

# Matlab 사용법

## A.1 Matlab의 개요

### 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.

### 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 전달 함수와 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 선형모델을 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.

## A.2 기본적인 사항

### Matlab 의 실행

Matlab프로그램을 실행하기 위해서는 Windows가 먼저 활성화되어 있어야 한다. 다음으로, Matlab을 표시하고 있는 아이콘(icon)을 더블 클릭(double click)함으로써(그림 A.1) Matlab프로그램을 실행할 수 있는 그림 A.2와 같은 명령창(command window)이 열리게 된다. 이 때

»

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

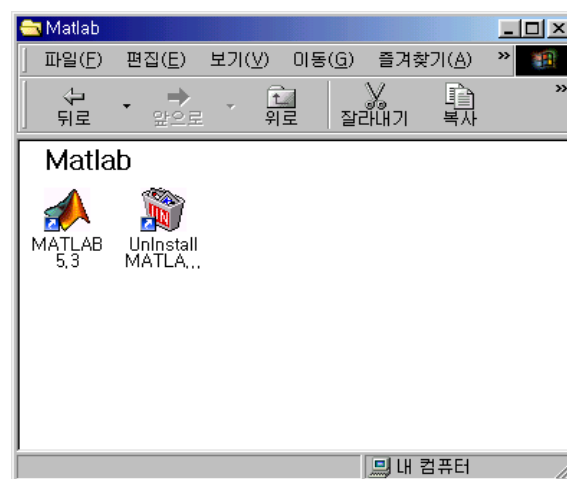


그림 A.1 Matlab 폴더의 아이콘(Icon)

Matlab을 끝내기 위해서는

» quit

혹은

» exit

를 사용한다.

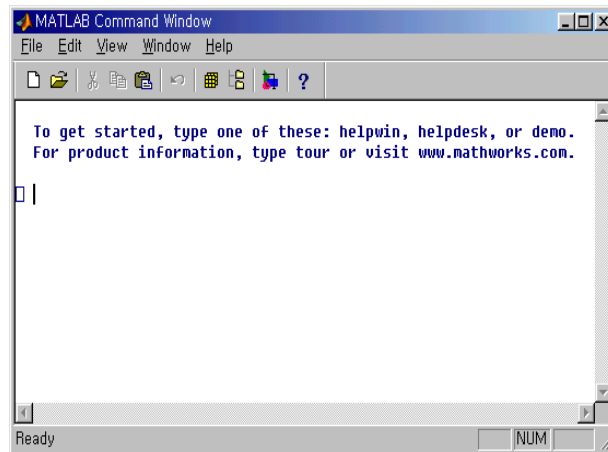


그림 A.2 Matlab 명령창(Command Window)

## 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 4.2 버전의 경우 자신이 사용하는 work라는 이름의 subdirectory를 등록하고 싶을 때는 path 명령을 이용해서 아래와 같이 실행하면 된다.

》 path(path, 'd:\tool\matlab\work')

또 다른 방법으로는 Matlab의 root directory 내에 있는 'matlabrc.m' 이름의 m-file을 편집기를 이용하여 work라는 subdirectory를 추가하면 된다. 이렇게 함으로써 항상 work라는 자신의 directory를 사용할 수 있다.

Matlab 5.x 버전에서는 그림 A.3과 같이 Matlab 명령창(command window)의 풀-다운 메뉴(File→Set Path)를 이용한 Path Browser에서 쉽게 path를 추가시키거나 제거할 수 있다.

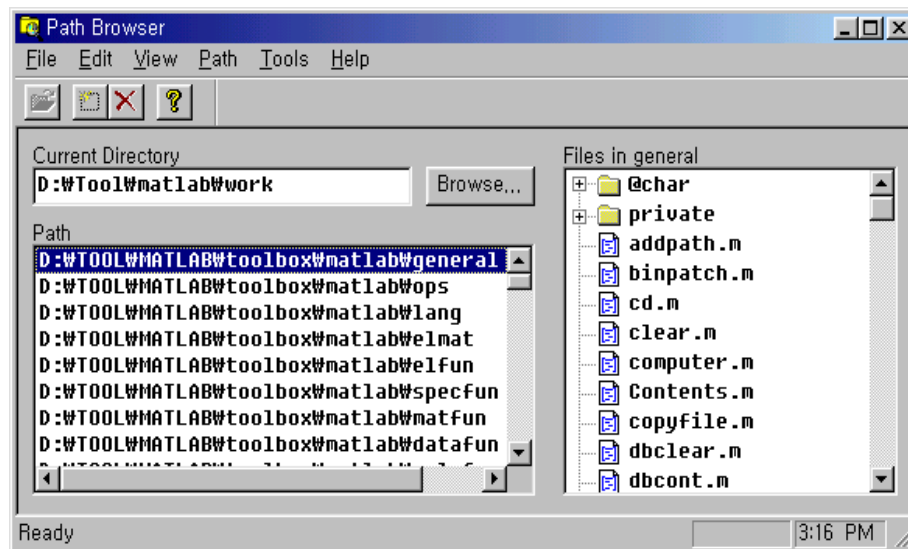


그림 A.3 Matlab의 Path Brower

## 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.

## 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 불러 들일 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

## A.3 기본 Matlab 명령어

### 행렬의 입력 방법

Matlab에서는 직각 수치 행렬을 기본적인 자료의 형식으로서 취급한다. 원소(element)로는 실수 및 복소수 그리고 문자도 사용 가능하다. 물론, 행렬에는 1×1의 스칼라와 하나의 행(1×n) 또는 하나의 열(n×1)만으로 구성된 벡터도 포함된다. Matlab에서 사용되는 연산들과 명령들은 가능한 한 자연적인 형태, 즉 축약 형태가 아닌 행렬 형식으로 표현하도록 되어 있다.

Matlab에서 사용되는 행렬 입력 방식에는 다음과 같이 몇 가지가 있다.

- ▶ Matlab 명령 행(command-line)에서 직접 행렬의 원소들을 하나씩

입력하여 행렬을 구성

- ▶ 내부 명령 또는 함수들을 사용하여 행렬을 생성
- ▶ M-파일 내에서 행렬을 구성
- ▶ 외부의 자료 파일로부터 행렬을 불러들임.

Matlab 언어는 다른 프로그래밍 언어들과는 달리 차원 선언이나 형선언이 없다. 현재 사용하고 있는 컴퓨터에서 사용 가능한 크기까지 자동적으로 저장 공간을 할당해 준다.

작은 크기의 행렬을 입력하는 가장 간단한 방법은 다음과 같이 직접적으로 원소들을 나열하여 입력하는 것이다.

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

아래의 예들을 살펴보면 위의 항목들이 쉽게 이해될 것이다.

```
»A=[1 2 3; 4, 5, 6; 7, 8 9]
```

또 다른 방법은 M-파일을 사용하여 행렬을 불러들이는 방법이다. 만일 test.m 파일이 다음과 같은 내용을 포함하고 있을 때

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

Matlab 명령 행에서 다음과 같이 입력하면

```
»test
```

Matlab은 M-파일 'test.m'을 읽어 행렬 A를 생성한다.

load 명령이나 fread 함수를 사용하면 이전의 Matlab 작업에서 생성하여 저장했던 행렬이나 다른 프로그램으로부터 만들어진 행렬들을 불러들일 수 있다.

## Matlab 산술문과 변수들

Matlab은 입력된 산술문(expression)을 해석하여 계산을 수행하는 언어이다. Matlab에서 산술문은 다음과 같은 형태를 갖는다.

```
»변수 = 수식
```

또는

```
»수식
```

수식은 연산자, 함수, 변수 등으로 구성될 수 있으며, 이 수식을 계산한 결과가 행렬로 생성된다. 수식 뒤에 ‘;’를 붙이지 않으면 수식의 결과가 변수에 할당되면서 화면에 표시되며, 변수를 지정하지 않았을 경우에는 수식의 결과가 ‘ans’라는 변수에 할당된다.

```
»y=log(5)
y = 1.6094
»log(7)
ans = 1.9459
»y=exp(4);
```

위에서와 같이 ‘;’를 사용하면 결과를 화면에 출력하지 않는다. 그러나 Matlab에서는 이 변수를 기억하고 있다.

현재 작업환경에서 사용되고 있는 변수는 who(s)라는 명령어를 사용해서 확인해볼 수 있다.

```
»whos
Name      Size      Bytes  Class
ans       1x1         8  double array
y         1x1         8  double array
z         1x1         8  double array
Grand total is 3 elements using 24 bytes
```

위와 같이 명령 창에서 whos를 입력하면 현재 사용중인 변수들의 name, size, bytes, class를 자세히 볼 수 있다.

하나의 수식이 너무 길어서 한 줄에 모두 나타낼 수 없을 경우에는 여러 줄에 표시해야 하는데 이 때에는 각 줄의 마지막에 하나 이상의 공백과 생략기호(‘...’)를 붙이고 를 친다.

```
»S=1 + 1/2 + 1/3 + 1/4 + 1/5 +1/6 + 1/7 ...
+ 1/8 + 1/9 + 1/10
```

Matlab에서는 영문자로 시작되는 19자 이하의 문자 및 숫자(특수기호들을 제외한 ‘\_’과 같은 기호도 포함)를 사용하여 변수 또는 함수의 이름으로 지정할 수 있다. 또한 Matlab에서는 대문자와 소문자를 구별하며, 함수의 이름은 항상 소문자이어야 한다. 즉 ‘A’와 ‘a’는 서로 다른 변수로 인식되면, ‘sum(A)’는

행렬 A의 원소들의 합을 의미하지만, 'SUM(A)'는 사용자가 특별히 정의하지 않는 한 정의되지 않은 함수로 취급한다.

Matlab에서는 다음 방법으로 입력된 변수를 다른 변수로 할당할 수 있다.

변수 이름 = 입력된 변수 이름

이 때 입력된 변수와 새로운 변수는 같은 내용을 포함한다.

```
»x=10; y=x  
y = 10
```

Matlab 작업 공간에 어떤 변수가 정의되어 있는지를 알아보기 위해서는 명령어 'exist'를 사용한다.

```
»exist('y')
```

만일 변수 y가 정의되어 있으면 1이 출력되고 그렇지 않으면 0이 출력된다. Matlab에서 '%'시작되는 줄은 실행되지 않는다. 따라서 '%'를 사용하여 주석이나 설명을 Matlab 명령 창이나 M-파일에 기록할 수 있다.

## 출력 형식

Matlab에서는 산술문의 결과가 화면에 표시되며, 이 때 표현되는 수치의 자릿수는 Matlab의 기본 설정 값에 따라 정해져 있다. Matlab 시작시 기본적인 수치적 자릿수는 항상 short로 설정되어 있다.

```
»pi  
ans = 3.1416
```

사용자가 화면에 출력하는 형식을 변경하기 위해서는 그림 A.4와 같이 풀다운 메뉴의 File→Preferences를 선택해서 Numeric Format을 바꿀 수 있고, 또한 그림 A.5와 같이 명령 창에서 직접 format이라는 명령어를 사용할 수 있다.

주의할 점은 format 명령은 단지 화면상에 표시되는 수치의 표현 형식만을 변경시킬 뿐이며, 변수값 자체의 정밀도는 바꾸는 것이 아니다.

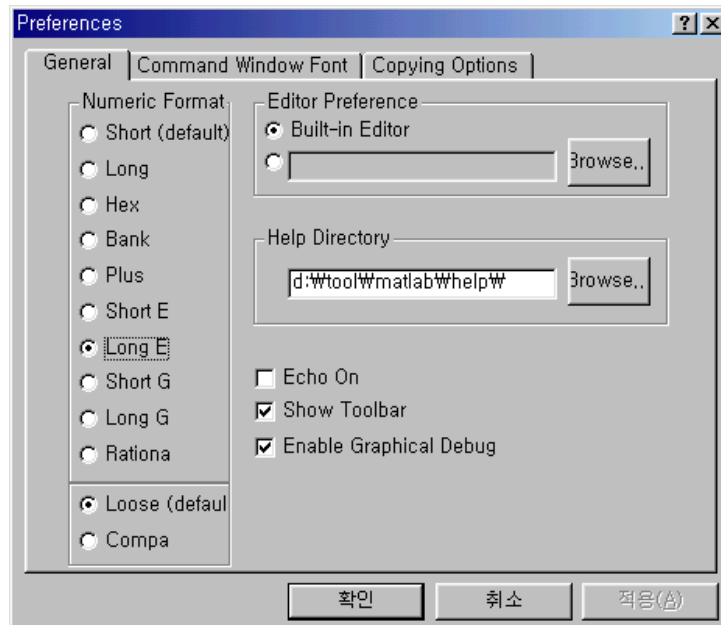


그림 A.4 Matlab의 Preferences

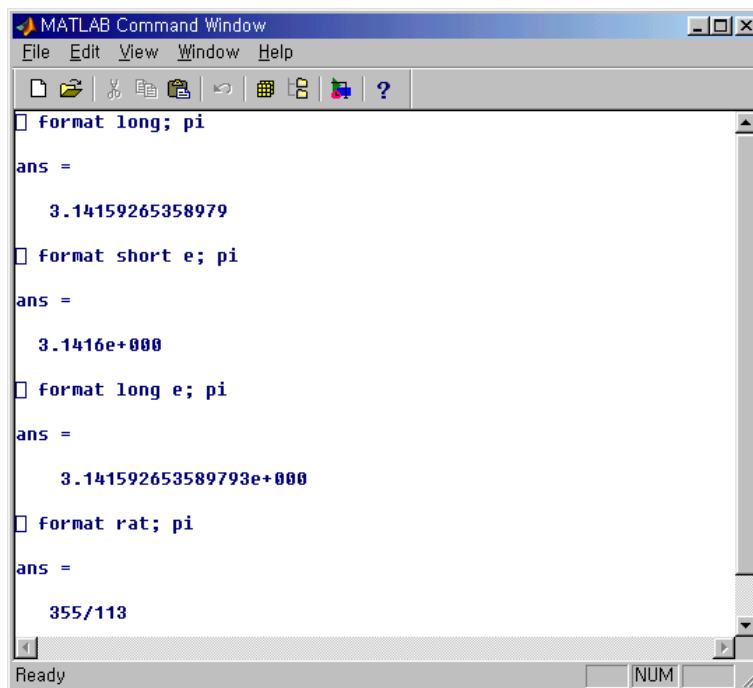


그림 A.5 Matlab 명령창에서의 Format 변경

## Matlab에서의 행렬 연산

Matlab에서 사용하는 행렬의 연산자는 다음과 같다.

|   |                                            |
|---|--------------------------------------------|
| + | 덧셈 (addition)                              |
| - | 뺄셈 (subtraction)                           |
| * | 곱셈 (multiplication)                        |
| ^ | 거듭제곱 (power)                               |
| ' | 전치 (transpose)                             |
| \ | 왼쪽 나눗셈 (left division)                     |
| / | 오른쪽 나눗셈 (right division)                   |
| . | 배열 연산 (elementary by elementary operation) |

### ■ 행렬의 덧셈과 뺄셈

같은 크기의 행렬의 덧셈 및 뺄셈에 +와 -기호를 사용한다.

```
»A = [1 2 3 ; 4 5 6];
»B = [2 4 6 ; 8 10 12];
»C = A+B
    C =     3         6         9
         12        15        18
»D = A-B
    D =    -1        -2        -3
         -4        -5        -6
```

1×1행렬인 스칼라의 경우에는 어떠한 행렬과도 덧셈과 뺄셈연산이 가능하다.  
이 경우 행렬의 모든 원소에 스칼라를 더하거나 뺀다.

```
»E = C-5
    E =    -2         1         4
         7        10        13
```

즉, E는 다음 연산과 같다.

```
»E = C-[5 5 5; 5 5 5]
```

## ■ 행렬의 곱셈

행렬 A가  $m \times p$ 이고 행렬 B가  $q \times n$ 이면 행렬 A와 B의 곱셈은  $p=q$ 일 때 가능하며 두 행렬의 곱 AB는  $m \times n$  행렬이며, Matlab에서는 '\*'를 사용한다.

```
>>A = [1 2 3; 4 5 6];
>>B = [1 2; 3 4; 5 6];
>>A*B
ans = 22    28
      49    64
```

## ■ 왼쪽 나눗셈과 오른쪽 나눗셈

Matlab에서 행렬의 나눗셈은 사용 용도가 다른 왼쪽 나눗셈과 오른쪽 나눗셈 두가지가 있다. 행렬 A가 역행렬을 갖는 정방 행렬이면 선형 방정식  $Ax=b$ 의 해를 구하기 위해 다음 왼쪽 나눗셈을 사용한다.

$A*x=b$ 의 해 :  $x = A \backslash b$  - 왼쪽 나눗셈 (left division)

다음과 같은 선형 방정식  $Ax=b$ 의 해를 구하기 위한 명령어는 아래와 같다.

```
3x+5y-3z = 8
5x+3y+2z = 22
2x+y-3z = 1

>>A = [ 3 5 -3; 5 3 2; 2 1 -3];
>>b = [8 22 1]';
>>x = A\b
x= 3
    1
    2
```

오른쪽 나눗셈은 행렬 방정식  $X*A = B$ 의 해 X를 구하기 위해 사용된다.

$X*A=B$ 의 해 :  $X=B/A$  - 오른쪽 나눗셈(right division)

오른쪽 나눗셈 ' $B/A$ '는 다음과 같은 왼쪽 나눗셈과 같다.

$B/A = (A' \backslash B')'$

따라서 3차 정방행렬 A의 역행렬을 구하기 위하여 다음 3가지 명령어 중 하나를 사용하면 된다.

```
» eye(3)/A
» A\eye(3)
» inv(A)
```

### ■ 행렬의 거듭제곱

행렬 A와 자연수 n에 대하여 연산 ' $A^n$ '은 행렬 A의 n 거듭제곱, A를 n번 곱한 것을 의미한다.

```
» A=[1 2 ; 3 4];
» A^3
ans = 37    54
      81   118
» A*A*A
ans = 37    54
      81   118
```

### ■ 전치행렬

행렬  $A=(a_{ij})$ 의 전치 행렬은

$$A^T = (\overline{a_{ji}})$$

이다. Matlab에서 전치 행렬을 구하기 위해서 '를 사용한다.

```
» A = [1 1-i ; i 2+2i ; 2i 3-3i] ;
» A'
ans = 1          0 - 1i      0 - 2i
      1 + 1i     2 - 2i      3 + 3i
```

### 배열 연산

배열 연산이란 \*, \, /, ^ 또는 ' 기호에 의하여 이루어지는 일반적인 선형 행렬 연산(linear algebraic matrix operation)이 아닌 행렬의 원소와 원소 사이

에 성립되는 산술 연산(element-by-element arithmetic operation)을 가리킨다. 연산자 앞에 점(.)을 찍음으로써 배열 연산 또는 원소 대 원소 연산을 나타낸다.

## ■ 배열 연산의 곱셈 및 나눗셈

두 행렬의 차원이 같을 때 연산자 ‘.’는 각 행렬의 같은 위치에 있는 원소들끼리의 곱셈을 나타낸다.

```
»A = [1 2 3; 4 5 6]; B = [1 3 6; 7 8 9];
»A.*B
ans = 1    6    8
      28   40   54
```

배열 연산의 나눗셈에는 ‘./’와 ‘.\’ 두 가지가 있다.

```
»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
```

## ■ 배열 연산의 거듭제곱

‘.^’은 원소간의 거듭제곱을 나타낸다.

```
»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
```

## 내장 행렬

### ■ 영 행렬과 단위 행렬

Matlab에서 다음 명령어를 사용하여 모든 원소가 0 또는 1로 이루어진 행렬 및 단위행렬을 생성할 수 있다.

|                |                           |
|----------------|---------------------------|
| zeros(n)       | 모든 원소가 0인 $n \times n$ 행렬 |
| zeros(m,n)     | 모든 원소가 0인 $m \times n$ 행렬 |
| zeros(size(A)) | 행렬 A와 크기가 같고 모든 원소가 0인 행렬 |
| ones(n)        | 모든 원소가 1인 $n \times n$ 행렬 |
| ones(m,n)      | 모든 원소가 1인 $m \times n$ 행렬 |
| ones(size(A))  | 행렬 A와 크기가 같고 모든 원소가 1인 행렬 |
| eye(n)         | $n \times n$ 의 단위 행렬      |
| eye(m,n)       | $m \times n$ 의 단위 행렬      |
| eye(size(A))   | 행렬 A와 크기가 같은 단위 행렬        |

예를 들면 다음과 같다.

```
>> zeros(3)
ans = 0  0  0
      0  0  0
      0  0  0
>> ones(2,3)
ans = 1  1  1
      1  1  1
>> A=[1 2 3; 4 5 6];
>> eye(size(A))
ans = 1  0  0
      0  1  0
```

### ■ 랜덤 행렬

Matlab에서는 여러가지 방법으로 임의의(random) 숫자를 생성한다. 'rand'는 0과 1 사이의 임의의 숫자와 균일하게 분포한 원소를 가지는 행렬을 생성하고, 'randn'은 정규분포, 즉 평균이 0이고 편차가 1인 Gaussian 분포를 가지는 행렬을 만든다.

|                |                                   |
|----------------|-----------------------------------|
| rand(n)        | 임의의 숫자를 원소로 하는 $n \times n$ 행렬    |
| randn(n)       |                                   |
| rand(m,n)      | 임의의 숫자를 원소로 하는 $m \times n$ 행렬    |
| randn(m,n)     |                                   |
| rand(size(A))  | 임의의 숫자를 원소로 하는 행렬 A와 크기가<br>같은 행렬 |
| randn(size(A)) |                                   |

예를 들면 다음과 같다.

```

>>rand(2)
ans =    0.9501    0.6068
        0.2311    0.4860
>>randn(2,3)
ans =  -0.4326    0.1253   -1.1465
        -1.6656    0.2877    1.1909

```

## 벡터와 행렬의 조작

### ■ 벡터의 생성

벡터를 만들고자 하면 ‘시작값:증분값:최종값 (a:h:b)’의 형식으로 지정하면 된다. 증분값을 지정하지 않고 ‘시작값:최종값 (a:b)’의 형식으로 입력하면 자동적으로 증분값이 1로 지정된다. 다음의 예를 보자.

```

>>x = 1:0.5:3
x = 1.0000    1.5000    2.0000    2.5000    3.0000
>>y = -3:3
y = -3    -2    -1     0     1     2     3

```

또한 linspace(x1,x2,n)을 이용하면 x1에서 x2까지를 n개의 등간격으로 분할해서 벡터를 만들 수 있다.

```

>>z = linspace(-1,1,6)
z = -1.0000   -0.6000   -0.2000    0.2000    0.6000    1.0000

```

위와 같이 생성된 벡터는 기본적으로 행(row)벡터가 된다. 만약에 열(column) 벡터를 만들고자 한다면 콜론을 사용하여 행벡터를 만든 후 작은 따옴표(')를 사용하여 열 벡터로 전환시키면 된다.

```
»x = linspace(1,2,8)';
»y = log(x);
»[x y]
z = 1.0000      0
    1.1429    0.1335
    1.2857    0.2513
    1.4286    0.3567
    1.5714    0.4520
    1.7143    0.5390
    1.8571    0.6190
    2.0000    0.6931
```

## ■ 행렬의 합성

행렬 A에 새로운 행을 추가하고 싶으면 세미콜론(;)을 사용하며, 새로운 열을 추가하고 싶으면 A뒤에 행의 크기가 같은 벡터를 그냥 적어 주면 된다.

```
»A = [1 2; 3 4]
A = 1 2
    3 4
»[A; [ 5 6]]
ans = 1 2
      3 4
      5 6
»y = [ 5 6]'
y = 5
    6
»[A y]
ans = 1 2 5
      3 4 6
```

## ■ 부행렬

Matlab에서는 행렬  $A=(a_{ij})$ 의 원소 또는 일부 원소들을 다음 명령어로 선택하여 부행렬(submatrix)을 구성할 수 있다.

|              |                                              |
|--------------|----------------------------------------------|
| $A(i,j)$     | 행렬 $A$ 의 원소 $a_{ij}$ 를 선택                    |
| $A(:,j)$     | 행렬 $A$ 의 $j$ 번째 열을 선택                        |
| $A(i,:)$     | 행렬 $A$ 의 $i$ 번째 행을 선택                        |
| $A(:,j:k)$   | 행렬 $A$ 의 $j$ 번째 열부터 $k$ 번째 열까지 선택            |
| $A(j:k,:)$   | 행렬 $A$ 의 $j$ 번째 행부터 $k$ 번째 행까지 선택            |
| $A(j:k,q:r)$ | 행렬 $A$ 의 $j$ 부터 $k$ 까지 행과 $q$ 부터 $r$ 까지 열 선택 |

```
»A = [ 1 2 3 4 ; 5 6 7 8 ; 9 10 11 12; 13 14 15 16]
```

```
A =  1   2   3   4
      5   6   7   8
      9  10  11  12
     13  14  15  16
```

```
»A(2,3)
```

```
ans = 7
```

```
»A(4,:)
```

```
ans = 13  14  15  16
```

## A.4 Matlab의 프로그래밍

### 관계 연산자와 논리 연산자

#### ■ 관계연산자

같은 크기의 두 행렬( $1 \times 1$  행렬 포함)의 서로 대응되는 원소의 크기를 비교하기 위하여 사용되는 관계 연산자(relational operator)는 아래와 같이 6가지가 있으며, 두 원소를 비교하여 참이면(true) 1, 거짓(false)이면 0을 비교한 원소의 위치에 나타낸다.

|            |                |
|------------|----------------|
| $A < B$    | A가 B보다 작다.     |
| $A \leq B$ | A가 B보다 작거나 같다. |
| $A == B$   | A가 B와 같다.      |
| $A \sim B$ | A가 B와 같지 않다.   |
| $A \geq B$ | A가 B보다 크거나 같다. |
| $A > B$    | A가 B보다 크다.     |

```

>>x = [1 4 7 -2 1];
>>y = [3 1 7 -3 1];
>>x > y
ans = 0 1 0 1 0
>>x ~ = y
ans = 1 1 0 1 0
>>x <= y
ans = 1 0 1 0 1

```

관계 연산자는 find와 같이 유용하게 쓰일 수 있다. 주어진

```
>>x = [ 3 -3 2 7 0 5 2];
```

에 대하여 4보다 큰 원소의 위치를 알고 싶으면

```

>>find(x>4)
ans = 4 6

```

를 사용하고 4보다 큰 원소의 개수를 알고자 하면

```

>>length(find(x>4))
ans = 2

```

를 사용한다.

## ■ 논리 연산자

Matlab에서 사용되는 논리 연산자는 &, |, ~가 있으며, 그 결과가 참이면 1, 거짓이면 0을 나타낸다.

|       |            |
|-------|------------|
| A & B | 논리곱(and)   |
| A   B | 논리합(or)    |
| ~ A   | 논리 부정(not) |

주어진

```
»x = [7 -2 6 3 5 10 8 7];
```

에 대하여

```
»(x > 5) & (x < 9)
ans = 1 0 1 0 0 0 1 1
```

이다. 1이 나타난 원소의 위치의 x원소들은 5보다 크고 9보다 작음을 의미한다. 또 다음 명령어를 사용하면 x의 원소중 5보다 크지 않은 원소의 위치에 1이 나타난다.

```
»~(x > 5)
ans = 0 1 0 1 1 0 0 0
```

## 흐름 제어문

Matlab은 일반적인 프로그래밍 언어들과 마찬가지로 흐름 제어문(flow control statement)을 포함하고 있다. 이러한 흐름 제어문들을 사용함으로써 단순한 전자 계산기 수준 이상의 기능을 Matlab에서 수행할 수 있으며, 아울러 완벽한 고급 프로그래밍 언어로서도 사용될 수 있다.

### ■ FOR 루프

Matlab에서 'FOR'는 Fortran 언어 'DO'에 해당하는 것으로 하나 이상의 문장을 미리 정해진 회수만큼 반복 수행하도록 하는 명령어이다.

```
for n = 수식(expression)
    수행문장(statement group)
end
```

1부터 10까지의 숫자들을 벡터 x에 차례대로 채워 넣는 작업은 아래와 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]
```

만약에 행렬의 인수(행과 열) 모두를 변화시키면서 어떤 기능을 수행하고자 할 때에는 중첩된 루프(nested loop)를 사용할 수 있다.

```
»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
```

## ■ WHILE 루프

WHILE 루프는 다음과 같은 형식을 가지는 데, 논리 수식(logical expression)이 만족될 때 까지 수행 문장(statement group)을 실행한다.

```
while 논리 수식(logical expression)
    수행 문장(statement group)
end
```

$n!$ ( $n$  계승)의 값이 처음으로 십진수 10000보다 커지는  $n$ 의 값을 구하고자 할 때 다음과 같이 while 루프를 사용할 수 있다.

```

>>n=1;
>>while prod(1:n) <= 10000
    n = n+1;
end
>>n
n= 8

```

위의 prod(1:n)은 1부터 n까지의 곱을 나타내는 Matlab명령어이다.  
또한, 함수  $f(x)=\ln(x+1)$ 의 Maclaurin 급수가 다음과 같을 때

$$\ln(x+1) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} x^n}{n} = x - \frac{1}{2} x^2 + \frac{1}{3} x^3 - \frac{1}{4} x^4 + \frac{1}{5} x^5 + \dots$$

x=0.5에서 뒤의 항의 절대값이  $10^{-10}$ 보다 작게 될 때까지 위 급수의 합을 구하기 위해서

```

>>format long
>>s=0; x=0.5; n=1;
>>while (x^n)/n >= 10^(-10)
    s=s+((-1)^(n+1))*((x^n)/n);
    n=n+1;
end, n, s

```

을 사용하면 다음 결과를 얻는다.

```

>>n=29
>>s=0.40546510806486

```

따라서 29번째 항까지 급수의 합은 0.40546510806486이다.

## ■ *IF, ELSE, ELSEIF*

IF 문은 프로그램의 흐름을 제어하는 가장 기본적인 형태로 조건적으로 문장을 실행하는 명령어이다. 가장 간단 IF 문은 다음과 같다.

```

if 수식(expression)
    수행 문장(statement group)
end

```

수식(expression)이 사실이면 수행 문장을 실행하고, 그렇지 않으면 무시된다.

IF 문이 한 문장을 수행할 것인지 아닌지를 선택할 수 있도록 하는 반면에 IF-ELSE문은 두 문장 간의 선택을 할 수 있게 해 준다.

```

if 수식(expression)
    수행 문장 1 (statement group 1)
else
    수행 문장 2 (statement group 2)
end

```

수식이 사실이면 수행 문장 1을 실행하고, 그렇지 않으면 수행 문장 2을 실행한다.

주어진 행렬 A에 대하여  $\det \neq 0$ 이면 행렬 A의 역행렬을 구하고, 그렇지 않으면  $A^2$ 을 계산하는 명령어는 다음과 같다.

```

>> if det(A) ~= 0
    inv(A)
else
    A^2
end

```

ELSEIF를 사용하면 두 가지 이상의 많은 문장들 중에서 선택할 수 있도록 하여 준다.

```

if      수식 a (expression a)
    수행 문장 1 (statement group 1)
elseif 수식 b (expression b)
    수행 문장 2 (statement group 2)
end

```

수식 a가 사실이면 수행 문장 1을 실행하고, 수식 a가 거짓이고 수식 b가 사실이면 수행 문장 2를 실행한다.

다음 행렬을 생성하는 명령어는

$$A = \begin{bmatrix} 5 & 1 & 0 & 0 \\ 1 & 5 & 1 & 0 \\ 0 & 1 & 5 & 1 \\ 0 & 0 & 1 & 5 \end{bmatrix}$$

```

>>for i=1:4
    for j=1:4
        if i==j
            A(i,j)=5;
        elseif abs(i-j)==1
            A(i,j)=1;
        end
    end
end
end

```

이다.

위에서 보는 바와 같이 ELSEIF는 자신의 END를 요구하지 않는다.

ELSEIF는 다음과 같이 자신의 ELSE 절을 가질 수 있다.

```

if      수식 a (expression a)
    수행 문장 1 (statement group 1)
elseif 수식 b (expression b)
    수행 문장 2 (statement group 2)
else
    수행 문장 3 (statement group 3)
end

```

이 경우 수식 a와 b가 모두 거짓이면 수행 문장 3을 실행한다.

## A.5 그래픽과 M-파일

### Matlab 그래프

Matlab의 그래픽 시스템은 자료를 가시적으로 표현(visualization)하기 위한 정교한 기법들을 다양하게 제공하고 있는데, 이 시스템은 객체 특성들(object properties)의 값을 조정함으로써 형태를 변형시킬 수 있는 선 또는 면들과 같은 ‘그래픽 객체들(graphics objects)’의 집합으로 구성되어 있다. 그러나 Matlab에서는 높은 수준의 2차원 3차원 그래픽 함수들을 풍부하게 제공하고

있기 때문에, 대개의 경우에는 이와 같은 저급(low-level)의 그래픽 객체를 직접 다룰 필요는 없을 것이다.

Matlab 명령어 `plot`은 기본적인 그래프를 그리는 데 사용되며, 입력 자료에 따라 자동적으로 축의 배율을 설정하여 그래프를 그려 준다. 구간  $[-4,4]$ 에서 함수  $y=\sin x$ 의 그래프를 그리기 위한 명령어는 아래와 같고 출력 결과는 그림 A.6과 같다.

```

>>x = -4:0.05:4;
>>y = sin(x);
>>plot(x,y)

```

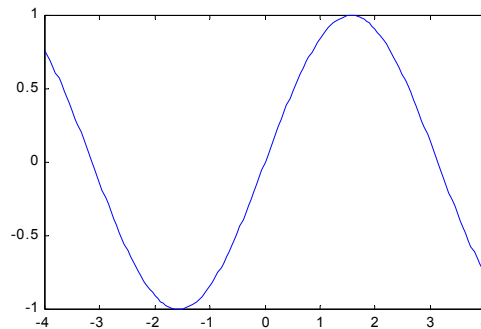


그림 A.6 Sin 함수의 그래프 출력

위에서 본 바와 같이 `plot(x,y)`는 두 개의 인수  $x, y$ 에 대하여 앞의 인수를  $x$ 축으로 뒤의 인수를  $y$ 축으로 지정하여 그래프를 작성한다.

`plot`을 사용하여 그래프를 작성한 후 다음 명령어들을 사용하면 그래프의 제목, 축의 이름, 눈금 격자(grid line) 및 그래프에 대한 설명 등을 추가할 수 있다.

|                     |                                 |
|---------------------|---------------------------------|
| <code>title</code>  | 그래프의 제목을 붙여 준다.                 |
| <code>xlabel</code> | $x$ 축에 축의 이름을 붙여 준다.            |
| <code>ylabel</code> | $y$ 축에 축의 이름을 붙여 준다.            |
| <code>text</code>   | 지정한 위치에 원하는 글을 표시해 준다.          |
| <code>gtext</code>  | 마우스를 사용하여 글의 위치를 지정한다.          |
| <code>grid</code>   | 눈금 격자(grid line)들을 그래프에 나타내 준다. |
| <code>axis</code>   | $x$ - $y$ 축의 배율을 조정한다.          |

다음을 실행하면 그림 A.7를 그릴 수 있다.

```

>>x = -3:0.05:3;
>>y = cos(2*x).*log(x+4);
>>plot(x,y)
>>xlabel('x-axis'); ylabel('y-axis');
>>title('y=cos(2x)log(x+4)')
>>grid

```

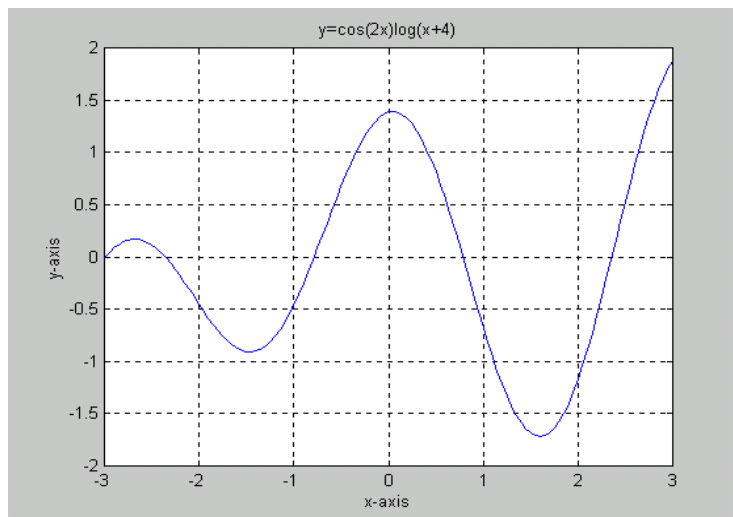


그림 A.7  $\cos(2x)\log(x+4)$ 의 그래프 출력

서로 다른 자료들을 사용하여 여러 개의 그래프를 그릴 때 서로 구별할 필요가 있다. 이 때 사용 가능한 선의 색상과 형태는 다음과 같다.

| 기호 | 색상           | 기호 | 선의 형태          |
|----|--------------|----|----------------|
| y  | yellow(노란색)  | .  | point(점)       |
| m  | magenta(자홍색) | o  | circle(원)      |
| c  | cyan(하늘색)    | x  | x-mark(x 표시)   |
| r  | red(빨간색)     | +  | plus(덧셈 기호)    |
| g  | green(녹색)    | *  | star(별표)       |
| b  | blue(파란색)    | —  | solid(실선)      |
| w  | white(흰색)    | :  | dotted(점선)     |
| k  | black(검정색)   | -. | dashdot(일점 쇄선) |
|    |              | -- | dashed(단선)     |

구간  $[0, 4\pi]$ 에서 함수  $y_1 = \sin x$ 는 실선으로  $y_2 = \cos x$ 는 점선으로 그래프를 그리려면 다음과 같다. text의 첫 번째 숫자는 x좌표를 두 번째 숫자는 y 좌표를 의미한다.

```

>>x = 0:0.05:4*pi;
>>y1 = sin(x);
>>y2 = cos(x);
>>plot(x,y1,'-',x,y2,':')
>>text(2.8, 0.5,'sin(x)')
>>text(11.8, 0.5,'cos(x)')

```

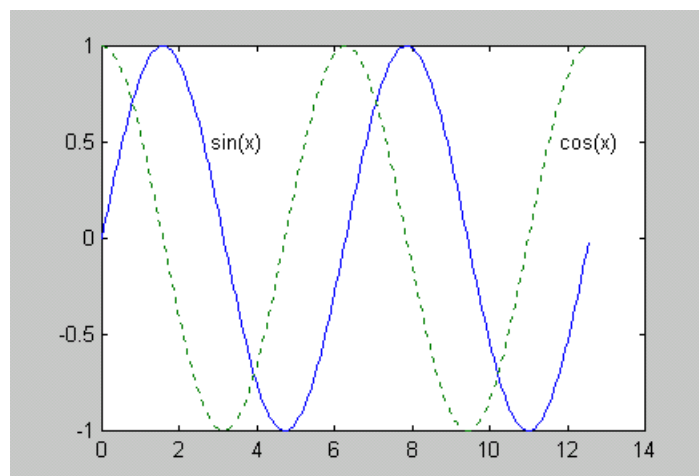


그림 A.8 Sin 함수와 Cos 함수의 텍스트 삽입

Matlab에서 작성된 그림을 워드프로세서나 다른 윈도우용 응용 프로그램(Application)에서 사용하고자 할 때는, Matlab에서 그림을 복사(copy)해서 사용하고자 하는 응용 프로그램에서 붙이기(paste)를 해야 한다.

복사 방법은 우선 Matlab의 그림창에서 풀-다운 메뉴(edit→copy options...)를 이용해 복사하고자 하는 그림을 그림 A.9와 같이, meta file 형태(format)로 할 것인지, 또는 bitmap 형태(format)로 할 것인지를 선택해야 한다. 또한 그 외 복사 옵션을 선택할 수도 있다.

원하는 복사 옵션을 선택했으면 그림창의 풀-다운 메뉴에서 그림 A.10과 같이 Copy Figure를 선택하면 윈도우 클립보드(clipboard)로 그림이 복사된다.

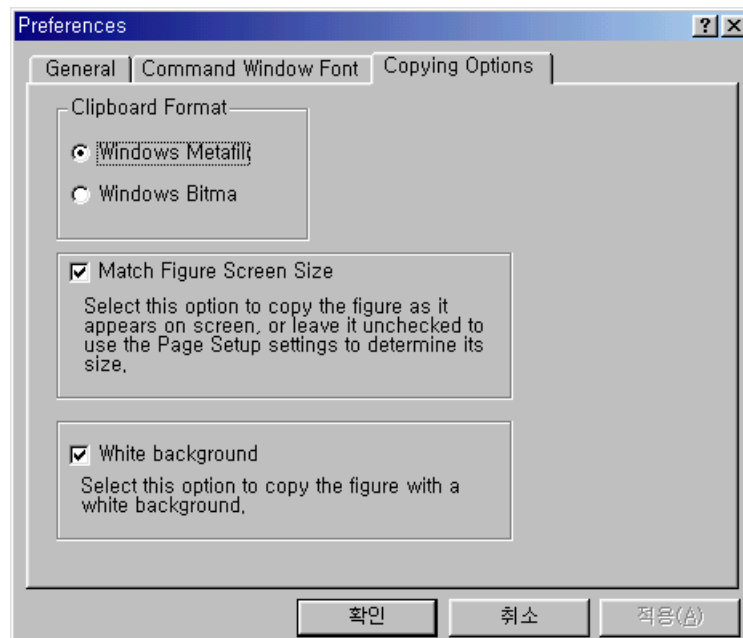


그림 A.9 Preferences의 Copying Options

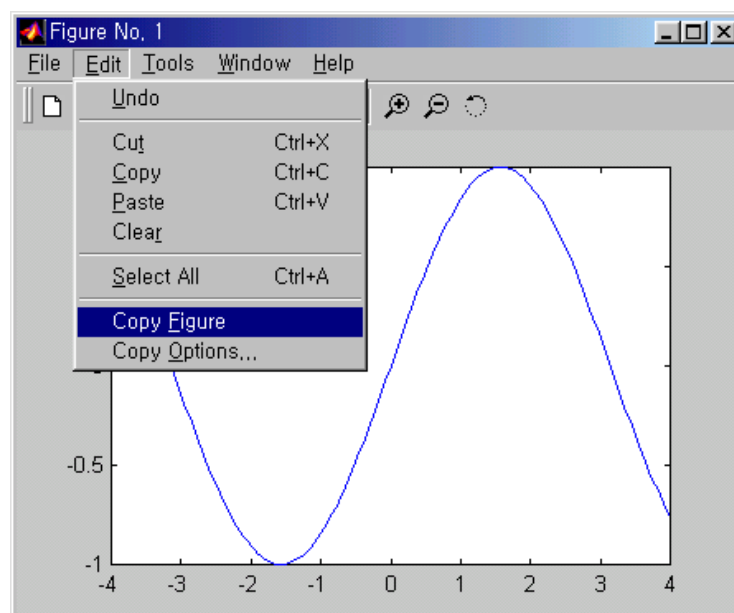


그림 A.10 그림(Figure) 창에서의 그림 복사

예로써 이 그림을 한글 워드프로세서에서 사용하고자 할 때는 그림 A.11과 같이 한글 워드프로세서의 풀-다운 메뉴에서 ‘편집→콜라 붙이기’를 선택하면 된다.

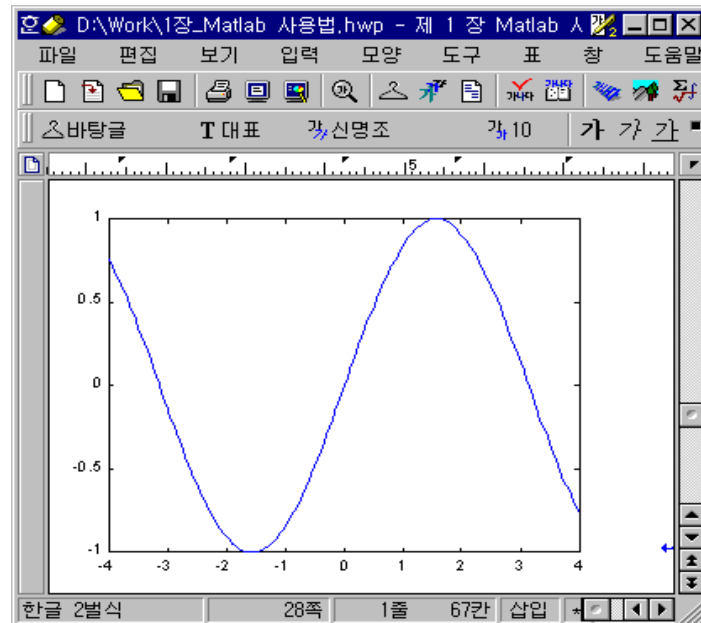


그림 A.11 한글 워드프로세서에서의 그림 붙여넣기

## M-파일

일반적으로 Matlab은 명령 구동 방식(command-drive 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 실행 할 수도 있다. 아울러 이러한 두 가지 방식은 해석기 방식의 환경(interpretive environment)을 구성한다.

Matlab의 실행 문장들을 포함하고 있는 파일들을 ‘M-파일(macro-file)’이라고 부르는데, M-파일은 일련의 정상적인 Matlab 실행 문장들로 구성되어 있으며, 실행 문장들 중에는 다른 M-파일을 참조하는 문장도 있을 수 있다. 또한 M-파일은 자체 M-파일 내에서 그 자신을 부를 수도 있다. 이러한 M-파일들은 일반적인 문자 편집기나 워드프로세서를 사용하여 작성할 수 있으며, ASCII 코드로 구성되어 있다.

## ■ 스크립트 파일(*Script File*)

스크립트 파일을 실행 시키면, Matlab은 단순히 파일 내에 들어 있는 명령들을 차례대로 실행한다. 스크립트 파일 내의 문장들은 Matlab 작업 공간상의 자료들을 전역 변수(global variable)로 취급하여 처리한다. 이러한 스크립트 파일은 일일이 한 줄씩 입력하면서 그 수행 결과를 알아보기에는 번거로운 자료 해석의 처리, 문제 풀이 또는 일련의 긴 명령어들의 구성에 유용하다.

만일 'test.m'파일이 다음과 같은 내용을 포함하고, Matlab 경로에 있는 디렉토리에 저장되었다고 하자.

```
x=(0:0.2:1)';  
y1=sin(x);  
y2=cos(x);  
[x y1 y2 y1-y2]
```

Matlab 명령창에서 'test'라고 입력하면 Matlab은 위의 모든 명령을 실행하고 다음 결과를 출력한다.

```
ans =  
      0      0  1.0000 -1.0000  
      0.2000  0.1987  0.9801 -0.7814  
      0.4000  0.3894  0.9211 -0.5316  
      0.6000  0.5646  0.8253 -0.2607  
      0.8000  0.7174  0.6967  0.0206  
      1.0000  0.8415  0.5403  0.3012
```

즉, 첫 번째 열에는  $x=0, 0.2, 0.4, 0.6, 0.8, 1$ 을 두 번째 열과 세 번째 열에는 각각의  $x$ 에서의  $\sin x$ 와  $\cos x$ 의 값을, 마지막 열에는  $\sin x - \cos x$ 의 값을 표시해 준다. 만일 위와 다른  $x$ 에서의 함수  $\sin x$ 와  $\cos x$ 의 값을 계산하고자 할 때나, 다른 함수의 값을 구하고자 할 때는 간단히 'test.m' 파일을 편집한 후 다시 Matlab 명령창에서 'test'이라고 입력하면 원하고자 하는 결과를 다시 출력한다.

## ■ 함수 파일(*Function File*)

파일이 시작되는 첫 줄에 'function'이라는 단어가 들어 있는 M-파일이 바로 '함수 파일(function file)'이다. '함수 파일'에서는 인수가 사용될 수 있으며, 파

일 내에서 정의되고 조작된 변수들은 그 함수에 국한될 뿐이고 Matlab의 작업 공간에 저장되어 전역 변수(global variable)가 아닌 지역변수(local variable)로 사용된다는 점에서 ‘스크립트 파일’과는 다르다. Matlab에서는 Matlab의 자체 언어를 사용하여 새로운 함수들을 생성할 수 있기 때문에 함수 파일은 Matlab의 기능을 확장하는데 아주 유용하다.

어떤 행렬 또는 벡터 자료에 대한 평균값을 구하는 ‘mean1.m’이라는 함수를 작성해 보면 다음과 같다.

```
function y = mean1(x)
% MEAN1 Average or mean value.
% For vectors, MEAN1(X) is the mean value of the elements in X.
% For matrices, MEAN1(X) is a row vector containing the mean value
%   of each cloum.

[m,n] = size(x);
if m==1
    m=n;
end
y = sum(x)/m;
```

‘mean1.m’ 파일을 이용해서 1부터 100까지의 정수로 구성된 벡터 t가 있을 때, 다음과 같이 입력하여 이 벡터의 평균값을 구할 수 있다.

```
»t=1:100;
»mean1(t)
ans = 50.5000
```

여기서 작성한 ‘mean1.m’에 대하여 구체적으로 살펴보자.

- ▶ 파일의 첫 번째 줄에 함수의 이름, 입력 인수 및 출력 인수가 선언되며, 이 부분이 ‘함수 파일’과 ‘스크립트 파일’의 가장 큰 차이점이다. 즉 만약 이 부분이 존재하지 않으면 ‘스크립트 파일’로 인식되고, 이 부분이 존재하면 ‘함수 파일’로 인식된다.
- ▶ ‘%’ 기호의 뒤에 나오는 문장들은 주석(comment)으로 인식되어 실행시 무시된다.

- ▶ 파일의 윗 부분에 '%' 기호와 함께 적어 놓은 설명 부분은 Matlab 프롬프트에서 'help mean1'이라고 입력했을 때 모니터 화면 상에 표시된다.
- ▶ 파일의 첫 번째 줄에 나타나는 변수들 y, m 및 n은 함수 'mean1'에 국한된 변수들로서 'mean1'이 실행되고 난 후에는 작업 공간 상에 존재하지 않게 된다. 만약 함수 'mean1'을 실행시키기 전에 같은 이름의 변수가 작업 공간상에 존재 하였다 해도 'mean1'의 실행이 완료된 후에 그 변수들은 그대로 보존된다.
- ▶ 함수 파일의 변수들은 지역 변수(logical variable)들이기 때문에, 함수 'mean1'에 사용되는 입력 인수의 이름이 반드시 M-파일 내의 변수인 'x'와 같을 필요는 없다. 즉, 위의 예에서 볼 수 있듯이 't'라는 벡터를 'mean1'의 인수로 입력하면 그 값이 'mean1'에 복사되면서 지역 변수 x로 전달되어 처리된다.