가 있을 수 있는데, 가장 단순하지만 효과적인 방법으로는 원하는 구간을 수천 개의 작은 구간으로 분할하여 각 점마다 함수값을 구한 후에 그 값들을 도시하는 것이다.

다음과 같은 함수는 0≤x≤1에서 무한대로 진동하는 결과를 나타낸다.

```
》 x=(0:1/1500:1)';
》 plot(x,sin(tan(pi*x)))
```



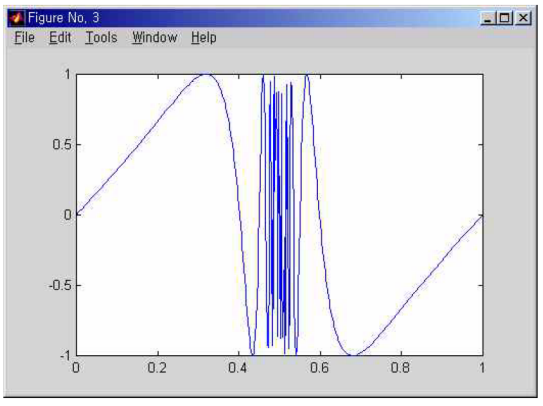

```
》 x=(0:1/150:1)';
》 plot(x,sin(tan(pi*x)))
```

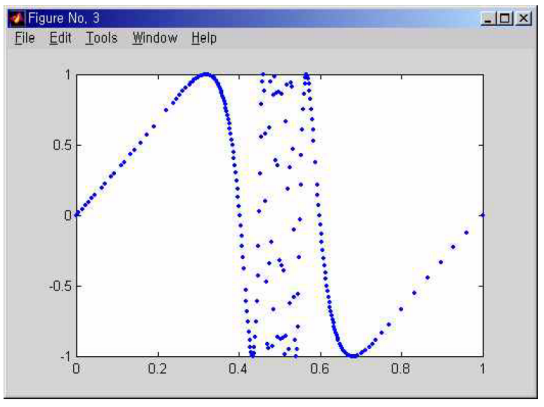


그러나 이러한 방법은 구간을 얼마나 많이 작게 쪼개어 계산하느냐에 따라 값의 정확도에 큰 차이가 발생하게 되며, 또한 위의 경우와 같이 부분적으로 많은 점들이 밀집되어 있는 경우에는 상당히 비효율적이다.

이러한 단점에 대한 보완책으로 함수 'fplot'을 사용하는 방법이 있다. 'fplot' 함수는 선택적으로 함수의 값의 취하는데, 함수 값의 변화가 심한 부분에서는 많은 점들을 선택하여 계산을 하고 함수 값의 변화가 적은 부분에서는 몇 개의 점만을 선택하여 계산을 수행한다.

```
   <sin_tan.m>
function y=sin_tan(x)
y=sin(tan(pi*x));

》fplot('sin_tan',[0 1])
```





1.12 그래픽 입력

MATLAB에서는 'ginput' 함수를 사용함으로써 마우스나 방향키를 사용하여 그래프 상의 한점을 선택하는 것이 가능하다. 이 함수는 지시자(pointer)의 현재 위치 또는 마우스나 키보드의 키가 눌려졌을 때의 위치를 좌표값으로 되돌려 준다.

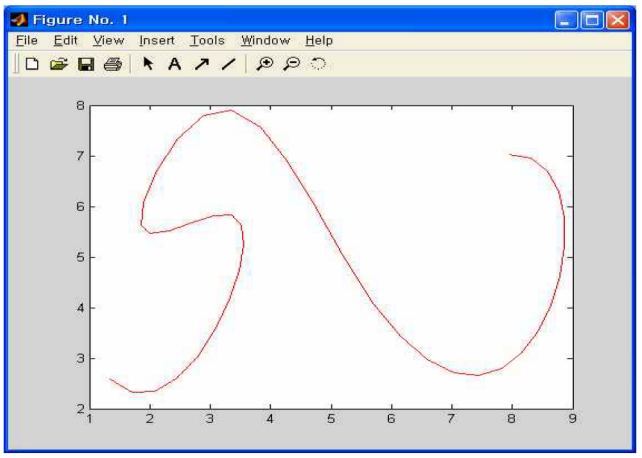
다음의 예는 2차원에서 내삽에 의하여 매끄러운 곡선을 구성하는 'spline' 함수를 통하여

'ginput'을 사용하는 방법에 대하여 설명하고 있다.

먼저 함수 'ginput'을 통하여 평면상의 일련의 점들 (x,y)를 지정하면, 원래의 두 점 사이의 간격에 대하여 1/5의 간격으로 내삽된 각각의 점들을 통과하는 운형곡선(spline)이 그려진다.

<mouse_test.m>

```
% 그림창을 지우고 그래프의 크기를 지정
clf                          % 그림창을 지움
axis([0 10 0 10])
hold on

 % 점의 좌표에 대한 목록을 빈 상태로 지정
x = [ ];                     % x 좌표값의 차원
y = [ ];                     % y 좌표값의 차원
n = 0;                       % 점이 찍힌 순서의 초기값

% 오른쪽 마우스 버튼이 눌리기 전까지 계속적으로 그래프 상에 찍히는 점을 기록
leftright = 1;               % 왼쪽 또는 오른쪽 마우스 버튼의 눌림 여부
while leftright == 1
   [xi,yi,leftright] = ginput(1); % 마우스 버튼 입력
   plot(xi,yi,'go')          % 그래프 상에 점의 표시
   n=n+1;                    % 찍힌 점의 번호 증가
   x(n,1)=xi;                %  x 좌표값의 입력
   y(n,1)=yi;                %  y 좌표값의 입력
end

% 두 점 사이를 5개의 구간으로 분할한 후 각각에 대하여 운형 곡선(spline)으로 연결
t=1:n;
tt=1:0.2:n;
xx=spline(t,x,tt);
yy=spline(t,y,tt);

% 내삽된 곡선을 그림
plot(xx,yy,'r-')
hold off
```

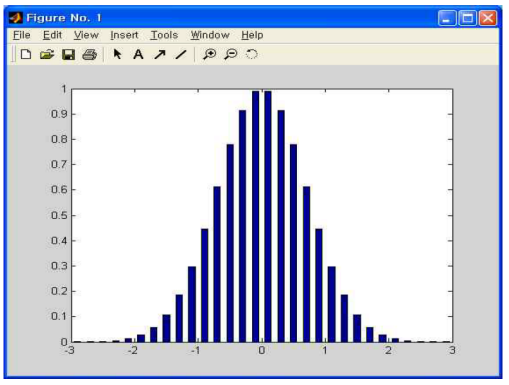


1.13 2차원 그래프들의 종류

① bar(막대 그래프) - 일차적인 막대 그래프의 형태로 표시

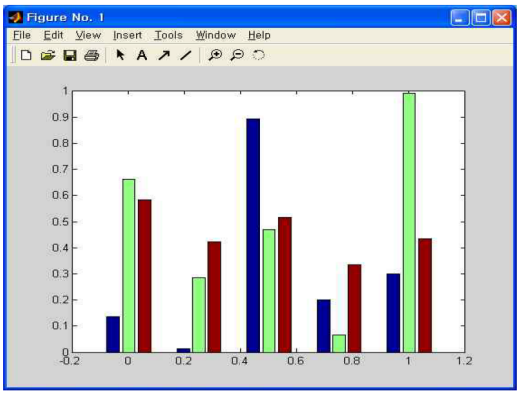
》 x=-2.9:0.2:2.9;

》 bar(x,exp(-x.*x),0.5)



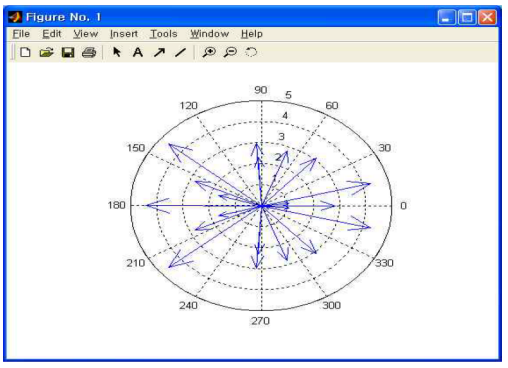
》 bar(0:.25:1,rand(5,3),0.8)

》 bar(rand(5,3),'stacked'), colormap(cool)



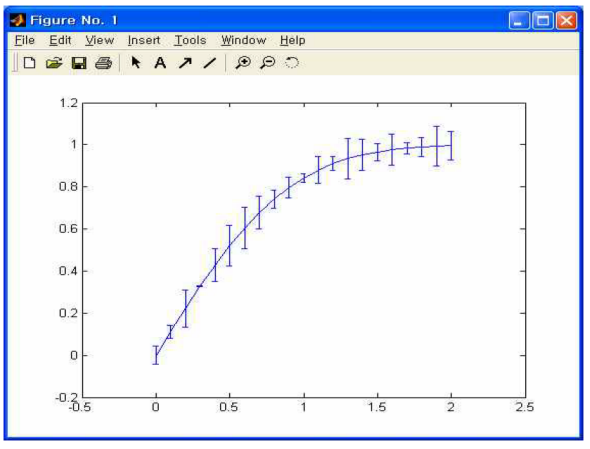
② compass(화살표 그래프) - 복소수 또는 벡터를 표시할 때와 같이 각도와 크기를 가지면서 원점으로부터 방사되는 화살표를 사용하여 자료를 표시

```
》z=eig(randn(20,20));
》compass(z)
```



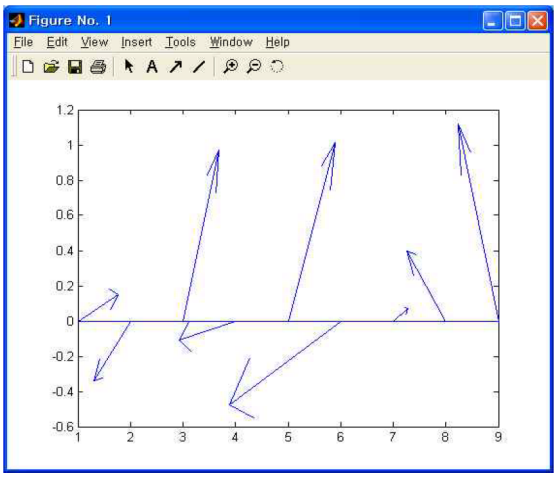
③ errorbar(오차 막대) - 오차 막대를 사용한 그래프를 작성

```
》x=0:0.1:2;
》y=erf(x);
》e=rand(size(x))/10;
》errorbar(x,y,e)
```



④ feather(화살표 그래프) - 크기와 각도를 갖는 화살표를 사용한다는 점은 'compass' 그래프와 같지만, 화살표의 시작점이 원점에 위치하지 않고 x축을 따라 일정한 자료의 개수만큼 분포된다는 점이 다르다.

```
》z=randn(3,3) + randn(3,3)*j
   z =
       -0.0715 - 0.5081i   0.1798 + 0.6250i   0.8252 + 0.4344i
        0.2792 + 0.8564i  -0.5420 - 1.0473i   0.2308 - 1.9171i
        1.3733 + 0.2685i   1.6342 + 1.5357i   0.6716 + 0.4699i
》feather(z)
```



⑤ fplot(함수 그래프) - 함수와 구간을 지정하면 그 구간에서의 함수값을 그래프로 그려준다.

》 fplot('tanh',[-2 2])

⑥ hist(막대 그래프) - 'bar'와 같은 형태의 막대 그래프이지만 통계처리에 사용되는 히스토그램(histogram)을 그려 준다.

》 x=-2.9:0.1:2.9;

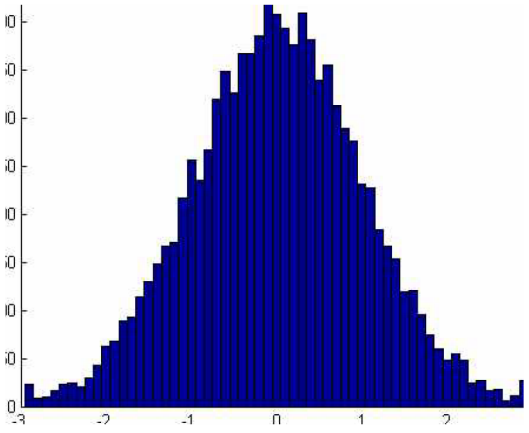
》 y=randn(10000,1);

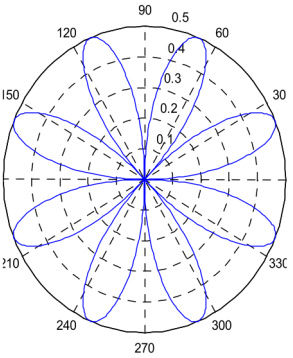
》 hist(y,x)

⑦ polar(극좌표 그래프) - 각도와 반경을 갖는 극좌표의 형태를 사용하여 자료점들을 연속적으로 나타낸다.

》 t=0:0.01:2*pi;

》 polar(t,sin(2*t).*cos(2*t))



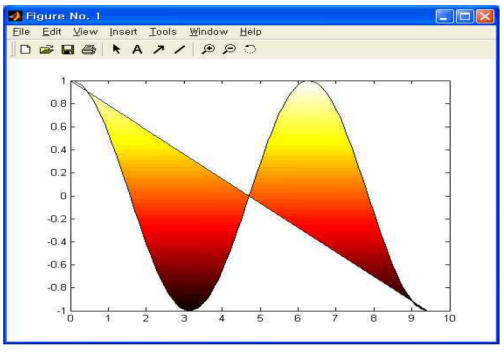


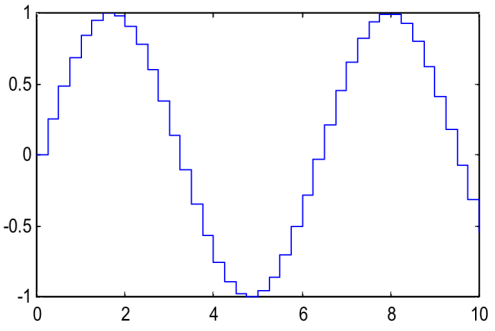
⑧ fill(다각형 그래프) - 일반적인 형태의 선(line) 그래프에서 첫 번째 자료점과 마지막 자료점을 연결하여 다각형을 만든 후 내부를 원하는 색상으로 채워 준다.

```
》 t=0:0.1:3*pi;
》 x=cos(t);
》 fill(t,x,'r')
》 colormap(hot)
》 fill(t,x,x)
》 colormenu
```

⑨ stairs(계단형 그래프)

```
》 x=0:0.25:10;
》 stairs(x,sin(x))
```





⑩기타 - quiver(화살표 그래프), rose(원형 막대 그래프)

2. 3차원 그래픽

MATLAB 프로그램에서 제공하는 3차원 그래프는 격자 생성, 등고선, 영상처리, 은선 제거, 동화상과 색 지정 및 관찰점의 이동 등과 같은 기능을 가지고 있으며, 좌표축이 3개인 3차원의 데이터를 처리할 수 있는 독자적인 도형용 함수를 가지고 있다.

3차원 그래프에의 추가 사항은 다음과 같다.

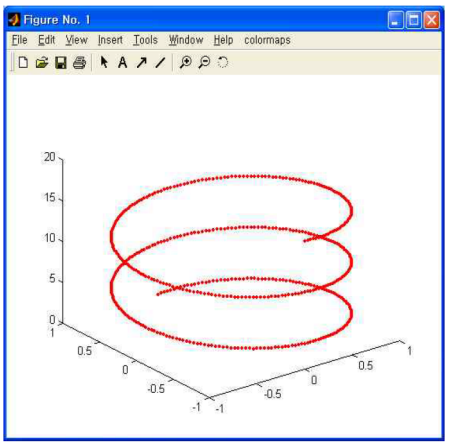
zlabel : z축에 대한 축 이름 지정.
clabel : 윤곽선(contour) 그래프의 경우에 각각의 윤곽선에 대한 기호를 붙여 줌.
view : 방위각과 고도를 설정하거나 변환 행렬을 지정함으로써 관찰 위치 설정.
viewmtx : 직각 투영(orthogrphic) 및 원근 투영(perspective) 변환에 대하여 4X4의 변환 행렬 계산

2.1 선 그래프

2차원 그래프의 'plot' 명령어에 해당하는 것으로 3차원 그래프에서는 'plot3' 명령어가 있다. 'plot3(x,y,z)' 명령어는 기본적으로 3개의 요소를 필요로 하며, x와 y의 값에 대해서 z의 값을 그리는 것으로써, 벡터 x, y, z 대신에 행렬 X, Y, Z를 지정해도 된다. 행렬일 경우에는 대응하는 열마다 그래프가 그려진다.

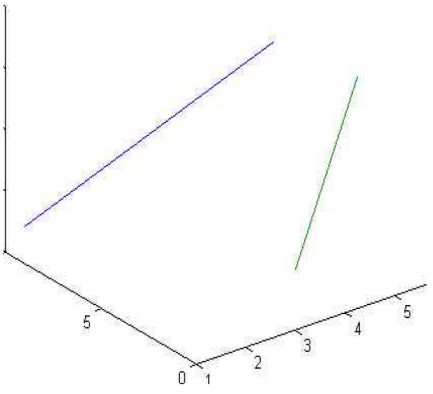
x ,y, z가 서로 차원이 같은 벡터일 경우, x, y, z에 의하여 표시되는 점들을 지나는 3차원 공간 상의 선을 구성하게 된다.

```
》t=0:pi/100:5*pi;  % t의 차원 - 1×501
》plot3(sin(t),cos(t),t,'.r')
```



또한 x, y, z가 차원이 같은 3개의 행렬일 경우, 각각 행렬 x, y, z의 열의 원소들에 의하여 표시되는 점들을 통과하는 선들을 그려준다.

```
》x=[1 3;6 5], y=[9 0; 9 2], z=[3 5; 11 16]
》plot3(x,y,z)
```



2.2 그물 격자 그래프(mesh)

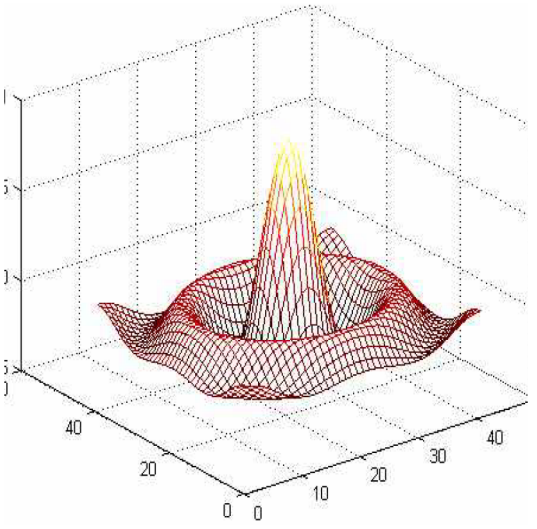
x, y 평면에서의 눈금 격자(grid line)에 대하여 z 좌표를 지정함으로써 3차원 표면으로 구성되는 그물 격자(meshgrid)가 있는 면을 생성할 수 있다. 그물 표면(mesh surface)은 수치적인 형태로 표현하기에는 너무 큰 행렬을 나타내거나, 두 개의 변수로 구성된 함수를 그래프로 나타내는데 유용하다. 'meshgrid' 명령어의 사용법은 아래와 같다.

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

이것은, 두 벡터 x와 y에 의하여 지정된 정의역을 행렬 X, Y로 변환하고 하는데, 이때 행렬 X의 행은 벡터 x와 같고 행렬 Y의 열은 벡터 y와 같다.

아래의 예는 meshgrid를 이용하여 그물격자 그래프를 도시한 것이다.(멕고 모자, sombrero)

```
》x=-10:.5:10;
》y=x;
》[X,Y]=meshgrid(x,y);
》r=sqrt(X.^2+Y.^2)+eps;    % eps : Z를 구할 때 분모의 r값으로 인하여 불능(NaN)인
》Z=sin(r)./r;              %          경우가 발생하지 않도록 한다.
》mesh(Z)
```



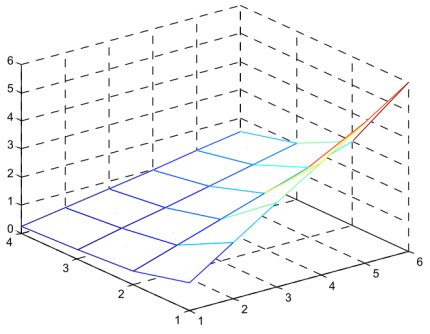
위의 프로그램에 아래와 같이, 5개의 원소를 갖는 x를 가지고 예를 들어 보았다.

```
x =
    -2    -1     0     1     2

y =
    -2    -1     0     1     2

X =
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2

Y =
    -2    -2    -2    -2    -2
    -1    -1    -1    -1    -1
```

```
     0     0     0     0     0
     1     1     1     1     1
     2     2     2     2     2

r =
    2.8284    2.2361    2.0000    2.2361    2.8284
    2.2361    1.4142    1.0000    1.4142    2.2361
    2.0000    1.0000         0    1.0000    2.0000
    2.2361    1.4142    1.0000    1.4142    2.2361
    2.8284    2.2361    2.0000    2.2361    2.8284

z =
    0.1089    0.3518    0.4546    0.3518    0.1089
    0.3518    0.6985    0.8415    0.6985    0.3518
    0.4546    0.8415    1.0000    0.8415    0.4546
    0.3518    0.6985    0.8415    0.6985    0.3518
    0.1089    0.3518    0.4546    0.3518    0.1089
```

<plot3_test1.m>

```
clear all
m=4;n=6;
for i=1:m
   for j=1:n
      z(i,j)=j/i;
   end
end
mesh(z)
```

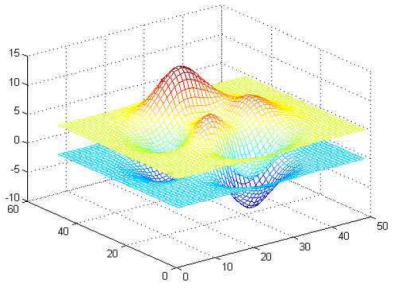


2.3 은선 제거

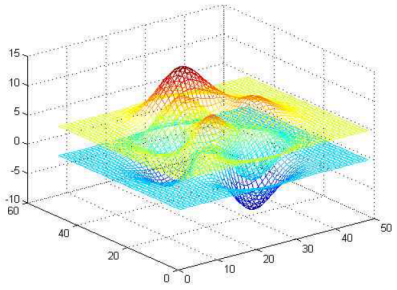
MATLAB 프로그램에서는 3차원 도형의 그물 격자에 대한 은선(hidden)을 제거하기 위해서 'hidden' 명령어를 사용한다. 이 명령어는 'hidden on'과 'hidden off'와 같이 토글 방식으로 사용되는데 'hidden off'를 사용하면 그물 격자를 투과해서 다른 그래프의 형상을 관찰할 수 있다.

```
》mesh(peaks+5)
》hold on
```

》 hidden on
》 mesh(peaks)
》 hold off
》 hidden off



Hidden On

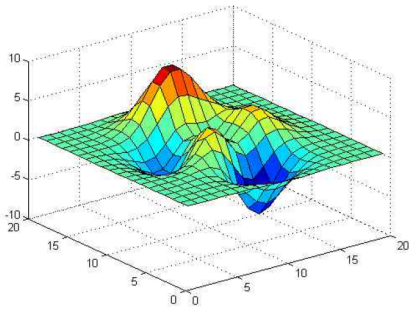


Hidden Off

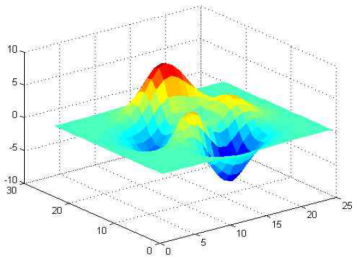
2.4 표면 그래프(surf)

'mesh'가 그래프를 구성하는 면에서 색상을 가지지 않는 반면, 'surf'는 색상이 지정된 평면(facet)들로 구성된 표면을 원근법을 사용하여 보여준다. 대개 이러한 작은 평면들(facets)은 검은 색의 그물 격자로 외곽선을 두른 일정한 색상의 사각형들이지만 'shading' 함수를 사용하면 그물 격자를 제거할 수 있으며, 작은 평면들 사이에도 두 색상간의 내삽(interpolation)에 의한 묘영법을 적용할 수 있다. '

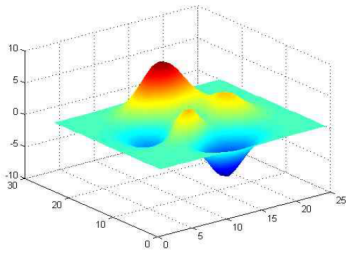
》 surf(peaks(25))



》 shading flat
》 shading interp



flat



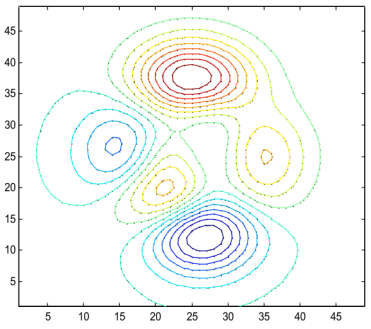
interp

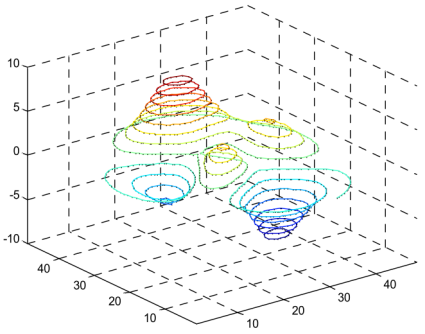
2.5 윤곽선 그래프(contour, contour3)

윤곽선(contour)의 형태를 나타낼 수 있는 것으로 행렬 입력으로부터 값이 같은 점들을 연결하는 선을 그려준다. 2차원 윤곽선 그래프는 'contour' 명령어를 사용하고, 3차원 윤곽선 그래프는 'contour3' 명령어를 사용한다.

아래의 예는 'peaks' 함수에 의하여 구성된 행렬에 대하여 15개의 윤곽선을 갖는 그래프를 2차원과 3차원으로 그린 것이다.

```
》 contour(peaks,15)
》 contour3(peaks,15)
```





2.6 'surf'와 'mesh'의 변형들

```
》 surfc(peaks(25))  % 2차원 contour 포함
》 meshc(peaks(25)) % 2차원 contour 포함
》 surfl(peaks(25),[-15 45]) % 명암을 갖는 표면 표현(방위각 -15도, 고도 45도)
```

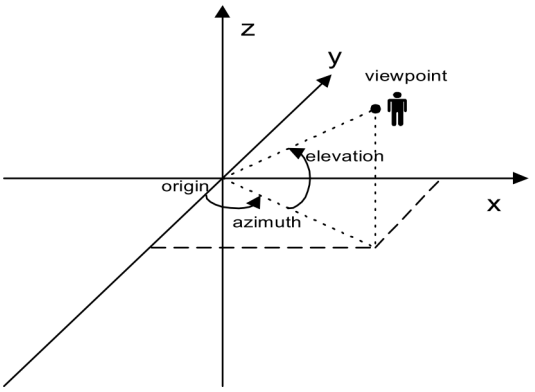
2.7 관찰점(View Point)

3차원 그래프의 관찰 방향을 사용자가 원하는 경우에 어떠한 관찰점에서 그래프의 형상을 관찰하도록 지정할 수 있다. 이와 같은 것을 수행하는 것으로 'view' 명령어가 있다. 이러한

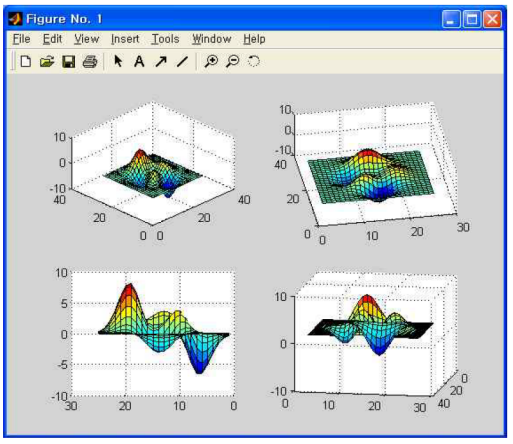
'view' 명령어에 대한 일반적인 형태는 다음과 같다.

view(azimuth, elevation)

여기서 방위각(azimuth)은 반시계 방향으로 양의 값을 갖고, 고도(elevation)는 x, y 평면의 위쪽이 양이 되고 아래쪽이 음이 된다.



》 colormap(hsv)
》 subplot(221), surf(peaks(25)), view(-45,45)
》 subplot(222), surf(peaks(25)), view(-10,60)
》 subplot(223), surf(peaks(25)), view(-90,0)
》 subplot(224), surf(peaks(25)), view(-10,-15)



3. Handle Graphics

Mathworks사에서는 MATLAB에서 이용되는 그래프들의 체계를 위해 'Handle Graphics System'이라는 새로운 용어를 만들어 냈다. 또한 컴퓨터 screen을 포함한 개개의 모든 그림창(figure window)들과 좌표축들 그리고 patch(그림 성분)들은 'graphics object'라고 부른다. 앞으로 배울 line, text, image, user interface controls, 그리고 user interface pulldown menus 등등은 모두 'graphics object'이다.

'Graphics object'는 모두 9가지로 분류되는데, Handle graphics system에서 복잡한 그림을 만들기 위해 사용되는 성분들이라고 생각하면 된다.

figure object
line object
axes object
patch object
surface object
text object
image object
uicontrol object
uimenu ojbect

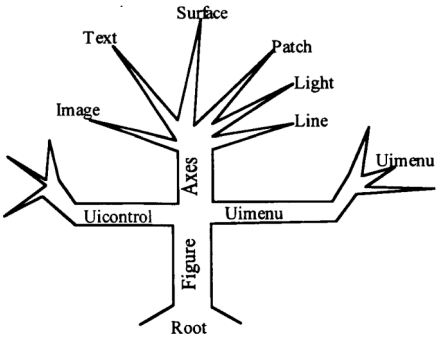


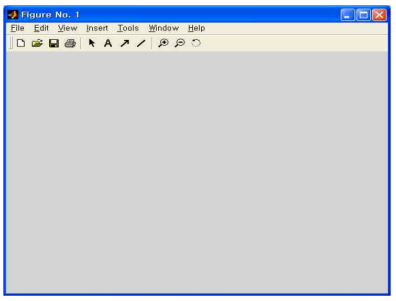
Figure 3-1: Graphics object hierarchy.

① MATLAB 그래프들은 객체들의 조합으로 만들어진다.
② 객체들은 계층도에 의해 분류된다.
③ 상위객체는 하위 객체들을 제어할 수 있다. 그러나 하위 객체는 상위 객체에 영향을 주지 못한다.
④ 객체들은 같은 level의 객체들을 소유할 수 없다.

1) Handle Graphics

각각의 MATLAB 그래프 함수는 객체들을 제어 할 수 있도록, 새로 만들어지는 객체마다 고유한 숫자를 반환한다. 이들 고유한 숫자들을 handle이라고 부른다. handle은 보통 figure object인 경우에는 양의 정수를 반환하고, 다른 하위 level에서는 부동소수점(floating point number)을 반환한다.

```
》h_fig=figure
   h_fig = 1
```



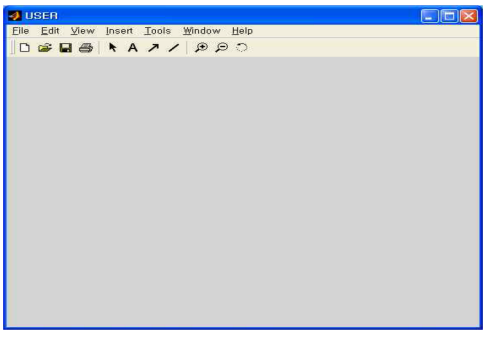
》 h_fig=figure(3);

일단, handle value가 할당된 객체는 자신의 handle value를 바꿀 수 없으며, 해당 객체가 사라질 때까지 handle value는 보존된다. 또한, 같은 종류의 객체라도 handle value는 서로 다르다. 그래야만, 각각의 객체를 독립적으로 제어할 수 있기 때문이다. 그러나, root object는 유일하기 때문에 handle value는 '0'의 값을 갖는다.

'set.m' 함수는 handle을 이용하여 각각의 객체 성질(propertyname)을 조정해주고 'get.m' 함수는 해당 객체의 성질에 할당된 값(propertyvalue)을 얻게 해주는 함수이다.

》 set(handle, propertyname, propertyvalue)
》 propertyvalue=get(handle, propertyname)

》 close all
》 h_fig1=figure(1);
》 set(h_fig1,'numbertitle','off','name','USER')



numbertitle, name : 주어진 figure의 propertyname
off, USER : 주어진 figure의 property value

만일 handle이 h_fig1인 figure의 크기를 알고 싶다면, 다음과 같이 하면 된다.

》 h_size=get(h_fig1,'position')
h_size = 360 808 291 126

```
》 set(h_fig1)
        BackingStore: [ {on} | off ]
        CloseRequestFcn
        Color
        Colormap
        CurrentAxes
        CurrentObject
        CurrentPoint
        Dithermap
        DithermapMode: [ auto | {manual} ]
        DoubleBuffer: [ on | {off} ]
        FileName
        IntegerHandle: [ {on} | off ]
        InvertHardcopy: [ {on} | off ]
        KeyPressFcn
        MenuBar: [ none | {figure} ]
        MinColormap
                :
                :
                :
        ButtonDownFcn
        Children
        Clipping: [ {on} | off ]
        CreateFcn
        DeleteFcn
        BusyAction: [ {queue} | cancel ]
        HandleVisibility: [ {on} | callback | off ]
        HitTest: [ {on} | off ]
        Interruptible: [ {on} | off ]
        Parent
        Selected: [ on | off ]
        SelectionHighlight: [ {on} | off ]
        Tag
        UIContextMenu
        UserData
        Visible: [ {on} | off ]

》 get(h_fig1);
```

2) Low_level functions

Low_level functions는 객체(object) 지향적인 MATLAB의 graphics 체계를 의미하는데, 이들 low_level graphic functions를 이용하여 모든 high_level graphic functions를 만들 수 있으며, 또한, 사용자가 원하는 다른 graphic functions를 작성할 수 있다.

① Root Object

Root object에 해당하는 직접적인 함수는 없다. 또한, 다른 low_level functions보다 상당히 적은 propertyname/propertyvalue pairs를 갖는데, MATLAB command window에서도 'set.m', 'get.m' 함수를 이용하여 직접 확인 할 수도 있다.

```
》 set(0)
```

위와 같이 명령을 실행하면 root object에 대한 자료를 볼 수 있다.

Diary : [on | off]
ShowHiddenHandles : [on | {off}]

'[ ]'은 해당 propertyname(여기서는 diary와 showhiddenhandles을 의미한다.)에서 이용할 수 있는 지정어 (즉, property value)이다. 다시 말해서, 해당 propertyname을 제어 할 수 있는 지정어로서 다른 비슷한 용어를 사용하면 안되고, 반드시 지정해준 'on, off'만 사용하여야 한다. { }는 default propertyvalue를 의미하는데 MATLAB이 자동으로 propertyvalue를 설정한 경우를 의미한다. 이것은 'set.m' 함수를 이용해서 'default propertyvalue'를 바꿀 수도 있다.

```
》get(0,'units')
   ans = pixels
》set(0,'units','nor')
》get(0,'units')
   ans = normalized
```

ShowHiddenHandles: [on | {off}]
Units: [inches | centimeters | normalized | points | pixels | characters]

ButtonDownFcn

한 줄 띄운 아래에 나오는 모든 propertyname은 'universal properties'를 의미한다. 즉 모든 object가 다 갖고 있는 공통된 properties이다. ButtonDownFcn은 옆에 아무런 지정어나, 또는 사용할 수 있는 format(예를 들어서, 'string','number' 등등)이 없다. 그것은 현 object인 root object에서는 사용할 수 없는 propertyname을 의미한다.

② Figure Object
아래는 새로운 그림창을 생성하고 그림창의 handle을 변경하는 예이다.

```
》h=figure
   h = 1
》set(h,'numbertitle','off','name','USER','menubar','none','resize','off')
```


```
》fig_size=get(h,'pos')
    fig_size =   150    410    560    420
》get(h,'units')
    ans = pixels
```

MATLAB에서 figure의 크기는 screen의 왼쪽 하단을 원점으로 하여 폭과 높이로 정해진다. 즉, [left bottom width height]와 같은 형식을 갖는다. 또한, MATLAB에서 지원하는 모든 units 체계는 window의 왼쪽하단을 원점으로 정하고 측정하는데, Normalized units는 왼쪽 하단이 (0,0)에 해당하면, 오른쪽 상단은 (1,1)에 해당한다.

③ Axes Obect

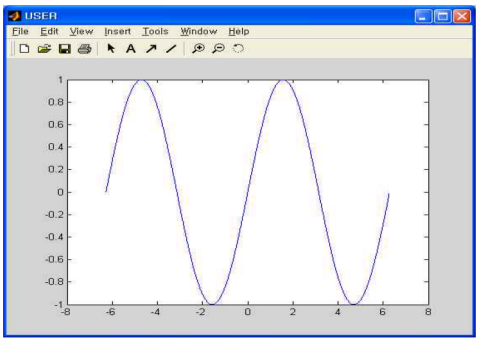
사용자가 여러 개의 객체를 생성한 후, 원하는 객체의 특성을 바꾸고자 할 때는 직접 해당 객체의 handle value를 반환하는 function을 이용할 수 있다.

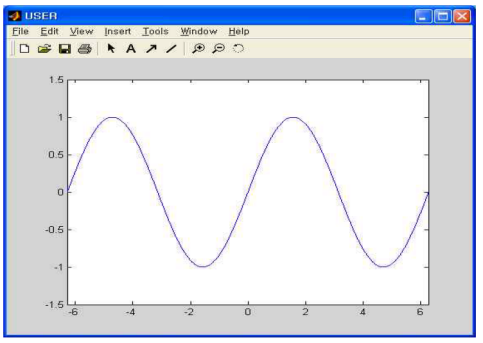
gcf(get current figure) : 현재 active 상태인 figure handle을 반환한다.

gca(get current axes) : 현재 active 상태의 figure 안에 있는 axes의 해당 handle을 반환한다.

gco(get current object) : active figure에서 가장 최근에 생성된 객체 또는, mouse로 가장 최근에 클릭된 객체의 handle을 돌려준다.

```
》t=-2*pi:1/100:2*pi;
》y=sin(t);
》plot(t,y)
```





》 set(gca,'xlim',[min(t) max(t)],'ylim',[-1.5 1.5])

propertyname : xlim, ylim, zlim

propertyvalue : 각각의 좌표축에 대한 경계를 정한다.

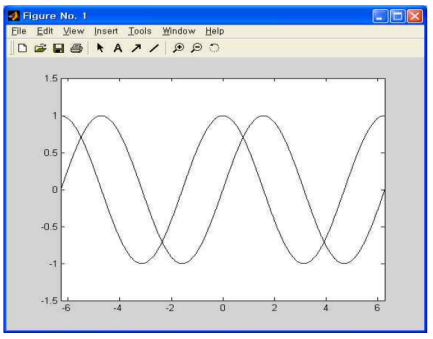
》 t=-2*pi:1/100:2*pi;

》 y1=sin(t);

》 y2=cos(t);

》 plot(t,y1,'k',t,y2,'k')

》 set(gca,'xlim',[min(t) max(t)],'ylim',[-1.5 1.5])



》 set(gco,'color','r','linewidth',15)

propertyname : color

propertyvalue : 선의 색을 변경하여 준다.

》 set(gca,'nextplot','add','xlim',[min(t) max(t)])

propertyname : nextplot

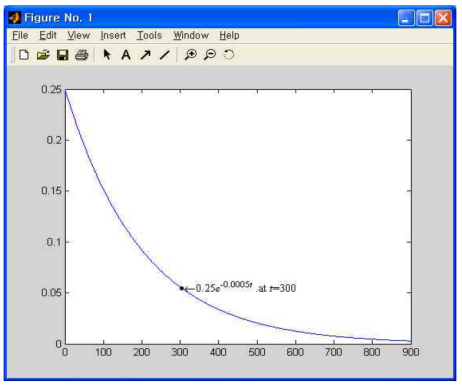
propertyvalue : 'add'인 경우에는 'hold on' 효과를,
'replace'이면 'hold off'의 효과를 준다.

④ Text Object

Text object의 전형적인 사용법(syntax)는 다음과 같다.

》 text_handle = text(x,y,z,'propertyname','propertyvalue')

```
》 t=0:900;
》 plot(t,.25*exp(-0.005*t));
》 h=text(300,.25*exp(-0.005*300),...
      '\bullet\leftarrow\fontname{times}0.25{\ite}^{-0.0005{\itt}} .at {\itt}=300');
```



이들 문자들은 'Latex' 문자 체계들로써, text object의 interpreter가 'tex'인 경우에만 특수문자로 변환한다.

```
》 get(h,'interpreter')
     ans = tex
```

만일, 다음과 같이 명령을 하면, 'string'이 특수문자로 변환되지 않고, typing한 그대로 figure안에 display된다.

```
》 set(h,'interpreter','none');
```

font의 종류는 다음과 같다

\bf	bold font
\it	italics font
\sl	oblique font
\rm	normal font
\fontanme{fontname}	위에 주어진 font외의 다른 font들

이들 특수문자 지정어들은 '{ }'으로 구분한다. 또한, 아래 첨자는 '_'으로 만들고, 윗 첨자는 '^'으로 만든다.

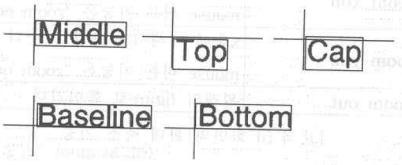
Character	Symbol	Character	Symbol	Character	Symbol
\alpha	α	\upsilon	υ	\sim	~
\beta	β	\phi	ϕ	\leq	≤
\gamma	ɣ	\chi	χ	\infty	∞
\delta	δ	\psi	ψ	\clubsuit	♣
\epsilon	ε	\omega	ω	\diamodsuit	◆
\zeta	ζ	\Gamma	Γ	\heartsuit	♥
\eta	η	\Delta	Δ	\spadesuit	♠
\theta	Θ	\Theta	Θ	\leftrightarrow	↔
\vantheta	υ	\Lambda	Λ	\leftarrow	←
\iota	ι	\Xi	Ξ	\uparrow	↑
\kappa	κ	\Pi	Π	\rightarrow	→
\lambda	λ	\Sigma	Σ	\downarrow	↓
\mu	μ	\Upsilon	Y	\circ	°
\nu	ν	\Phi	Φ	\pm	±
\xi	ξ	\Psi	Ψ	\geq	≥
\pi	π	\Omega	Ω	\propto	∝
\rho	ρ	\forall	∀	\partial	∂
\sigma	σ	\exist	∃	\bullet	•
\varsigma	ζ	\ni	Θ	\div	÷
\tau	τ	\cong	≅	\neq	≠
\equiv	≡	\approx	≈	\aleph	א
\lm	ℑ	\Re	ℜ	\wp	℘
\otimes	⊗	\oplus	⊕	\oslash	∅
\cap	∩	\cup	∪	\supseteq	⊇
\supset	⊃	\subseteq	⊆	\subset	⊂
\int	∫	\in	∈	\o	o

문자열(string)의 정렬을 관장하는 propertyname으로는 'HorizontalAlignment', 'VerticalAlignment'가 있다.

Text Object	
propertyname	propertyvalue
HorizontalAlignment VerticalAlignment	{left} | center | right top | cap | {middle} | baseline | Bottom

문자열의 정렬 방법은 아래와 같다.

(a) "HorizontalAlignment"의 경우.

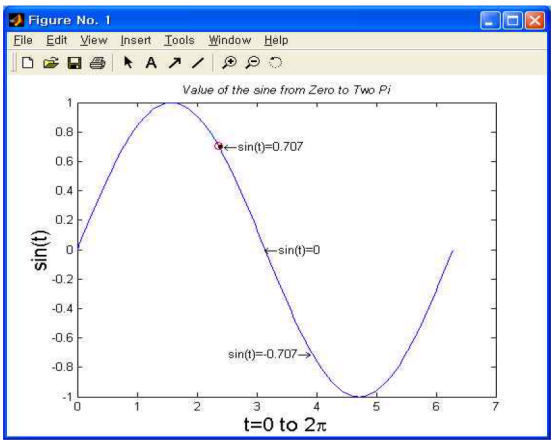


(b) "VerticalAlignment"의 경우.

[그림4-16]문자열의 정렬 방법

<text1.m>

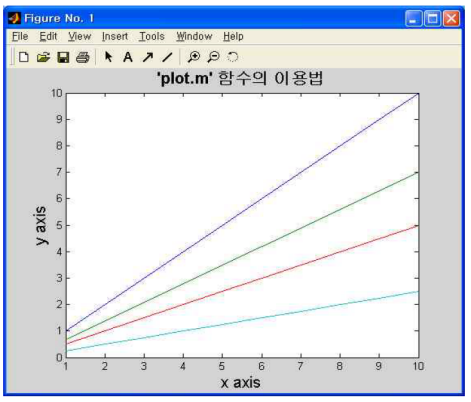
```
close all
clear all

t=0:pi/100:2*pi;
y=sin(t);
plot(t,y)
xlabel('t=0 to 2\pi','fontsize',16)
ylabel('sin(t)','fontsize',16)
title('\it{Value of the sine from Zero to Two Pi}');
set(gcf,'color','w')
text(3*pi/4,sin(3*pi/4),'\bullet\leftarrowsin(t)=0.707')
set(gca,'nextplot','add')
plot(3*pi/4,sin(3*pi/4),'ro')
text(pi,sin(pi),'\leftarrowsin(t)=0')
text(5*pi/4,sin(5*pi/4),...
    'sin(t)=-0.707\rightarrow','Horizontalalignment','right')
set(gca,'nextplot','replace')
```



3) High_level functions

High_level function에 속하는 그래픽 함수들은 일반적으로 데이터를 화면에 display하기 위해 사용하는 함수이다. 상당히 많은 종류의 그래픽 함수들이 존재하는데, 이들 함수들은 Low_level function처럼 데이터를 display 하는데, 필요한 여러 가지 propertyname과 propertyvalue를 묻지 않고, 자동으로 좌표의 크기 조절(axis scaling)과 선의 색 등을 설정해준다.

```
》 x=([1:10; 0.7*[1:10]; 0.5*[1:10]; 0.25*[1:10]])';
》 plot(x)
》 xlabel('x axis','fontsize',15)
》 ylabel('y axis','fontsize',15)
》 title('\bf{''plot.m'' 함수의 이용법}', 'fontsize',15)
```

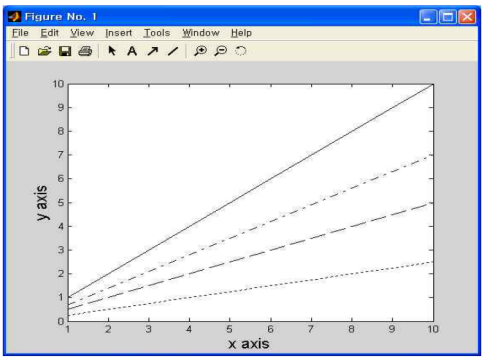


이들 선들의 색에 대한 default값은 다음과 같은 방법으로 얻을 수 있다.

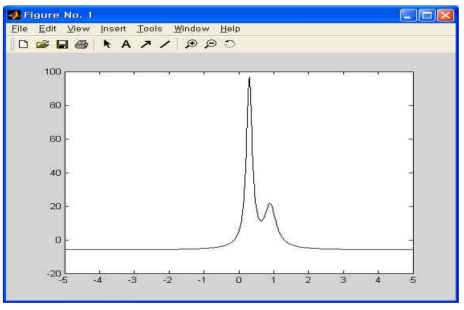
```
》 line_color=get(gca,'colororder')
    line_color =
         0         0    1.0000
         0    0.5000         0
    1.0000         0         0
         0    0.7500    0.7500
    0.7500         0    0.7500
    0.7500    0.7500         0
    0.2500    0.2500    0.2500
```

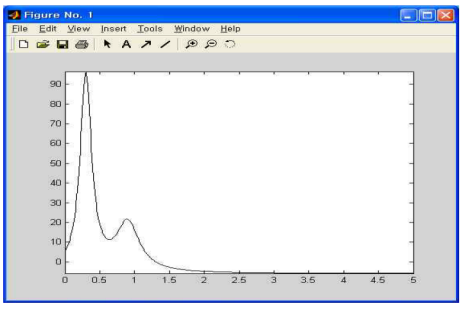
또한, 선들의 type에 대한 default값은 다음과 같은 방법으로 역시 얻을 수 있다.

```
》 line_type=get(gca,'linestyleorder')
      line_type =
                  -
》 set(0,'defaultaxescolororder',[0 0 0],'defaultaxeslinestyleorder','-|-.|--|:')
```



》 t=-5:1/100:5;
》 y=1./((t-0.3).^2+0.01)+1./((t-0.9).^2+0.04)-6;
》 plot(t,y)



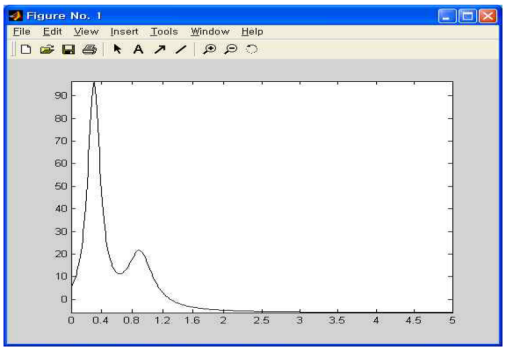


t축의 범위를 [0 ∞]으로 바꾸는 명령은 아래와 같다.

》 axis([0 inf -inf inf]) %(or set(gca,'xlim',[0 inf],'ylim',[-inf inf]))

위의 그림을 보면 데이터가 0≤t≤2 사이에 많이 분포하고 있는 것을 알 수 있다. 그러므로 0≤t≤2 구간에서는 0.4간격마다 tick을 표시하고 싶다. 그러기 위해서는 다음과 같이 한다.

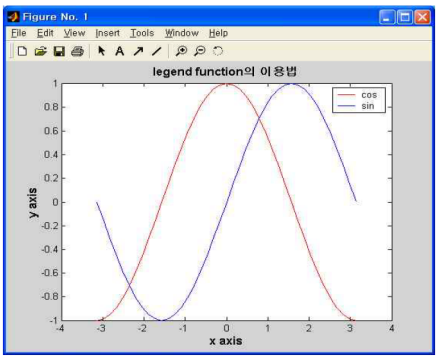
》 set(gca,'xtick',[0:0.4:2 2.5 3 3.5 4 4.5 5])



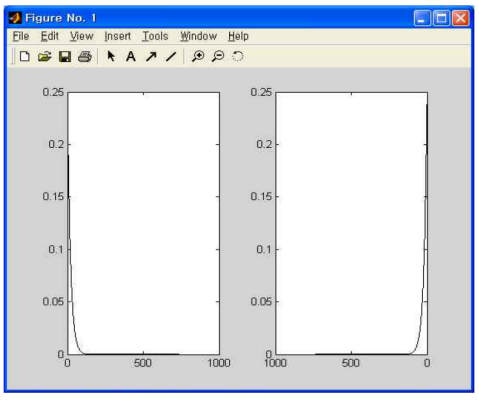
다음은 'legend.m'에 대해서 알아보자.

h=legend('str1','str2',...)
h=legend('off')
h=legend(...., pos)
pos -1 : 좌표계의 오른쪽 밖에 놓는다.
pos 0 : 좌표계의 안쪽에 놓는다.
pos 1 : 좌표계의 upper-right corner에 놓는다.
pos 2 : 좌표계의 upper-left corner에 놓는다.
pos 3 : 좌표계의 lower-left corner에 놓는다.
pos 4 : 좌표계의 lower-right corner에 놓는다.

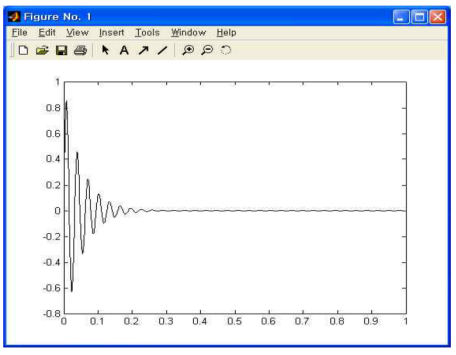
```
》 x=-pi:pi/20:pi;
》 plot(x,cos(x),'r-',x,sin(x),'b-');
》 xlabel('\bf{x axis}','fontsize',12)
》 ylabel('\bf{y axis}','fontsize',12)
》 title('\bf{legend function의 이용법}', 'fontsize',12)
》 h=legend('cos','sin',0)
```

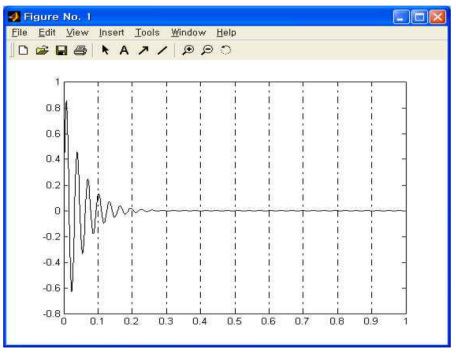



```
》 t=1:900;
》 subplot(121),plot(t,0.25*exp(-0.05*t))
》 subplot(122),plot(t,0.25*exp(-0.05*t))
》 set(gca,'xdir','rev')
```



```
》 t=0:1/1000:1;
》 beta=200;
》 alpha=20;
》 y=exp(-alpha*t).*sin(beta*t);
》 plot(t,y)
》 set(gcf,'color','w')
```

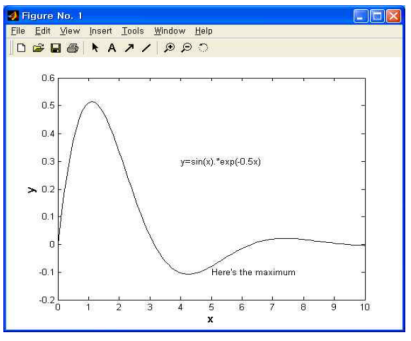





```
》 set(gca,'xgrid','on','gridlinestyle','-.')
```

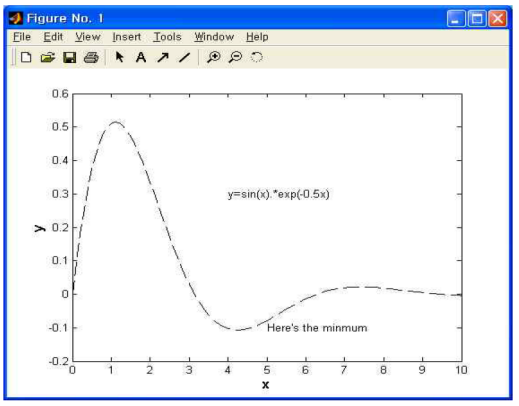
'findobj.m' 함수는 여러 개의 object가 섞여 있는 figure에서 각각의 object를 구별해주는 함수이다. 이 함수는 GUI 프로그램을 작성하는 경우에도 상당히 유용하게 사용되는 함수이므로 자세히 알아둘 필요가 있다.

```
》 x=0:.1:10;
》 plot(x,sin(x).*exp(-0.5*x));
》 xlabel('\bf{x}','fontsize',12)
》 ylabel('\bf{y}','fontsize',12)
》 text(4,.3,'y=sin(x).*exp(-0.5x)');
》 text(5,-0.1,'Here''s the maximum');
》 set(gcf,'color','w')
```



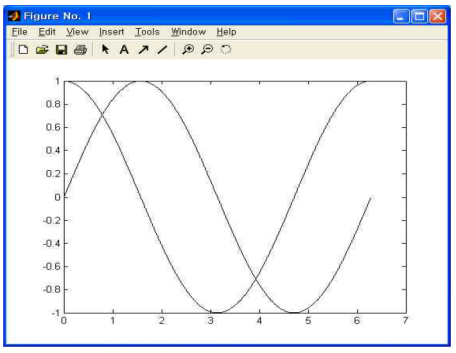
여기서, 'Here's the maximum'을 'Here's the minimum'이라고 바꾸고자 한다. 또한 선(line)을 점선(dashed line)으로 바꾸겠다.

```
》 line_handle=findobj('type','line');
》 set(line_handle,'linestyle','--');
》 text_handle=findobj('string','Here''s the maximum');
》 set(text_handle,'string','Here''s the minmum')
```



사용된 'findobj.m' 함수는 특별히, 검색 object를 지정하지 않았으므로, Root object부터 하위 level의 object까지, 해당 object의 handle을 찾는다. 'type'이라는 propertyname은 'uiversal properties'에 속하는 propertyname이다. 그러므로, 모든 9개의 object는 'type'이라는 propertyname을 소유하고 있다. 'type'의 propertyvalue는 해당 object의 이름인 문자열(string)이 된다. 보통 'findobj.m' 함수는 'tag'라는 propertyname과 함께 사용되는데, 'tag'는 임의의 object를 창조 할 때, 해당 object에 이름(label)을 부여 할 수 있는 기능을 말한다. 결국, 여러 개의 objects 가 혼합되어 있는 figure에서 원하는 object의 이름만 알면 쉽게 해당 object의 handle을 얻을 수 있다.

```
》 x=0:1/100:2*pi;
》 y1=sin(x);
》 y2=cos(x);
》 plot(x,y1,'tag','sin_plot')
》 set(gca,'nextplot','add');
》 plot(x,y2,'tag','cos_plot')
》 set(gcf,'color','w');
```



다음에 나오는 명령어는 위 그림의 lines 중에서 'cos'을 나타내는 선을 점선으로 바꾸는 과정을 나타낸 것이다.

```
》cos_line=findobj('tag','cos_plot');
》set(cos_line,'linestyle','--');
```

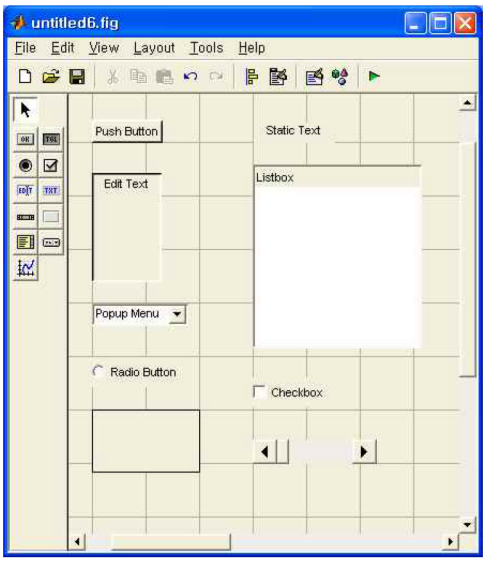
4) GUI 프로그램의 작성

대부분의 MATLAB GUI 프로그램은 앞에서 살펴본 9개의 객체(object) 중에서 'Uicontrol, Uimenu'라는 objects에 의해서 설정할 수 있다. 이들 객체들은 서로 다른 성질과 모양을 갖는 요소들을 포함하고 있는데, 각각의 요소들은 해당 객체의 propertyname에 속한다.

① Uicontrol object의 GUI 요소 사용방법

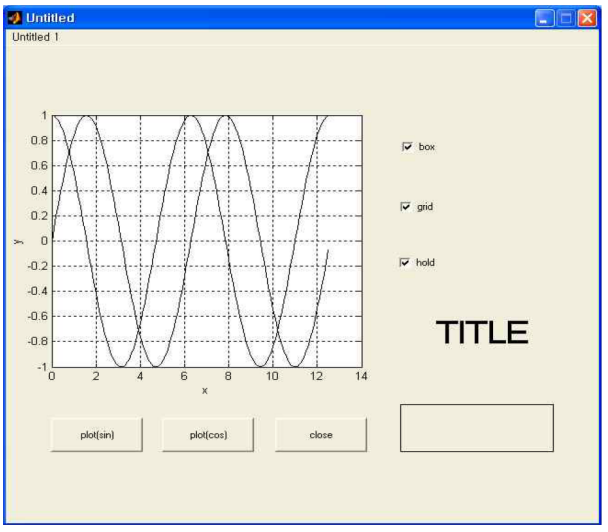
Uicontrol object에 속하는 GUI 요소(propertyname)들은 사용자가 앞으로 어떠한 선택이나, 명령(action)을 하고자 원할 때, 능동적으로 대처하기 위해서 제작된 것이다. 선택된 uicontrol은 대부분 'callback' propertyname에 의해서 대응하는 함수나 m-file을 실행하게 된다. Uicontrol object가 제공하는 요소(propertyname)들은 다음과 같은 것들이 있다.

- Push button
- Static text
- Editable text
- List box
- Pop-up menus
- Check box
- Radio buttons
- Slider
- Frame



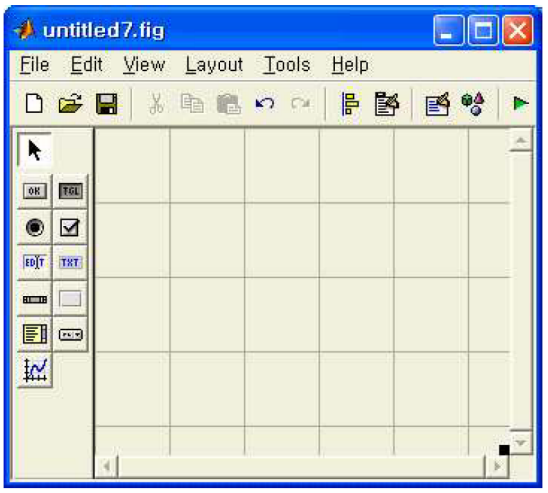
이들 9개의 요소들은 각각 고유한 성질들과 공통된 성질들을 가지고 있는데, GUI 프로그램을 작성하면서 설명을 해 나갈 것이다.

실제로 작성할 프로그램은 아래와 같다.



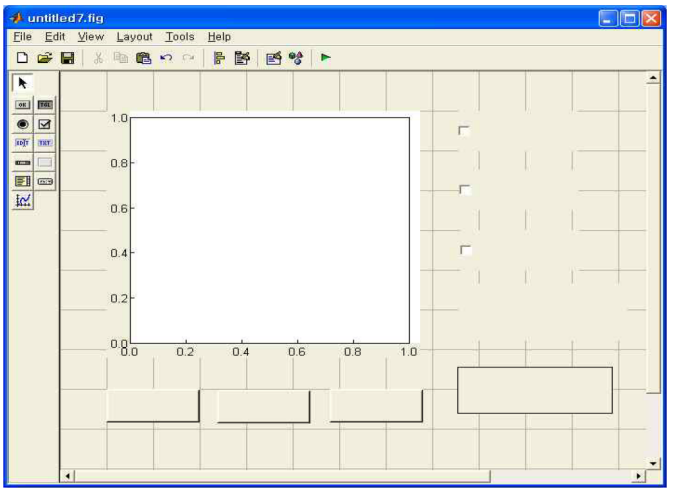
MATLAB 5.0 이후 버전에서는 Visual C++처럼 해당 객체들의 design을 담당하는 'guide.m'이라는 함수가 제공되면서 design을 손쉽게 할 수 있게 되었다.
우선, MATLAB command window에서 guide 함수를 호출하면, 다음과 같이 새로운 Figure 창과 Guide control panel창이 활성화 된다.

》guide

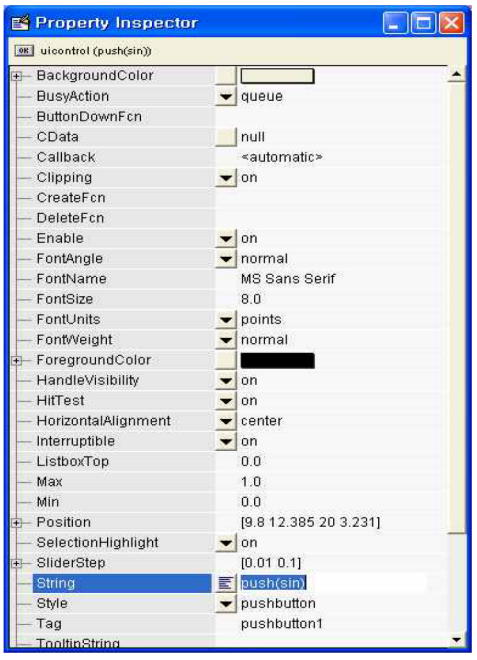


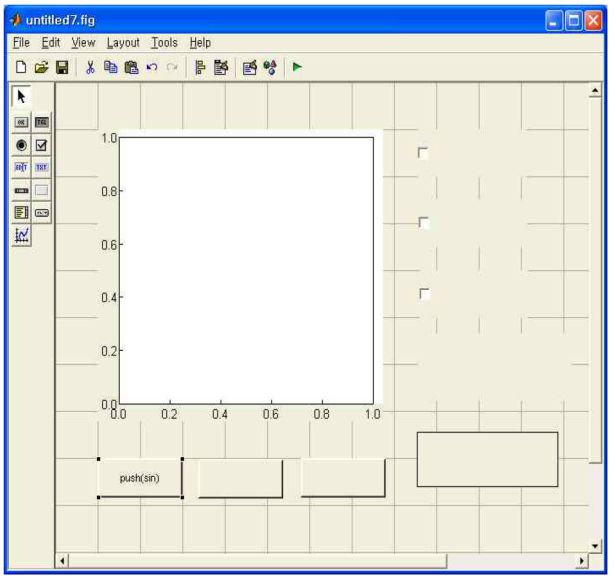
각각의 object를 controlled state에 있는 figure에 위치시키기 위해서는, 왼쪽에 있는 객체를 mouse로 클릭한 후 controlled figure로 가져가면 된다.

axes 객체를 선택한 후 controlled figure에서 mouse의 왼쪽 버튼을 누른 다음 controlled figure의 원하는 위치에서 다시 클릭하면 다음과 같이 axes object가 생성될 것이다. 같은 방법으로 push button, check box, static text, Editable text 객체를 생성한다.



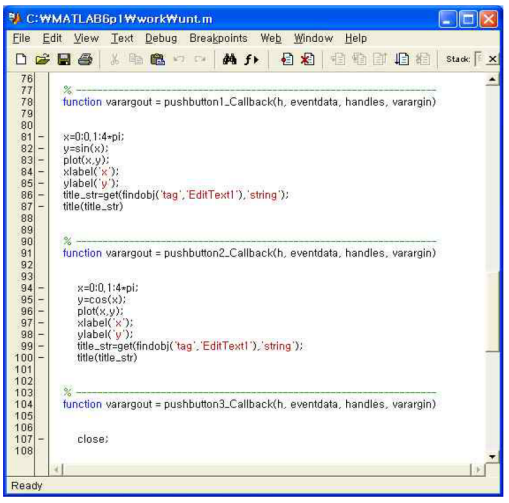
다음은 push button이나 check box 등에 label을 다는 작업이다. 먼저 push button을 더블 클릭 하면 다음과 같은 창을 볼 수 있는데, 아래 그림과 같이 label 값을 설정할 수 있다.

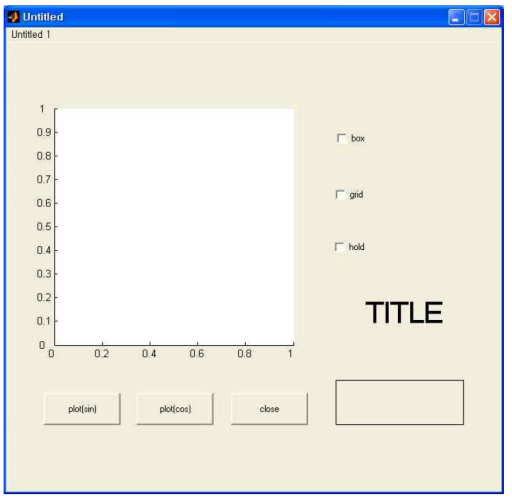




마찬가지로 나머지 객체에도 label을 할당한다. label 값을 할당한 후에는 각각의 객체가 동작할 수 있도록 함수적 기능을 부여해야 한다.

push button를 선택한 후, View에 있는 Object callback 의 Callback을 클릭하면 아래와 같은 그림을 볼 수 있는데, 여기에 해당되는 함수 값을 지정하면 된다.



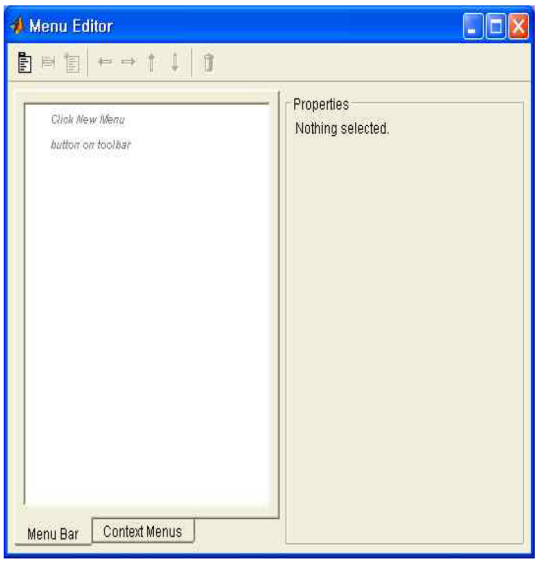


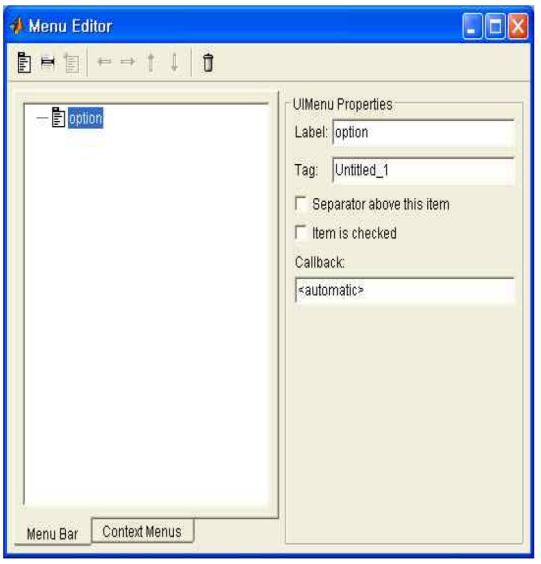

```
% --------------------------------------------------------------------
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)

    x=0:0.1:4*pi;
    y=sin(x);
    plot(x,y);
    xlabel('x');
    ylabel('y');
    title_str=get(findobj('tag','EditText1'),'string');
    title(title_str)

% --------------------------------------------------------------------
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)

    x=0:0.1:4*pi;
    y=cos(x);
    plot(x,y);
    xlabel('x');
    ylabel('y');
    title_str=get(findobj('tag','EditText1'),'string');
    title(title_str)

% --------------------------------------------------------------------
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)

    close;
```

```
% --------------------------------------------------------------------
function varargout = checkbox6_Callback(h, eventdata, handles, varargin)

    if get(findobj('tag','checkbox6'),'value') ==1
        set(gca,'box','on')
    else
        set(gca,'box','off')
    end

% --------------------------------------------------------------------
function varargout = checkbox7_Callback(h, eventdata, handles, varargin)

    if get(findobj('tag','checkbox7'),'value') ==1
        grid on;
    else
        grid off
    end

% --------------------------------------------------------------------
function varargout = checkbox8_Callback(h, eventdata, handles, varargin)
    if get(findobj('tag','checkbox8'),'value') ==1
        hold on;
    else
        hold off
    end

% --------------------------------------------------------------------
function varargout = Untitled_1_Callback(h, eventdata, handles, varargin)

   title(get(findobj('tag','EditText1'),'string'))
```

② Uimenu Object의 GUI요소 사용 방법

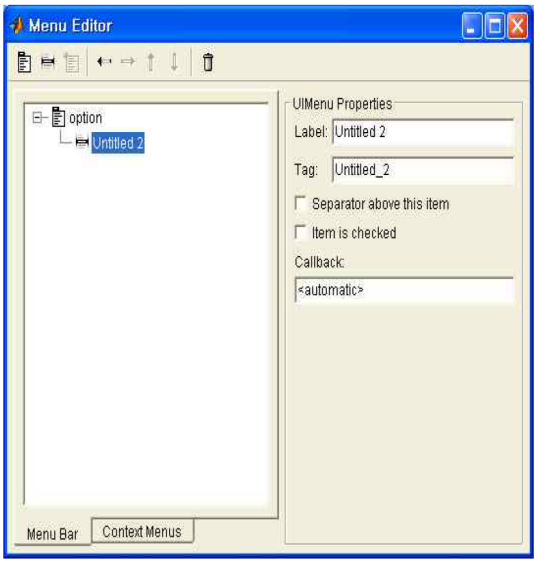
Uimenu object는 figure object에 menus를 첨가하는 기능을 갖고 있다. guide control panel의 guide tools의 'menu editor'를 클릭하면 다음과 같은 그림을 볼 수 있다.





new menu를 클릭하면 새로운 menu가 생성되는데 label을 'option'으로 하면 오른쪽 그림과 같이 된다.

여기에서 다시 new menu item를 클릭하면 submenu가 생성된다.



제 6 장 Simulink

시뮬링크(simulink)는 동적인 시스템을 묘사하는데 사용하는 MATLAB의 확장도구로써 GUI(Graphic User Interface) 기능을 제공하고 있다. MATLAB과는 달리 simulink는 광범위하고 강력한 프로그램으로써 다음과 같은 특징을 가지고 있다.

- 동적인 시스템의 모델링, 모의실험, 분석을 위한 MATLAB의 패키지이다. 이러한 simulink는 연속시간과 이산시간으로 모델화된 선형, 비선형 시스템을 제공한다.
- GUI 환경에서 마우스를 이용하여 블록화된 모듈(module)들을 통해 시스템을 구성할 수 있다.
- Simulink는 각 부문별로 블록화 되어 있으며, 각각의 부문에는 모델화된 도구들이 들어있다.
- 모델이나 시스템을 구성한 후에 MATLAB의 명령창이나 simulink의 풀다운 메뉴를 이용한다.
  하여 시뮬레이션을 할 수 있고, 시뮬레이션의 결과는 scope나 그 이외의 display block을 이용하여 확인할 수 있다.

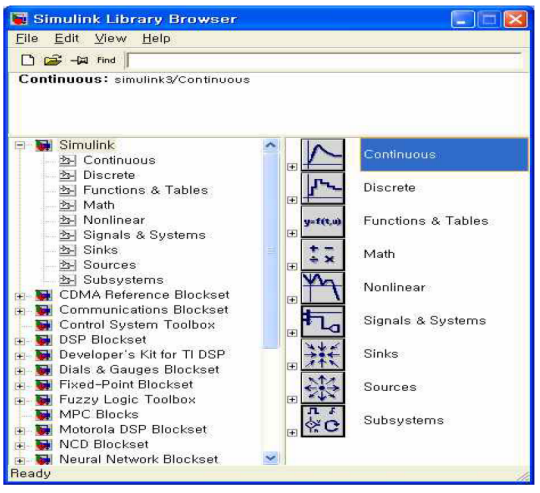
1. Simulink의 시작

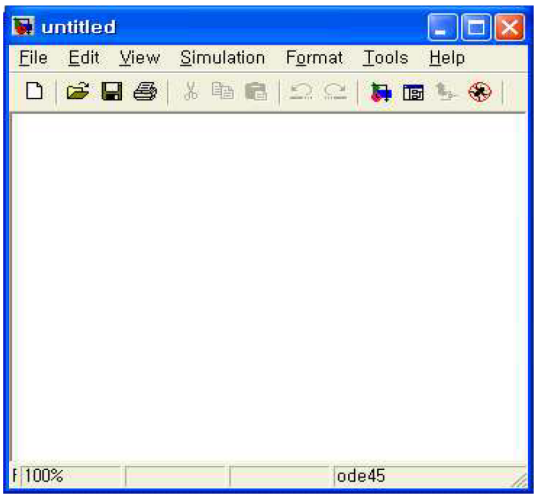
Simulink만 단독으로 실행할 수 없기 때문에 simulink를 시작하기 위해서는 먼저 MATLAB을 실행해야한다. 실행방법은 다음과 같다.

- MATLAB 툴바 메뉴에 있는 simulink 아이콘을 클릭한다.
- MATLAB 명령창에서 'simulink'라고 작성한 후 Enter키를 누른다.

》simulink

위와 같이 입력하면 다음과 같은 'Simulink Library Browser' 창이 뜬다. 왼쪽상단에 있는 첫 번째 아이콘을 클릭하면 제목이 'untitled'라고 되어있는 새로운 창이 뜨게 된다.





Simulink의 라이브러리는 다음과 같은 것들이 존재한다.

① Continuous : 연속시간 함수 성분을 나타내는 블록을 포함
- Derivative : 입력의 미분시간을 출력한다.
- Integrator : 신호를 적분한다.
- Memory : 이전 time step에서의 입력 값을 출력
- State space : 선형 상태 공간 시스템의 도구
- Transfer Fcn : 선형전달 함수 도구
- Transport Delay : 주어진 시간동안 입력 값을 지연
- Variable Transport Delay : 입력된 시간동안 입력 값을 지연
- Zero-Pole : 전달 함수를 극(pole)과 영(zero)을 사용하여 나타냄

② Discrete : 이산 함수 성분을 나타내는 블록을 포함
- Discrete Transfer Fcn : 불연속부의 전달함수의 도구
- Discrete Zero-pole : 극과 제로 점의 구간에서 불연속부의 전달함수 도구
- Discrete Filter : 이산 IIR가 FIR 필터 도구
- Discrete State Space : 불연속부의 상태 공간 시스템의 도구
- Discrete-Time Integrator : 입력 값에 이산 시간 적분을 취하여 출력
- First-Order Hold : First-Order sample과 hold룰 수행
- Unit Delay : 신호에서 샘플구간마다의 지연시간
- Zero-Order Hold : 샘플링 시간동안 Zero-Order Hold 를 수행한다.

③ Function & Tables
- Direct Look-Up Table (n-D) : Index into an N차원 테이블을 스칼라, 벡터, 또는 2-D 매트릭스로 채운다.
- Fcn : 입력에서 열거된 논리조작을 적용한다.
- Look-Up Table : 입력된 테이블을 사용하여 입력 값에 1차원 선형 보간을 수행하여 출력 값으로 한다.
- Look-Up Table(2-D) : 입력된 테이블을 사용하여 입력 값에 2차원 선형 보간을 수행하여 출력 값으로 한다.
- Look-Up Table (n-D) : 입력된 테이블을 사용하여 입력 값에 n차원 선형 보간을 수행하여 출력 값으로 한다.
- S-Function : S-function에 접근한다.
- MATLAB Fcn : 블록 입력 값을 MATLAB 함수에 대입하여 원하는 연산을 수행한 후 블록의 출력으로 이동한다.
- Polynomial : 다항식에 접근
- PreLook-Up Index Search : 미리보기 인덱스로 검색
- S-Function Builder : S-function 수행자

④ Math : 수학함수 블럭

- Abs : 절대값을 출력
- Algebraic Constraint : 입력신호 f(z)가 0이 되도록 구속하여 그 때의 z를 출력 값으로 한다.
- Bitwise Logical Operator:
- Combinatorial Logic : 진리표를 만든다.
- Complex to Magnitude-Angle : 복소수 입력을 받아서 크기와 위상각을 또는 크기만을 또는 위상각만을 계산하여 출력한다.
- Complex to Real-Imag : 복소수 입력을 받아서 실수부와 허수부를 출력한다.
- Dot Product : 두 벡터를 입력받아 내적을 계산한다.
- Gain : 이득 값을 나타낸다
- Logical Operator : 입력 값에 논리 연산을 취한 후 그 결과를 출력한다.
- Magnitude-Angle to Complex : 크기와 위상각을 입력받아 복소수로 출력한다.
- Math Function : 수학함수를 사용한다.
- Matrix Gain : 입력에 행렬을 곱한다.
- MinMax : 최대 또는 최소 입력 값을 출력한다.
- Product : 입력 값의 곱을 계산하여 그 결과를 출력한다.
- Real-Imag to Complex : 실수부와 허수부를 복소수로 출력한다.
- Relational Operator : 관계 연산 수행 후 0또는 1로 출력한다.
- Rounding Function : 회전 함수를 표현한다.
- Sign : 양수이면 1, 음수이면 -1, 0이면 0을 출력한다.
- Slider Gain : 스칼라 이득 값을 스크롤바로 조정한다.
- Sum : 입력 값들을 더하거나 뺀 후 출력한다.
- Trigonometric Function : 삼각 함수를 나타낸다.

⑤ Nonlinear : 비선형 함수를 표현하는 블록을 포함

- Backlash : Backlash를 모델링하기 위해서 사용되는 블록이다.
- Coulomb & Viscous Friction : 쿨롱과 점성마찰을 모델링하기 위해 사용된다.
- Dead Zone : 입력 값이 특정한 영역에 해당될 경우 0을 출력한다.
- Manual Switch : 두 입력 중에서 하나를 선택하기 위해서 사용된다.
- Multiport Switch : 제어 입력 값에 따라서 여러 입력 중에서 하나를 선택하는데 사용된다.
- Quantizer : 특정 간격으로 입력 값을 이산화하기 위해서 사용된다.
- Rate Limiter : 입력 신호의 기울기인 1차 미분 값을 제한하는데 사용된다.
- Relay : 어떤 조건에 따라 두 가지의 값 중에서 하나를 출력하기 위해서 사용된다.
- Saturation : 입력 값에 한계 값을 주기 위해서 사용된다.
- Switch : 제어입력값에 따라 두 입력 중에서 1개의 입력을 선택하기 위해서 사용된다.

⑥ Signals & Systems

⑦ Sinks : display와 출력의 블록을 포함
- Display : 입력신호의 현재 값을 보여준다.
- Scope, Floating Scope : 시뮬레이션하는 동안 신호발생을 보여준다.
- Stop Simulation : 입력신호가 0이 아닐 때 시뮬레이션을 멈춘다.
- Terminator : 어떤 블록의 출력 단자가 다른 블록과 연결되어 있지 않는 경우에 그 단자를 봉하기 위해서 사용된다.
- To File : MATLAB에 .mat 형식의 파일로 입력신호가 저장되고 신호는 scalar나 vector
- To Workspace : workspace에 행렬로 데이터를 저장한다.
- XY Graph : 두 개의 입력에 대한 신호 발생을 나타낸다.

⑧ Sources : 신호를 생성하는 블록들을 포함
- Band-Limited White Noise : 이산시스템이 아닌 연속시스템에 백색 노이즈를 가한다.
- Chirp Signal : 주파수를 증가시켜 사인(sine)파를 발생시킨다.
- Clock : 현재 시뮬레이션 시간을 나타내고 공급한다.
- Constant : 상수 값을 출력한다.
- Digital Clock : 시뮬레이션 시간을 출력한다.
- From workspace :작업공간에서 데이터를 읽는다.
- From File : 파일로부터 데이터를 읽는다.
- Ground : 접지
- Pulse Generator : 일정간격의 펄스 발생
- Ramp : 시간 도함수가 일정할 때의 신호가 발생
- Random Number : 가우시안 분포를 갖는 random 신호를 발생
- Repeating Sequence : 반복적인 불규칙 신호를 발생
- Signal Generator : 여러 가지 신호 파형을 출력
- Sine Wave : 사인파를 발생
- Step : 계단함수를 발생시키고, 매개변수로는 step 시간, 초기치와 최종치가 있다.
- Uniform Random Number : Uniform한 분포를 갖는 random 신호를 발생

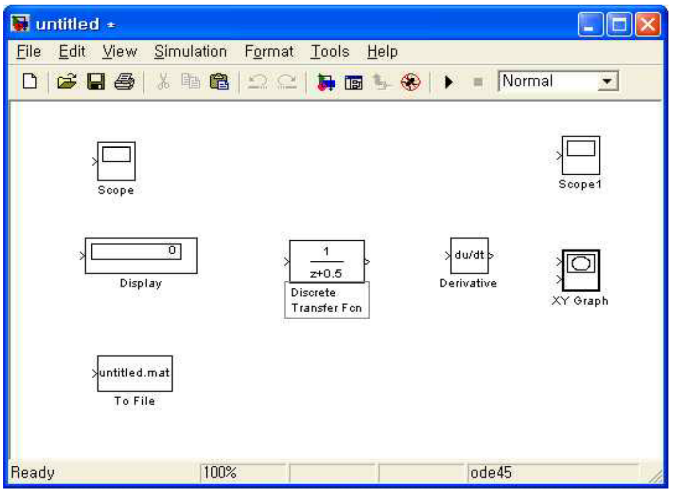
⑨ Subsystems : 기타의 특별한 블록을 포함하고 있다.

2. 객체(Object)

Library browser 창에서 시뮬링크의 하위 메뉴('+'로 표시)를 클릭하고, 해당 객체를 클릭하면 simulink library browser 하단에 객체를 설명하는 도움말과 객체의 심볼(symbol)을 볼 수 있다.

사용하고자 하는 객체는 마우스 왼쪽 버튼으로 클릭한 후, untitled 창(작업창)으로 드래그하면 객체가 생성된다. 같은 객체를 여러 개 사용할 경우에는 작업창에서 객체를 선택한 후, Ctrl키를 누른 상태에서 드래그하면 객체가 복사된다. 각각의 객체는 고유의 이름을 가지고 있는데, 객체의 이름부분에 마우스 커서를 옮기고 더블 클릭하면 객체의 이름을 바꿀 수 있

는 상태로 전환된다.



1) 블록지우기

작업창으로 복사하여 선택된 객체는 Delete나 또는 Backspace를 이용하여 제거할 수 있다. Edit 메뉴에서의 Clear와 Cut으로도 제거할 수 있는데 Cut은 클립보드에 저장되어 붙여넣기하면 다시 생성된다.

2) 블록의 회전

객체의 자세를 변화시키기 위해서는 작업창의 Format 메뉴에서 Flip block과 Rotate Block을 이용하면 된다. 단축키는 각각 Ctrl + F와 Ctrl + R이다.

3) 블록 크기 변경

객체의 크기는 핸들을 이용하여 마우스를 드래그하면 사용자 임의대로 조정이 가능하다.

4) 블록 이름 변경

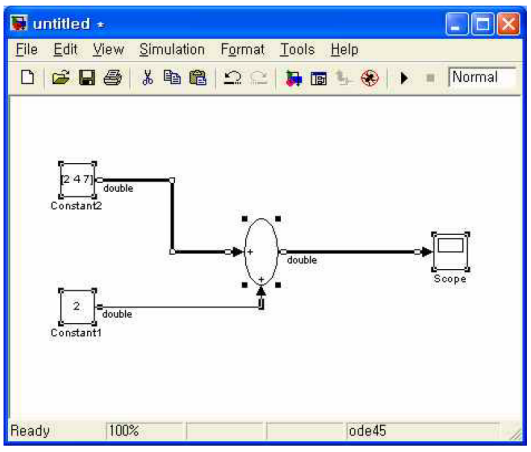
블록으로 형성된 객체의 이름을 바꾸기 위해서는 이름을 클릭하면 된다. 그러면 새로운 이름을 쓸 수 있는 상태가 되어 수정할 수 있다.

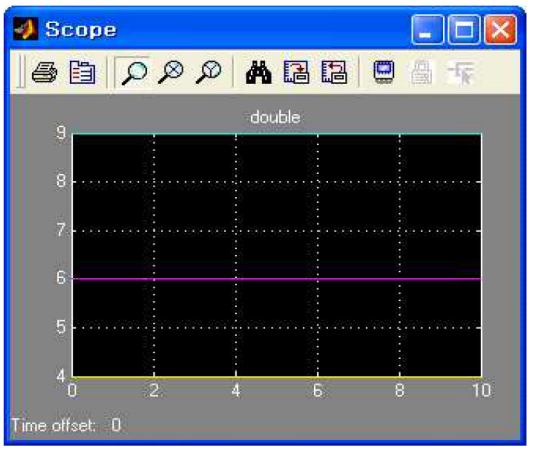
5) 블록 이름의 위치변경

블록 이름의 위치는 상하로 변경될 수 있으며 방법은 두 가지가 있다. 마우스로 블록의 이름을 선택한 후 블록의 위나 아래로 옮기거나 작업창의 Format 메뉴에서 Flip name을 선택한다. 이때는 블록이 선택된 후이어야 한다. 블록의 이름을 제거할 때는 이름을 선택한 후에 키보드의 Delete키를 누르거나 블록을 선택한 상태에서 작업창의 Format 메뉴의 Hide name을 선택한다. 다시 보이기 위해서는 메뉴의 Show name을 선택한다. 작업창의 어느 곳에서라도 마우스를 더블클릭하면 그 위치에서 사용자 임의의 문장을 쓸 수 있다. 작업창의 Format 메뉴에서 text와 객체들 그리고 배경화면까지 색상을 조정할 수 있다.

6) 벡터의 입·출력

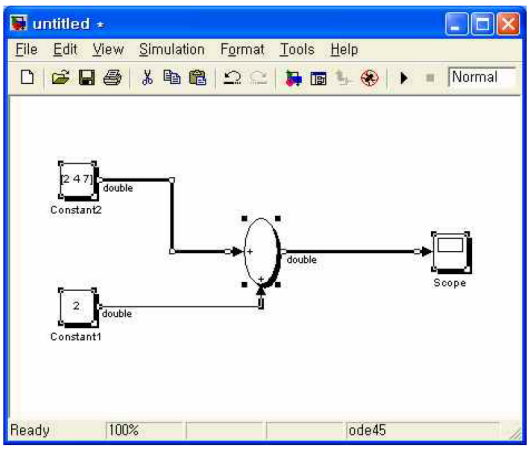
Simulink에서는 스칼라(scalar)와 벡터(vector)의 입력과 출력을 할 수 있다. sources 라이브러리에서 constant 블록을 드래그하여 작업창으로 옮긴 후 아래와 같이 구성한 다음 Format 메뉴에서 wide vector lines를 선택하면 벡터 입력이나 출력은 굵은선으로 표현되고 스칼라는 가는 선으로 표현된다.





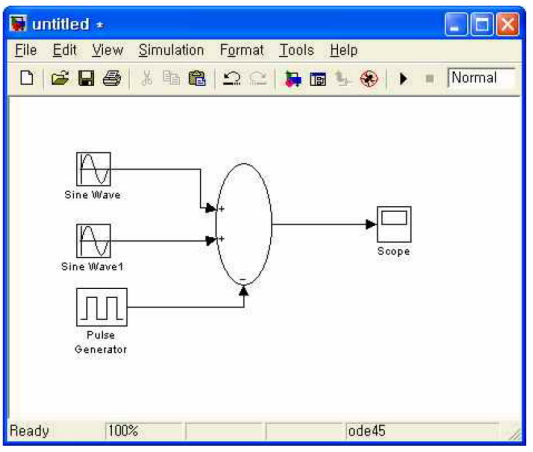
7) 블록에 그림자 넣기

블록에 입체감을 나타내주기 위해서, 임의의 블록을 선택한 후에 Format 메뉴의 Show Drop Shadow를 선택하면, 선택한 블록에 그림자가 형성된다.



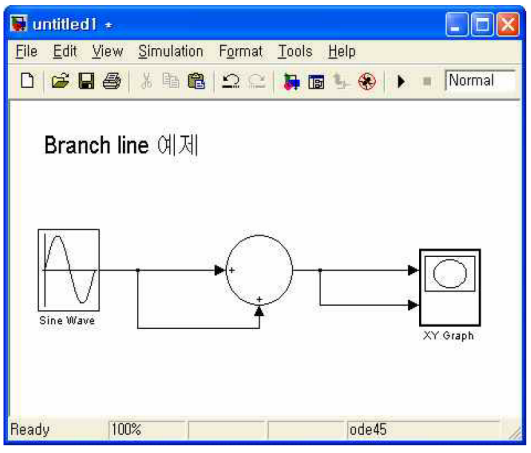
8) 블록을 선으로 잇기

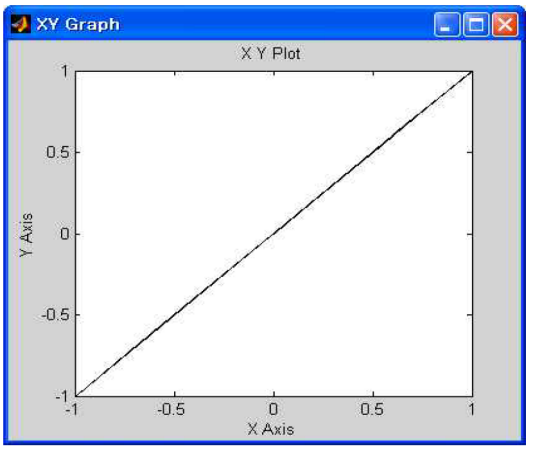
신호는 연결한 선들(lines)을 통해서 전달된다. 각 연결선은 스칼라와 벡터 신호를 전달할 수 있다. 한 블록의 출력을 다른 한 블록의 입력으로 연결하는 방법은 출력하고자 하는 블록의 출력 모서리에서 마우스를 클릭한 상태로 입력할 곳으로 드래그를 하면 된다. simulink에서는 연결선을 그릴 때 수직·수평으로만 그려지는데 대각으로 그리기 위해서는 키보드의 shift키를 누른 상태에서 마우스를 드래그하면 된다.



9) 분선(Branch Line) 그리기

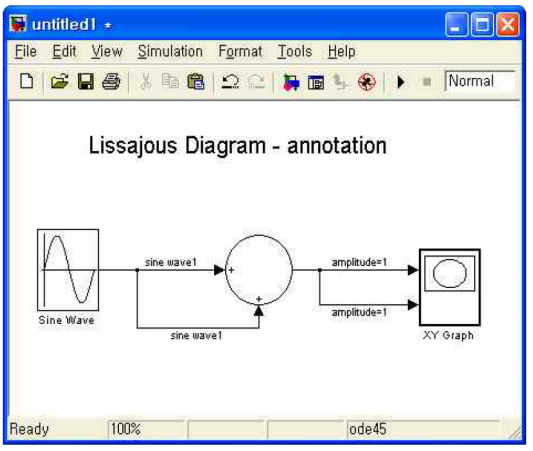
분선은 한 라인에서 동일한 신호를 실어 보내는 또 다른 라인을 말한다. 분선을 그리기 위해서는 분선이 시작하는 라인에서 마우스의 오른쪽 버튼을 클릭한 상태에서 드래그 한다. 또는 키보드의 Ctrl키를 누른 상태에서 마우스 왼쪽버튼을 누르고 드래그 하면 된다.





10) 신호레이블(label) 사용하기

신호의 레이블을 생성하기 위해서는 라인의 원하는 위치에서 마우스를 더블클릭하면 된다. 주의할 점은 라인에서 더블클릭하지 않고 임의의 공백에서 더블클릭하면 주석(annotation)이 된다. 선(line)에 레이블을 붙을 수 있으면, 위치는 라인상의 고정된 왼쪽이나 가운데 또는 오른쪽으로 위치하게 된다.



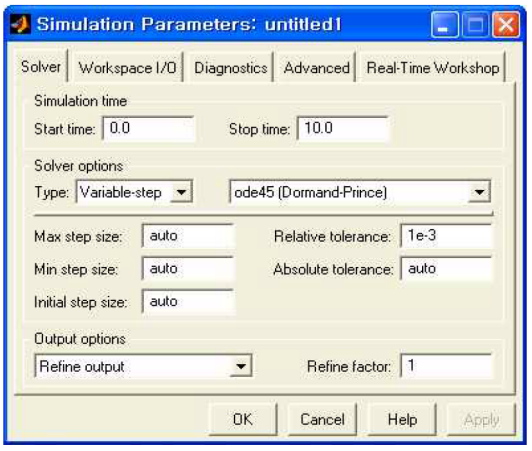
선에 위치한 레이블의 복사는 Ctrl키를 누른 상태에서 마우스로 드래그하면 된다. 단, 신호의 흐름이 같은 자신의 라인에서만 복사가 가능하다. 이미 존재하는 레이블의 편집은 레이블을 마우스로 클릭하면 편집상태로 된다. 레이블을 삭제할 때는 레이블의 외각 테두리를 선택한 후 Delete나 Backspace키를 누른다.

3. Simulation Parameters 옵션

작업창의 시뮬레이션에 매개변수들을 이용하여 시뮬레이션 파라미터와 solver를 설정할 수 있다.

1) Solver 매개변수

Solver 창에서는 시뮬레이션의 시작과 종료시간, solver와 solver의 parameter를 선택하고 또한 출력의 옵션을 선택한다. 각종 옵션은 아래 그림과 같다.



① 시뮬레이션 시간

시뮬레이션할 때 시작시간과 종료시간을 설정한다. 기본값은 start time은 0.0초 stop time은 10.0초로 설정되어 있다.

② Solver

모델의 시뮬레이션은 상미분 방정식(ODE)의 수치적 집합을 포함하고 있다. simulink는 이런 방정식의 시뮬레이션을 위해 solver를 제공하는데, 여기에는 variable-step과 fixed-step가 있다. variable-step solver는 시뮬레이션 되는 동안의 step 사이즈를 변경할 수 있다. 이 상태에서는 오차제어(error control)와 영점교차(zero crossing) 검출을 제공한다. fixed-step solver는 시뮬레이션 되는 동안의 step 사이즈를 설정하는데 이 상태에서는 오차제어와 영점교차 검출을 제공하지 않는다.

③ Output 옵션

Output option은 시뮬레이션이 생성한 출력을 어떻게 제어할 것인가를 결정하는 영역이다. 다음과 같은 시간에서 출력을 갖는 시뮬레이션이 있을 때 각각의 옵션의 차이를 보면

0, 2.5, 5, 8.5, 10

- Refine factor (refine factor=2)

  0, 1.25, 2.5, 3.75, 5, 6.75, 8.5, 9.25, 10

- Produce additional output (output times=[0:10])

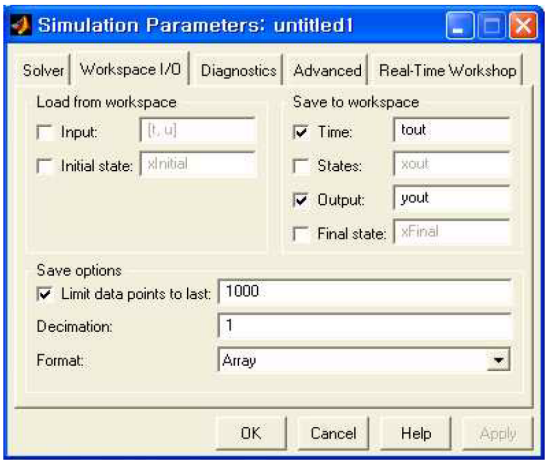
  0, 1, 2, 2.5, 3, 4, 5, 6, 7, 8, 8.5, 9, 10

- Produce specified output only (output times=[0:10])

  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

2) Workspace I/O

workspace I/O 에서는 MATLAB 작업공간상에서 시뮬레이션의 결과를 출력할 수 있고, 입력과 초기상태를 작업공간으로부터 얻을 수 있다.



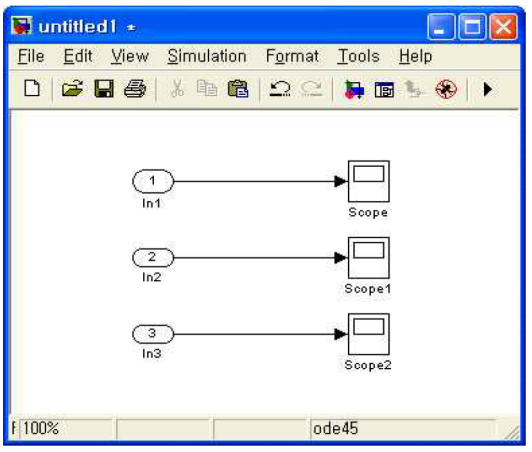
① 기본 workspace에서의 입력 설정

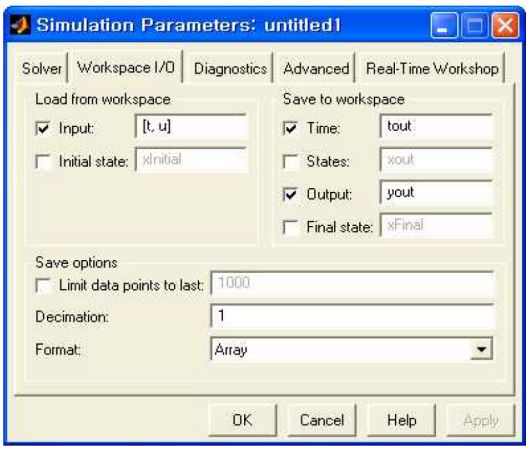
signals & systems의 ln1 블록은 시스템의 외부로부터 입력을 제공한다. 내부로부터의 입력은 시뮬레이션 time(t)의 용어로 표현된 MATLAB 명령이나, 행렬로써 표현할 수 있다.

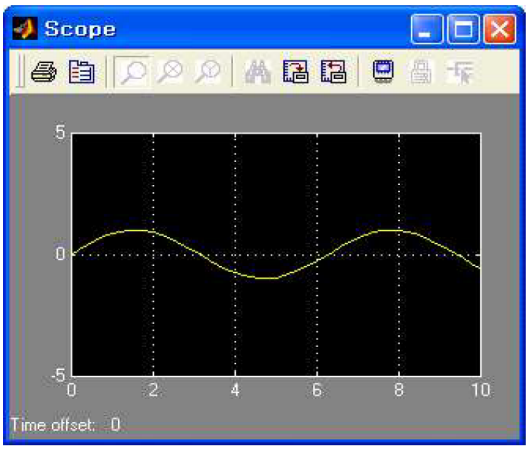
simulink의 작업창에 아래와 같이 블록을 구성한 후 MATLAB 명령창에서 다음과 같이 입

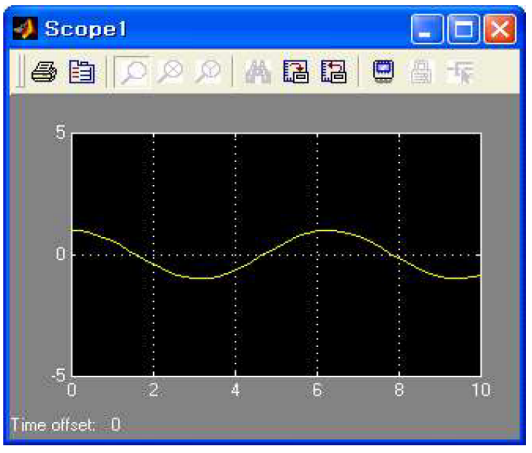
력한 후 시뮬레이션을 시작한다.

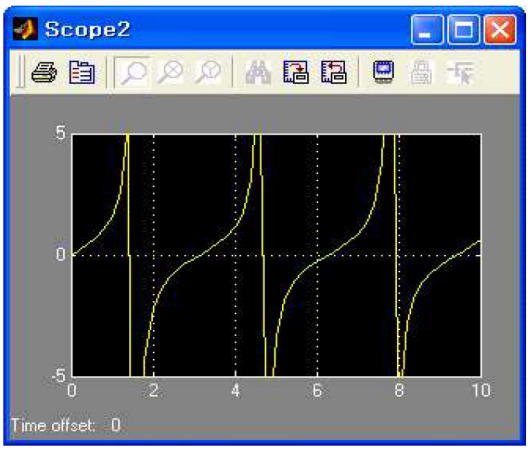
```
》 t=[0:0.1:10]';
》 u=[sin(t), cos(t), tan(t)];
```









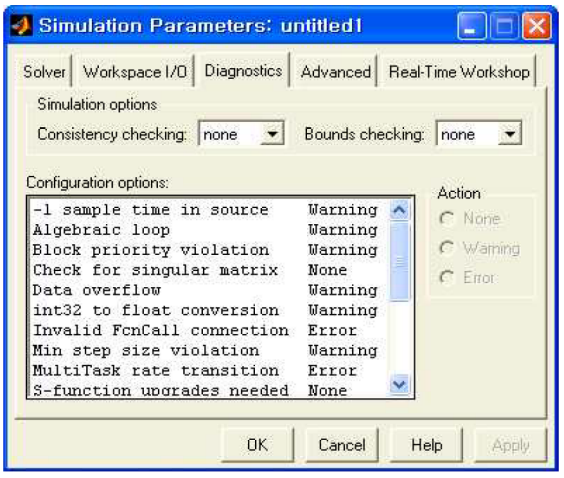


② workspace로 출력 저장하기

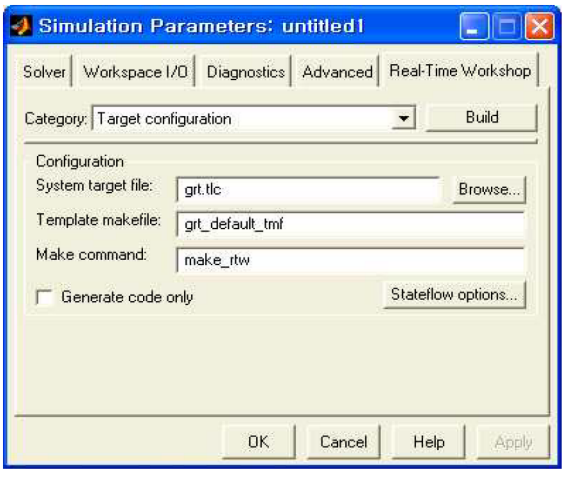
save to workspace 블록에서 time과 states와 output을 선택함으로써 MATLAB 작업공간으로 변수명을 부여하면서 값을 저장할 수 있다. 여러 개의 출력을 MATLAB 작업 공간상으로 저장하고자 할 때는 콤마(,)를 이용하여 출력변수를 분리하여 주면 된다. save options은 저장되는 출력의 양을 제안한다. data행의 수를 제안하기 위해서 limit rows to last를 선택하고 행의 수를 설정한다. decimation은 출력되는 양의 수를 제안하는데, 출력의 원소가 50개일 때 decimation을 2로 하게 되면 출력이 25개로 균등한 간격을 유지하는 출력을 얻게 된다.

3) Diagnostics 페이지

시뮬레이션이 되는 동안 일어나는 여러 가지 상황에 대한 메시지의 수준을 결정한다. 각각의 상황에 대한 메시지의 수준을 결정하는데 warning은 시뮬레이션을 중단시키지는 않지만 error의 선택은 시뮬레이션을 중단시킨다.



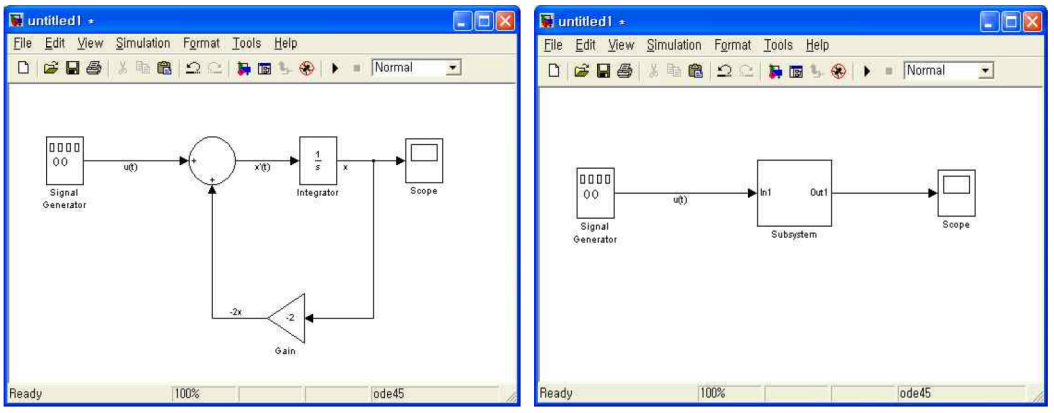
4) Real-Time Workshop



4. 서브시스템과 마스킹(Subsystems & Masking)

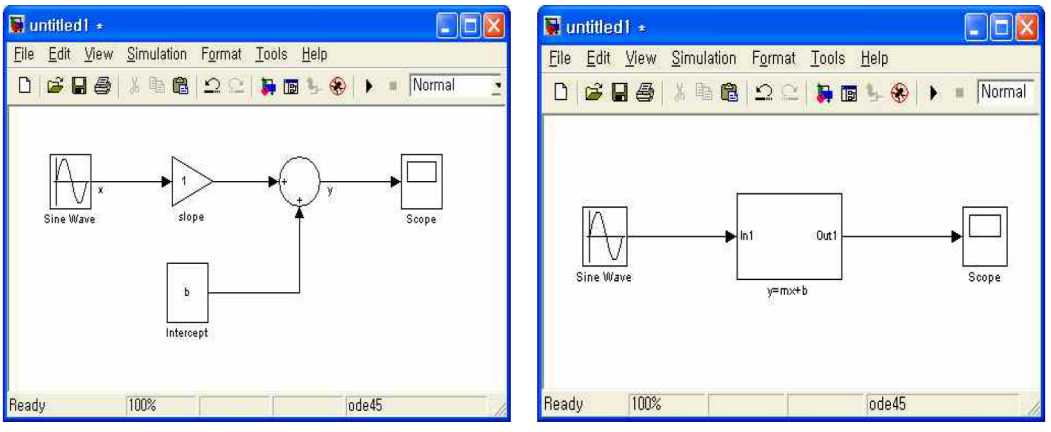
1) 서브시스템(Subsystem)

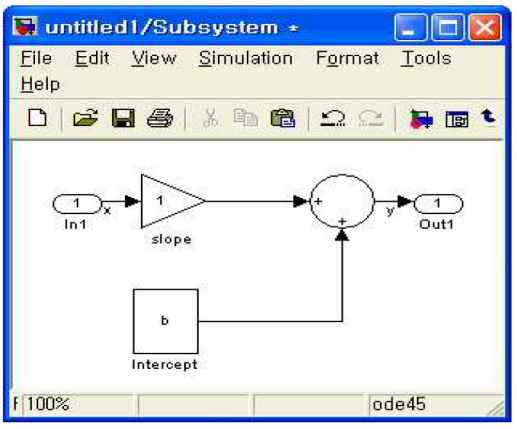
모델이 커지거나 복잡함에 따라서 쉽게 이것을 이해하거나 유지하기가 어렵게 된다. 서브시스템은 이러한 모델들을 계층적으로 축소하여 해결할 수 있다. 서브시스템을 만드는 작업은 먼저 블록을 선택하여 edit 메뉴에서 'create subsystems'을 선택하거나 단축키 ctrl+G로 설정하면 되고 Block properties를 선택하여 캡슐화 된 블록에 대한 제원을 설정할 수 있다.



2) 마스크 블록(Masked Block)

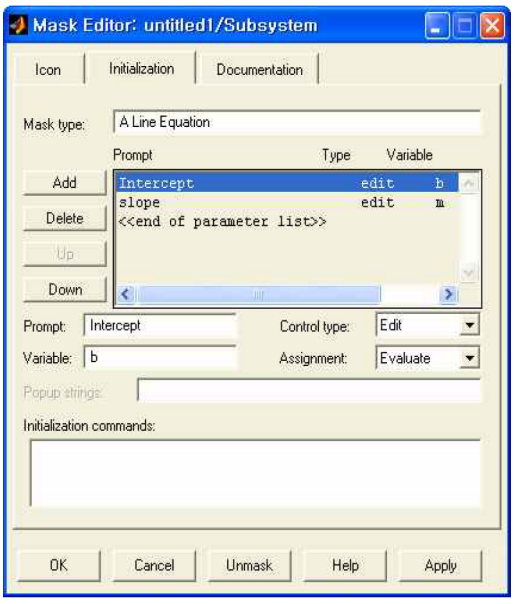
아래 그림은 직선의 방정식 y=m+b의 간단한 모델을 서브시스템으로 만들고, 서브시스템을 더블클릭하게 되면 서브시스템의 블록이 열리면서 상세한 모델의 시스템을 볼 수 있다. 여기서 게인 매개변수는 m으로 상수블록이고, slope의 레이블을 붙여주었고, 또한 b도 상수블록으로 intercept의 레이블로 표시 되었다. 이들 매개변수는 각각 직선의 기울기와 절편값이다.





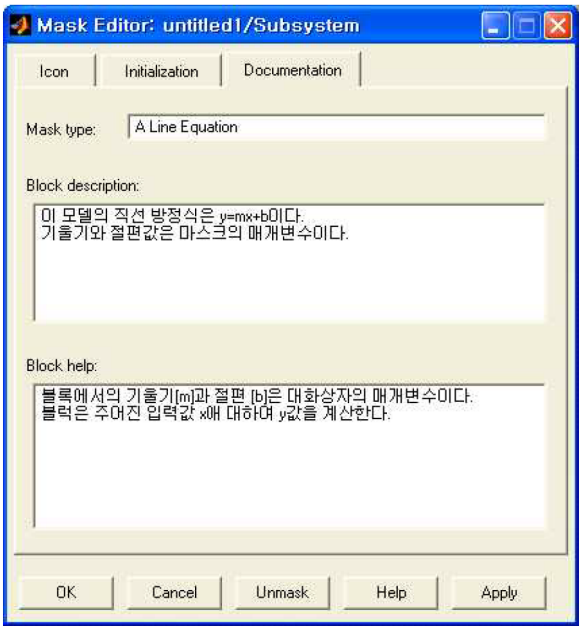
① 마스크 대화상자 prompt 기입

이 서브시스템을 마스크로 만들기 위해 서브시스템을 선택하고 Edit 메뉴에서 Mask subsystem을 선택하면 아래와 같은 창이 활성화 된다. 활성화 시킨 후 Mask type과 그 밖의 내용을 입력한다.



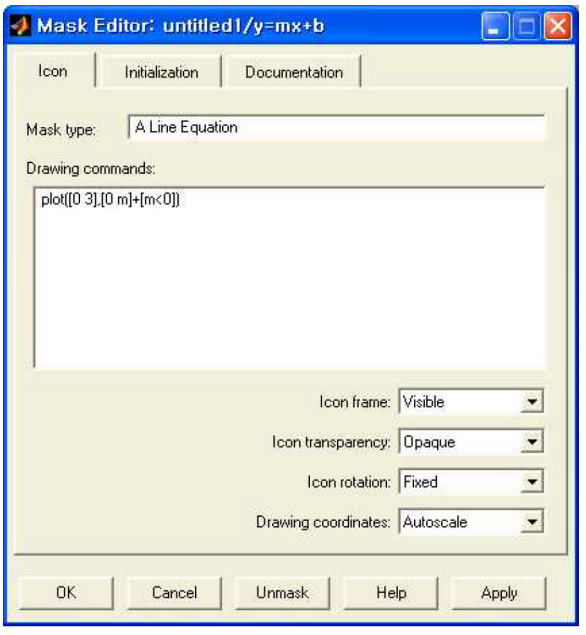
② 블록서술과 도움말 만들기

블록서술과 도움말의 형태는Mask editor의 Documentation에서 설정한다.



③ 블록 아이콘 만들기

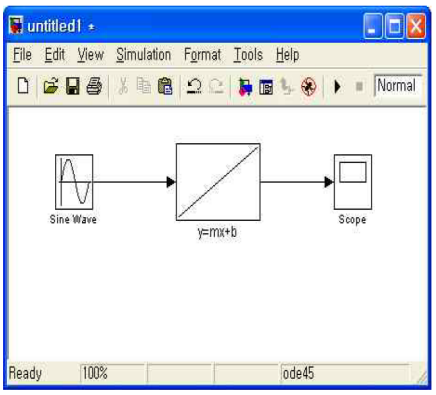
지금까지 대화상자에서 작성한 서브시스템은 mx+b이다. 그러나 서브시스템 블록은 특유한 simulink 서브시스템의 아이콘을 나타내어야 한다. 다음은 마스크 블록에 사용할 아이콘의 모양을 직선의 기울기로 나타내는 drawing commands를 입력한 상태이다.

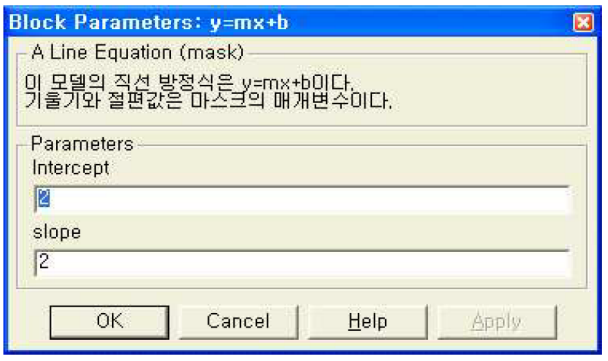


드로잉 명령은 m이 양수일 때 두 점 (0,0)과 (3,m)의 직선을 그리고, m이 음수일 때는 두 점 (0,1)과 (3,m+1)의 직선을 그린다.

④ 완성된 마스크

완성된 마스크는 다음과 같다. 마스크 블록을 더블클릭하면 Block parameters창이 활성화되어 Intercept값과 Slope값을 입력할 수 있다.

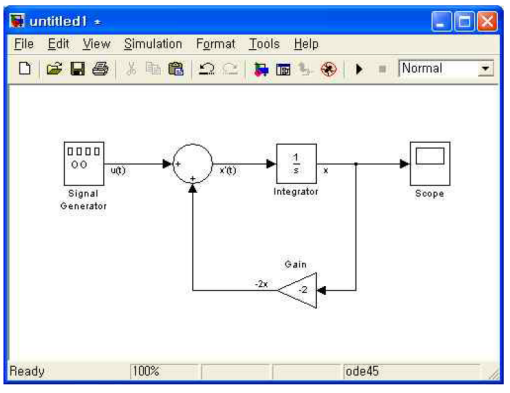


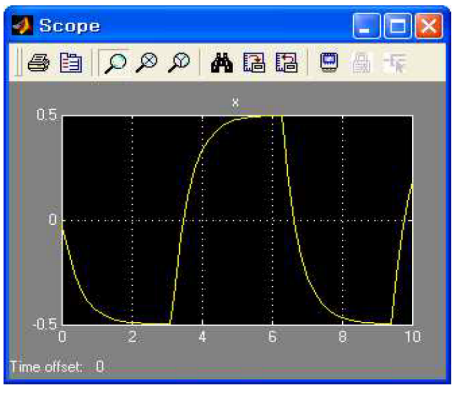


5. 모델링 방정식(Modeling Equation)

1) 간단한 연속 시스템의 모델링

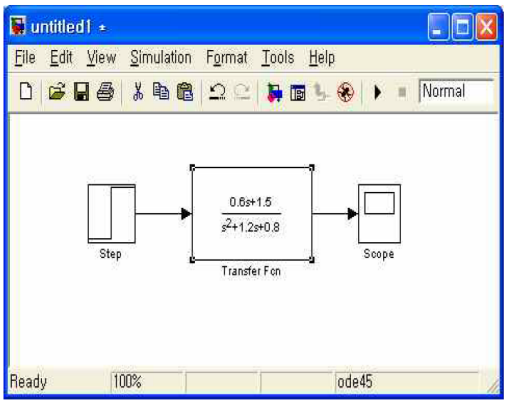
모델링하기 위한 방정식이 $\dot{x}(t) = -2x(t) + u(t)$와 같을 때 u(t)는 크기가 1이고 주파수는 1 rad/sec인 구형파이다.

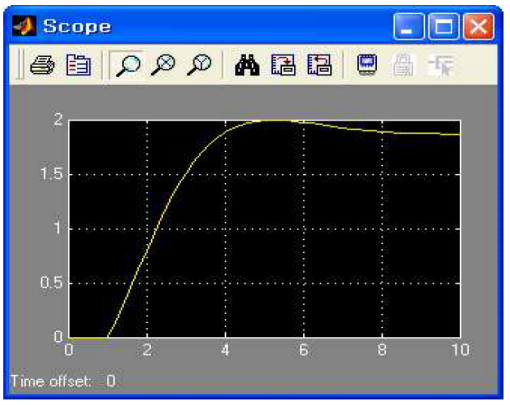




2) 전달함수 모델(Transfer Function Model)

단위 계단 입력함수가 $G(s)=\frac{0.6s+1.5}{s^2+1.2s+0.8}$ 일 때 시뮬레이션 결과는 다음과 같다.



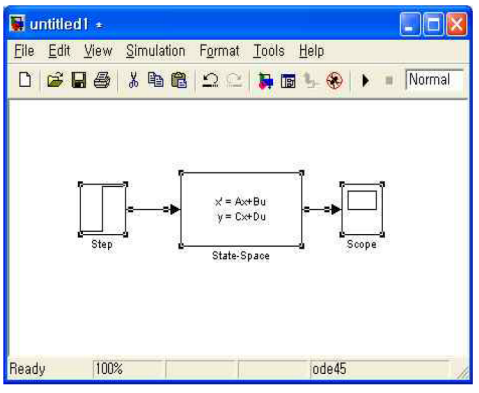


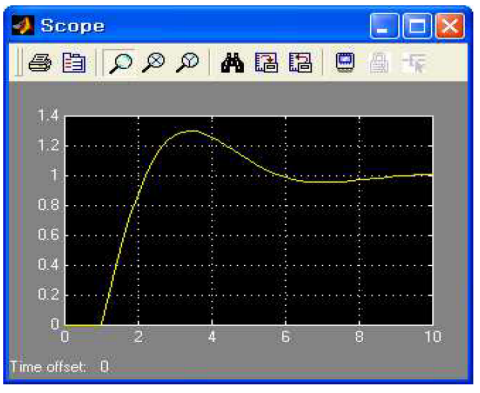
3) 상태 공간 모델(State-Space Model)

상태 공간 모델에서 단위계단 입력함수가 다음과 같은 때, simulink의 시뮬레이션 결과는 아래와 같다.

$$\dot{x}=\begin{bmatrix}0 & 1\\ -1 & -1\end{bmatrix}x+\begin{bmatrix}0\\ 1\end{bmatrix}u$$

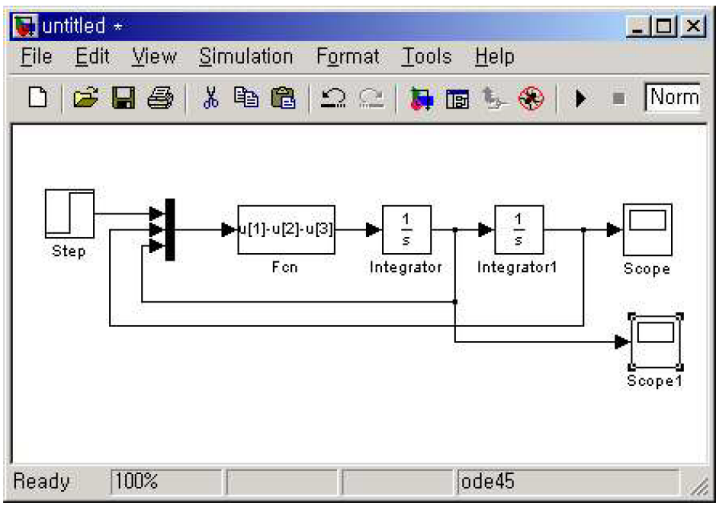
$$y=[1 \quad 1]x+[0]u$$

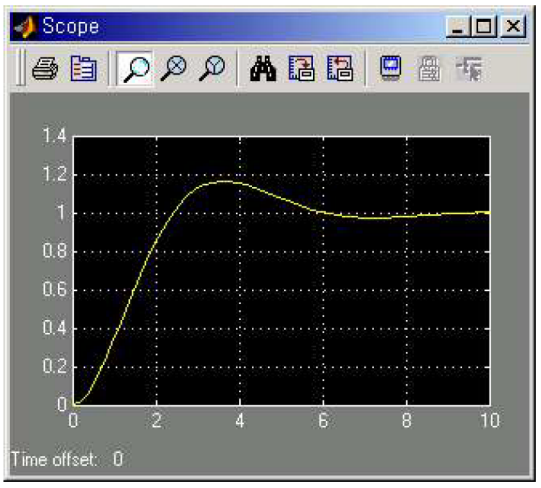


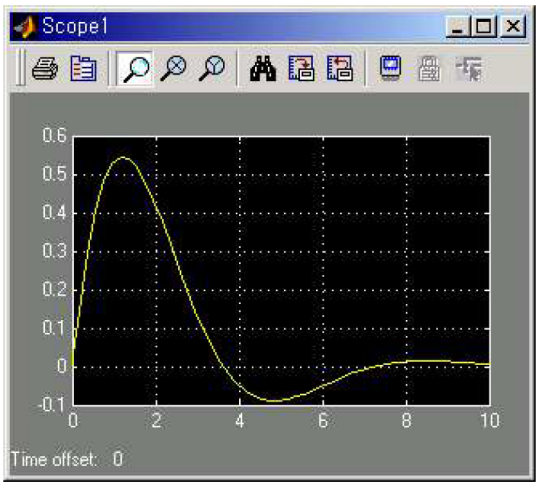


4)운동 방정식 모델

운동방정식 $m\ddot{x}+b\dot{x}+ks=F$ ($m=1$, $b=1$, $k=1$)에 대한







6. 시뮬레이션 구동

1) 메뉴 명령을 사용한 시뮬레이션 구동

- Solver : stop time과 start time을 설정하고 출력의 옵션등을 지정한다.
- Workspace I/O : 입력의 형태와 MATLAB 작업공간으로의 출력을 관리한다.
- Diagnostic : simulation되는 동안의 경고 메시지의 수준을 선택한다.

2) 매개변수(Parameters) 대화 상자

- The solver page

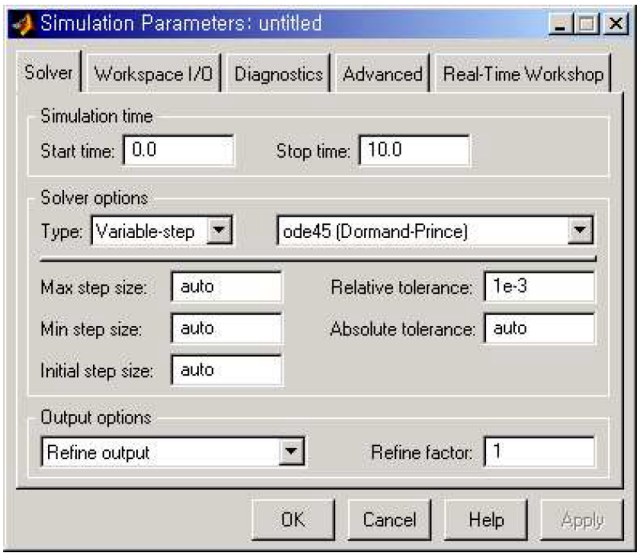


그림 4 매개변수 대화상자의 Solver page

◦ 시뮬레이션의 시작과 종료시간을 설정
◦ Solver와 solver의 parameter를 선택
◦ 출력의 옵션을 선택

○ The Workspace I/O Page

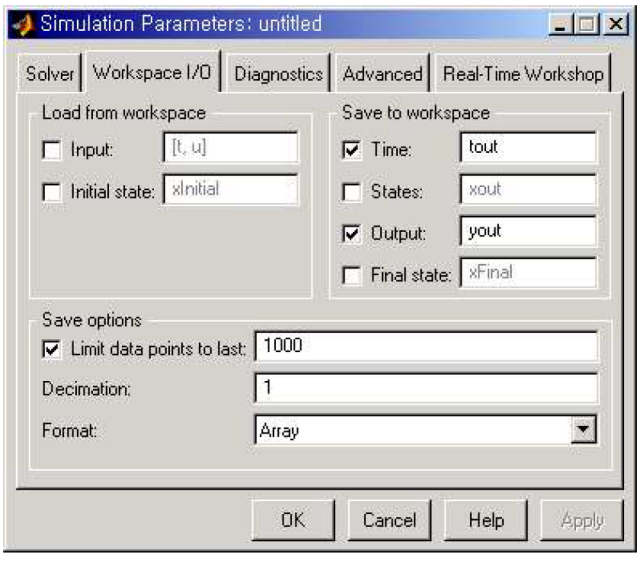


그림 5 Workspace I/O page

◦ 기본 Workspace에서의 입력설정
◦ Workspace로 출력을 저장하기
◦ 로드문과 저장문

○ Diagnostics page

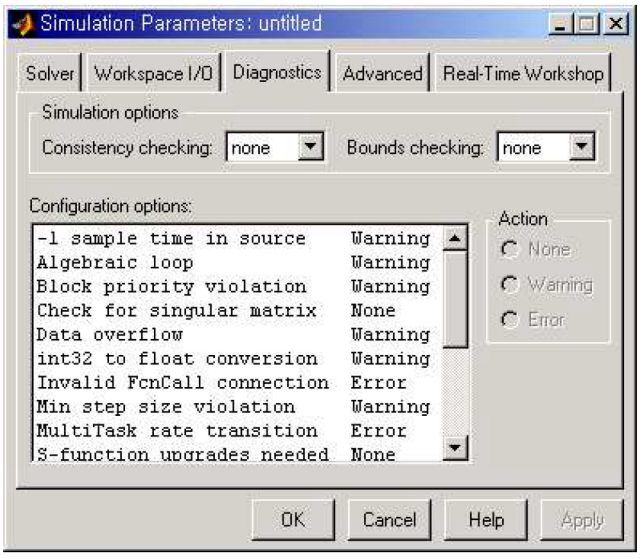


그림 6 Diagnostics page

◦ 시뮬레이션이 되는 동안 일어나는 여러 가지 상황(event)에 대한 메시지의 수준을 결정

◦ Warning은 시뮬레이션을 중단시키지 않지만 Error의 선택은 시뮬레이션을 중단시킴

7. 예제

그림과 같은 기계 장치 시스템을 고찰해 보자. 여기서 질량 $m=1$, $ks=100$, 그리고 $fdf=100$이다. 감쇠계수 $b=[0,\ 0.4,\ 0.8,\ 10]$가 변화 한다. 초기에 질량이 m인 물체가 $x(0)=1$, $\dot{x}(0)=0$으로 작용할 때 초기조건에 영향을 받는 물체의 운동을 구하라.

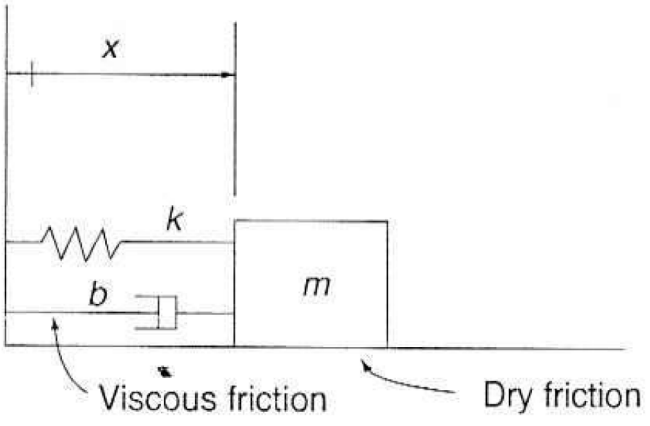


그림 7 기계 장치 시스템

(풀이)

이해를 돕기 위해서 MATLAB에서의 방법과 Simlulink를 사용하여 2가지 방법으로 풀이

이 시스템의 방정식은 다음과 같다.

$$m\ddot{x} + b\dot{x} + kx = F$$

초기값이 $x(0) = 1, \quad \dot{x}(0) = 0$이다. 이 시스템 방정식의 Laplace 변환은 아래와 같다.

$$X(s) = \frac{1}{s} \cdot \frac{m^2 + bs + F}{ms^2 + bs + k}$$

그러므로 질량 m인 물체의 운동은 아래와 같은 시스템의 단위 계단 응답과 같이 구해질 수

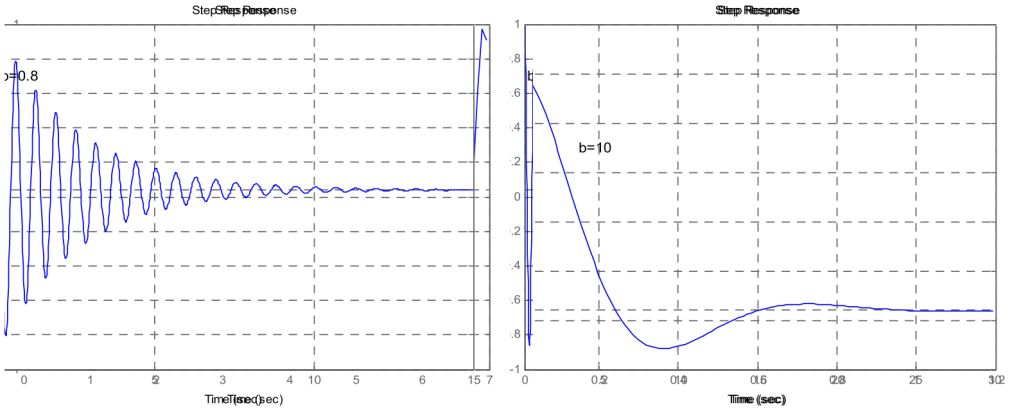
있다.

$$G(s) = \frac{m^2 + bs + F}{ms^2 + bs + k}$$

MATLAB Program으로부터 물체의 운동의 그래프를 얻을 수 있다.

방법1. MATLAB 프로그램

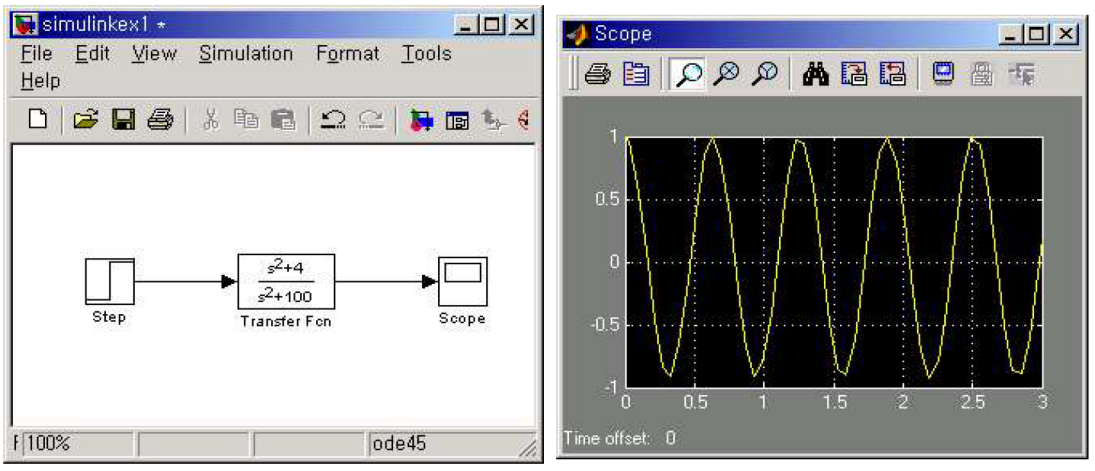
```
>> num1=[1 0 4]; den1=[1 0 100];
>> num2=[1 0.4 4]; den2=[1 0.4 100];
>> num3=[1 0.8 4]; den3=[1 0.8 100];
>> num4=[ 1 10 4]; den4=[1 10 100];
>> h1=step(num1, den1);
>> h2=step(num2, den2);
>> h3=step(num3, den3);
>> h4=step(num4, den4);
>> step(num1, den1); text(0.15,0.7,'b=0');grid
>> step(num2, den2); text(0.15,0.7,'b=0.4');grid
>> step(num3, den3); text(0.15,0.7,'b=0.8');grid
>> step(num4, den4); text(0.15,0.7,'b=10');grid
```



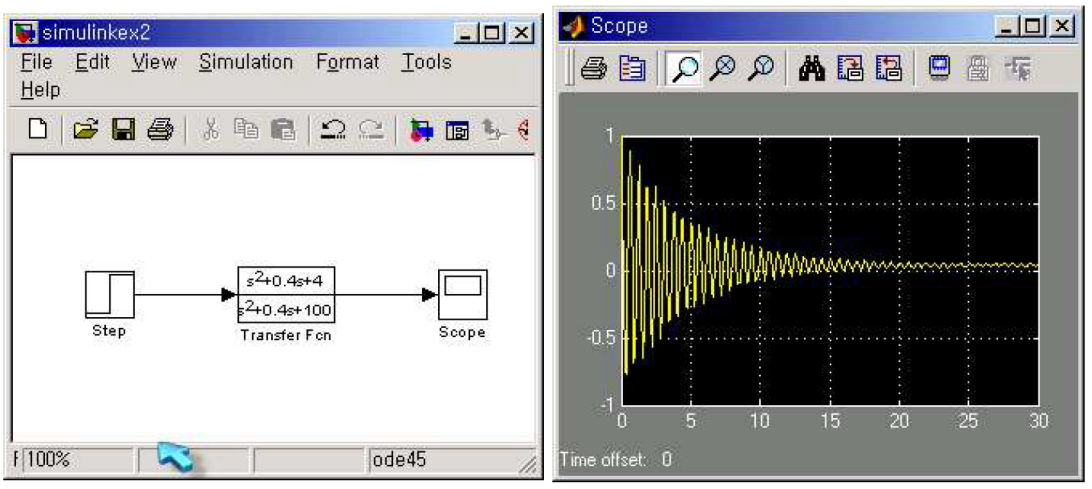
C

방법 2. Simulink에서의 step response를 이용한 결과

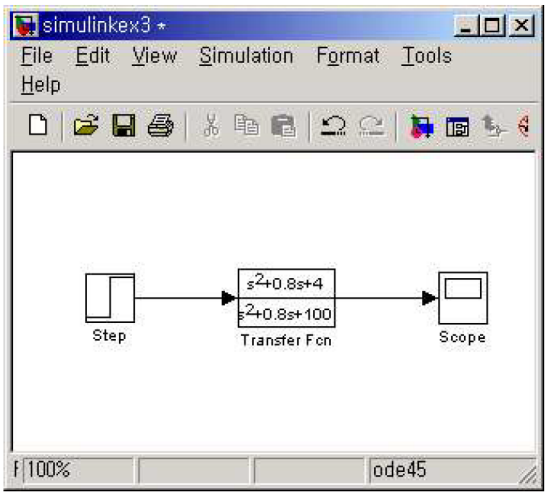
1) 감쇠계수 $b=0$일 때

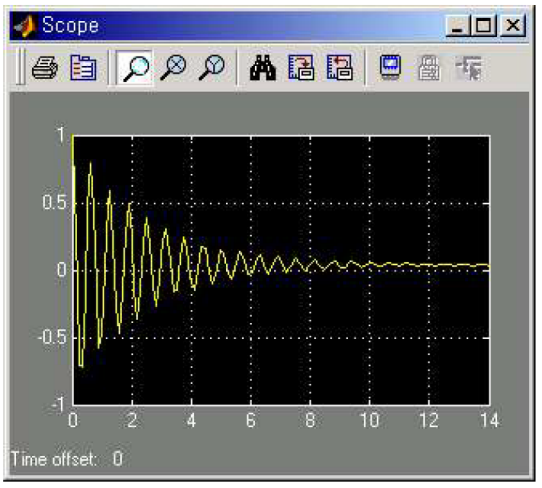


2) 감쇠계수 $b=0.4$일 때



3) 감쇠계수 $b=0.8$일 때





4) 감쇠계수 $b=10$ 일 때