1. MATLAB 시작

원래 최초의 MATLAB은 Cleve Moler에 의해 포트란으로 작성되었으나, 현재는 미국의 MathWorks (http://www.mathworks.com) 사가 계속해서 개발하고 있다. MATLAB의 코딩 체계는 일반적인 프로그래밍 언어와 유사한 문법으로 이루어져 있는데, 주로 이용되는 범위는 다음과 같다.

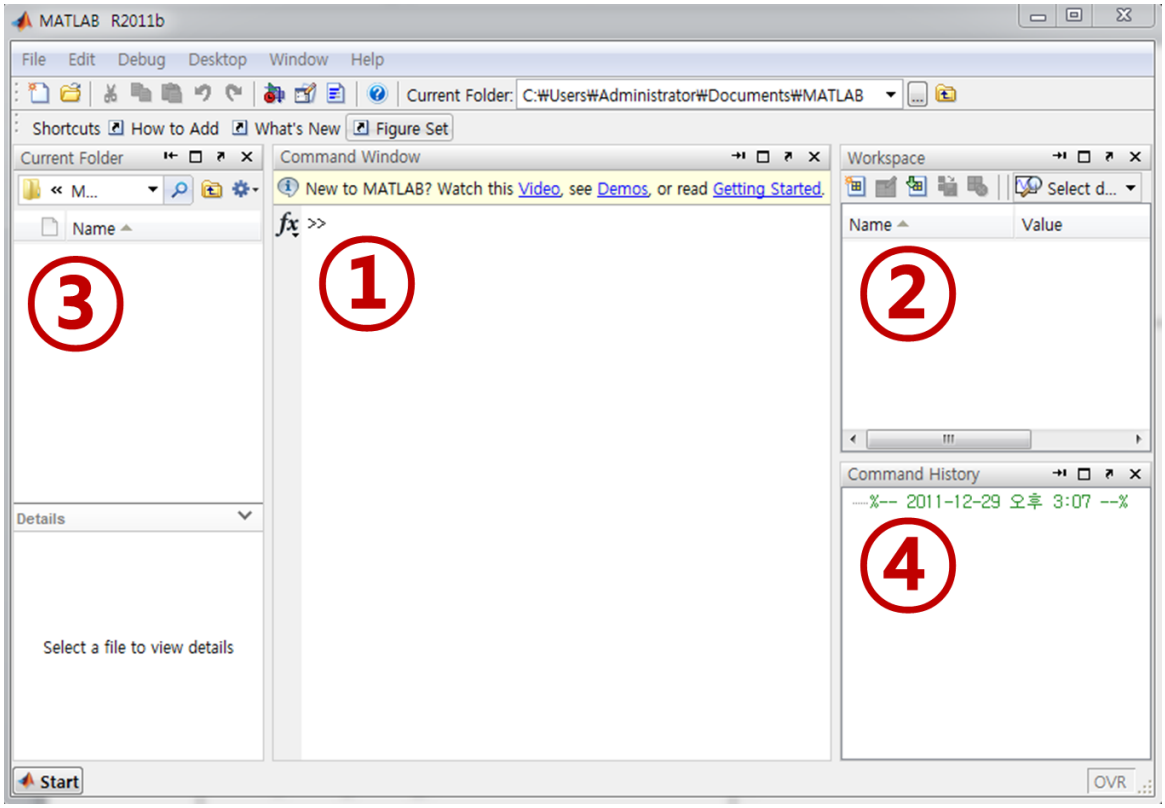
① 수학과 관련된 수치계산
② 알고리즘 개발
③ 시스템 모델링과 데이터 분석
④ 여러 가지 그래픽 표현
⑤ GUI를 기반으로 하는 어플리케이션 개발
⑥ 임베디드 시스템 개발

MATLAB의 기본 데이터 요소는 차원의 제한이 없는 배열이다. 기존의 프로그래밍 언어에서 행렬 연산을 하는 경우 데이터 구조를 따로 만들어주어야 했지만, MATLAB에서는 기본 변수 형태가 행렬이기 때문에 행렬을 다루기 쉬운 형태로 되어 있다. 이는 많은 양의 데이터를 일괄적으로 처리하는데 유리하다는 것을 뜻한다.

MATLAB은 수학적인 도구를 필요로 하는 각 전문 분야에 도움이 되고자 여러 툴박스(toolbox)를 제공한다. 신호 처리, 통계학, 영상 처리, 제어, 퍼지 논리, 금융, 화학 공정, CDMA, 하드웨어 인터페이스 등, 해당 분야에서 주요하게 쓰이는 기능을 미리 함수로 구성한 것이다. 특히, Simulink라는 MATLAB 내부에서 실행되는 프로그램은 동적 시스템의 시뮬레이션에 매우 적합하며, 자주 쓰인다. 이 외에 C/C++나 포트란 등 외부 프로그램과 연동에서 이용할 수 있는 기능도 제공한다.

본 자료는 <임종주의 MATLAB7>을 주로 참조하여 작성되었으며, 그 외 경험을 통해 얻은 내용도 다수 들어 있다. 중간에 막히는 부분이 있을 경우, MATLAB의 도움말 기능이나 도서, 인터넷 검색 등을 참조하는 것이 큰 도움이 될 것이다.

1.1. MATLAB 기본 창



1) Command Window

MATLAB에서 가장 기본이 되는 창으로, 명령어를 이곳에 입력하여 실행한다. 여기서 생성한 모든 데이터는 workspace에 저장된다.

2) Workspace

Command window에서 생성한 데이터가 저장되는 공간으로, 변수 이름과 내용에 대한 간략한 정보를 제공한다(size, min, max 등). 변수를 더블 클릭하면 스프레드 시트와 같은 방식으로 변수 내용을 확인할 수 있다.

3) Current Folder

현재 작업하고 있는 폴더를 보여주며, 폴더 내에 있는 m-file은 별도의 경로 지정 없이 command window에서 실행시킬 수 있다. M-file이란 MATLAB의 function과 같은 것이며, 자세한 사항은 뒤에서 다룰 것이다. 다른 폴더의 m-file을 실행하려면 current folder를 바꾸거나, File >> Set Path에 경로를 추가하여야 한다. 일반적으로 쓰는 MATLAB의 함수는 이미 set path에 경로 설정이 되어 있다.

4) Command History

Command window에서 실행한 명령어가 차례대로 기록된다. 날짜와 시간 별로 정리되어 있으며, 필요한 명령어를 더블 클릭하거나 command window로 드래그하여 동일한 명령어를 다시 실행할 수 있다. Command window에서 키보드 방향키를 통해서도 command history에 접근할 수 있다.

1.2. 명령어 실행

MATLAB 명령어의 기본 형태는 다음과 같다.

>> Variable = Expression

즉, 우변에 있는 expression의 실행 결과를 좌변의 변수에 저장하는 것이다. 만약 변수가 주어져

있지 않다면(명령어에 등호가 없다면), 실행결과는 자동으로 'ans'라는 이름의 변수에 저장된다. 다음 예제를 실행해보자.

예제1.1. 명령어 실행과 변수 생성
>> x = 1; >> x + 1 ans = 2 >> y = ans;
첫 번째 명령어는 x라는 이름의 변수에 1을 저장한다는 것이다. 이 때 변수 x는 다른 언어와 달리 미리 type과 변수명을 선언해주지 않아도 자동으로 double type으로 생성된다. 두 번째 명령어는 등호 없이 수식만 나타나 있는데, 이를 실행한 결과는 ans에 저장된다. Workspace의 변수 리스트를 보면 x와 ans에 각각 1과 2가 저장된 것을 알 수 있다. 마지막 명령어는 앞에서 계산한 결과인 ans를 y에 대입하는 것이다. 자동으로 저장된 ans도 변수처럼 사용할 수 있다. 위의 세 명령어의 끝부분을 살펴보면 x=1과 y=ans에 ';'가 추가된 것을 알 수 있다. 이는 명령어의 실행 결과를 command window에 바로 표시할 것인지를 결정하는 것으로, ';'가 없는 x+1은 실행 결과가 곧바로 아래에 나타난 것을 볼 수 있다.

1.3. 행렬 만들기

MATLAB의 변수는 기본적으로 double type의 값을 지닌 행렬이다. 즉, 예제1의 x와 y도 1X1 크기의 행렬로 본다. 또한, 1X3이나 5X1과 같이 행이나 열의 크기가 1인 행렬을 따로 벡터라 부르기도 한다. 그러나 생성 방법은 행렬과 동일하다. 행렬을 만드는 방법은 크게 3가지가 있는데, 다음과 같다.

① 직접 값을 입력하기
② m-file 함수를 실행하기
③ 외부 데이터(엑셀, 텍스트, Simulink 등) 불러오기

마지막의 외부 데이터로부터 불러오는 방법은 상황에 따라 매우 다양하므로 여기서 다루지 않으며, 다음 예제는 직접 값을 입력하는 방법과 함수를 이용하는 방법을 소개한다.

예제1.2. 직접 값을 입력하여 행렬 만들기	
>> A = [1 2 3] A = 1 2 3 >> B = [6:-1:4]	행렬의 값을 직접 입력할 때, 각 요소는 띄어쓰기로 열을 구분하며, 전체를 [ ]로 감싸주어야 한다. ':'을 두 번 사용하여 동일한 간격의 벡터를

B = 6 5 4	만들 수 있다. [6:-1:4]는 6부터 1씩 감소하여 4까지, 라는 의미이다. 다른 예로, [3:0.5:5]는 3, 3.5, 4, 4.5, 5의 5개의 요소를 만들어낸다.
>> C = [1 2; 3 4] C = 1 2 3 4	행을 구분할 때는 ';'를 사용한다. 이 때, [ ] 안의 모든 행의 크기가 동일해야 에러 없이 행렬을 만들 수 있다. 만약 [1 2; 3 4; 5]라고 쓴다면, 세 번째 행의 요소가 부족하여 행렬을 만들 수 없다.
>> X = [A; B] X = 1 2 3 6 5 4	[ ] 안에 미리 생성한 변수를 넣을 수 있다. 당연히 행과 열 방향의 크기에 문제가 없어야 한다.
>> Y = [X X] Y = 1 2 3 1 2 3 4 5 6 4 5 6	
>> Z = [C [A; B]] Z = 1 2 1 2 3 3 4 4 5 6	Z와 같이 [ ] 속에 또 다른 [ ]를 만들어도 크기에 문제가 없다면 행렬을 만들 수 있다. C의 크기는 2X2이고 A와 B는 1X3인데, A와 B를 묶어 2X3으로 만들고 이를 C에 붙인 형태이다.

예제1.3. 함수를 이용하여 행렬 만들기	
>> eye(3) ans = 1 0 0 0 1 0 0 0 1	eye, ones, zeros, rand 등과 같은 함수는 사용법이 거의 동일하다. eye(3)과 같이 인자를 하나만 쓰면 그 크기의 정사각행렬을 출력하며, zero(1, 5)와 ones(3, 2)와 같이 두 개의 인자를 쓰면 그 크기의 행렬을 출력한다.
>> zeros(1, 5) ans =	eye는 단위행렬을 출력하며 zeros는 영행렬, ones는 1로 이루어진 행렬, rand는 0에서 1 사이의 난수로 이루어진 행렬을 만든다.

0 0 0 0 0 >> ones(3, 2) ans = 1 1 1 1 1 1 >> rand(3) ans = 0.8147 0.9134 0.2785 0.9058 0.6324 0.5469 0.1270 0.0975 0.9575 >> linspace(0, 10, 5)' ans = 0 2.5000 5.0000 7.5000 10.0000 >> logspace(0, 1, 5)' ans = 1.0000 1.7783 3.1623 5.6234 10.0000	linspace(a, b, n)은 a에서 b까지 일정한 간격으로, 전체 요소의 수가 n개가 되도록 나누어주는 함수이다. 이와 비슷하게, logspace(a, b, n)은 지수가 a에서 b까지 n개로 나눈 후, 밑을 10으로 계산한 함수이다. 즉, 왼쪽의 예는 10^0, 10^0.25, 10^0.5, 10^0.75, 10^1을 계산한 결과이다. 왼쪽의 명령어에서 linspace와 logspace 끝을 보면 '가 적힌 것을 확인할 수 있다. 이는 전치행렬을 만드는 연산자로, 행과 열을 서로 바꾸어준다. 즉, 원래 linspace와 logspace의 출력은 1X5 크기이지만, '를 써서 5X1로 변환된 것이다. MATLAB을 사용하면서 굉장히 많이 사용되는 연산자이다.

1.4. 행렬 추출하기

행렬을 자유롭게 다루기 위해서는 내부 요소 중 필요한 값을 가져올 수 있어야 한다. 필요한 값의 의미는 상황에 따라 다양하지만, 크게 행렬의 어떤 열이나 행, 또는 더 작은 부분을 가져오는 것과 특정 조건을 만족하는 부분을 가져오는 것으로 나눌 수 있다.

예제1.4. 주어진 인덱스로 추출하기	
>> M = magic(4) M = 16 2 3 13 5 11 10 8 9 7 6 12 4 14 15 1	MATLAB은 행렬의 어떤 요소를 좌표와 같이 ','로 구분하여 쓴다. 예를 들면 세 번째 행의 두 번째 열을 추출 할 때, M(3, 2)와 같이 쓴다. 이 때 (3, 2)는 메모리 상의 위치를 나타내며, 인덱스(index)라고 부른다. 함수 magic은 주어진 크기의 마방진을 출력해준다. 여기서 M(3, 2)를 입력하면 세 번째 행의 두 번째 열의 값인 7을 얻을 수 있다.
>> M(3, 2) ans = 7	
>> M(end, 1) ans = 4	행렬의 크기에 상관없이 마지막 위치를 의미할 때, 인덱스에 end를 쓸 수 있다.
>> M(2, :) ans = 5 11 10 8	앞에서 벡터를 만들 때 쓰였던 ':'는 인덱스에서 전체를 의미한다. 즉, M(2, :)는 행렬에서 두 번째 행 전부를 추출한다는 것이다.
>> M(:, 4) ans = 13 8 12 1	네 번째 열 전부를 가져온다.
>> M(:, [1 3]) ans = 16 3 5 10 9 6 4 15	인덱스 안에 벡터를 만들면 그 숫자에 맞는 행 또는 열을 추출할 수 있다. M(:, [1 3])은 첫 번째와 세 번째 열 전체를 가져온다.

>> M(2:4, :) ans = 5 11 10 8 9 7 6 12 4 14 15 1 >> M([2 1], [4 1]) ans = 8 5 13 16	2:4는 [2 3 4]와 같으므로, 2, 3, 4 행 전체를 가져온다. 인덱스의 행과 열에 벡터를 응용하면 다양한 방식으로 행렬의 부분을 추출할 수 있다. ([2 1], [4 1])은 첫 번째와 두 번째 행의 네 번째와 첫 번째 열을 추출하여, 위치를 서로 바꾼 것이다.

예제1.5. 벡터에서 추출하기	
>> t = 0:0.1:1 t = 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 >> t(1) ans = 0 >> t(end) ans = 1 >> t([2:3:8]) ans = 0.1000 0.4000 0.7000 >> t = t' t = 0	1Xn이거나 nX1 형태의 벡터는 인덱스를 사용할 때 숫자 하나로 위치를 나타낼 수 있다. 편의상 1Xn을 행벡터, nX1을 열벡터라 부른다. 첫 번째 요소의 값이다. 마지막 요소의 값이다. [2:3:8]은 [2 5 8]을 나타내므로, 그 인덱스에 해당하는 값만 추출한다. 앞에서 언급한 전치행렬 연산자 '를 사용하면 행벡터를 열벡터로 바꿀 수 있다.

0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000	
>> [t(1) t(end)] ans = 0 1	벡터는 그 형태에 관계없이 모두 숫자 하나로 인덱스를 나타낸다.
>> M = magic(4) M = 16^{1} 2^{5} 3^{9} 13^{13} 5^{2} 11^{6} 10^{10} 8^{14} 9^{3} 7^{7} 6^{11} 12^{15} 4^{4} 14^{8} 15^{12} 1^{16} >> M([1 5 10]) ans = 16 2 10	행렬에서 인덱스를 숫자 하나로 나타낼 경우, 열 방향부터 센 위치를 의미한다. 즉, 4X4 행렬에서 1은 (1, 1), 5는 (1, 2), 10은 (2, 3)이다. 이 경우, 행렬의 크기가 바뀌면 동일한 숫자라도 가리키는 위치가 다를 수 있다. 5X5 크기의 행렬이라면, 5는 (5, 1)을 나타낼 것이다.

예제1.6. 행렬 지우기	
>> R = [] R = [] >> R = rand(4) R = 0.2769 0.6948 0.4387 0.1869	[ ]은 행렬의 값을 직접 입력할 때 쓰이는 문자이다. 만약 아무 내용 없이 [ ]만 쓸 경우, 이는 빈 행렬을 의미한다.

```
    0.0462    0.3171    0.3816    0.4898
    0.0971    0.9502    0.7655    0.4456
    0.8235    0.0344    0.7952    0.6463

>> R = R([1 3 4], :)

R =
    0.2769    0.6948    0.4387    0.1869
    0.0971    0.9502    0.7655    0.4456
    0.8235    0.0344    0.7952    0.6463

>> R(:, 3) = []

R =
    0.2769    0.6948    0.1869
    0.0971    0.9502    0.4456
    0.8235    0.0344    0.6463
```

앞에서 배운 벡터 인덱스를 통한 행렬 추출을 이용하면 왼쪽과 같이 두 번째 행만 지울 수 있다.

더 쉬운 방법으로, 왼쪽과 같이 세 번째 열에 빈 행렬을 대입하면 세 번째 열만 지울 수 있다.

예제1.7. 조건으로 추출하기

```
>> M = magic(4)

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

>> M > 10

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

>> M <= 4 | M > 10
```

행렬에 조건을 취하면, 참인 값은 1, 거짓인 값은 0으로 대체된다. 즉, 위치에 따라 독립적으로 실행되는 것이다.

M > 10이라고 하면, M에서 10보다 큰 값인 위치는 1, 이하는 0으로 바뀐다.

M <= 4와 M > 10 사이에 or 연산(|)을 하면 두 조건 중 하나라도 만족하는 위치가 1로

ans = 1 1 1 1 0 1 0 0 0 0 0 1 1 1 1 1	바뀐다.
>> L = M > 7 & M <= 10 L = 0^{1} 0^{5} 0^{9} 0^{13} 0^{2} 0^{6} 1^{10} 1^{14} 1^{3} 0^{7} 0^{11} 0^{15} 0^{4} 0^{8} 0^{12} 0^{16}	and 연산(&)을 사용하면 둘 다 만족하는 위치가 1이다.
>> m = find(L) m = 3 10 14	0과 1로 이루어진 행렬(논리행렬)을 find 함수에 넣으면 1인 위치의 인덱스가 출력된다. 이 인덱스는 앞에서 언급한 벡터 인덱스와 같이, 숫자 하나로 나타낸 방식으로 출력된다.
>> M(m) ans = 9 10 8	이 인덱스를 다시 원래의 행렬 인덱스로 지정하면 그 위치의 값이 출력된다.

1.5. 산술 연산자

MATLAB의 산술 연산자는 행렬을 기반으로 이루어져 있다. 따라서, '+' '-' '*' '/' '₩' '^' 와 같은 연산자는 모두 피연산자가 적합한 형태라는 조건을 필요로 한다. A□B를 가정하여 설명하면 다음과 같다.

① +와 -는 A와 B 행렬의 크기가 같아야 한다.
② *는 A의 열과 B의 행의 길이가 같아야 한다. 행렬 곱을 의미한다.
③ /는 B가 정사각행렬이고 역행렬을 구할 수 있어야 하며, A의 열의 길이가 B의 크기와 같아야 한다. A에 B의 역행렬을 뒤에서 곱하는 연산이다.
④ ₩는 3번과 반대로 A가 정사각행렬에 역행렬이 가능해야 하며, B의 행의 길이가 A의 크기와 같아야 한다. B에 A의 역행렬을 앞에서 곱하는 연산이다.

⑤ ^는 A가 정사각행렬로 자기 자신끼리 곱할 수 있어야 하며, B가 스칼라여야 한다.

그러나 이 연산자만으로는 변수 내의 값을 행렬로 보지 않고 독립된 데이터로 처리해야 할 때, '*' '/' '₩' '^'와 같이 곱하기 개념을 포함한 연산자에서 문제가 생길 수 있다. 예를 들어 A*B/C^D₩E 와 같은 식을 계산할 때, 행렬 사이의 연산이 아니라 '+' '-' 처럼 동일한 자리의 요소끼리 연산을 하고자 한다면 A.*B./C.^D.₩E 와 같이 '.'을 붙인 연산자를 사용해야 한다.

⑥ .* ./ .₩ .^와 같이 점 .이 붙은 연산자는 A와 B 행렬의 크기가 같아야 한다.

연산 중 변수가 행렬이나 데이터 집합이 아니라 상수로 취급해야 할 경우도 있다. 예를 들어 3*A의 경우, 3을 행렬로 생각하면 1X1이므로 A가 1Xn의 행벡터가 아니라면 연산이 불가능하지만, 이 명령어의 의도는 A의 모든 요소에 3을 곱하라는 것이다. 따라서 A의 각 요소에 3배한 결과를 출력한다. 이와 같이 MATLAB의 모든 연산자는 A와 B 중 하나가 스칼라라면 이를 상수로 취급하여 연산한다. 이를 따로 스칼라 확장(scalar expansion)이라 부르기도 한다.

⑦ A, B 중 하나가 스칼라일 경우 이를 상수로 가정하여 연산한다. 이를 스칼라 확장이라 부른다.

예제1.8. 산술 연산자	
>> A = pascal(3) A = 1 1 1 1 2 3 1 3 6 >> B = magic(3) B = 8 1 6 3 5 7 4 9 2 >> X = A + B X = 9 2 7 4 7 10 5 12 8	함수 pascal은 파스칼의 삼각형을 만들어준다. 두 행렬의 크기가 같으므로 덧셈에 문제가 없다.

>> C = ones(2, 3) C = 1 1 1 1 1 1	
>> Y = A + C Error using + Matrix dimensions must agree.	C는 2X3 크기이므로, 3X3인 A와 크기가 맞지 않아 에러를 출력한다.
>> Y = A + 1 Y = 2 2 2 2 3 4 2 4 7	피연산자 중 하나가 스칼라일 경우, 스칼라 확장이 일어나 모든 요소에 적용된다.
>> Z = A*C Error using * Inner matrix dimensions must agree.	3X3 행렬과 2X3 행렬의 곱이므로 크기가 맞지 않아 에러를 출력한다.
>> Z = C*A Z = 3 6 10 3 6 10	2X3 행렬과 3X3 행렬은 행렬 곱이 가능하다.
>> A^2 ans = 3 6 10 6 14 25 10 25 46	A에 ^을 사용하면 행렬을 거듭제곱한다. 이 때, A는 정사각행렬이어야 한다.
>> A.^2 ans = 1 1 1 1 4 9	.^는 요소 스스로의 제곱을 구한다. 즉, A^2는 A*A를 의미하지만, A.^2는 A의 각 요소를 제곱한다.

1 9 36	

1.6. 수학적 상수와 다양한 함수

MATLAB에는 수학 연산에서 쓰이는 특수한 상수와 여러 가지 함수가 정의되어 있다. 그 중에서 자주 쓰이는 값과 함수를 소개한다.

MATLAB 명령	설명
pi	원주율 π
i 또는 j	복소수 (-1)^0.5
sqrt( )	제곱근
exp( )	자연상수 e의 지수함수
log( ) 와 log10( )	자연상수 e의 로그함수와 상용로그
sin( ) cos( ) tan( ) 등	삼각함수
asin( ) acos( ) atan( ) 등	역삼각함수
abs( )	절대값
length( ) size( )	벡터와 행렬의 크기
max( ) min( ) mean( )	최대, 최소, 평균
sum( ) prod( )	합, 곱
poly( ) roots( )	다항식의 계수와 근

예제1.9. 자주 쓰는 기능	
>> pi ans = 3.1416	원주율이 미리 정의되어 있다.
>> cos(pi) ans = -1	cos π =-1 이다. 삼각함수는 rad 단위로 입력한다.
>> i^2 ans = -1	허수인 -1의 제곱근이 정의되어 있다.
>> (1 + i) + (3 - 4*i)	i를 활용한 복소수 사이의 연산도 가능하다.

ans = 4.0000 - 3.0000i >> abs(4-3*i)	절대값을 취하면 음수, 복소수 관계없이 양수로 변환해준다.
ans = 5 >> exp(1)	자연상수 e의 지수함수가 정의되어 있다.
ans = 2.7183 >> log(1)	자연상수 e의 로그함수가 정의되어 있다.
ans = 0 >> atan(1)*4	역삼각함수의 출력은 rad 단위이다.
ans = 3.1416 >> M = magic(4) M = 16 2 3 13 5 11 10 8 9 7 6 12 4 14 15 1 >> L = M(:, [2 3]) L = 2 3 11 10 7 6 14 15	

>> size(M) ans = 4 4 >> size(L) ans = 4 2	함수 size는 행과 열의 크기를 출력한다.
>> length(M) ans = 4 >> length(L) ans = 4	함수 length는 행과 열의 크기 중 큰 값을 출력한다.
>> max(M) ans = 16 14 15 13	함수 max는 각 열의 최대값을 행벡터로 출력한다.
>> max(max(M)) ans = 16	행렬의 최대값을 구하려면 max를 두 번 쓰면 된다.
>> sum(L) ans = 34 34	함수 sum도 각 열 방향의 모든 요소를 더하여 행벡터로 출력한다.
>> sum(L, 2) ans = 5	함수 sum의 두 번째 인자로 2를 넣어주면 행 방향으로 더하여 열 벡터로 출력한다. 이러한 종류의 함수는 인자에 DIM(차원) 값을 받아 계산할 방향을 정해주는 경우가 많다. Default

21 13 29 >> f = [1 0 -7 6] f = 1 0 -7 6 >> r = roots(f) r = -3.0000 2.0000 1.0000 >> ff = poly(r) ff = 1.0000 0.0000 -7.0000 6.0000	값은 1로, 열 방향 연산을 나타낸다. 함수 roots와 poly는 다항식의 계수와 근에 대한 기능을 제공한다. 함수 roots는 주어진 벡터를 다항식의 계수로 보고 그 근을 계산한다. 반대로 poly는 벡터를 근으로 보고 다항식의 계수를 구해준다. 왼쪽에서 f는 $x^3 - 7x + 6$과 같은 의미로 쓰인 것이다. 이를 roots에 입력하면 -3, 2, 1이 나오는데, 세 수가 삼차식 f의 근에 해당한다. 구한 근을 다시 poly에 입력하면 원래 다항식의 계수 1 0 -7 6을 구할 수 있다.

1.7. 도움말 기능

MATLAB에는 지금까지 소개한 것 외에도 다양한 함수가 있으나, 지면에 모두 담기에는 무리가 따른다. 따라서 필요한 기능을 찾는 방법에 대하여 소개한다. MATLAB은 도움말 기능이 매우 뛰어나 원하는 함수를 쉽게 찾을 수 있다.

① 함수의 기능이 궁금할 경우, >> help search_function 을 사용한다.
② 검색어를 통해 원하는 기능을 찾고 싶을 경우, >> lookfor search_function 을 사용한다.
③ command window보다 자세한 설명을 원할 경우, >> doc search_function 을 사용한다.
④ 아무것도 모르겠으면 >> doc matlab 부터 입력한다.

MATLAB에서 사용자가 직접 만든 m-file을 제외한 모든 기능은 문서화가 잘 되어 있으므로, 인터넷이나 책을 찾는 것보다 MATLAB의 doc를 검색하는 것이 훨씬 편하며, 무엇보다 정확하다. 이 이후로 간단한 기능의 함수는 자세한 설명 없이 진행할 텐데, 반드시 도움말 기능으로 내용을 확인하기 바란다.

예제1.10. 도움말 기능	
>> lookfor arithmetic	
integerMath	- Integer Arithmetic
arith	- metic operators.

arithdeco - Decode binary code using arithmetic decoding.
arithenco - Encode a sequence of symbols using arithmetic coding.
tocsourcecoding - Differential, arithmetic, Huffman, DPCM, quantization
fncmb - Arithmetic with function(s).
bitsra - Shift Right Arithmetic.
fiarithmeticdemo - Fixed-Point Arithmetic
leftshift - Arithmetic left-shift.
rightshift - Arithmetic right-shift.
vpa - Variable precision arithmetic.
symvpademo - Variable Precision Arithmetic

>> help arith
Arithmetic operators.

+ Plus.
X + Y adds matrices X and Y. X and Y must have the same
dimensions unless one is a scalar (a 1-by-1 matrix).
A scalar can be added to anything.

- Minus.
X - Y subtracts matrix X from Y. X and Y must have the same
dimensions unless one is a scalar. A scalar can be subtracted
from anything.

* Matrix multiplication.
X*Y is the matrix product of X and Y. Any scalar (a 1-by-1 matrix)
may multiply anything. Otherwise, the number of columns of X must
equal the number of rows of Y.

.* Array multiplication
X.*Y denotes element-by-element multiplication. X and Y
must have the same dimensions unless one is a scalar.
A scalar can be multiplied into anything.

^ Matrix power.
Z = X^y is X to the y power if y is a scalar and X is square. If y is an
integer greater than one, the power is computed by repeated
multiplication. For other values of y the calculation
involves eigenvalues and eigenvectors.
Z = x^Y is x to the Y power, if Y is a square matrix and x is a scalar,

computed using eigenvalues and eigenvectors.
Z = X^Y, where both X and Y are matrices, is an error.

.^ Array power.
Z = X.^Y denotes element-by-element powers. X and Y must have the same dimensions unless one is a scalar. A scalar can operate into anything.

앞에서 설명한 산술 연산자에 대하여 자세히 소개하고 있다.

1.8. 연습문제

① x = 3이고 y = 4 일 때, 다음 수식의 값을 구하여라.

$$\left(1-\frac{1}{x^3}\right)^{-1}, \quad 5\pi x^3, \quad \frac{4(y-5)}{3x-6}, \quad \frac{x^5}{x^5-1}, \quad e^x - e^y$$

```
>> x = 3;
>> y = 4;
>> (1-1/x^3)^-1

ans =
    1.0385

>> 5*pi*x^3

ans =
  424.1150

>> 4*(y-5)/(3*x-6)

ans =
   -1.3333

>> x^5/(x^5-1)

ans =
    1.0041

>> exp(x)-exp(y)

ans =
```

```
-34.5126
```

② 다음과 같이 회로의 각 저항에 걸린 전압을 측정한 데이터 표가 주어졌다. 각 저항에 흐르는 전류의 크기를 v=iR에 의하여 계산하라.

	1	2	3	4	5
저항	10^4	$2X10^4$	$3.5X10^4$	10^5	$2X10^5$
전압	120	80	110	200	350

```
>> R = [10^4   2*10^4   3.5*10^4   10^5   2*10^5];
>> v = [120   80   110   200   350];
>> i = v./R

i =
    0.0120      0.0040      0.0031      0.0020      0.0018
```

③ $f(x) = 3x^2 + 2x + 1$ 라는 이차함수에서, x = 5, 7, 9일 때의 함수 값 f(x)를 구하라. 단, MATLAB 함수 polyval을 사용한다.

```
>> help polyval
 polyval Evaluate polynomial.
    Y = polyval(P,X) returns the value of a polynomial P evaluated at X. P
    is a vector of length N+1 whose elements are the coefficients of the
    polynomial in descending powers.

        Y = P(1)*X^N + P(2)*X^(N-1) + ... + P(N)*X + P(N+1)

    If X is a matrix or vector, the polynomial is evaluated at all
    points in X.   See POLYVALM for evaluation in a matrix sense.

    [Y,DELTA] = polyval(P,X,S) uses the optional output structure S created
    by POLYFIT to generate prediction error estimates DELTA.   DELTA is an
    estimate of the standard deviation of the error in predicting a future
    observation at X by P(X).

    If the coefficients in P are least squares estimates computed by
    POLYFIT, and the errors in the data input to POLYFIT are independent,
    normal, with constant variance, then Y +/- DELTA will contain at least
    50% of future observations at X.
```

```
    Y = polyval(P,X,[],MU) or [Y,DELTA] = polyval(P,X,S,MU) uses XHAT =
    (X-MU(1))/MU(2) in place of X. The centering and scaling parameters MU
    are optional output computed by POLYFIT.

>> f = [3 2 1];
>> x = [5 7 9];
>> y = polyval(f, x)

y =

    86    162    262
```