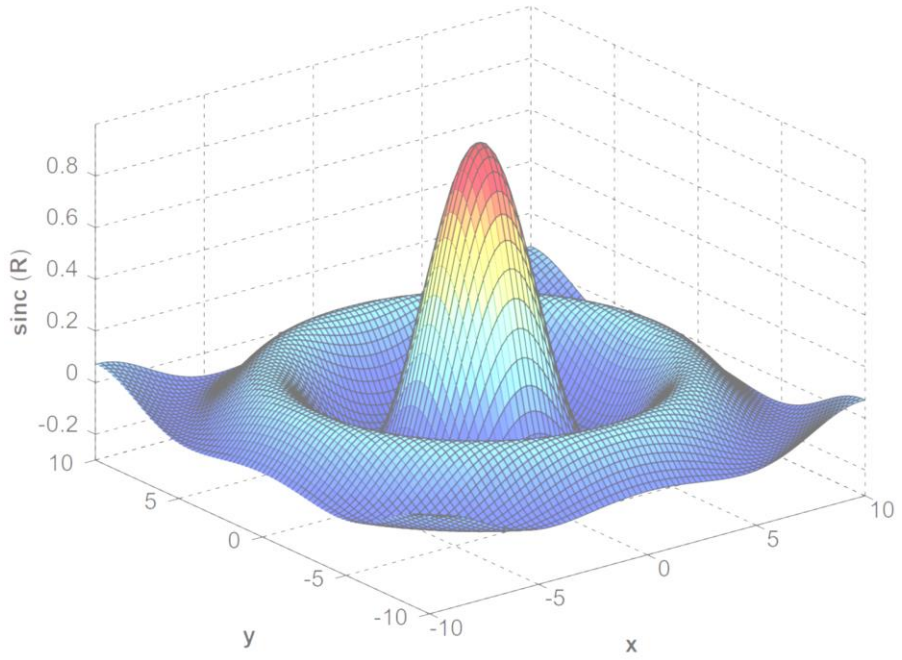
응용전산및실습 I

(강의자료 #4)



교과목명 : 응용전산 및 실습 I

담당교수 : 이 수 형

E-mail : soohyong@uu.ac.kr

교재명 : 유인물

수강생 안내

- 본 교과목은 기본적으로 “대면”수업입니다.
- ‘코로나바이러스감염증-19’ 등의 이유로 대면수업을 받지 못하는 학생들을 위해서 ‘비대면 실시간 수업’을 병행합니다. 기본적으로 대면 수업이므로 대면 수업에 참석한 학생들은 교실에서 출석을 체크하면 되며, 참석하지 못하고 ‘비대면 수업’을 듣는 학생들은 ‘실시간 온라인 강의’에 참석하면 출석이 반영됩니다.
- 수강생 여러분들은 강의 자료를 온라인 배포 및 게재 등의 행위를 할 시 저작권 문제가 발생할 수 있사오니 이에 유념하여 강의 자료를 공유하는 행위는 삼가 바랍니다.
- 교재는 따로 없으며, 유인물로 수업을 진행하니 각자 다운로드하여 보기 바랍니다.

지난시간에는?

Matlab기초

- Matlab이란?
  - Matlab은 Matworks사의 테크니컬 컴퓨팅 (공학계산용) 언어
  - MATLAB = MATrix + LABoratory : 행렬연산을 손쉽게
  - Matlab의 대안 : Matlab과 호환성이 뛰어나면서 무려인 Octave

- Matlab의 기초
  - 명령창(command windows)에서 명령어 입력 및 결과 확인
  - 스크립트 M-file : 수행할 명령어를 모아놓은 파일 (프로그램 소스)
  - 자주 사용하는 명령어 : clc (clear console), help, cd (change directory), pwd (print working directory)

사칙연산

- 사칙연산자

Operation	Symbol	Example
Addition, a + b	+	3 + 22
Subtraction, a – b	-	90 – 54
Multiplication, a • b	*	3.14 * 0.85
Division, a ÷ b	/ or \	56/8 = 8\56
Exponentiation, a^b	^	2^8

- 일반적인 프로그래밍 언어와 유사
- 나누기 연산자
  - / : right division (오른쪽 나눗셈) → 일반적인 실수에서는 동일한 결과, 행렬 연산에서 차이
  - \ : left division (왼쪽 나눗셈)

변수

- 변수(variable)란?
  - 변수 : 하나의 이름을 가지는 기억장소, '=' 기호를 사용하여 값을 저장 (`a = 5`)
  - 기억된 변수는 그 자체로 기억된 숫자로서 수식, 함수 등에 바로 사용
  - 숫자 하나를 저장할 수도 있고, 벡터 또는 배열을 저장할 수도 있음
- 변수 이름 규칙
  - 대소문자 구별 (Cost, cost, CoST, COST 는 서로 다른 변수)
  - 최대 31글자까지 허용. 단, 최근 Matlab과 Octave에서는 63글자까지 허용
  - 변수 이름은 문자, 숫자, 밑줄(_)이 사용된 하나의 단어로 구성되며, 문자로 시작
  - 키워드는 변수로 사용할 수 없으며, 함수 이름은 변수로 사용하지 않도록 주의.
- 미리 초기화된 변수
  - ans, pi (π), I & j ($\sqrt{-1}$), eps, inf, realmin, realmax

복소수 (Complex Number)

- 복소수 사용법

Declaration
a ± bi, a ± bj

Related Function	Description
abs(x)	Magnitude of x (= a ± bi) = $\sqrt{a^2 + b^2}$
angle(x)	Angle in radians of x (= a ± bi) = $\tan^{-1}(b/a)$
real(x)	Real part of x (= a ± bi) = a
imag(x)	Imaginary part of x (= a ± bi) = b
conj(x)	Complex conjugate of x (= a + bi) = a - bi

```
>> a = sqrt(3) + 1j
a =  1.7321 + 1.0000i

>> m = abs(a)
m =  2.0000

>> theta = angle(a)
theta =  0.52360

>> degree = theta * 180 / pi
degree =  30.000

>> r = real(a)
r =  1.7321

>> im = imag(a)
im =  1
```

수학 함수

Exponential Function	Description
exp(x)	Exponential
log(x)	Natural logarithm
log10(x)	Base 10 logarithm
log2(x)	Base 2 logarithm and floating-point dissection
pow2(x)	2^x
sqrt(x)	Square root

Trigonometric Function	Description
sin(x)	Sine
cos(x)	Cosine
tan(x)	Tangent
csc(x)	Cosecant of x = 1 / sin(x)
sec(x)	Secant of x = 1 / cos(x)
cot(x)	Cotangent of x = 1 / tan(x)
asin(x)	Inverse sine of x = arcsin(x) = $\sin^{-1}(x)$
acos(x)	Inverse cosine of x = arccos(x) = $\cos^{-1}(x)$
atan(x)	Inverse tangent of x = arctan(x) = $\tan^{-1}(x)$

M 파일의 작성

- M파일 : Matlab에서 사용하는 프로그램 저장용 파일
- 스크립트 파일
  - 명령어들을 미리 모아놓은 파일
  - 명령창에서 파일 이름 지정 → 저장된 명령어들을 순차적으로 실행

Keyword / Function	Meaning	Example
;	Suppress output	radius = 5;
%	Comment	radius = 5; % 반지름 값
disp(variable)	Display results without identifying variable names	disp(radius)
input	Prompt user for input	radius = input('반지름을 입력하세요> ');
pause	Pause until user presses any key	pause
fprintf(format, var1, var2,,,)	Print msg	fprintf('계산이 완료되었습니다.'); fprintf('반지름은 %f 입니다.\n', radius);

행렬의 정의

- 행렬의 정의

Keyword	Meaning
[ ]	배열의 시작과 끝
;	행 구분
,	열 구분

```
>> A = [1, 2; 3, 5]
A =
    1    2
    3    5
```

- 벡터 : 1차원 행렬, 행벡터/열벡터

```
>> y = [1, 3, 2]
y =
    1    3    2
```

행렬

- 행렬의 계산

Operation	Symbol	Example
Addition, a + b	+	A + B
Subtraction, a – b	-	A – B
Multiplication, a · b	*	A * B
Division, a ÷ b	/ or inv()	A / B = A * inv(B)
Exponentiation, a^b	^	A ^ 2
Transposition, a^T	’	A’

```
>> A = [1,2; 3,4]
A =
     1     2
     3     4

>> B = [5,6; 7,8]
B =
     5     6
     7     8

>> C = A + B
C =
     6     8
    10    12
```

```
>> D = A - B
D =
    -4    -4
    -4    -4

>> H = A * B
H =
    19    22
    43    50

>> G = A ^ 2
G =
     7    10
    15    22
```

```
>> E = A / B
E =
    3.0000   -2.0000
    2.0000   -1.0000

>> F = A * inv(B)
F =
    3.0000   -2.0000
    2.0000   -1.0000

>> J = A - 3
J =
    -2    -1
     0     1
```

행렬연산의 응용

- 선형방정식의 해 구하기

$$x_1 - x_2 + 2x_3 = 10$$
$$3x_1 + 2x_2 + 9x_3 = 9$$
$$x_2 - 4x_3 = -3$$

$$\begin{bmatrix} 1 & -1 & 2 \\ 3 & 2 & 9 \\ 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10 \\ 9 \\ -3 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 2 \\ 3 & 2 & 9 \\ 0 & 1 & -4 \end{bmatrix}^{-1} \begin{bmatrix} 10 \\ 9 \\ -3 \end{bmatrix}$$

```
>> A = [1, -1, 2; 3, 2, 9; 0, 1, -4]
A =
     1    -1     2
     3     2     9
     0     1    -4

>> B = [10; 9; -3]
B =
    10
     9
    -3

>> x = inv(A) * B
x =
    6.4783
   -4.0435
   -0.2609
```

행렬연산자

- 스칼라와 행렬의 연산

Element-By-Element Operation	Representative Data $a = [a_1, a_2, \cdots, a_n], b = [b_1, b_2, \cdots, b_n], c =< a\ scalar >$
Scalar Addition	$a + c = [a_1 + c, a_2 + c, \cdots, a_n + c]$
Scalar Multiplication	$a * c = [a_1 * c, a_2 * c, \cdots, a_n * c]$
Array Addition	$a + b = [a_1 + b_1, a_2 + b_2, \cdots, a_n + b_n]$
Array Multiplication	$a .* b = [a_1 * b_1, a_2 * b_2, \cdots, a_n * b_n]$
Array Division	$a ./ b = \left[\frac{a_1}{b_1}, \frac{a_2}{b_2}, \cdots, \frac{a_n}{b_n}\right]$
Array Exponentiation	$a .\hat{}\, c = [a_1^c, a_2^c, \cdots, a_n^c]$
	$\mathrm{c} .\hat{}\, a = [c^{a_1}, c^{a_2}, \cdots, c^{a_n}]$
	$a .\hat{}\, b = \left[a_1^{b_1}, a_2^{b_2}, \cdots, a_n^{b_n}\right]$

특수 행렬 만들기

- 자주 사용되는 특수 행렬의 생성

Keyword	Description
ones(n) ones(m, n)	Create n x n arrays containing all ones Create m x n arrays containing all ones
zeros(n) zeros(m, n)	Create n x n arrays containing all zeros Create m x n arrays containing all zeros
eye(n, n) eye(m, n)	Produce n x n identity matrices Produce m x n identity matrices
rand(n) rand(m, n)	Uniformly distributed n x n random arrays Uniformly distributed m x n random arrays
randn(n) randn(m, n)	Zero-mean, unit-variance normal distribution n x n, m x n random arrays
size(A)	Return row & column size

특수 행렬

- 특수 행렬 만들기 연습

```
>> ones(3)
ans =
   1   1   1
   1   1   1
   1   1   1

>> zeros(2)
ans =
   0   0
   0   0

>> zeros(2,5)
ans =
   0   0   0   0   0
   0   0   0   0   0

>>
```

```
>> eye(2)
ans =

Diagonal Matrix

   1   0
   0   1

>> eye(2,4)
ans =

Diagonal Matrix

   1   0   0   0
   0   1   0   0

>>
```

```
>> rand(2)
ans =
   0.22906   0.12572
   0.84617   0.42582

>> randn(2,4)
ans =
  -1.25628   0.55138   0.91693  -1.39879
  -0.49975   1.60643   1.50290  -1.12196

>> A=[1,2,3; 4,5,6]
A =
   1   2   3
   4   5   6

>> size(A)
ans =
   2   3
```

행렬(Matrix)의 사용 #2

행렬 만들기

- 콜론(:) 연산자를 사용하여 반복문의 형태로 행렬을 만드는 방법
  - 시작값, 증가값, 마지막값을 이용하여 한번에 배열을 지정

Array Construction Technique	Description
x = [2, 2*pi, sqrt(5), 2-3j]	Create row vector x containing element specified
x = *first* : *last*	Create row vector x starting with *first*, counting by one, ending at or before *last*
x = *first* : *increment* : *last*	Create row vector x starting with *first*, counting by *increment*, ending at or before *last*
x = linspace(*first*, *last*, *n*)	Create row vector x starting with *first*, ending at *last*, having *n* elements
x = logspace(*first*, *last*, *n*)	Create logarithmically-spaced row vector x starting with 10^{first}, ending at 10^{last}, having *n* elements

행렬 만들기

- 콜론 연산자를 이용한 행렬 만들기

```
>> c = [1, 6, 9, 7]
c =
     1     6     9     7

>> a = 1 : 5
a =
     1     2     3     4     5

>> b = 1 : 2 : 7
b =
     1     3     5     7

>> x = (0 : 0.1 : 0.3) * pi
x =
     0     0.3142     0.6238     0.9425
```

```
>> d = [ 1:2:5, 1, 0, 1 ]
d =
     1     3     5     1     0     1

>> a = 1 : 3,   b = 1 : 2 : 7
a =
     1     2     3
b =
     1     3     5     7

>> c = [ b, a ]
c =
     1     3     5     7     1     2     3
```

행렬의 주소 지정

- 콜론(:) 연산자 = 벡터

```
>> A=[1,2,3,4,5; 2,3,5,4,2; 2,4,6,8,10; 3,2,8,4,5]
A =
     1     2     3     4     5
     2     3     5     4     2
     2     4     6     8    10
     3     2     8     4     5

>> C = A(2:4, :)
C =
     2     3     5     4     2
     2     4     6     8    10
     3     2     8     4     5

>> C = A([2,3,4],:)
C =
     2     3     5     4     2
     2     4     6     8    10
     3     2     8     4     5
```

행렬 만들기

```
>> x = linspace(2, 6, 3)
x =
     2     4     6

>> y = logspace(0, 2, 11)
y =
    1.0000    1.5849    2.5119    3.9811    6.3096   10.0000   15.8489   25.1189
39.8107   63.0957  100.0000
```

x = logspace(*first*, *last*, *n*)	Create logarithmically-spaced row vector x starting with 10^{first}, ending at 10^{last}, having *n* elements

```
>> z = linspace(0, 2, 11)
z =
   0.00000   0.20000   0.40000   0.60000   0.80000   1.00000   1.20000   1.40000   1.60000
1.80000   2.00000

>> yy = 10.^(z)
yy =
    1.0000    1.5849    2.5119    3.9811    6.3096   10.0000   15.8489   25.1189
39.8107   63.0957  100.0000
```

행렬의 주소 지정

- 주소지정 방법

Array Addressing	Description
A(r, c)	Indicates the elements by the r'th row and the c'th column
A(r, :)	Addresses a subarray within A defined by the index vector of desired rows in r and all columns
A(: , c)	Addresses a subarray within A defined by all rows and the index vector of desired columns in c
A(a : b , c : d)	Addresses a subarray within A intersected by the rows from a to b and the columns from c to d

행렬의 주소 지정

- 벡터 또는 행렬의 원소의 지정
  - 벡터 : (k) 형식으로 지정
  - 행렬 : (r, c)의 형식으로 지정
    - ➢ 범위를 벗어나면 오류 또는 크기 변경

```
>> A=[1,2,3; 4,5,6]
A =
   1   2   3
   4   5   6

>> B=[35, 23, 34, 62, 12]
B =
   35   23   34   62   12

>> B(3)
ans =  34
>> B(4) = 13
B =
   35   23   34   13   12
```

```
>> A(1,1)
ans =  1
>> A(1,2) = 3
A =
   1   3   3
   4   5   6

>> A(3,1) = 1
A =
   1   3   3
   4   5   6
   1   0   0

>> B(7) = 10
B =
   35   23   34   13   12    0   10

>> B(-1) = 12
error: B(-1): subscripts must be either integers 1
to (2^63)-1 or logicals
>>
```

행렬의 주소 지정

- 콜론(:)을 사용한 범위지정
  - 벡터 B(*n*:*m*) ⇒ n번째 원소에서 m번째 원소까지
    B(:) ⇒ 모든 원소
  - 행렬 A(*n*:*m*, *p*:*q*) ⇒ n번째 행에서 m번째 행과 p번째 행에서 q번째 행까지
    A(k, :) ⇒ k번째 행 전체

```
>> B=[1, 2, 3, 4, 5, 6, 7];
>> B(2:3)
ans =
   2   3

>> B(1:8)
error: B(8): out of bound 7

>> B(3:5)=[9, 8, 10]
B =
    1    2    9    8   10    6    7
```

행렬의 주소 지정

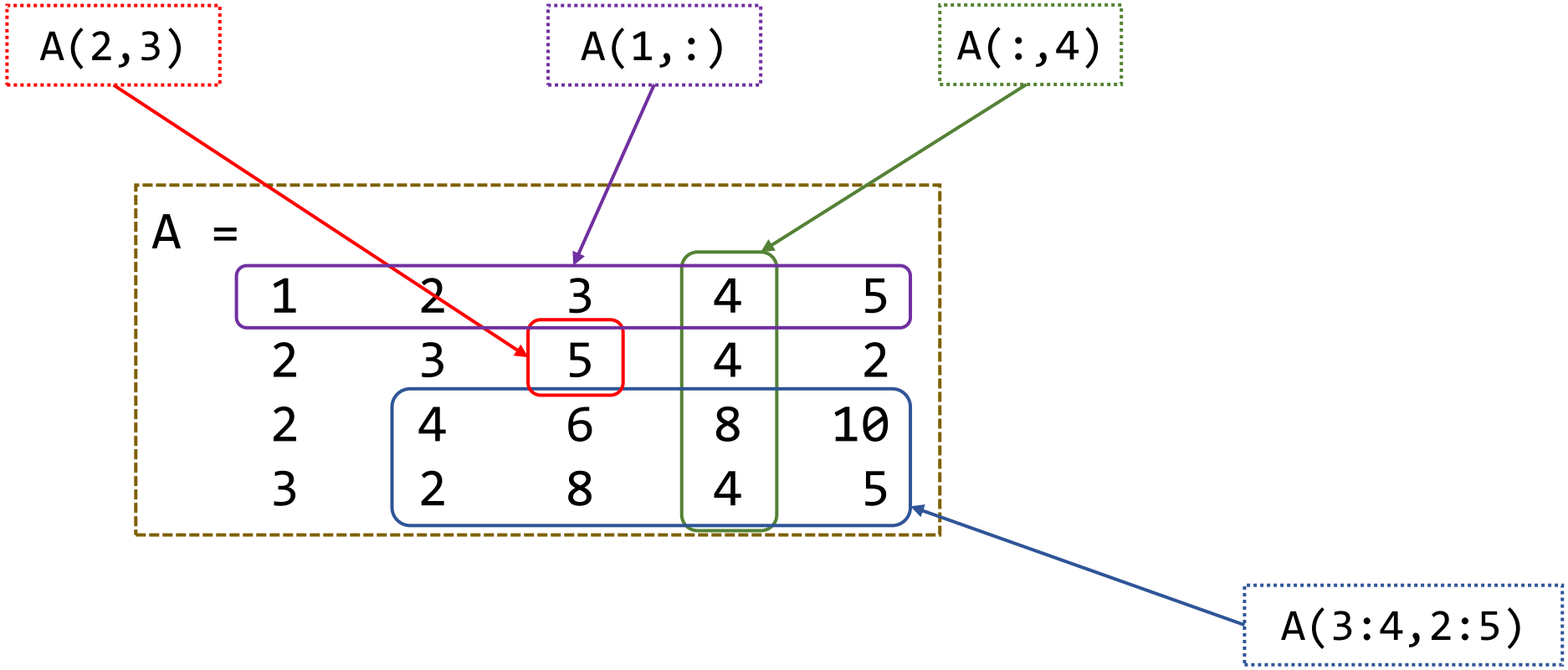
- 행렬의 범위 지정

```
>> A=[1,2,3,4,5; 2,3,5,4,2; 2,4,6,8,10; 3,2,8,4,5]
A =

     1     2     3     4     5
     2     3     5     4     2
     2     4     6     8    10
     3     2     8     4     5

>> C = A(:,3)
C =

     3
     5
     6
     8
```

```
>> C = A(1,:)
C =

     1     2     3     4     5

>> C = A(2:4,:)
C =

     2     3     5     4     2
     2     4     6     8    10
     3     2     8     4     5

>> C = A(3:4,1:3)
C =

     2     4     6
     3     2     8
```

행렬의 주소지정

- 콜론(:)을 사용한 행렬의 주소



행렬의 원소 추가

- 행렬 크기를 벗어난 주소 지정 / 행렬에 다른 행렬 포함

```
>> B(4) = 4
B =

   1   2   3   4

>> B(5:7) = [10, 20, 30]
B =

    1    2    3    4   10   20   30

>> B(10) = 100
B =

     1     2     3     4    10    20    30     0     0   100

>>
```

```
>> A = [1, 2; 3 4]
A =
   1   2
   3   4

>> A(3, 1) = 10
A =
    1    2
    3    4
   10    0

>> A(:, 3) = [5; 10; 15]
A =
    1    2    5
    3    4   10
   10    0   15
```

행렬의 원소 추가

- 행렬 크기를 벗어난 주소 지정 / 행렬에 다른 행렬 포함

```
>> A = [1, 2; 3 4]
A =

   1   2
   3   4

>> B = [5, 6; 7, 8]
B =

   5   6
   7   8

>> C = [A B]
C =

   1   2   5   6
   3   4   7   8
```

```
>> D = [A, B; B, A]
D =
   1   2   5   6
   3   4   7   8
   5   6   1   2
   7   8   3   4

>> E = [A, B; B, A]
E =
   1   2   5   6
   3   4   7   8
   5   6   1   2
   7   8   3   4

>> F = [A, B; ones(2, 4)]
F =
   1   2   5   6
   3   4   7   8
   1   1   1   1
   1   1   1   1
```

행렬의 원소 삭제

- 범위 지정 후 [] 이용

```
>> B = [1 2 3 4 5 6 7]
B =

   1   2   3   4   5   6   7

>> B(3) = []
B =

   1   2   4   5   6   7

>> B(2:4) = []
B =

   1   6   7

>>
```

```
>> A = [1 2 3 4; 5 6 7 8; 9 10 11 12]
A =
    1    2    3    4
    5    6    7    8
    9   10   11   12

>> A(:, 2) = []
A =
    1    3    4
    5    7    8
    9   11   12

>> A(3, :) = []
A =
   1   3   4
   5   7   8
```

연습

- ones() 함수와 zeros()함수를 이용하여 다음 행렬을 만들어라.
  - 처음 두 행은 0이고 다음 두 행은 1인 4x5 행렬을 만들어라.

```
A =
     0     0     0     0     0
     0     0     0     0     0
     1     1     1     1     1
     1     1     1     1     1
```

  - 6x6 행령에서 가운데 두 행과 가운데 두 열의 원소가 1이고, 나머지 원소는 0인 행렬을 만들어라.

```
B =
     0     0     1     1     0     0
     0     0     1     1     0     0
     1     1     1     1     1     1
     1     1     1     1     1     1
     0     0     1     1     0     0
     0     0     1     1     0     0
```

실습문제

- 강의중에 Octave를 이용하여 연습했던 내용들을 따라하고, 수행한 화면들을 캡처해서 제출하시오.
- [연습] 으로 수행했던 내용을 수행하고 결과를 제출하시오.

행렬(Matrix)의 사용 #3

복소 행렬

- 복소수로 이루어진 행렬의 전치 vs. 복소 공액 전치

```
>> a = 1 : 3
a =
     1     2     3

>> d = a + i * a
d =
     1.0000 + 1.0000i     2.0000 + 2.0000i     3.0000 + 3.0000i

>> e = d’  % Complex conjugate transpose of d
e =
     1.0000 - 1.0000i
     2.0000 - 2.0000i
     3.0000 - 3.0000i

>> f = d.’ % Dot transpose = Transpose of d
     1.0000 + 1.0000i
     2.0000 + 2.0000i
     3.0000 + 3.0000i
```

행렬의 크기 구하기

- 행렬의 크기 구하는 함수
  - size(A) : 행렬의 크기 ⇒ A 행렬의 행과 열의 수를 되돌려

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

>> size(A)
ans =
   2   4

>> [row col] = size(A)
row =  2
col =  4

>> r = size(A, 1)
r =  2
>> r = size(A, 2)
r =  4
```

행렬의 크기 구하기

- 행렬의 크기 구하는 함수
  - size(A) : 행렬의 크기 ⇒ A 행렬의 행과 열의 수를 되돌려 줌
  - length(A) : 행렬의 길이 ⇒ 행렬의 행과 열의 수 중에서 큰 값을 되돌려 줌

```
>> x = 0:0.1:pi
x =
 Columns 1 through 13:
    0.00000    0.10000    0.20000    0.30000    0.40000    0.50000   ...    1.20000
 Columns 14 through 26:
    1.30000    1.40000    1.50000    1.60000    1.70000    1.80000   ...    2.50000
 Columns 27 through 32:
    2.60000    2.70000    2.80000    2.90000    3.00000    3.10000
>> length(x)
ans =  32

>> size(x)
ans =
    1   32
>> length(A)
ans =  4
```

행렬 조작 함수

Function	Description
reshape(A, r, c)	Construct new matrix with dimension (r x c) using given matrix A
rot90(A) rot90(A, k)	Rotates vector A by 90 degrees Rotates vector A by 90 degrees * k (k > 0 = ccw, k < 0 = cw)
flipud(A)	Flips array in up-down direction
fliplr(A)	Flips array in the left-right direction
triu(A)	Extracts upper triangular part
tril(A)	Extracts lower triangular part
diag(A)	Extracts diagonal elements
cross(A, B)	Outer product of A and B : $A \times B = (\lvert A \rvert \lvert B \rvert \sin\theta)$
dot(A, B) = sum(A.*B)	Inner product of A and B : $A \cdot B = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ $A = [a_1 \;\; a_2 \;\; \cdots \;\; a_n], \quad B = [b_1 \;\; b_2 \;\; \cdots \;\; b_n]$

행렬의 변형

- reshape : 행렬의 원소들을 유지하면서 행과 열의 수를 변경
  - 순서 : 열 우선, (cf. C언어에서는 행 우선으로 저장되어 있음)

```
>> A = [1, 2, 3; 4, 5, 6]
A =

   1   2   3
   4   5   6

>> B = reshape(A, 3, 2)
B =

   1   5
   4   3
   2   6
```

행렬의 회전 / 대칭 변환

- rot90(A, k) : 행렬의 회전
  - K > 0 : CCW (반시계방향)
  - K < 0 : CW(시계방향)

- 행렬의 대칭 변환
  - flipud(A) : 상하(up/down) 대칭
  - fliplr(A) : 좌우(left/right) 대칭

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

>> flipud(A)
ans =
     7     8     9
     4     5     6
     1     2     3

>> fliplr(A)
ans =
     3     2     1
     6     5     4
     9     8     7
```

```
>> rot90(A)
ans =
     3     6     9
     2     5     8
     1     4     7

>> diag(A)
ans =
     1
     5
     9

>> diag(ans)
ans =
     1     0     0
     0     5     0
     0     0     9
```

기타 조작

- triu(A), trid(A)
  - 대각선 위(up)/아래(down) 삼각행렬
- cross(A, B) : 외적
- dot(A, B) : 내적

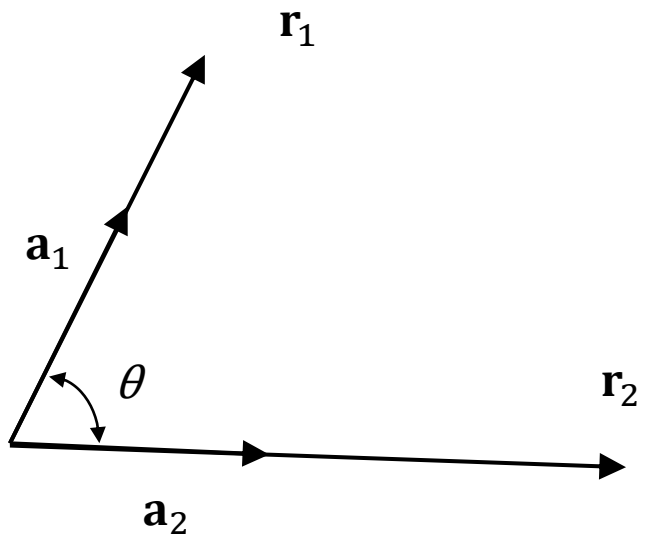
```
A =
     1     2     3
     4     5     6
     7     8     9

>> triu(A)
ans =
     1     2     3
     0     5     6
     0     0     9

>> A = [1, 2, 3];
>> B = [4, 5, 6];
>> dot(A, B)
ans =
     32

>> cross(A, B)
ans =
     -3     6     -3
```

연습

- 두 개의 단위 벡터 $\mathbf{a}_1, \mathbf{a}_2$의 내적은 두 벡터가 이루는 각도의 cosine 값이다. 이를 이용하여 두 벡터 $\mathbf{r}_1, \mathbf{r}_2$의 사이각을 구하라. $\mathbf{r}_1 = 3\mathbf{a}_x - 4\mathbf{a}_y + 2\mathbf{a}_z, \mathbf{r}_2 = 2\mathbf{a}_x + 9\mathbf{a}_y + 7\mathbf{a}_z$ 이다.

  - $\mathbf{a}_1 \cdot \mathbf{a}_2 = |\mathbf{a}_1||\mathbf{a}_2| \cos\theta = \cos\theta$
  - $\theta = \cos^{-1} \mathbf{a}_1 \cdot \mathbf{a}_2$

  - $\mathbf{r}_1 \cdot \mathbf{r}_2 = |\mathbf{r}_1||\mathbf{r}_2| \cos\theta$
  - $\theta = \cos^{-1}\left(\frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{|\mathbf{r}_1||\mathbf{r}_2|}\right), \ |\mathbf{r}| = \sqrt{\mathbf{r} \cdot \mathbf{r}}$



파일 입/출력

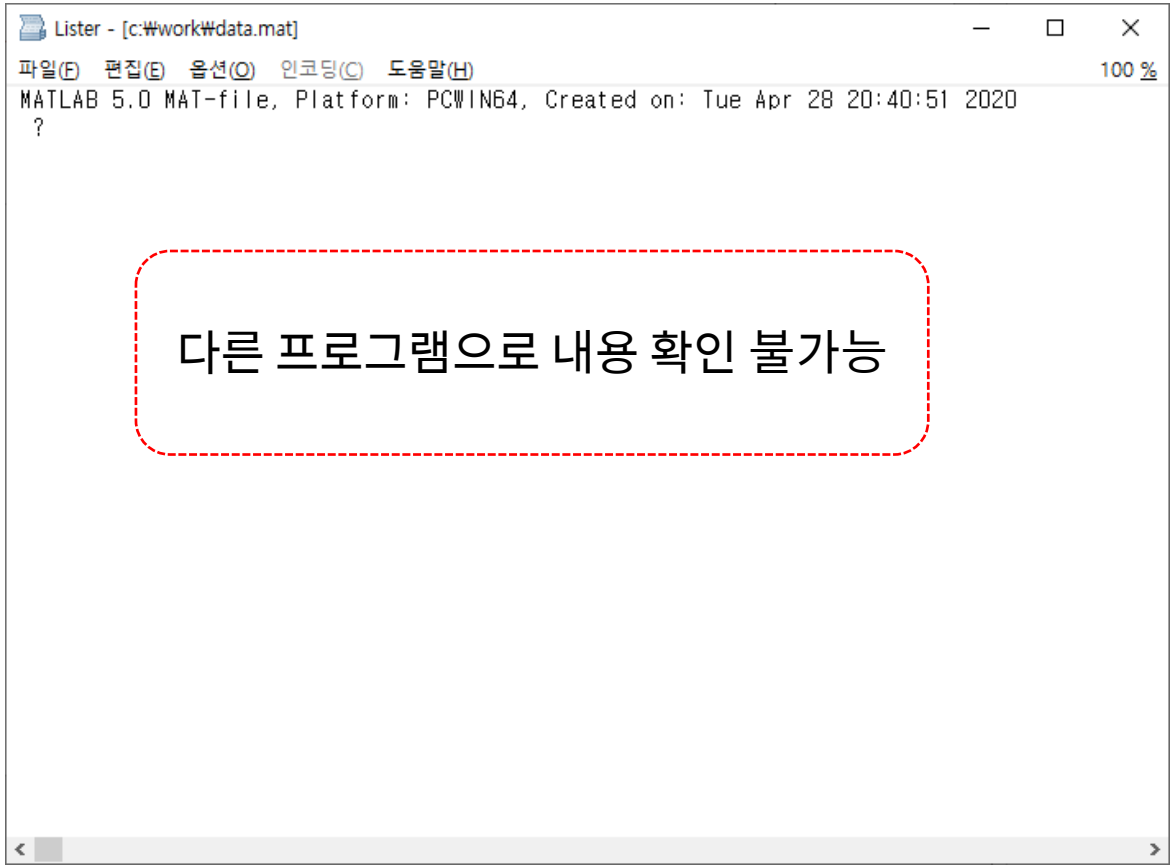
데이터 파일의 저장 및 불러오기

- save/load 명령어
  - fopen, fprintf 등의 함수보다 간단하게 변수들만 파일에 쓰고 읽는 방법
- save 명령어
  - 변수의 값을 파일에 저장
    `save 파일이름` 또는 `save('파일이름')` ⇒ save data
  - 지정한 변수들을 저장
    `save 파일이름 변수1 변수2`
  - 텍스트 파일로 저장 (`-ascii` 옵션 사용)
- load 명령어
  - save로 저장된 변수 읽어오기
    `load 파일이름` ⇒ load data

데이터 파일의 저장 및 불러오기

- Matlab : 확장자 없는 경우 - 파일이름.mat으로 저장 (고유형식)

```
>> A = [1,2,3; 4,5,6];
>> B= 1:5;
>> whos
  Name        Size            Bytes  Class     Attributes
  A           2x3                48  double
  B           1x5                40  double
>> save data
>> clear
>> whos
>> load data
>> whos
  Name        Size            Bytes  Class     Attributes
  A           2x3                48  double
  B           1x5                40  double
>> A
A =
     1     2     3
     4     5     6
>> B
B =
     1     2     3     4     5
```

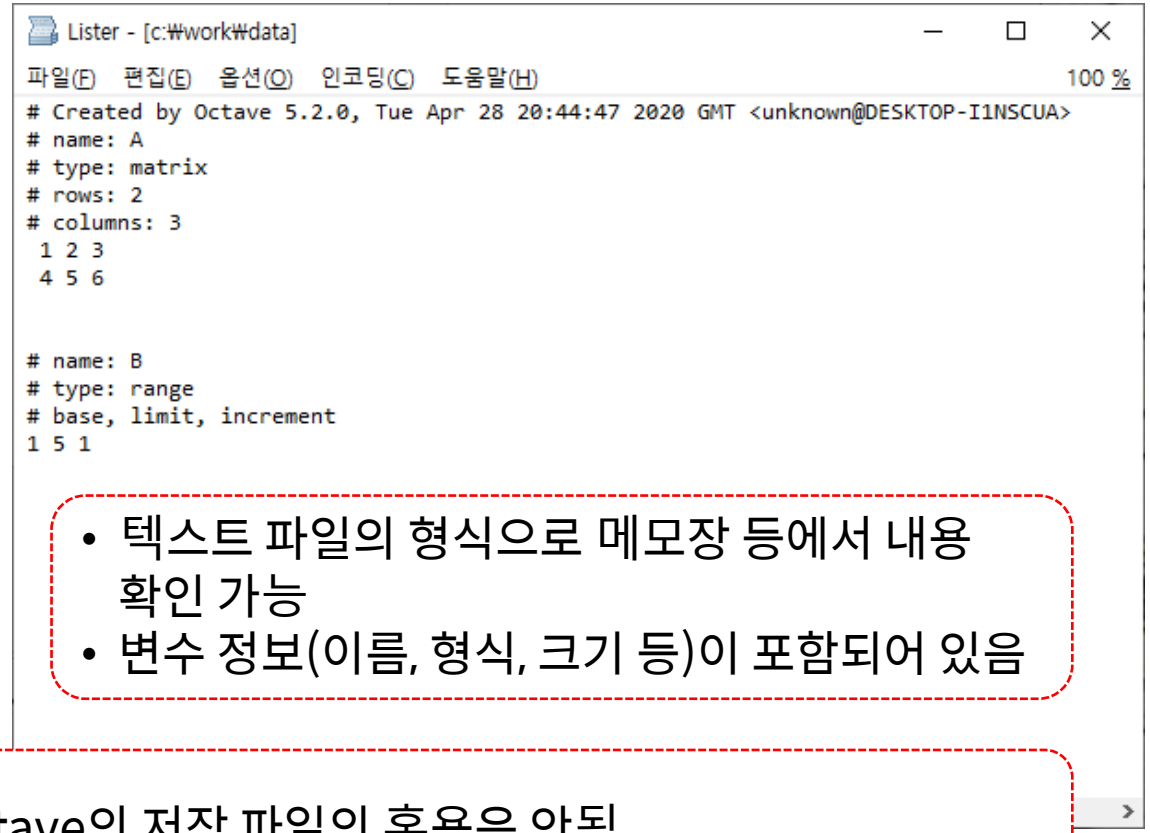


다른 프로그램으로 내용 확인 불가능

데이터 파일의 저장 및 불러오기

- Octave : 확장자 지정되지 않음 (텍스트 파일의 형식)

```
>> A = [1,2,3; 4,5,6];
>> B= 1:5;
>> whos
  Name       Size         Bytes  Class     Attributes
  A          2x3             48  double
  B          1x5             40  double
>> save data
>> clear
>> whos
>> load data
>> whos
  Name       Size         Bytes  Class     Attributes
  A          2x3             48  double
  B          1x5             40  double
>> A
A =
     1     2     3
     4     5     6
>> B
B =
     1     2     3     4     5
```



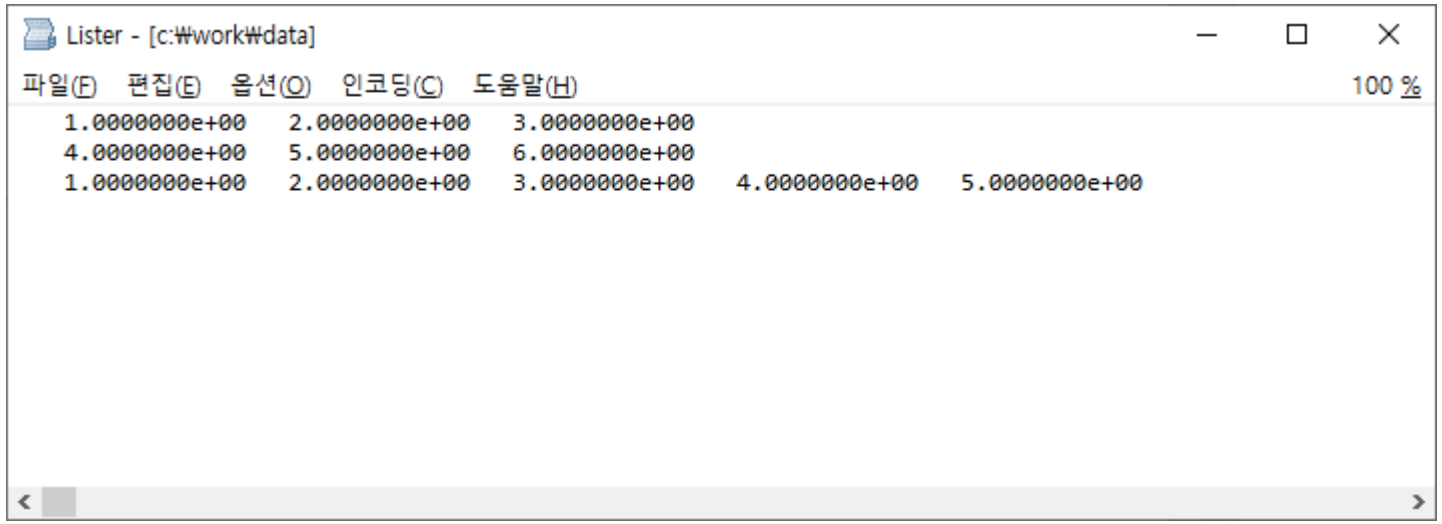
- 텍스트 파일의 형식으로 메모장 등에서 내용 확인 가능
- 변수 정보(이름, 형식, 크기 등)이 포함되어 있음

- Matlab과 Octave의 저장 파일의 혼용은 안됨
  예] Matlab 저장 → Octave 불러오기 등
- Octave에서 저장시 -7 옵션을 추가로 지정하면 Matlab 호환되는 데이터 파일로 저장 : 예] `save -7 data.mat`

데이터 파일의 저장 및 불러오기

- save -ascii 옵션
  - 고유형식 또는 추가 정보 없이 데이터만 텍스트 파일 형식으로 저장함
  - Matlab/Octave 동일

```
>> A = [1,2,3; 4,5,6];
>> B= 1:5;
>> save -ascii data
```



파일 입출력

- fprintf() 함수
  - 임의의 형식으로 텍스트와 데이터를 파일에 출력(저장)하기 위한 함수
  - 형식 : `fprintf(파일번호, '형식문자열', 변수들 ...);`
    - 파일번호는 fopen()함수를 사용하여 파일을 열 수 있음
  - C언어에서 파일에 출력하는 fprintf()함수와 유사하며, printf()함수처럼 스크린에도 출력이 가능
  - C언어와 다른 점
    - 문자열은 작은 따옴표를 사용함
    - 파일번호를 생략하는 경우 command window에 출력
  - 같이 사용하는 함수
    - fopen() : 파일 열기
    - fclose() : 파일 닫기

파일 입출력

- 파일 출력 순서
  - 파일 열기 : fopen('파일이름', '동작방식');
    - 동작방식
      - ✓ 'r' : 읽기(read) 모드, 'w' : 쓰기(write) 모드
      - ✓ 'a' : 추가(append) 모드 – 기존 파일에 추가
      - ✓ 'r+' : 읽고/쓰기 가능. 파일 생성은 안됨
      - ✓ 'w+' : 읽고/쓰기 가능. 파일 있으면 삭제되고 생성
      - ✓ 'a+' : 읽고/쓰기 가능. 파일 있으면 추가됨
  - 파일 일고/쓰기
    - fprintf(), fscanf() 등 파일 다루는 함수 사용
  - 파일 닫기 : fclose(fid);
    - 작업 종료 후에 파일 정리
- 추후 다시 자세히 다룸

```
>> fid = fopen('test.txt', 'w')
fid =  3

>> fprintf('sin(30) = %.5f\n', sind(30))
sin(30) = 0.50000

>> fclose(fid)

ans = 0
```

실습 문제

- 각도에 따른 삼각함수의 값을 계산하여 파일에 저장하는 스크립트를 완성하라.
  - 각도는 0도에서 360도까지 10도 단위로 한다.
  - 삼각함수는 sin, cos, tan의 세가지를 계산한다.
  - 행렬을 하나 만들어서 각 열마다 angle sin cos tan을 저장하고,파일 저장은 save 명령어에 -ascii 옵션을 이용한다.