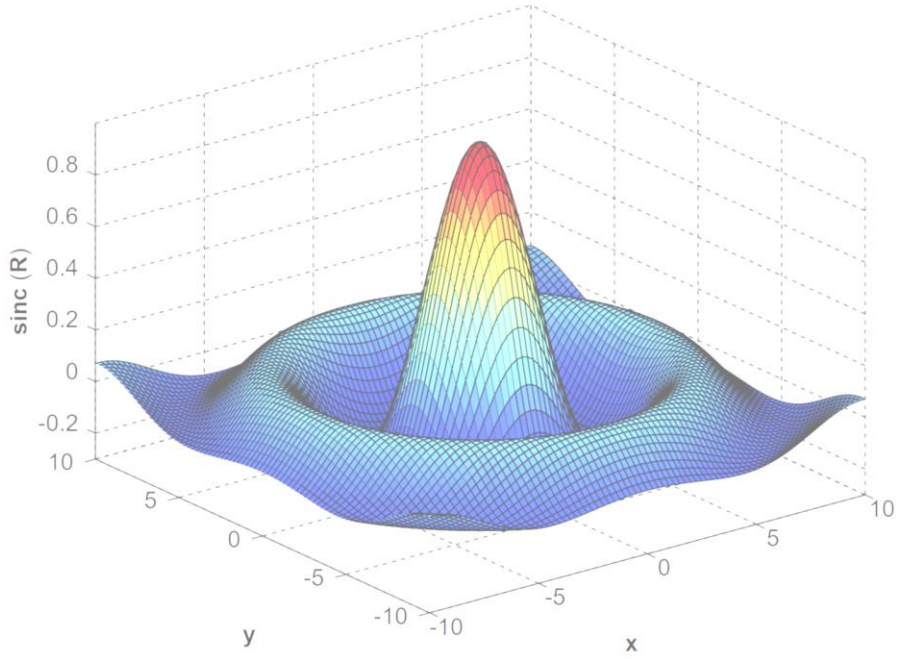
2021년도 2학기

응용전산및실습 II (02)

(강의자료 #1)



교과목명 : 응용전산 및 실습 II (02)

담당교수 : 이 수 형

E-mail : soohyong@uu.ac.kr

교재명 : 유인물

응용전산 및 실습 II 수업계획서

교과목명	**응용전산및실습II**	교과목번호	K3010C
담당교수	이 수 형	수강대상	에너지전기공학부 3학년
학점 (시수)	3(4)	이수구분	전공
강의시간, 강의실	목6,7,8,9 (갈마관 401)	교수연구실	갈마관 405
전화번호	054-760-1664	상담가능시간	사전약속 후 수시로 가능
핸드폰	010-2521-3227	E-mail 주소	soohyong@uu.ac.kr

수업 개요

- MATLAB은 전기, 전자 외 다양한 공학 분야에서 사용되는 소프트웨어로 범용성이 높고 사용하기 편리하며, SIMULINK 패키지를 사용할 경우 실험실에서 행하는 대부분의 실험은 모의를 할 수도 있을 것이다. 이와 같이 유용한 수학적 도구를 사용하여 복잡한 수치해석 문제와 전기해석에 관련된 문제를 학생들이 쉽게 해결할 수 있는 능력을 함양하는 데에 본 교과목은 목표를 두고 있다. 많은 분야에서 각광받고 있으며 그 정확성 또한 널리 검증되어 있는 상태이다. 기본적인 수치해석 문제를 손쉽게 해결할 수 있고 고차원의 수학적 문제 또한 MATLAB을 사용하여 간단히 해를 구할 수 있다.

수업 목표

- 수업 목표
  - 1학기 과정인 MATLAB의 기본적인 사용법 및 기본연산인 행렬 연산을 확인함.
  - MATLAB을 활용하여 그래프 작성하는 방법을 익히며, 응용법을 배움.
  - 3차원 그래프의 작성법을 익히며, 응용법을 배움.
  - 스크립트 파일의 작성법과 프로그램을 작성하는 방법을 익힘.

- 교재 및 참고문헌
  - 유인물 및 MATLAB 매뉴얼

성적 평가 및 참고사항

- 성적 평가 방법 및 기준

평가방법	반영비율	내용
출석 및 수업참여도	20%	• 결석 1회시 1점 감점, 지각 3회시 결석 1회 처리 • 수업시간 중에 참여하지 않을 시에 1점 감점 • 전체 수업 시간 중 1/3을 초과결석하는 경우 F 학점 처리함
중간고사	30%	• MATLAB의 기본적인 사용법과 연산방법 등의 이해 정도를 이론 시험을 통해서 평가
기말고사	30%	• MATLAB을 이용한여 그래프를 다루는 방법을 이해한 정도를 이론/실기 시험을 통해서 평가
과제물	20%	• 이론 강의 및 실습을 통해서 주어진 문제를 해결할 능력이 어느 정도인지 평가 • 과제물 제출은 그 다음 시간을 마감으로 함을 원칙으로 하며 늦을 경우, 감점 또는 0점 처리함

응용전산 및 실습 I (1학기) 복습하기

Matlab의 기초

Matlab 시작

- Matlab이란 무엇인가? (#1)
  - Matlab은 Matworks사의 테크니컬 컴퓨팅 언어로서, 공학계열에서 가장 많이 사용되는 계산용 언어이다. [https://kr.mathworks.com/]
  - MATLAB = MATrix + LABoratory : Mat이라는 용어가 Matrix에서 나왔으며 기본적인 계산방법이 행렬연산을 손쉽게 할 수 있게 한 일종의 계산용 언어이다.
  - 행렬연산을 이용하는 다양한 알고리즘들을 내장하고 있으므로 1줄로 간단하게 계산할 수 있다. (Matlab을 사용하지 않고 이러한 알고리즘을 C/C++, FORTRAN 등의 언어로 구현하려면 많은 노력이 필요하다.)
  - M-file이라는 형태로 일반적인 프로그래밍 언어로서의 사용이 가능하다. 즉, C/C++과 같이 프로그램을 작성하여 저장한 것을 불러서 수행할 수 있다.

Matlab 시작

- Matlab이란 무엇인가? (#2)
  - 구현하려면 많은 노력이 필요하다.)
  - M-file이라는 형태로 일반적인 프로그래밍 언어로서의 사용이 가능하다. 즉, C/C++과 같이 프로그램을 작성하여 저장한 것을 불러서 수행할 수 있다.
  - 프로그래밍 언어로서 윈도우 환경에서 그래프, 버튼, 메뉴 등의 GUI(Graphics User Interface) 프로그램을 작성하는 것이 가능하다.
  - 다양한 툴박스(toolbox)들을 통해서 다양한 분야에서의 응용이 가능하다.
    - ➢ 신호처리, 통계학, 영상처리, 제어, 재정, 화학 등
  - 심볼로 이루어진 수식을 계산하는 기호 계산이 가능하다.

Matlab 시작

- Matlab의 장점
  - 설치가 쉽다.
  - 사용하는 방법이 쉽다. 프로그래밍 언어로서의 문법도 쉬운 편이어서 처음 접하는 사람들도 쉽게 접할 수 있다.
  - 인터프리터 방식이므로 명령어 또는 계산식을 한 줄 입력하면 바로 결과를 확인할 수 있기 때문에 사용이 쉽다.
  - 다양한 그래프를 지원하므로 계산 결과를 시각화할 수 있는 기능이 우수하다.
  - CMEX라는 이름으로 C로 작성된 함수들을 불러와서 사용할 수 있다.
  - Matlab으로 작성된 것을 C언어로 변환하는 것이 가능하다. 단, 추가적인 라이브러리 등이 필요하기도 하며 Matlab 버전 및 컴파일러 버전에 따라 내용이 달라지기도 하므로 주의해야 한다.

Matlab 시작

- Matlab의 단점
  - 인터프리터 방식을 사용하므로 계산식 또는 프로그램을 읽으면서 해석한 다음 실행하기 때문에 C/C++과 같은 컴파일하는 언어에 비해서 느리다.
    - ➢ 최근 버전에서는 병렬처리 및 해석/컴파일 후 실행하기 때문에 처음 해석할 때를 제외하고는 실행 성능이 많이 향상되었다.
  - 상업용 소프트웨어이며 기업용은 매우 비싸다. 기본 패키지도 싼 편은 아니지만 각각의 툴박스들이 따로 구매하도록 하기 때문에 가격이 비싼 편이다.
    - ➢ 상업용은 기본 패키지가 260만원이며 필요한 툴박스들은 별도 구매이다.
    - ➢ 교육용은 기본적으로 사용하는 툴박스가 몇가지 포함되어 $55로 판매되고 있다.
  - 다른 프로그래밍 언어에 비해서 사용법이 쉽지만, 최근에 많이 사용되는 프로그래밍 언어로서의 방식에 비하면 기초적이다.

Matlab 시작

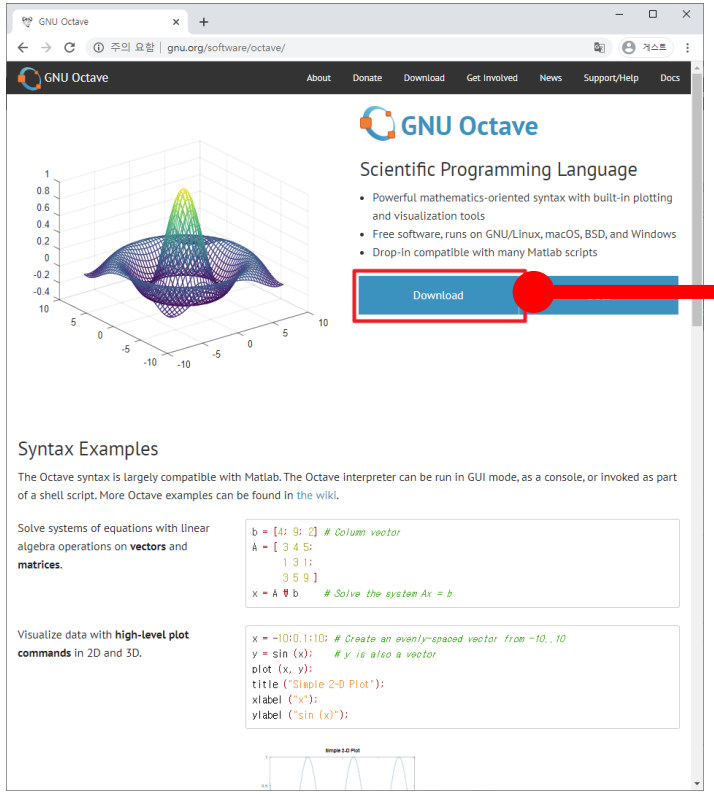
- Matlab 의 대안
  - Matlab이 공학용 계산 소프트웨어로 고가의 소프트웨어이므로 Matlab의 대안이 될 수 있는 비슷한 기능을 하는 다양한 무료 소프트웨어가 존재한다.
  - Matlab을 사용함으로서 사용하고자 하는 여러가지 기능들을 라이브러리 형태로 Python 등의 프로그래밍 언어로서 쉽게 사용할 수 있도록 만들어놓은 형태가 있고, Matlab처럼 별도의 소프트웨어로서 Matlab과 비슷하게 사용해놓은 S/W도 존재한다.
  - 프로그래밍 언어 (무료)
    - Python 라이브러리 (SymPy, Numpy) : 파이썬 프로그래밍 언어를 위한 수학, 과학, 공학용 계산 라이브러리 (아주 많은 기능을 제공한다.)
    - Julia 프로그래밍 언어 : Matlab과 비슷하게 행렬연산 및 다양한 과학/공학용 계산을 위한 프로그램 언어이며 속도 또한 매우 빠르다. 단, 아직 초기 개발 버전이다.
  - 무료 소프트웨어 패키지
    - Scilab, Octave : 둘 다 Matlab과 유시하게 사용할 수 있다. 벡터/행렬 연산을 기본으로 제공하며 Matlab과 유사하게 사용할 수 있다. 다만 Scilab 보다 Octave가 Matlab과 거의 동일한 명령어 및 계산식을 사용하므로 Octave가 여러가지 대안 중에서 가장 호환성이 뛰어나므로 대부분의 경우 동일하게 사용할 수 있다.

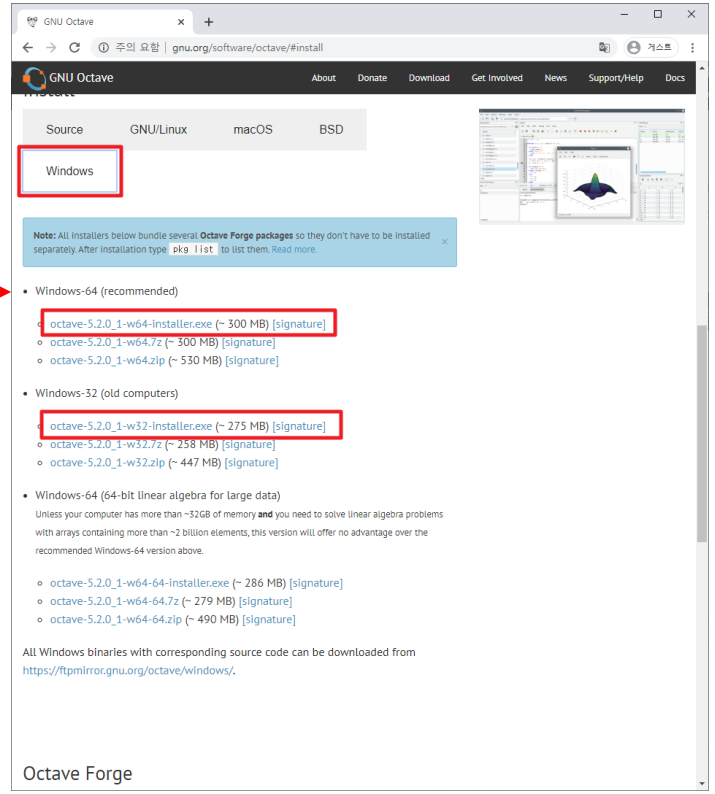
Octave 시작

- Octave
  - GNU재단에서 만든 수치해석용 소프트웨어로 누구나 무료로 사용이 가능하다.
  - 기본적으로 Matlab과 동일한 문법을 사용하므로 한 S/W에 익숙하면 다른 곳에 적용이 아주 쉽다.
  - Gnuplot 등의 그래프 툴과 연계해서 표, 그래프, 차트 등을 만들 수 있으며 Matlab과 동일한 명령어로 같은 내용의 그래프 등을 만들 수 있다. (다만, 그래프의 자세한 설정 방법은 Matlab과 다소 차이가 난다.)
  - Matlab이 사용 소프트웨어로 최적화 등에 많은 투자를 하기 때문에, Octave의 수행 속도는 Matlab에 비해서 느리다.
- Octave 사용 이유
  - 상용 소프트웨어이므로 구매해서 설치해 놓은 학교의 실험실에서는 사용이 가능하나, 예습/복습 등의 이유로 집에서는 따로 구매해서 사용하여야 한다.
  - 따라서 수강하는 학생들이 별도로 실습을 하기 위해서 사용할 수 있으므로 Octave를 설치하여 사용한다.

Octave 설치

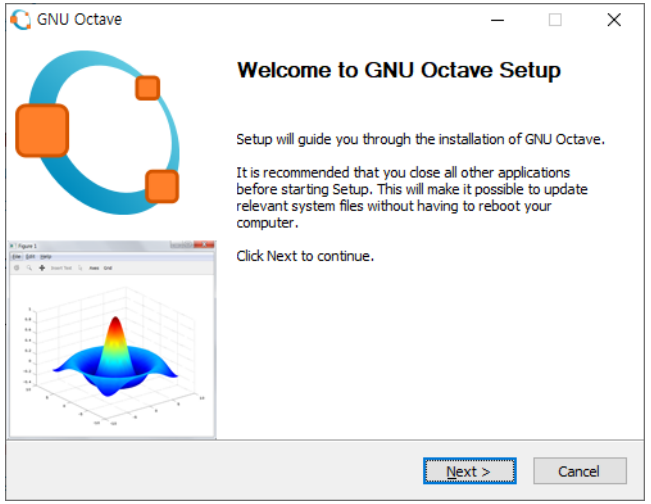
- Octave 설치 방법
  - 홈페이지 [http://www.gnu.org/software/octave/]에 접속한다.
  - 다운로드를 클릭하고, Windows를 선택한 후에 윈도우 버전에 알맞은 설치파일을 다운로드한다.

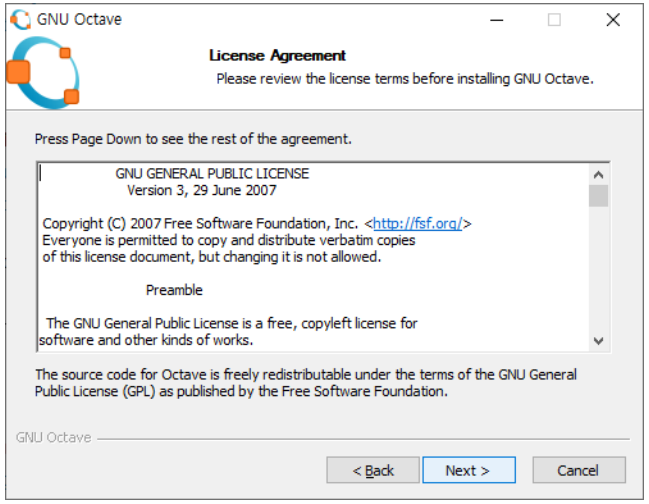


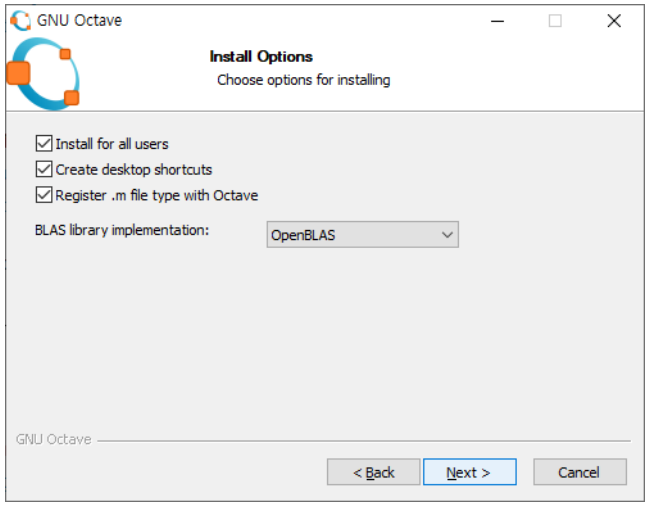


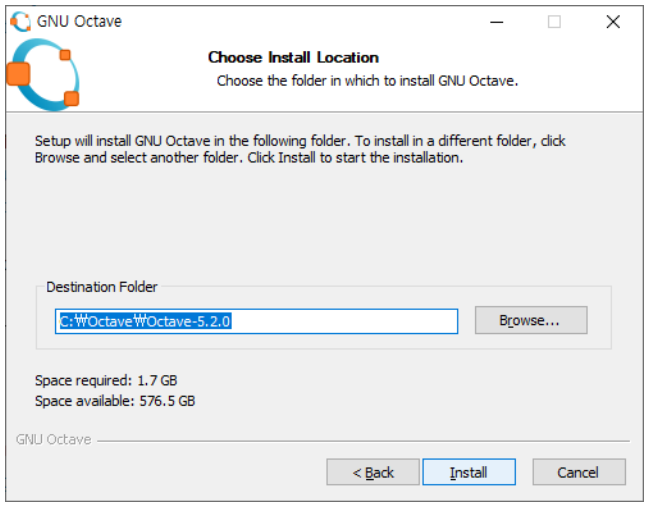
Octave 설치

- 설치 파일을 수행한 후에 기본으로 선택되어진대로 [Next]를 눌러서 설치한다.



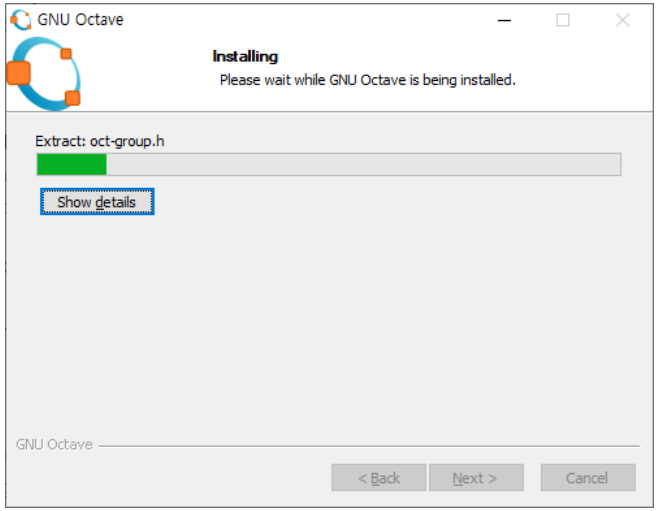


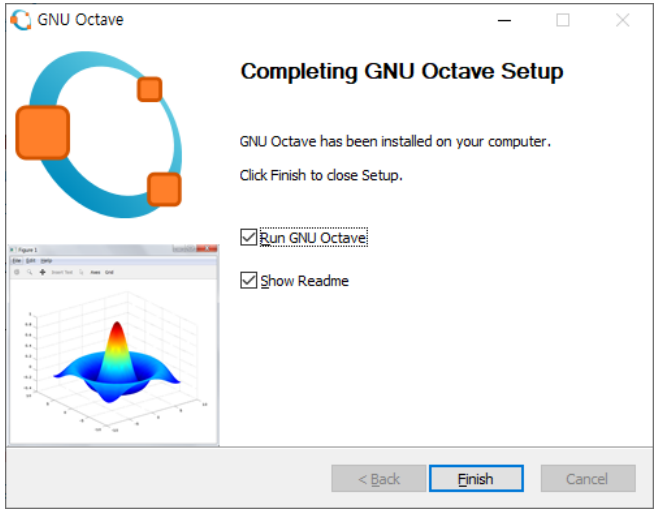


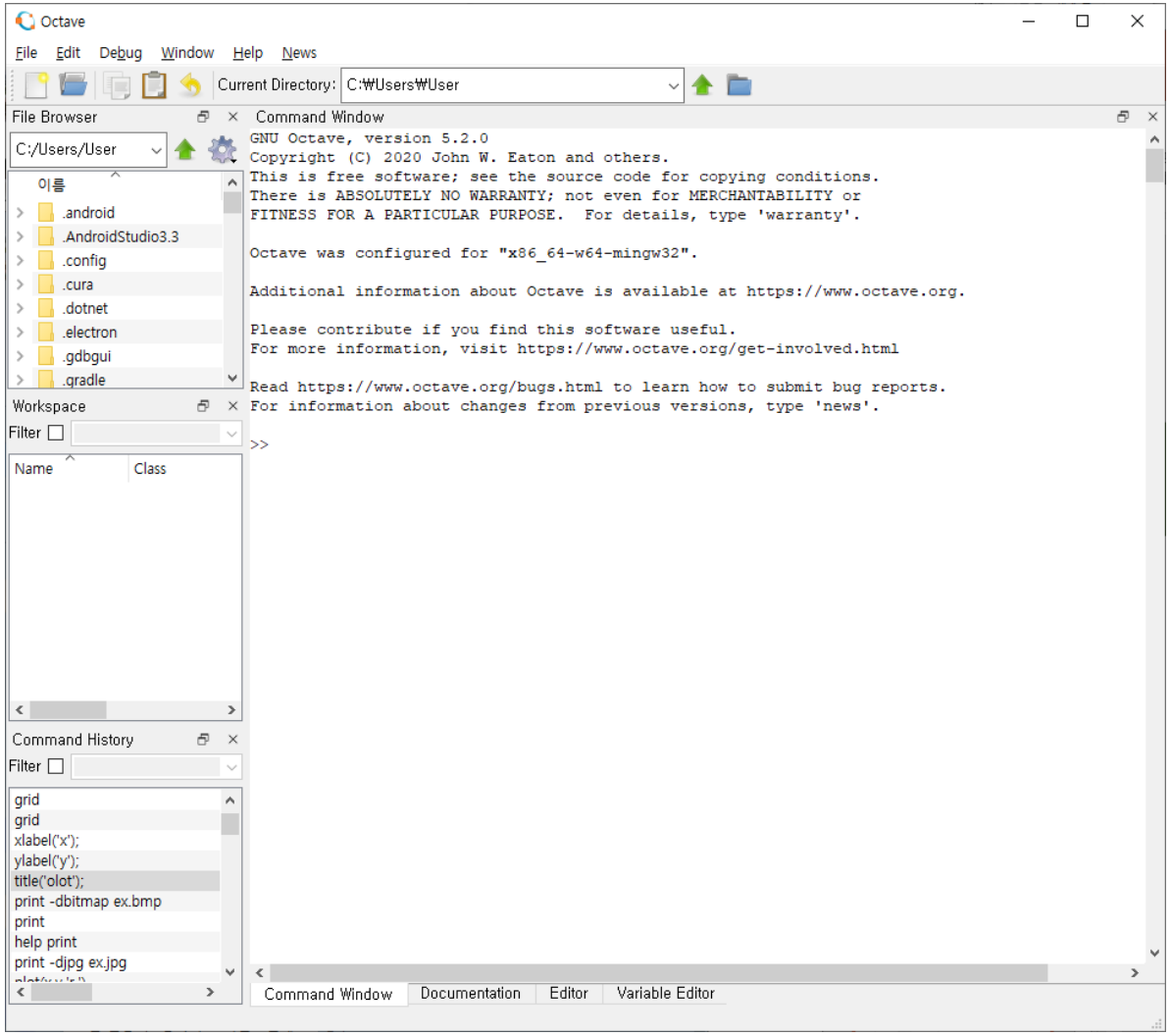


Octave 설치

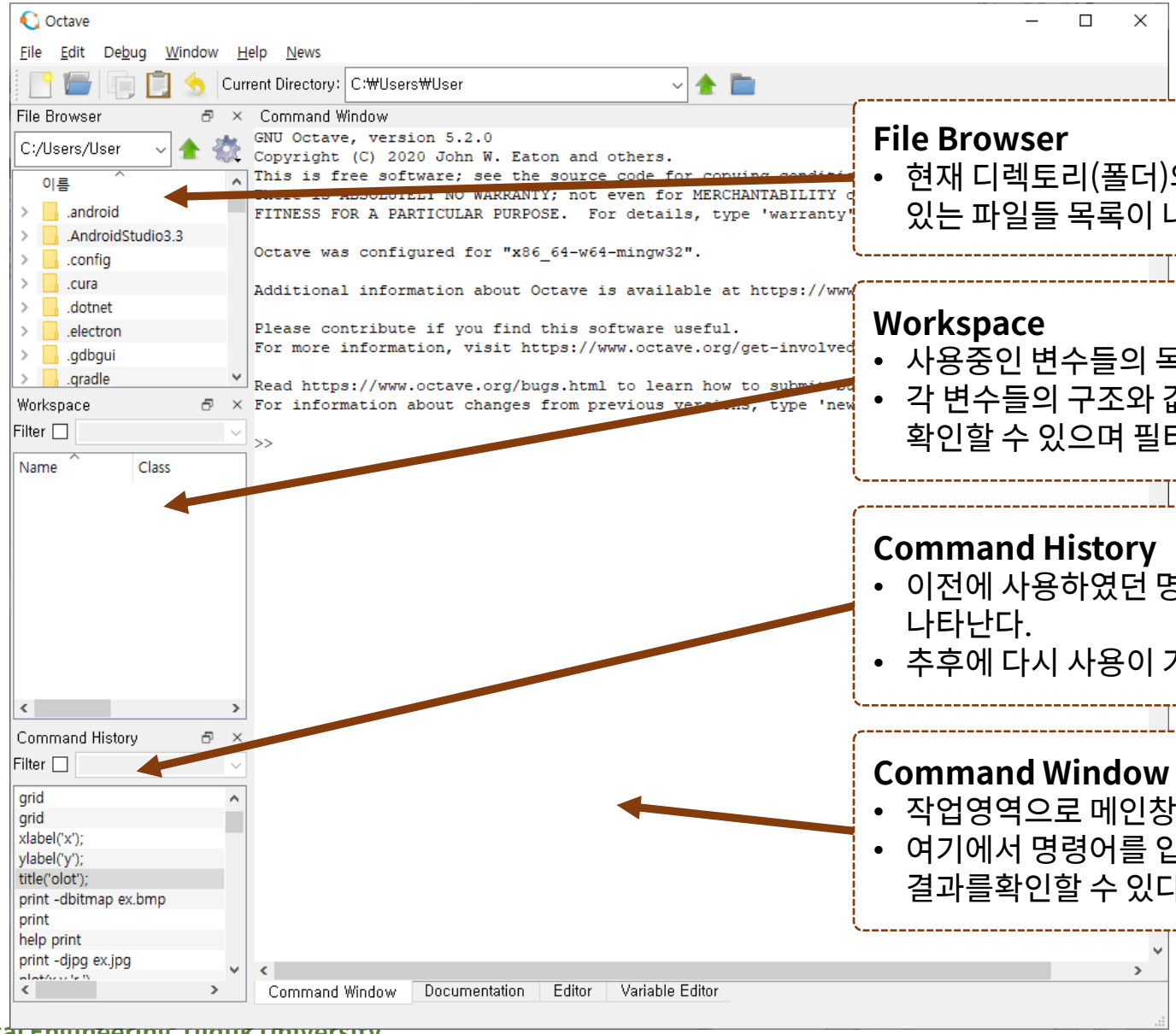
- 설치 후 수행하면 다음과 같은 화면이 나타난다.







Octave 화면



File Browser
- 현재 디렉토리(폴더)와 여기에 포함되어 있는 파일들 목록이 나타난다.

Workspace
- 사용중인 변수들의 목록이 나타난다.
- 각 변수들의 구조와 값을 포함한 정보들을 확인할 수 있으며 필터를 사용할 수 있다.

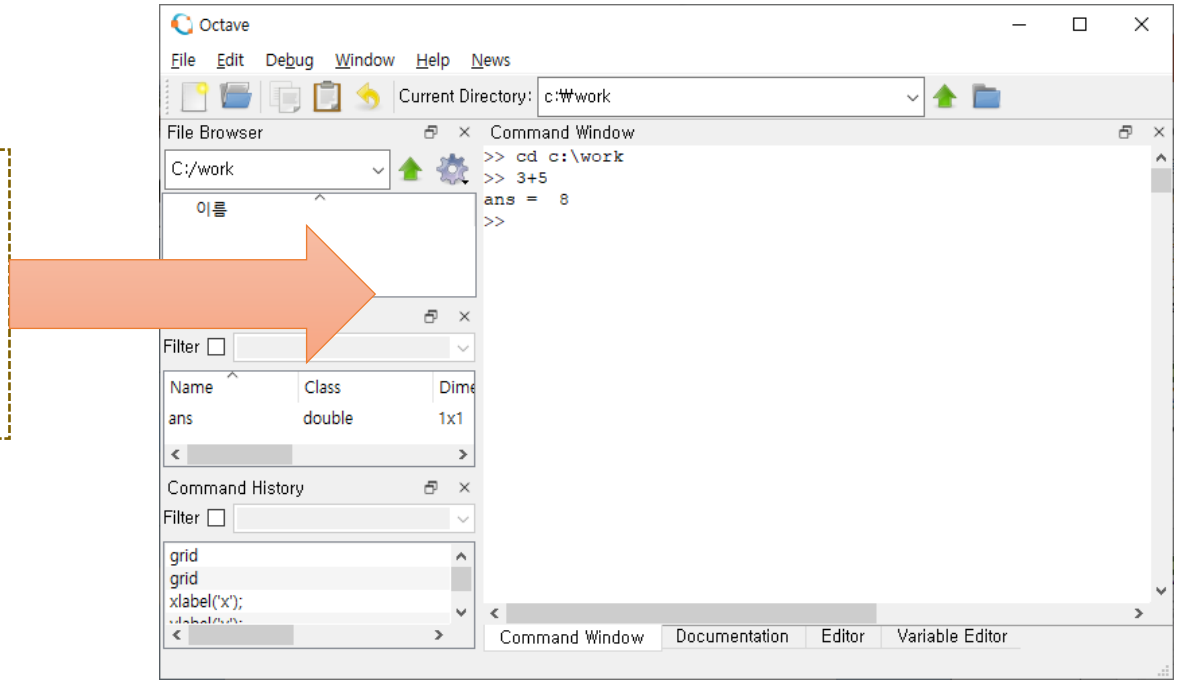
Command History
- 이전에 사용하였던 명령어/계산식들이 나타난다.
- 추후에 다시 사용이 가능하다.

Command Window
- 작업영역으로 메인창에 해당한다.
- 여기에서 명령어를 입력하고 수행 결과를확인할 수 있다.

Octave 시작

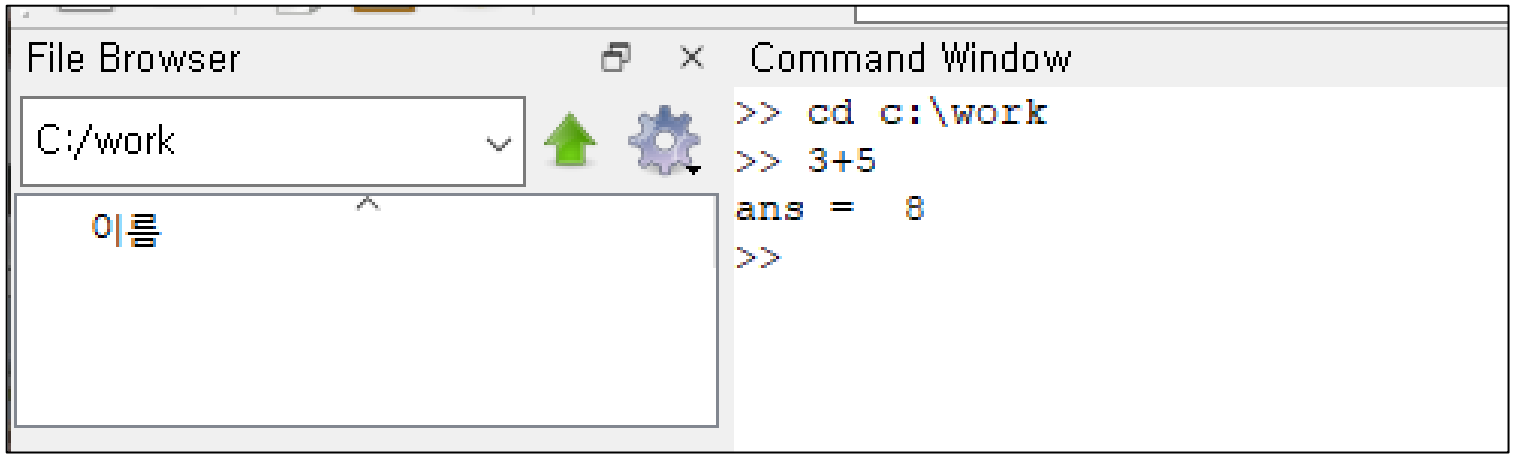
- 실습시에 작업영역은 C:\work로 지정한다.
  - File Browser에서도 이에 맞추어서 사용한다.
  - Matlab 또는 Octave의 명령어에서 cd 명령어를 사용하면 디렉토리 위치를 변경할 수 있다.
    - ➢ `cd c:\work` → c:\work 디렉토리(폴더)로 현재의 폴더를 변경한다.
- 명령창(Command Window)에서 명령어 입력

```
>> clc
>> cd c:\work
>> 3 + 5
ans = 8
>>
```



Octave 시작

- Octave 명령창의 사용 방법
  - [>>] 기호가 명령어를 입력할 수 있는 대기상태라는 것을 알린다.
  - [>>] 기호가 나타나면 앞의 명령어를 입력한다.
  - clc : clear screen으로 화면을 지우는 명령어이다.
    - ➢ 수행 후에는 그림과 같이 화면이 지워지며 정리된다.
  - 3+5 : 수식을 입력하면 수식을 계산한다.
    - ➢ 다른 변수에 입력하지 않는 계산 결과는 ans라는 변수에 자동으로 저장된다.
    - ➢ 즉, ans는 가장 최근의 계산결과를 의미한다.



사칙연산

- Matlab을 사용하여 계산식을 입력, 가장 간단한 계산식 → 사칙 연산

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

- 프로그래밍 언어와 비슷하게 더하기와 빼기는 수학 기호인 +, - 를 그냥 사용하고, 곱하기와 나누기는 *, / 로 사용한다.
- 특이하게 Matlab에서는 일반적인 나누기인 / 기호 외에 \ 기호를 사용하기도 하는데 / 기호는 right division 이라고 하고 \는 오른쪽의 값을 왼쪽으로 나누는 나눗셈을 수행하며 left division이라고 한다. 일반적인 숫자의 경우에는 순서만 주의하면 결과는 차이가 없으며, 행렬 연산에서 차이가 발생한다.

사칙연산

- 다음의 연산을 따라해 보도록 하자.
  - 먼저 +, -, *, / 는 C 언어에서 사용하던 연산자이므로 별 무리없이 이해가 될 것이다.
  - 왼쪽 나눗셈(left division)의 경우에 56 / 8과 8 \ 56의 결과가 동일한 것을 확인하도록 한다.
  - 거듭제곱(exponentiation)인 ^ 연산은 C 언어에서는 없지만 Matlab에서는 지원하며 FORTRAN 언어의 경우 **의 연산자로 지원한다.
    - 2^8 => 2^8
  - 잘 알다시피 괄호() 는 연산의 우선순위를 변경하는 용도로 사용된다. 오른쪽 아래 두 가지 경우를 비교해 보도록 하자.

```
>> 3 + 22
ans =  25

>> 90 - 54
ans =  36

>> 3.14 * 0.85
ans =  2.6690

>> 56 / 8
ans =  7

>> 8 \ 56
ans =  7

>> 2 ^ 8
ans =  256

>> ((1.043 + 2.01) * 3)^2
ans =  83.887

>> (1.043 + 2.01) * 3^2
ans =  27.477
```

사칙연산

- 계산식이 주어질 때 연산자들은 다음의 우선순위를 가지고 계산된다.

우선순위	연산자
1	괄호, 중첩되어 있는 경우에는 안쪽 괄호부터 수행된다.
2	거듭제곱 (^)
3	곱셈, 나눗셈
4	덧셈, 뺄셈

  - 동일한 우선순위의 연산자들은 동등한 순위를 가진다. 즉 곱셈과 나눗셈 기호는 서로 동등한 순위이다. 따라서 먼저 나오는 연산자를 먼저 수행한다.

- 위와 같이 복잡한 계산식의 경우에도 식만 입력하면 바로 계산되므로 탁상용 계산기의 용도로 사용해도 무방하다.

변수

- 변수(variable)란?
  - 변수는 하나의 이름을 가지는 기억장소로써 수치를 기억할 수 있으며, 이를 변경할 수 있다. 기억된 변수는 그 자체로 기억된 숫자로서 수식, 함수 등에 바로 사용할 수 있다.
  - Matlab에서는 숫자 하나(스칼라: scalar)를 기억할 수도 있고, 벡터 또는 배열을 저장할 수도 있다.
- 변수 이름 규칙
  - 대소문자를 구별한다.
    - ➢ Cost, cost, CoST, COST 는 서로 다른 변수의 이름이다.
  - 최대 31글자까지 허용한다. 단, 최근 Matlab과 Octave에서는 63글자까지 허용한다.
    - ➢ 참고로 namelengthname() 함수를 수행하면 최대 가능한 길이를 알 수 있다.
  - 변수 이름은 문자, 숫자, 밑줄(_)이 사용된 하나의 단어로 구성되며, 문자로 시작해야 한다. 다른 기호는 사용할 수 없다.
    - ➢ Octave에서는 '_' 로 시작할 수도 있다. (Matlab에서는 안된다.)
  - 키워드는 변수로 사용할 수 없으며, 함수 이름은 변수로 사용하지 않도록 주의한다.
    - ➢ 함수 이름을 변수로 사용해도 되나, 사용한 후에 해당하는 함수를 사용하지 못한다.

```
>> namelengthmax()
ans =   63
>>
```

변수

- 변수의 사용법
  - 변수에 값을 지정하고자 할 때는 = 기호를 사용.
    - 변수 a를 만들고 5를 대입한다.
    - 변수 b를 만들고 4를 대입한다.

```
>> a = 5
a =
     5
>> b = 4
b =
     4
```

  - 수식에 변수를 적으면 변수에 저장된 값이 사용됨
    - 기억되어 있는 a, b 값이 계산에 사용되어 a + b 는 5 + 4로 계산된다.

```
>> a + b
ans =
     9
>> a * b
ans =
    20
```

변수

- 미리정의된 특별한 변수들
  - Matlab에서는 시작할 때 미리 정의되어 있는 특별한 변수들이 있다. 이 변수들은 값의 변경이 가능하나, 변경 후에는 원래의 의미가 사라지므로 가능하면 변경하지 않도록 한다.

Special Variable	Description
ans	결과를 나타내는 기본 변수, 가장 최근에 계산된 결과를 저장하고 있음
pi	원주율 (π)
eps	인접한 두 수 사이의 최소값으로 이 숫자 이하로 가까이 있는 숫자들은 구분하지 못함 : 2.2204×10^{-16}
inf	무한대 (1 / 0)
i , j	허수를 표현하는데 사용되는 상수, i = j = $\sqrt{-1}$
realmin	저장할 수 있는 최소의 + 숫자 : 2.2251×10^{-308}
realmax	저장할 수 있는 최대의 + 숫자 : 1.7977×10^{308}

```
>> pi
ans =
    3.1416
>> eps
ans =
   2.2204e-16
>> inf
ans =
   Inf
>> i
ans =
   0.0000 + 1.0000i
>> j
ans =
   0.0000 + 1.0000i
>> realmin
ans =
  2.2251e-308
>> realmax
ans =
  1.7977e+308
```

변수

- 미리 정의된 키워드
  - 키워드(keyword) : Matlab에서 특정한 명령어로 사용되므로 변수이름으로 사용불가 프로그램을 작성하는 경우 제어문으로 사용되는 명령어가 대부분이다.
  - Matlab과 Octave의 경우 미리 정의된 키워드가 조금 다르나, 제어문과 같은 것들은 공통이므로 아래의 키워드를 보면서 구분해보자.
  - Matlab keywords
    - break, case, catch, classdef, continue, do, else, elseif, end, for, function, global, if, otherwise, parfor, persistent, return, spmd, switch
  - Octave keywords
    - __FILE__, __LINE__, break, case, catch, classdef, continue, do, else, elseif, end, end_try_catch, end_unwind_protect, endclassdef, endenumeration, endevents, endfor, endfunction, endif, endmethods, endparfor, endproperties, endswitch, endwhile, enumeration, events, for, function, global, if, methods, otherwise, parfor, persistent, properties, return, switch, try, until, unwind_protect, unwind_protect_cleanup, while
    - 특히 end로 시작되는 키워드들은 Matlab에서 사용하는 것처럼 end로 대신할 수 있다.

자주 사용하는 명령어

- 앞에서 본 화면을 지우는 clc 이외에도 자주 사용하는 명령어들이 있으므로, 반복 연습하여 익히도록 한다.
  - help : 함수 이름이나 명령어의 사용법을 익히는 명령어.
  - who : 현재 사용중인 변수들의 목록출력
  - whos : 변수들의 목록 및 크기, 데이터 형 등의 정보를 같이 출력.
  - clear : 사용중인 변수를 지움. 변수명을 지정하지 않으면 모든 변수를 제거.

```
>> help cos
 cos    Cosine of argument in radians.
    cos(X) is the cosine of the elements of X.

    See also acos, cosd.
    .... (생략)

>> a = 5
A =
     5
>> b = 6
B =
     6
>> who

사용자의 변수:
A  b
>> whos
  Name        Size              Bytes  Class     Attributes

  a           1x1                   8  double
  b           1x1                   8  double

>> clear a
>> a
'a'은(는) 정의되지 않은 함수 또는 변수입니다.
>> b
B =
     6
>> clear
```

자주 사용하는 명령어

- 아래의 명령어 중에서 pwd, cd 등은 자주 사용하므로 반드시 익히도록 한다.
  - pwd (print working directory)
    - ➢ 현재 디렉토리(폴더)를 보여준다.
  - cd (change directory)
    - ➢ 현재 디렉토리를 변경한다.
  - dir : 현재 디렉토리의 파일들을 보여준다.
  - quit : Matlab를 종료한다.
  - version : 현재 버전을 출력한다.

```
>> pwd
ans = c:\users\user
>> cd c:\work
>> pwd
ans = c:\work
>> dir
.   ..
>> version
ans = 5.2.0
```

[Octave 수행 결과]

```
>> pwd
ans =
C:\WINDOWS\system32
>> cd c:\work
>> pwd
ans =
c:\work
>> dir

.   ..

>> version
ans =
8.6.0.267246 (R2015b)
>>
```

[Matlab 수행 결과]

복소수 (Complex Number)

- 복소수 사용법

Declaration
a ± bi, a ± bj

Related Function	Description
abs(x)	Magnitude of x (= a ± bi) =
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

- 자주 사용되는 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

```
>> exp(1)
ans =  2.7183
```
e^1

```
>> log(2)
ans =  0.69315
```
$\ln 2$

```
>> log10(2)
ans =  0.30103
```
$\ln_{10} 2$

```
>> log2(2)
ans =  1
```
$\ln_2 2$

```
>> pow2(4)
ans =  16
```
2^4

```
>> sqrt(3)
ans =  1.7321
```
$\sqrt{3}$

삼각함수 (Trigonometric Functions)

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) = $\tan^{-1}(x)$
sinh(x)	Hyperbolic sine of x = sinh(x) = $(e^x – e^{-x}) / 2$
cosh(x)	Hyperbolic cosine of x = cosh(x) = $(e^x + e^{-x}) / 2$
tanh(x)	Hyperbolic tangent of x = tanh(x) = $(e^x - e^{-x}) / (e^x + e^{-x})$

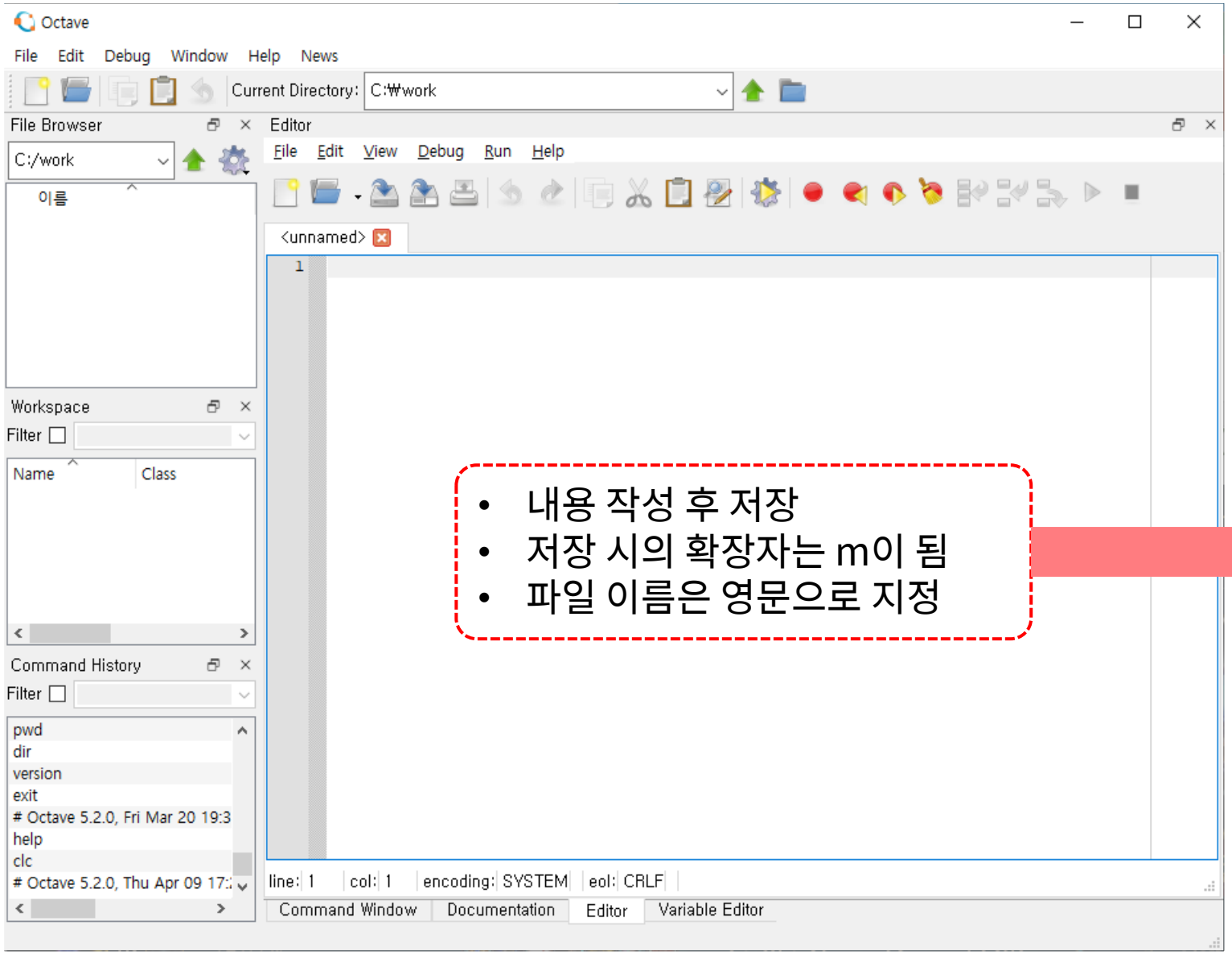
M 파일의 작성

- M파일 : Matlab에서 사용하는 프로그램 저장용 파일
  - 확장자가 'm'으로 되어 있음
  - 스크립트 파일과 함수 파일로 나누어짐 (함수 파일은 나중에 다룰 예정)

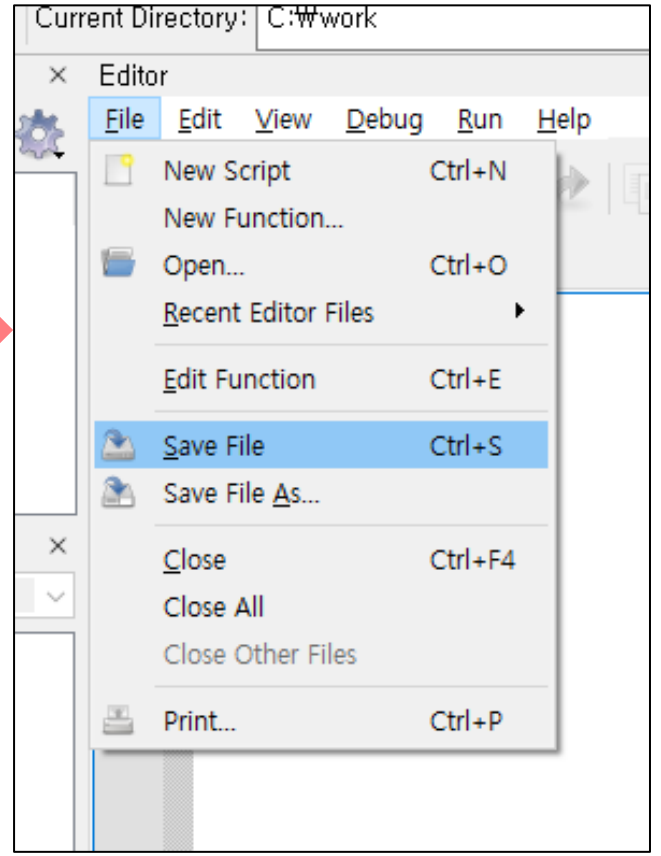
- 스크립트 파일
  - 명령어들을 미리 모아놓은 파일
  - 명령창(command window)에서 호출 → 저장된 명령어들을 순차적으로 실행

- 새파일 작성
  - 메뉴에서 [File → New → New Script]를 선택하여 실행

M 파일의 작성



- 내용 작성 후 저장
- 저장 시의 확장자는 m이 됨
- 파일 이름은 영문으로 지정

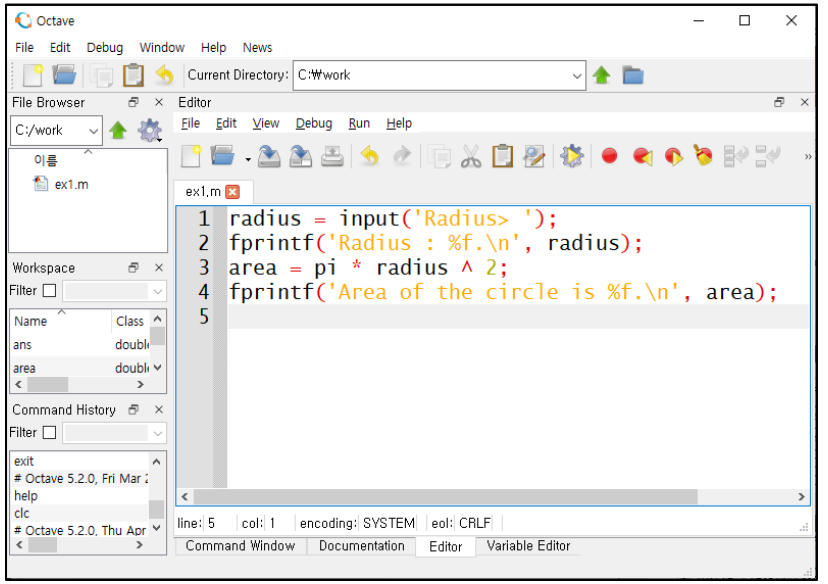


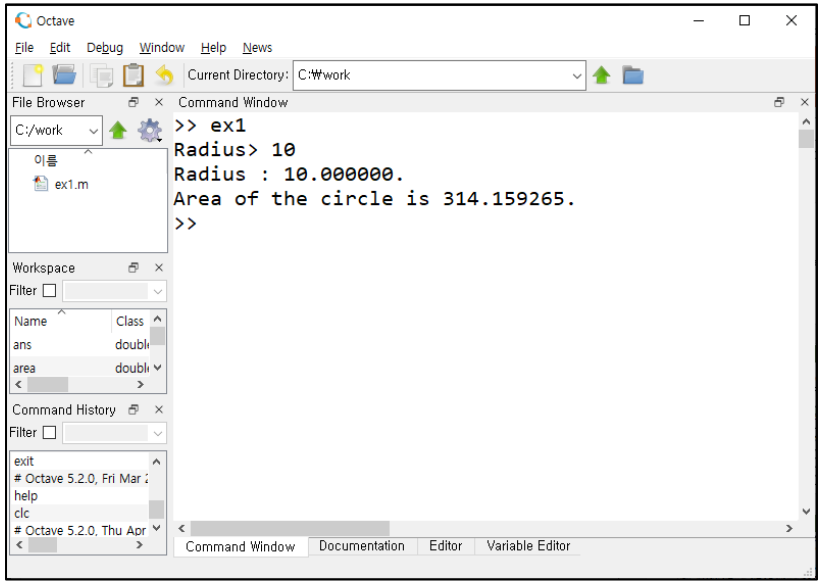
M 파일의 작성

- M-file 작성
  - 현재 폴더에 저장 (파일 이름은 영문으로)
    명령창에서 한글입력 불가능
  - 파일이름
- 수행
  - 명령창에서 파일 이름 (확장자 제외) 입력
    >> ex1

ex1.m

```
radius = input('Radius> ');
fprintf('Radius : %f.\n', radius);
area = pi * radius ^ 2;
fprintf('Area of the circle is %f.\n', area);
```





M 파일의 작성

- M 파일의 작성에 자주 사용되는 키워드/함수들

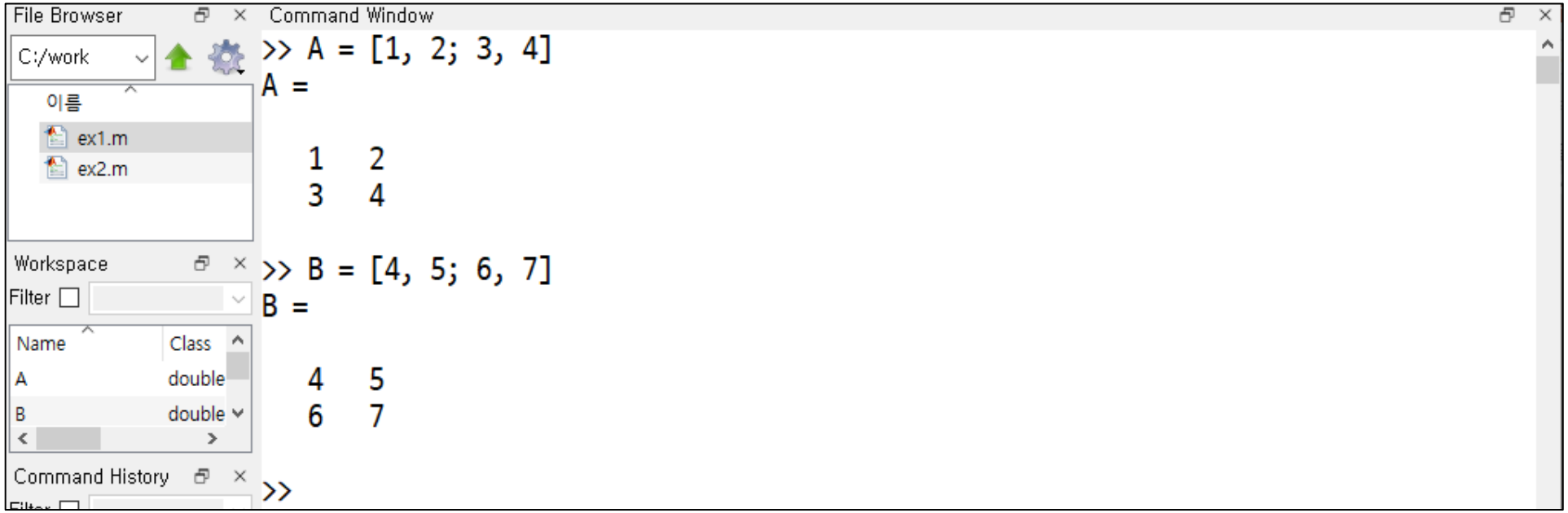
Keyword / Function	Meaning	Example
;	Suppress output	radius = 5;
%	Comment	radius = 5; % 반지름 값
disp(variable)	Display results without identifying variable names	disp(radius)
echo	Control the 'Command Window' echoing of script file commands	echo
input	Prompt user for input	radius = input('반지름을 입력하세요> ');
pause	Pause until user presses any key	pause
pause(n)	Pause for n seconds, then continue	pause(3)
fprintf(format, var1, var2,…)	Print msg	fprintf('계산이 완료되었습니다.'); fprintf('반지름은 %f 입니다.\n', radius);

행렬의 사용

행렬

- 행렬의 정의

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



행렬

- 벡터(vector)
  - 1차원 행렬, 행벡터(row vector), 열벡터(column vector)

```
>> x = [1, 2, 3, 5]
x =

   1   2   3   5

>> y = [1; 3; 2]
y =

   1
   3
   2
```

```
>> 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’

- 행렬의 계산

```
>> 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
```

연습

- 원소가 $8, 10/4, 12 \times 1.4, 0.15, \sin 60^{o}, \sqrt{12}$인 행벡터를 생성하라.

- 다음 행렬 A와 B를 정의하고, $A + B, A - B, AB, A \cdot B^{-1}, A^{2}$를 계산하는 스크립트를 작성하라.

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 9 \\ 1 & -1 & 0 \\ 2 & 1 & 5 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} -1 & 3 & 9 \\ 2 & 6 & -2 \\ 6 & 8 & 5 \end{bmatrix}$$

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	$a .\hat{}\, c = [a_1^c, a_2^c, \cdots, a_n^c]$
	$\mathrm{c} .\hat{}\, a = [c^{a_1}, c^{a_2}, \cdots, c^{a_n}]$
	$a .\hat{}\, b = \left[a_1^{b_1}, a_2^{b_2}, \cdots, a_n^{b_n}\right]$

행렬연산자

- 스칼라와 행렬 연산
  - $A = \begin{bmatrix} 2 & 4 \\ 5 & 6 \end{bmatrix}, B = \begin{bmatrix} 3 & 4 \\ 5 & 7 \end{bmatrix}, C = 4$
  - $\begin{bmatrix} 2*3 & 4*4 \\ 5*5 & 6*7 \end{bmatrix}$
  - $\begin{bmatrix} 2/3 & 4/4 \\ 5/5 & 6/7 \end{bmatrix}$
  - $\begin{bmatrix} 2*4 & 4*4 \\ 5*4 & 6*4 \end{bmatrix}$

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 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 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
```

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
```

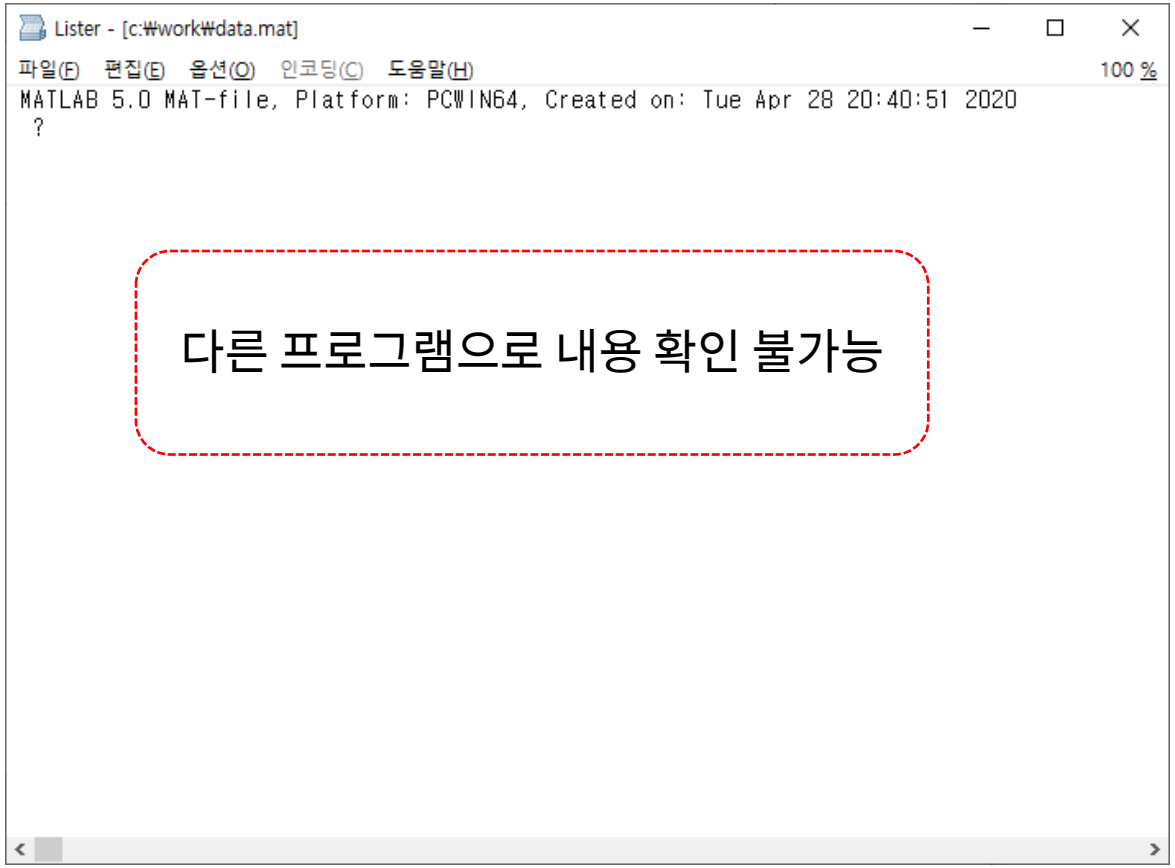
데이터 파일의 저장 및 불러오기

- 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)

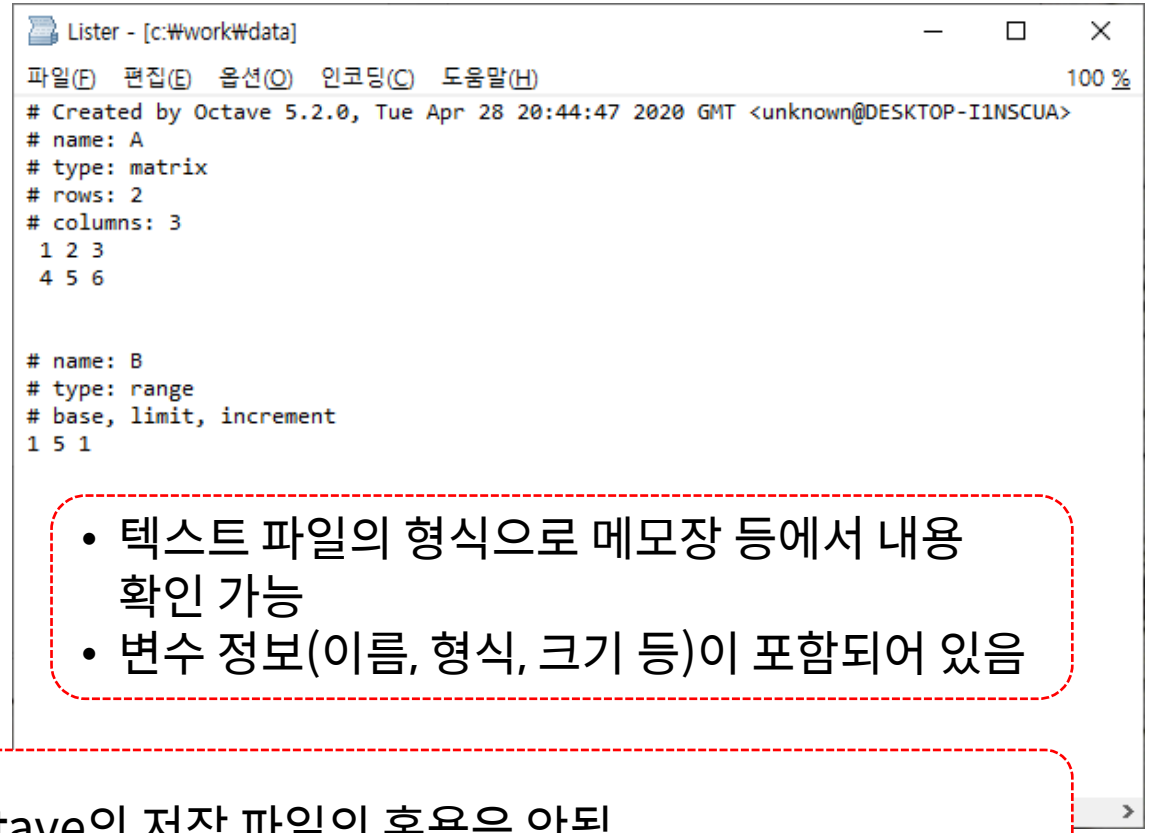
```
>> A = [1,2,3; 4,5,6];
>> B= 1:5;
>> whos
  Name        Size              Bytes  Class     Attributes
  A           2x3                  48  double
  B           1x5                  40  double
>> save data
>> clear
>> whos
>> load data
>> whos
  Name        Size              Bytes  Class     Attributes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



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
```



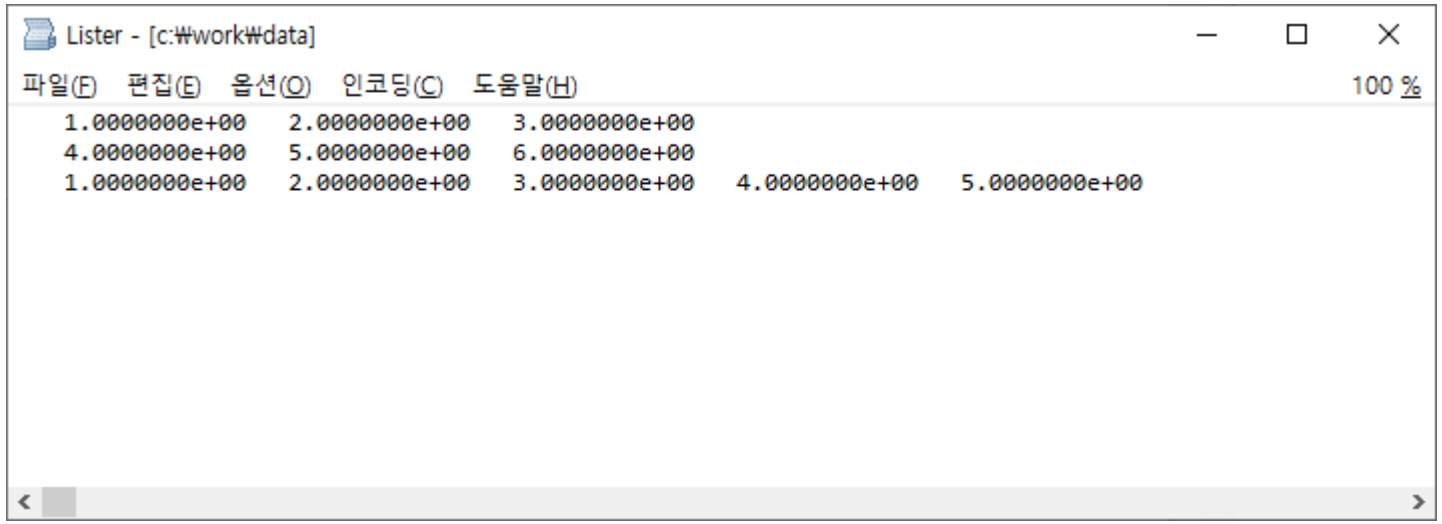
- 텍스트 파일의 형식으로 메모장 등에서 내용 확인 가능
- 변수 정보(이름, 형식, 크기 등)이 포함되어 있음

- 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

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
```

2차원 그래프

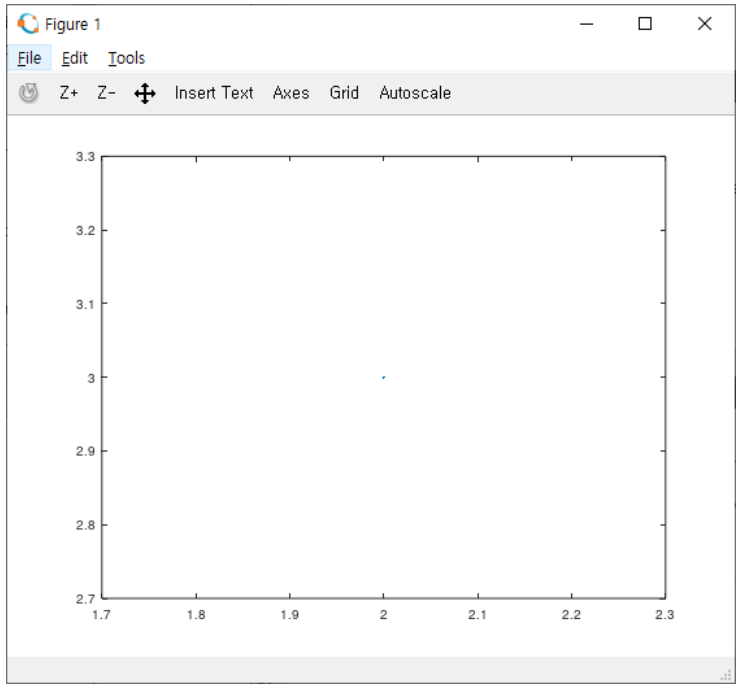
2차원 그래프

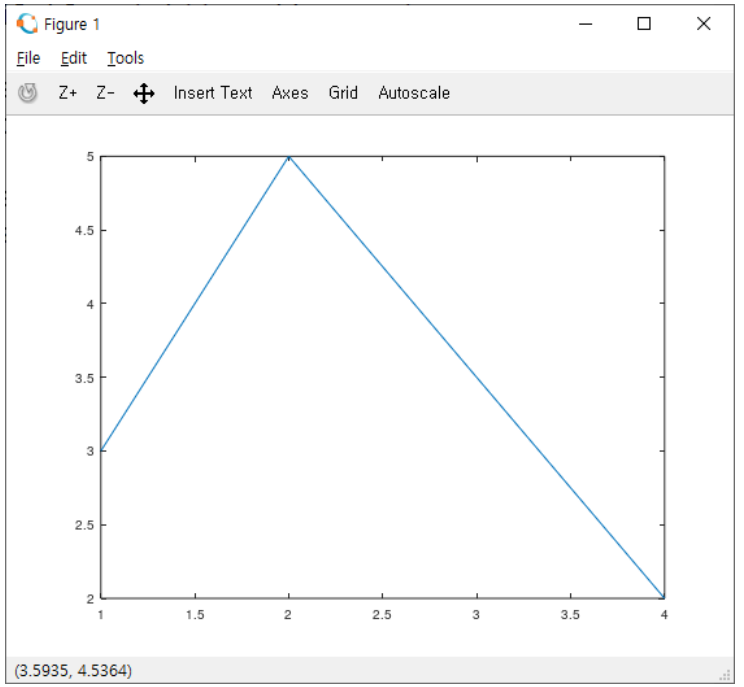
- 2차원 그래프 함수
  - `plot(x, y)` : 기본적인 2차원 그래프 함수
  - x : x축 데이터, y : y 축 데이터 → 두 데이터의 크기는 동일해야 함
  - x, y의 크기에 따른 그래프 표현
    - ➢ 스칼라 : 점으로 표시, Matlab에서는 표시 안됨 → 표식(mark) 지정해야 함
    - ➢ 벡터 : 선으로 표시
    - ➢ 행렬 : 여러 개의 선으로 표시

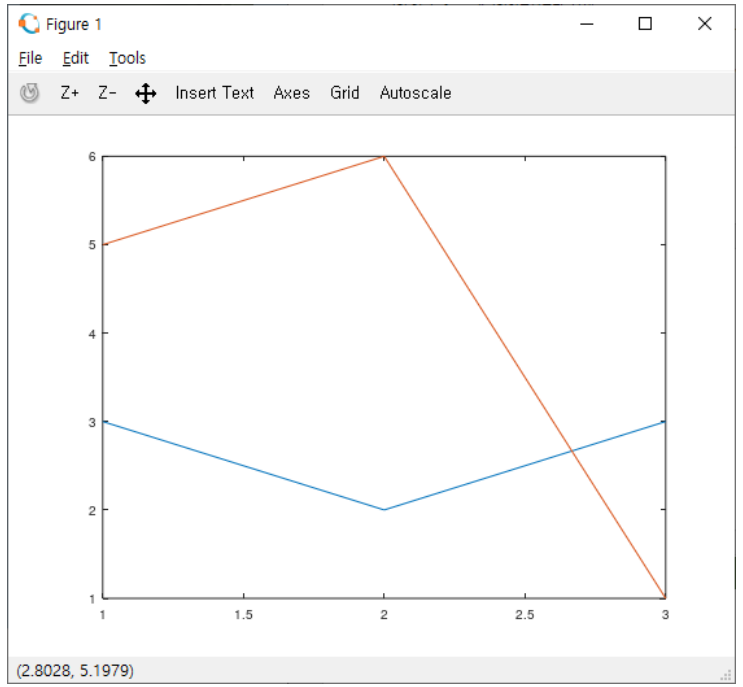
2차원 그래프

- plot() 함수 사용

```
>> plot(2, 3)
>> plot([1 2 4], [3 5 2])
>> plot([1 1; 2 2; 3 3], [3 5; 2 6; 3 1])
```







2차원 그래프

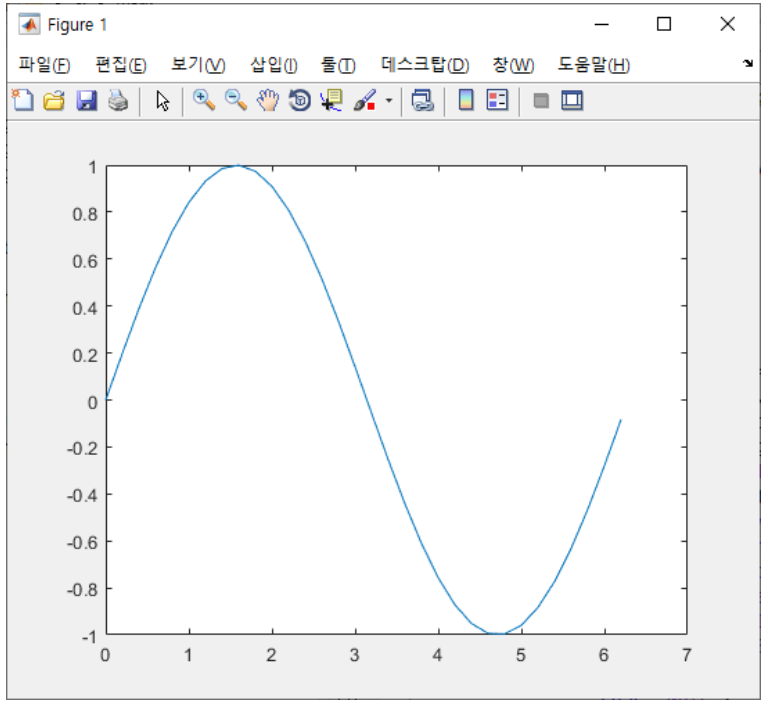
- 함수 그래프 그리기 : $f(x) = \sin x\,, 0 \le x \le 2\pi$
  - 데이터의 준비 : $x = [0, 0.2, 0.4, \ldots, 2\pi], y = [\sin 0\,, \sin 0.2\,, \ldots \sin 2\pi]$

```
>> x = 0:0.2:2*pi;
>> y = sin(x);
```

⇒ sin 함수 : 벡터 입력 → 벡터 출력

  - x축 데이터의 간격 ?

```
>> x = 0:0.5:2*pi;
>> y = sin(x);
>> x = 0:0.1:2*pi;
>> y = sin(x);
>> x = 0:0.01:2*pi;
>> y = sin(x);
```

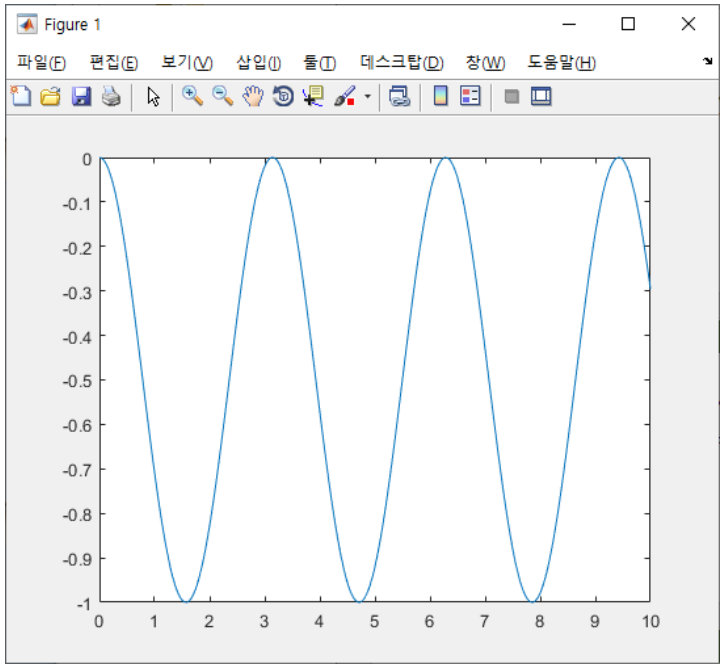


2차원 그래프

- 함수 그래프 그리기 : $f(x) = \sin x \cos\left(x + \frac{\pi}{2}\right), 0 \le x \le 10$
  - 데이터의 준비 : 벡터의 연산

```
>> y = sin(x) .* cos(x + pi / 2);
```
⇒ 벡터의 각 요소끼리 연산 .* .^ 등 사용

```
>> x = 0:0.01:10;
>> y = sin(x) .* cos(x + pi / 2);
>> size(x)
ans =
           1        1001
>> size(y)
ans =
           1        1001
>> plot(x, y)
>>
```



2차원 그래프

- 함수 그래프 그리기 : plot() 함수의 활용
  - figure() 함수 : 그래프 윈도우 생성, 선택
    - `figure;` ⇒ 새로운 그래프 윈도우 생성
    - `figure(1);` ⇒ 1번 그래프 윈도우 선택 및 활성화, 없으면 생성
      ⇒ plot() 함수 등은 활성화된 윈도우에 그래프를 표시함
  - plot() 함수 형식
    - `plot(x, y);` ⇒ 기본 형식
    - `plot(x, y, s);` ⇒ s : 형식(색상/선/표식) 지정
    - `plot(x1, y1, x2, y2);` ⇒ 2개의 그래프 동시에 그리기
    - `plot(x1, y1, s1, x2, y2, s2);` ⇒ 형식 및 2개의 그래프 동시에 그리기
    - `plot(x, y, '속성', '속성값');` ⇒ 그래프 여러 가지 속성 지정

2차원 그래프

- plot() 함수의 형식 설정
  - `plot(x, y, 형식);` ⇒ 문자열로 표현된 형식(색상, 표식 모양, 선 모양)
  - 형식 : 문자색상 + 표식 모양 + 선모양
    - ➢ 예] `plot(x, y, 'ro--');` ⇒ 빨간색 'o' 표식의 쇄선 모양

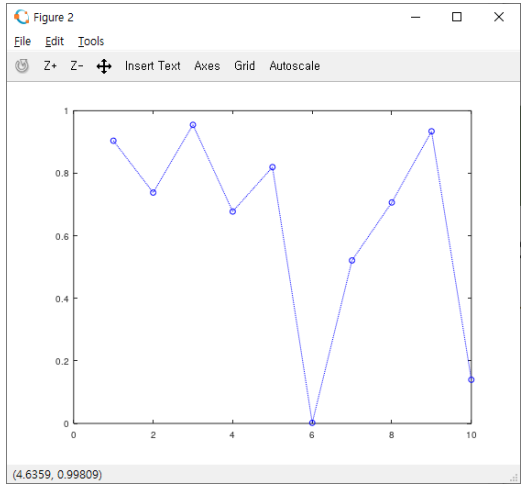
문자색상	표식모양	표식모양	선모양
Red	+	o	- (실선)
Green	*	.	--(쇄선)
Blue	x	^ △	-.(1점쇄선)
Magenta	v ▽	> ▷	: (점선)
Cyan	< ◁	s □	
black	d ◇	p ☆	
White	h (6점별)		

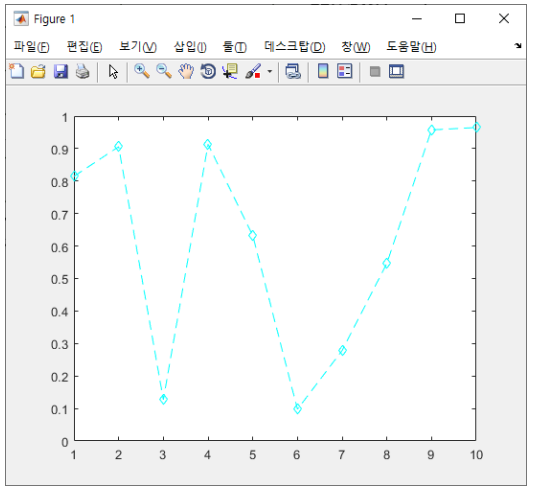
2차원 그래프

- plot() 함수의 형식 설정

```
>> x=1:10;
>> y=rand(1,10);
>>
>> plot(x,y);
>> plot(x,y,'r');
>> plot(x,y,'bo');
>> plot(x,y,'bo:');
>> plot(x,y,'cd--');
>> plot(x,y,'-mo');
```

문자색상	표식모양	표식모양	선모양
Red	+	o	- (실선)
Green	*	.	--(쇄선)
Blue	x	^ △	-.(1점쇄선)
Magenta	v ▽	> ▷	: (점선)
Cyan	< ◁	s □	
black	d ◇	p ☆	
White	h (6점별)		





2차원 그래프

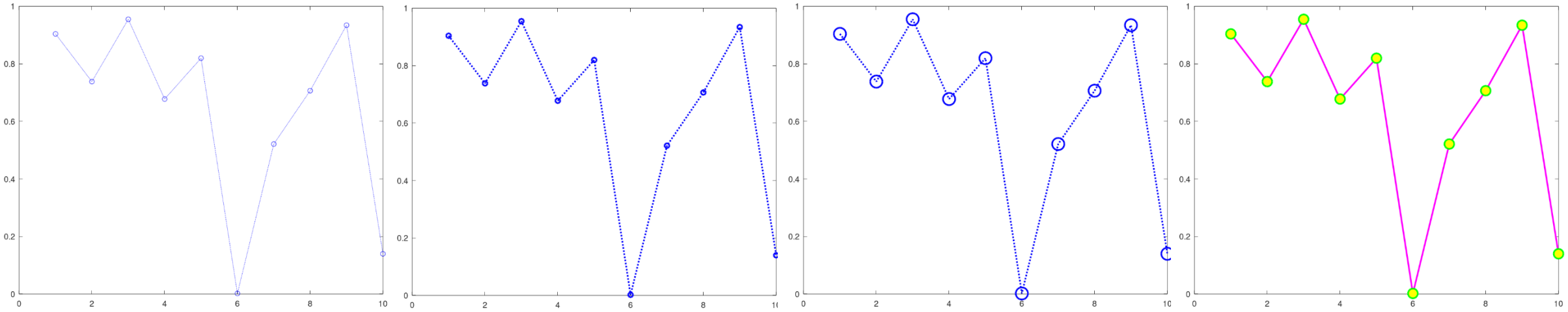
- plot() 함수의 속성 설정
  - 속성(property) : 그래프의 다양한 속성들을 지정할 수 있음
  - 속성 + 속성 값의 형식으로 인수들을 입력
    - ➢ 예] `plot(x, y, 'LineWidth', 2);` ⇒ 선의 굵기를 2로 지정 (기본 : 0.5)

Property	설명	속성값
LineWidth	선의 굵기	포인트 단위(0.5)
MarkerSize	표식의 크기	포인트 단위
MarkerEdgeColor	표식의 색 (테두리)	문자열 표시 (rgb..)
MarkerFaceColor	표식의 배경색	문자열 표시 (rgb..)

2차원 그래프

- plot() 함수의 속성 설정

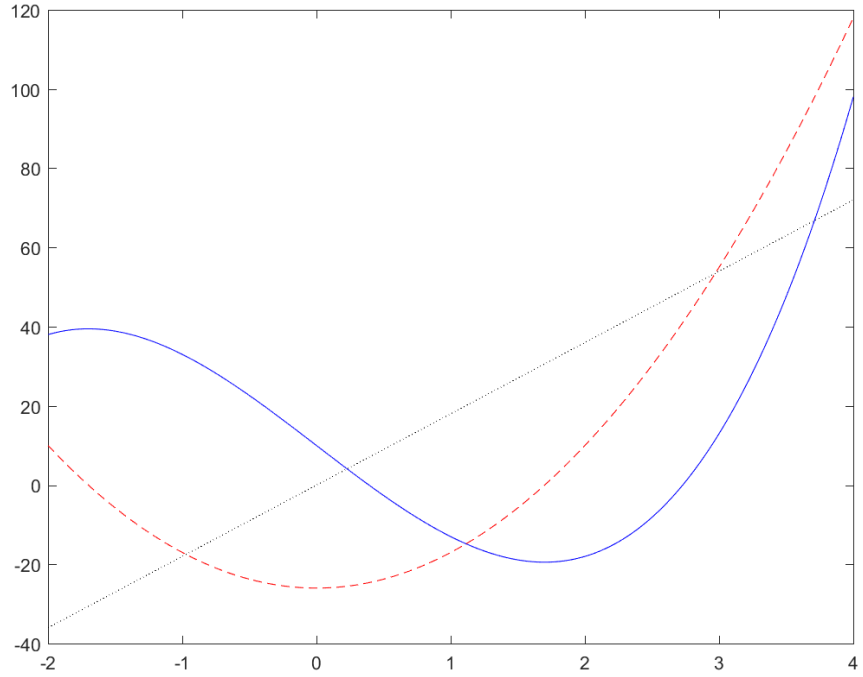
```
>> plot(x, y, 'bo:');
>> plot(x, y, 'bo:', 'LineWidth', 2);
>> plot(x, y, 'bo:', 'LineWidth', 2, 'MarkerSize', 5);
>> plot(x, y, 'bo:', 'LineWidth', 2, 'MarkerSize', 15);
>> plot(x, y, '-mo', 'LineWidth',2,'MarkerSize',12, 'MarkerEdgeColor','g',
'MarkerFaceColor','y')
```



2차원 그래프

- 2개 이상의 그래프 그리기
  - plot(x1, y1, x2, y2);
  - plot(x1, y1, s1, x2, y2, s2);
  - $y = 3x^3 - 26x + 10$의 함수와 y', y''의 함수를 같이 그려라.
    - $y' = 9x^2 - 26,\ y'' = 18x$

```
>> x = -2:0.01:4;
>> y = 3 * x.^3 - 26 * x + 10;
>> y1 = 9 * x.^2 - 26;
>> y2 = 18 * x;
>> plot(x, y, x, y1);
>> plot(x, y, '--', x, y1, 'g:');
>> plot(x, y, '--', x, y1, 'm:');
>> plot(x, y, 'b-', x, y1, 'r--', x, y2, 'k:');
```



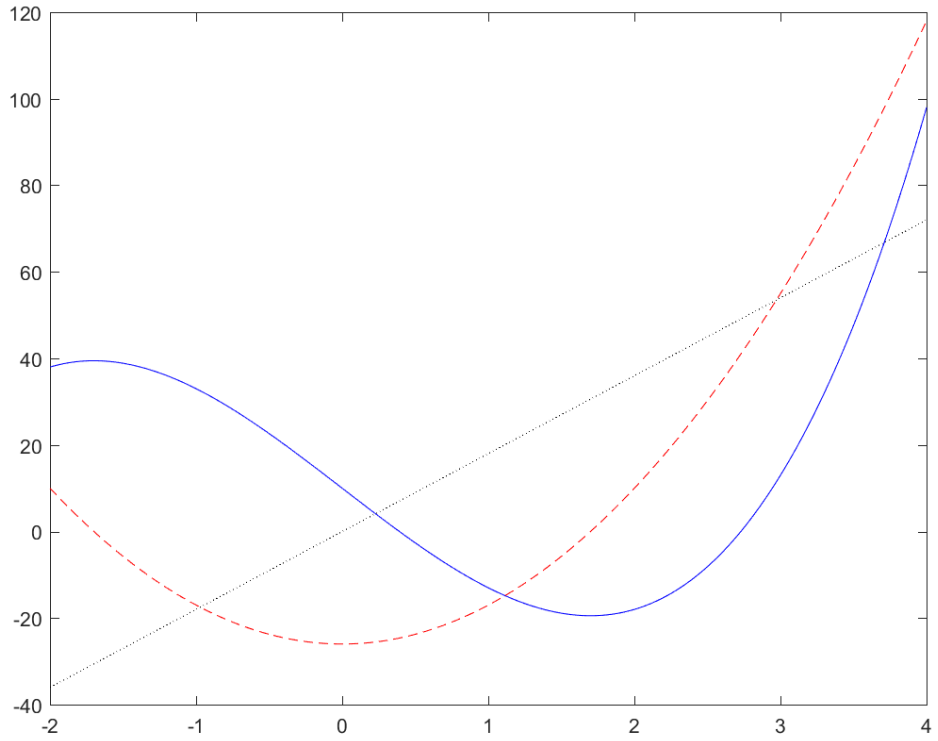
2차원 그래프

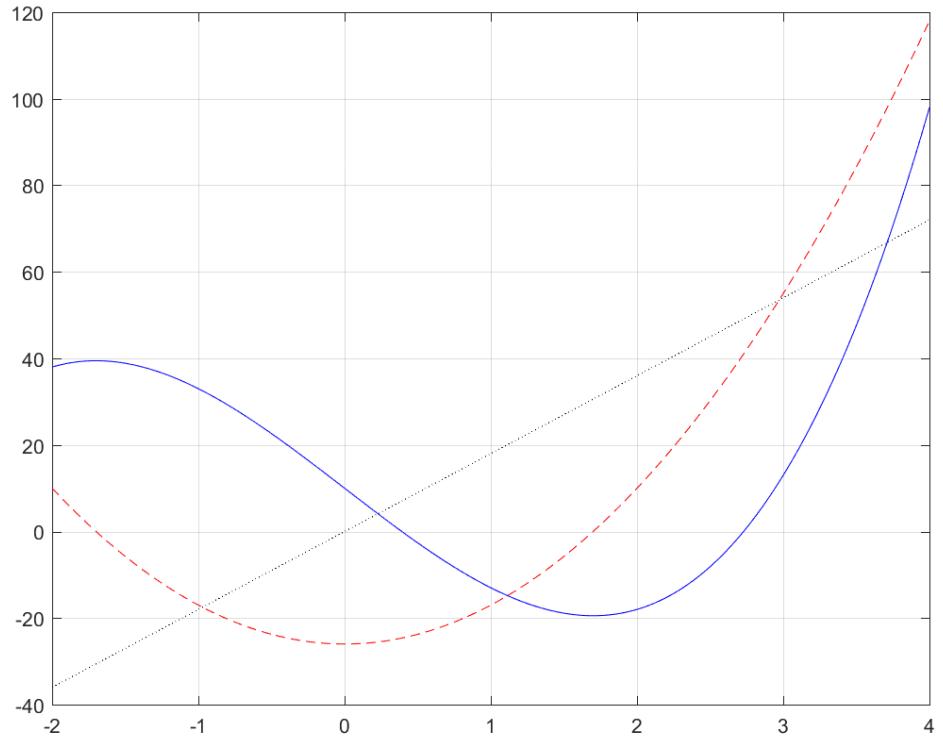
- 2개 이상의 그래프 그리기
  - hold 명령어 (hold on, hold off, hold : 토글방식)
    - ➢ 여러 개의 그래프를 한 그림에 나타낼 때 사용
    - ➢ hold on이 되어 있는 경우에 새로운 그래프를 그려도 이전의 그래프가 남아있음
    - ➢ hold off 되어 있는 경우, 새로운 그래프를 그리면 이전의 그래프는 지워짐

```
>> x = -2:0.01:4;
>> y = 3 * x.^3 - 26 * x + 10;
>> y1 = 9 * x.^2 - 26;
>> y2 = 18 * x;
>> plot(x, y, x, y1);
>> plot(x, y, '--', x, y1, 'g:');

>> plot(x, y, '--');
>> hold on;
>> plot(x, y1, 'g:');
```

2차원 그래프

- grid : 격자선 출력 (grid on, grid off, grid)

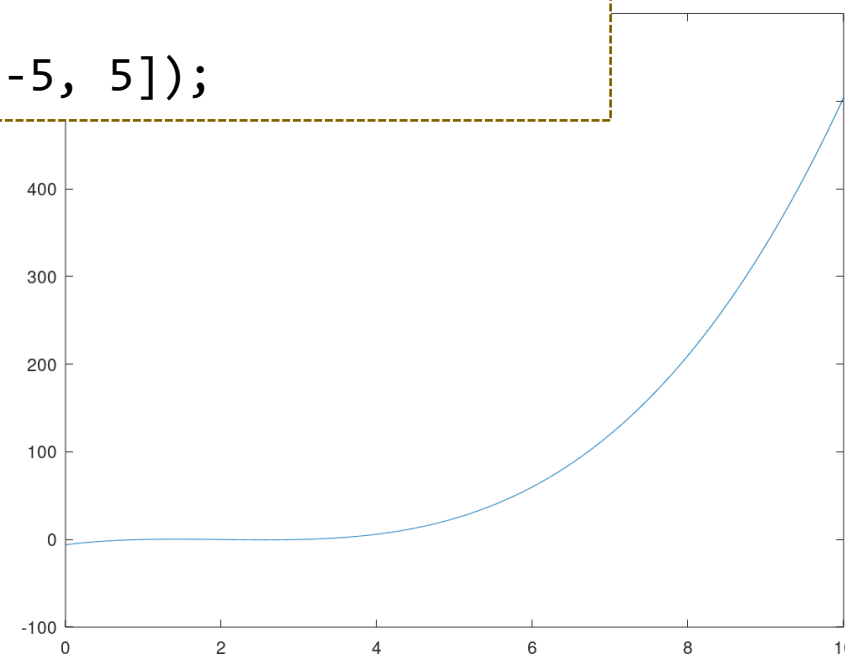


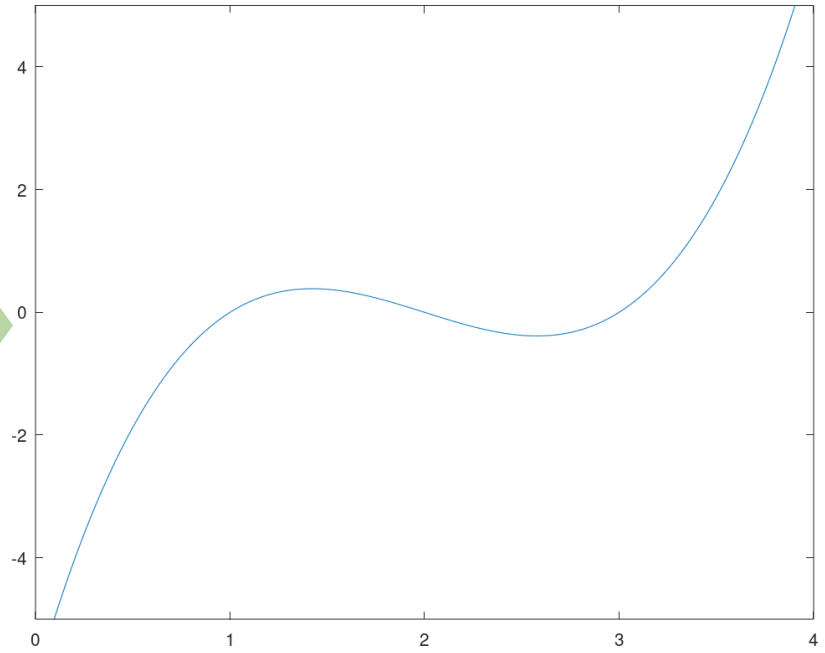


2차원 그래프

- axis 함수
  - 축의 범위와 모양을 변경할 때 사용
  - axis([xmin xmax ymin ymax]); ⇒ 그래프의 범위를 재지정

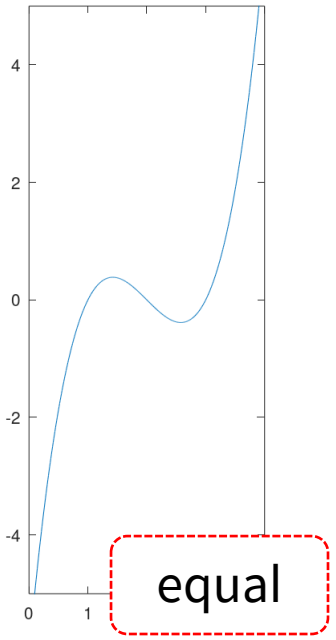
```
>> x = 0:0.01:10;
>> y = x.^3 - 6 * x.^2 + 11 * x - 6;
>> plot(x, y);
>> axis([0, 4, -5, 5]);
```

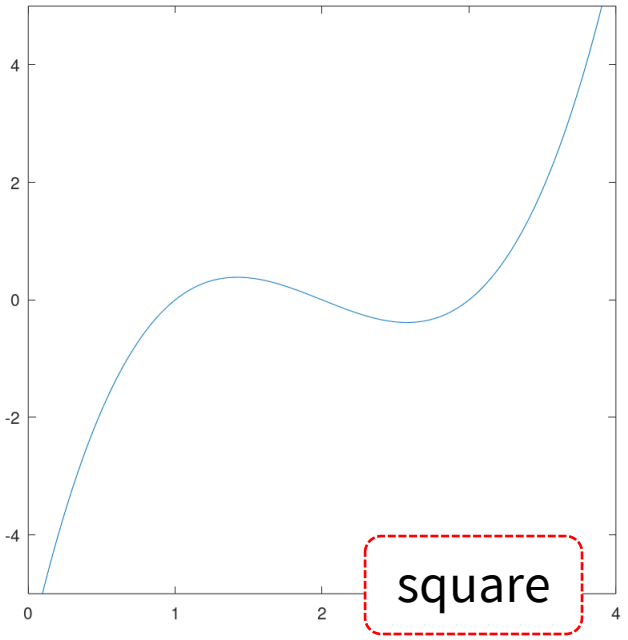


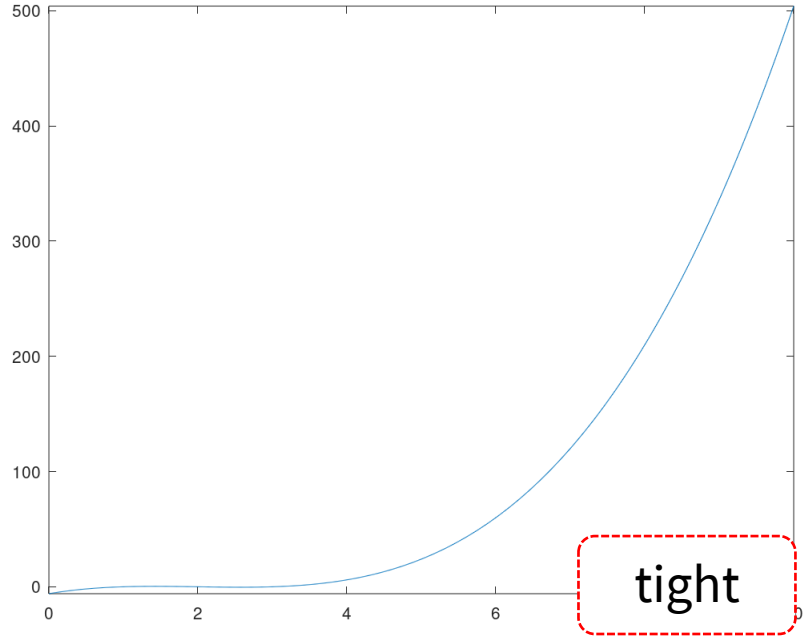


2차원 그래프

- axis 함수
  - axis equal : x축과 y축의 배율을 동일하게 설정
  - axis square : 좌표축의 영역을 정사각형으로 지정
  - axis tight : 좌표축의 범위를 데이터가 있는 영역으로 지정 (여백, 잘림 없음)







그래프 저장

- print() 함수 : 이미지 파일로 저장
  - `print -d<형식> <파일이름>`
  - `print –djpeg ex.jpg` ⇒ jpg 파일 형식으로 저장
  - `print –dpng ex.png` ⇒ png 파일 형식으로 저장
  - jpg 파일 : 손실 저장, 사진 등에 권장
  - png 파일 : 무손실 저장, 그래프 도표 등에 권장
  - bmp 파일 : 무손실, 무압축, 권장하지 않음

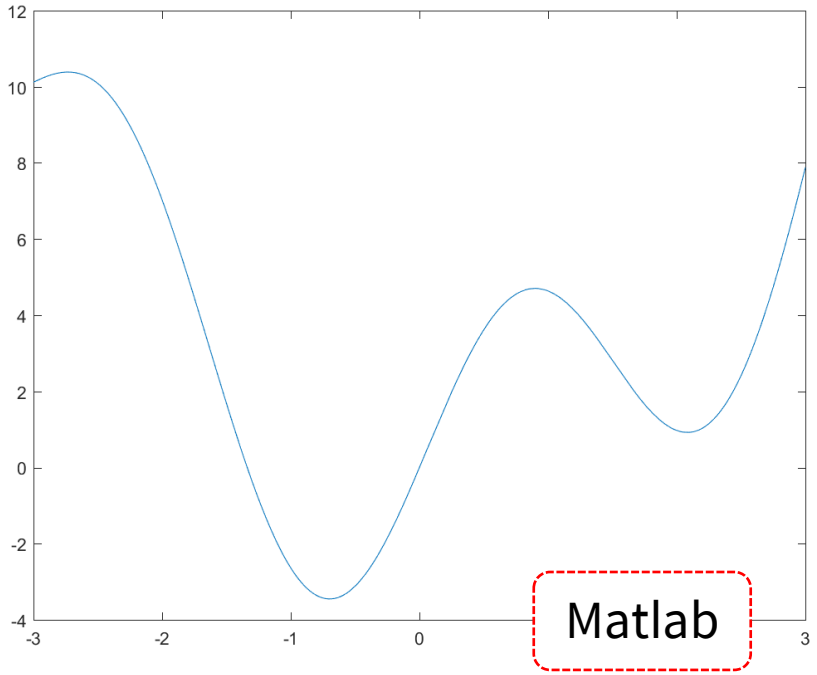
- Matlab 함수의 인수 처리 방식
  - 함수명 인수1 인수2 … ⇒ 함수명('인수1', '인수2', …)
    - ➢ 공백 후의 문장은 단어 단위로 문자열로 취급하여 인수로 변환
    - ➢ `print –dpng ex.png` ⇒ `print('-dpng', 'ex.png')`

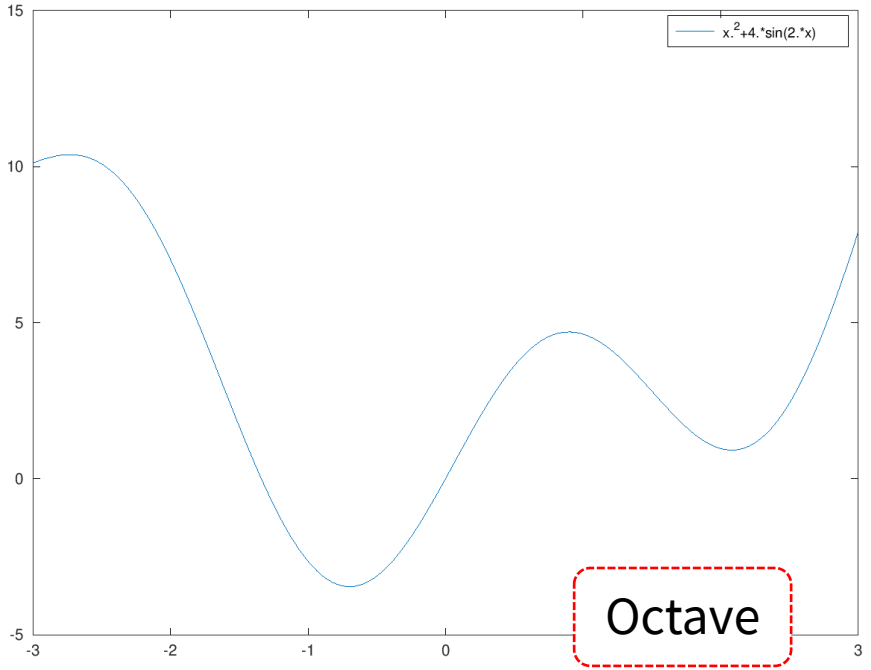
그래프 저장

- print() 함수 : 이미지 파일로 저장
  - `print -d<형식> <파일이름>`
  - 가능 형식
    - Matlab
      - ✓ Windows 전용 : win, winc, meta, bitmap
      - ✓ 범용 : ps, psc, ps2, psc2, eps, epsc, eps2, epsc2, pdf svg, jpg, jpeg<nn>, tiff, tiffnocompression, png, bmpmono, bmp256, bmp16m, pcxmono, pcx16, pcx256, pcx24b, bpm, bpmraw, pgm, pgmraw, ppm, ppmraw
    - Octave
      - ✓ pdf, pdfcrop, ps, ps2, psc, psc2, eps, eps2, epsc, epsc2, pslatex, epslatex, pdflatex, pslatexstandalone, epslatexstandalone, pdflatexstandalone, epscairo*, pdfcario*, epscairolatex*, pdfcairolatex*, epscairolatexstandalone*, pdfcairolatexstandalone*, svg, canvas*, cdr*, corel*, cgm*, dxf, emf, meta, fig, hpgl, ill, aifm, latex*, eepic*, mf*, tokz, tokzstandalone, png, jpg, jpeg, tif, tiff, tiffn, gif, pbm, dumb*

함수 그리기

- fplot : 주어진 함수를 주어진 정의역에서 그리기
  - fplot('함수', [정의역의 범위]);
  - `fplot('x^2+4*sin(2*x)', [-3 3])`
  - `print -dpng ex.png`





그래프 꾸미기

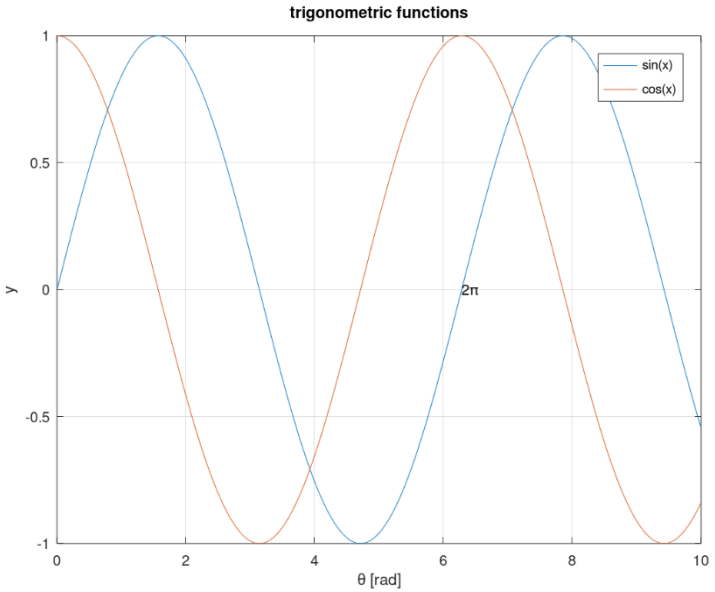
- 그래프 꾸미기
  - title(‘제목’)
  - xlabel(‘x 축 제목’)
  - ylabel(‘y 축 제목’)
  - legend(‘제목1’, ‘제목2’, ‘제목3’, …) : 범례
  - text(x, y, ‘그래프 위의 문자열’)

```
plot(x, ys, x, yc);
grid on;
title('trigonometric functions');
xlabel('\theta [rad]');
ylabel('y');
legend('sin(x)', 'cos(x)');
text(2 * pi, 0, '2\pi');
```

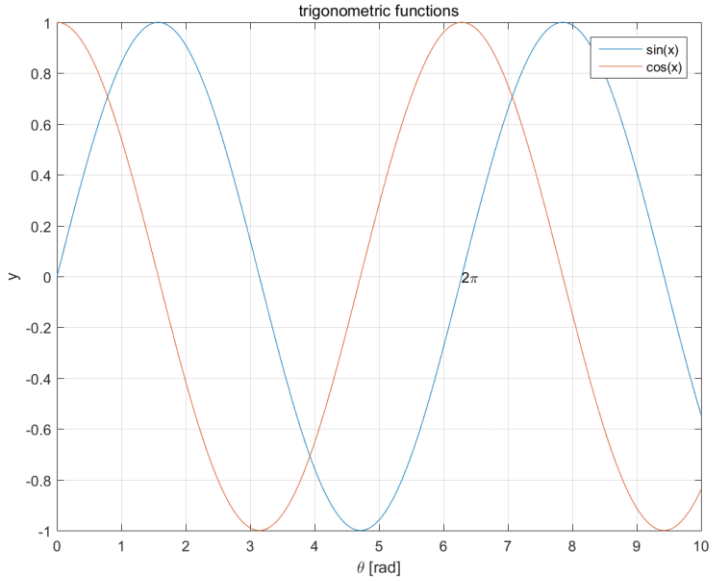
그래프 꾸미기

- 꾸미기 예제

```
x = 0:0.01:10;
ys = sin(x);
yc = cos(x);

plot(x, ys, x, yc);
grid on;
xlabel('\theta [rad]');
ylabel('y');
title('trigonometric functions');
legend('sin(x)', 'cos(x)');
text(2 * pi, 0, '2\pi');
print -dpng sincos.png
```



Octave



Matlab

그래프 꾸미기

- 텍스트 형식 (LaTeX-style)
  - title, xlabel, ylabel, legend, text 등에서 텍스트 내부에 사용되는 형식지정
  - 그리스 문자, 대문자로 시작하는 경우 그리스어 대문자 (예: \Gamma ⇒ Γ)
  - 함수의 추가 인수로 속성(property)를 지정
    - `text(2*pi, 0, '2\pi', 'fontsize', 20)`

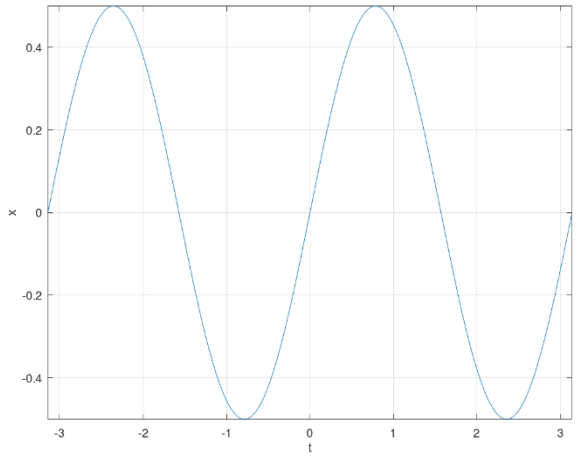
형식지정	내용
\bf	bold face
\it	Italic
\rm	Roman
_	Subscript
^	Superscript
속성	내용
fontname	폰트 설정
fontsize	글자 크기
color	글자 색상

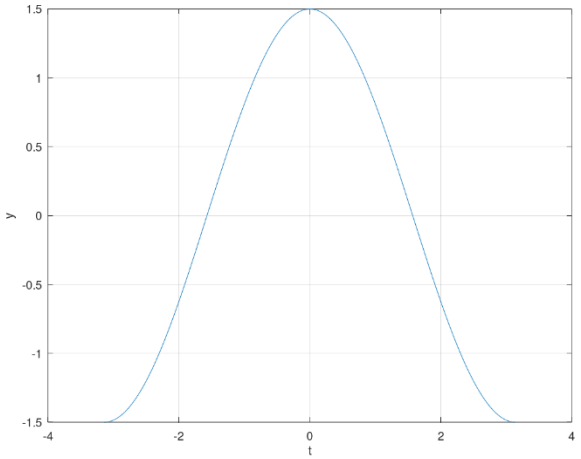
형식지정	내용	형식지정	내용	형식지정	내용
\alpha	α	\vartheta	ϑ	\sigma	σ
\beta	β	\iota	ι	\varsigma	ς
\gamma	γ	\kappa	κ	\tau	τ
\delta	δ	\mu	μ	\upsilon	υ
\epsilon	ϵ	\nu	ν	\chi	χ
\zeta	ζ	\xi	ξ	\psi	ψ
\eta	η	\pi	π	\omega	ω
\theta	θ	\rho	ρ	\varphi	φ
\varpi	ϖ	\lambda	λ	\varepsilon	ε

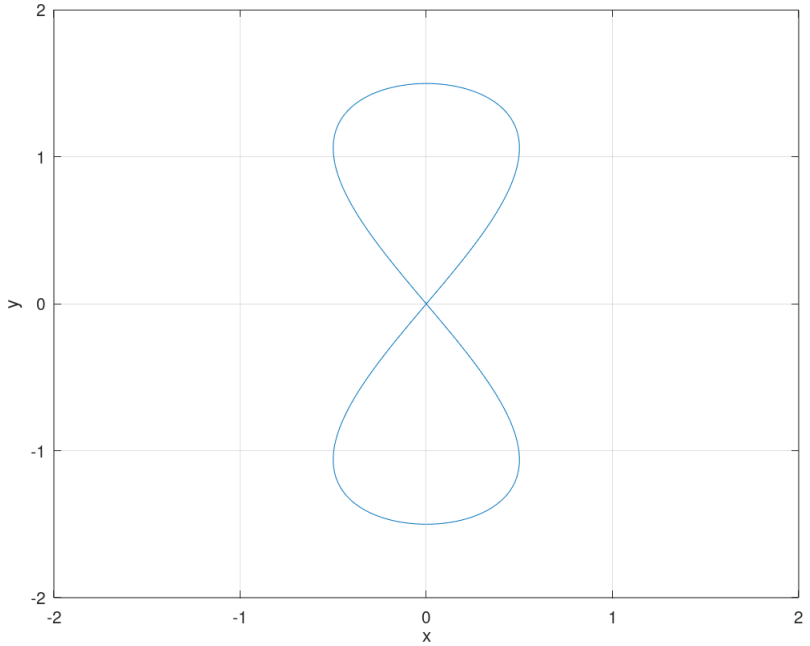
매개변수 방정식

- x와 y가 다른 변수의 함수로 구성되는 경우
  예] $x = \sin t \cdot \cos t\,,\ y = 1.5\cos t\,,\ -\pi \le t \le \pi$

```
>> figure;
>> plot(t, x);
>> xlabel('t');
>> ylabel('x');
>> axis tight;

>> figure;
>> plot(t, y);
>> xlabel('t');
>> ylabel('y');

>> figure;
>> plot(x, y);
>> xlabel('x');
>> ylabel('y');
>> axis([-2 2 -2 2]);
>> grid;
```

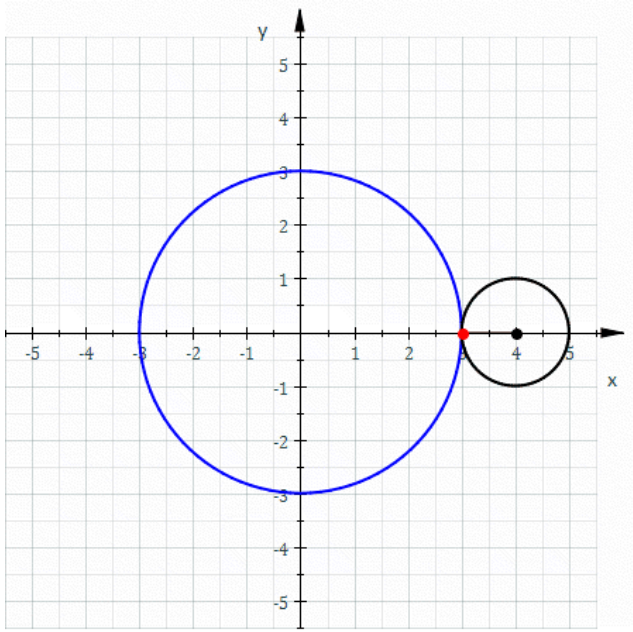






매개변수 방정식의 예

- 에피싸이클로이드(Epicycloid)
  - 서로 외접하는 원이 있을 때 외부의 원이 내부의 원과 접하면서 구르는 경우, 외부 원의 정점의 궤적
  - 참고 : https://suhak.tistory.com/687



- 사이클로이드(Cycloid)
  - 직선 위로 원을 굴렸을 때 원의 정점이 그리는 곡선
  - 참고 : https://ko.wikipedia.org/wiki/사이클로이드

매개변수 방정식의 예

- 에피싸이클로이드(Epicycloid)
  - 참고 : https://suhak.tistory.com/687

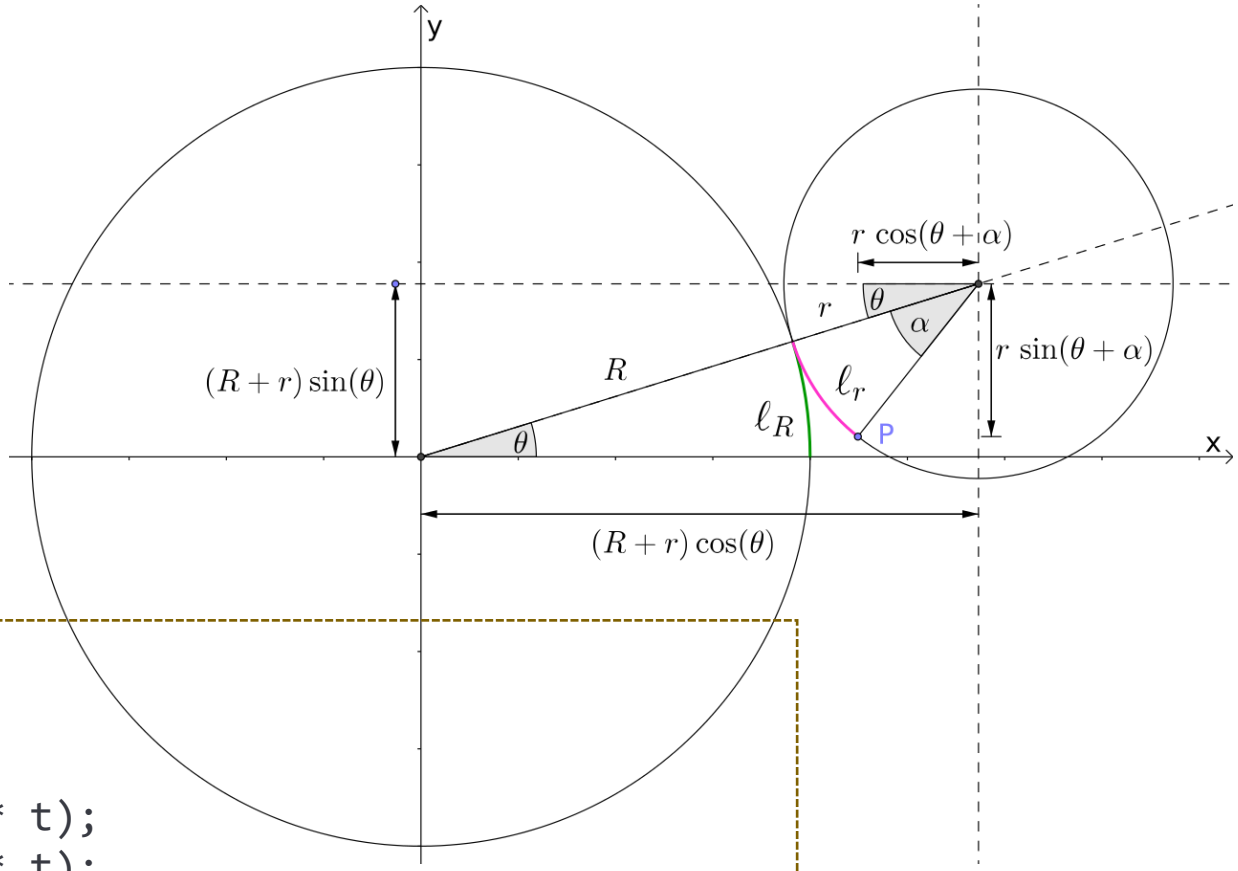
➢ $l_R = R\theta = r\alpha = l_r,\ \alpha = \frac{R\theta}{r}$

✓ $x(\theta) = (R+r)\cos\theta - r\cos(\theta+\alpha)$

$= (R+r)\cos\theta - r\cos\left(\frac{R+r}{r}\theta\right)$

✓ $y(\theta) = (R+r)\sin\theta - r\sin(\theta+\alpha)$

$= (R+r)\sin\theta - r\sin\left(\frac{R+r}{r}\theta\right)$



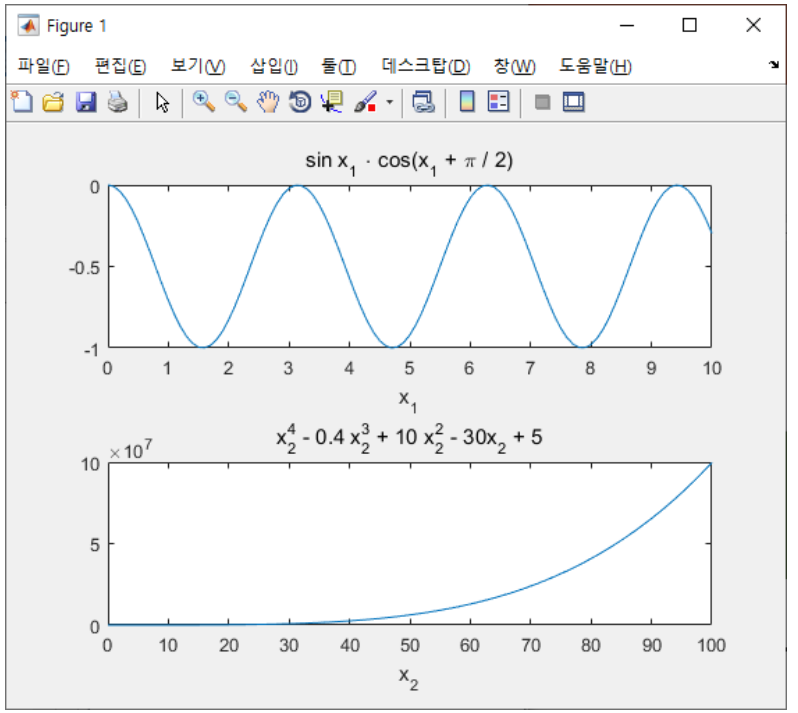

```
R = 11;
r = 2;
t = 0:0.01:4*pi;
x = (R + r) * cos(t) - r * cos((R + r) / r * t);
y = (R + r) * sin(t) - r * sin((R + r) / r * t);

plot(x, y);
axis equal;
grid;
title(sprintf('Epicycloid, {\\it R} = %g, {\\it r} = %g', R, r));
```

2차원 그래프

- subplot : 여러 개의 그래프를 하나의 윈도우에 그리기

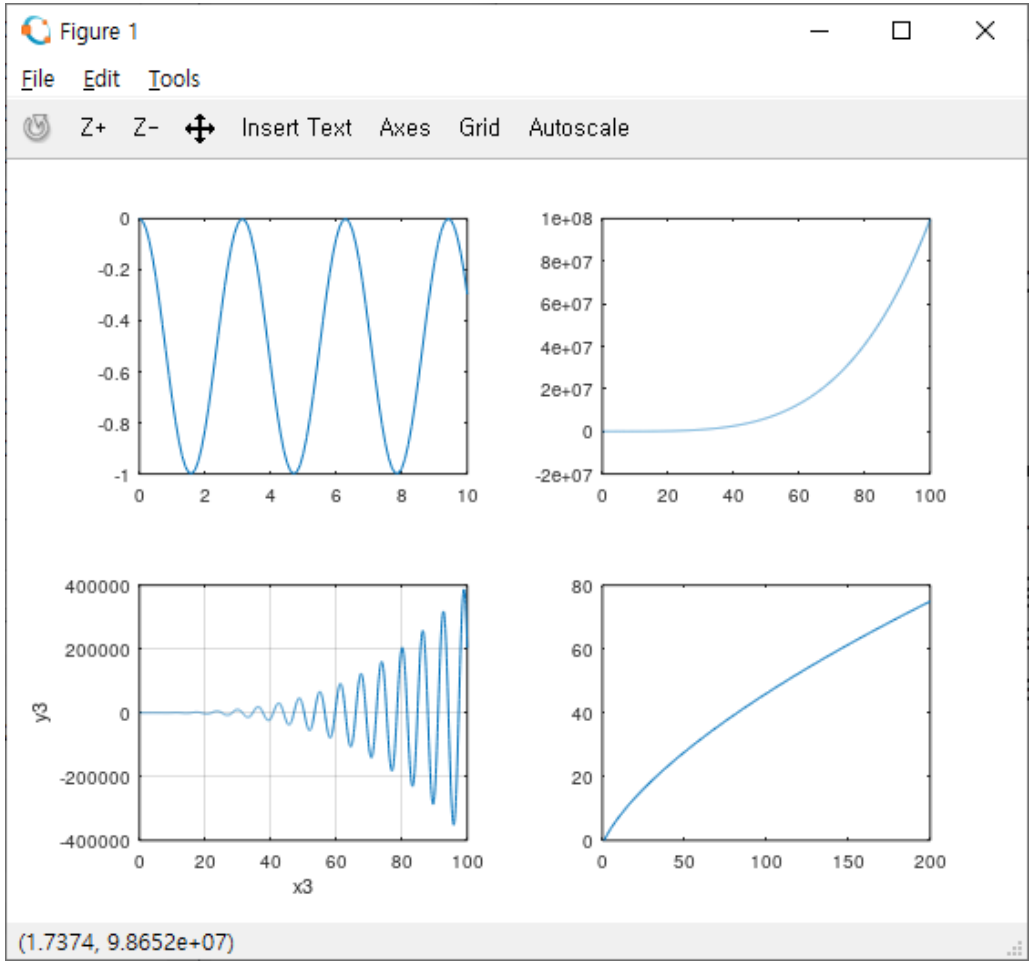
Form	Meaning
subplot(m, n, i)	Breaks the Figure window into an m X n matrix of small axis and selects the i'th axis for the current plot
subplot(mni)	

```
x1 = 0:0.1:10;
f1 = sin(x1) .* cos(x1 + pi / 2);
subplot(221);
plot(x1, f1);
xlabel('x_1');
title('sin x_1 \cdot cos(x_1 + \pi / 2)');

x2 = 0:0.1:100;
f2 = x2.^4 - 0.4*x2.^3 + 10*x2.^2 - 30*x2 + 5;
subplot(222);
plot(x2, f2);
xlabel('x_2');
title('x_2^4 - 0.4 x_2^3 + 10 x_2^2 - 30x_2 + 5');
```



2차원 그래프

```
>> x1 = 0:0.1:10;
>> f1 = sin(x1) .* cos(x1 + pi / 2);
>> subplot(221);
>> plot(x1, f1);

>> x2 = 0:0.1:100;
>> f2 = x2.^4 - 0.4*x2.^3 + 10*x2.^2 - 30*x2 + 5;
>> subplot(222);
>> plot(x2, f2);

>> x3 = 0:0.1:100;
>> f3 = -(0.4*x3.^3).*sin(x3);
>> subplot(223);
>> plot(x3, f3);
>> grid on;
>> xlabel('x3');
>> ylabel('y3');

>> x4 = 1:1:200;
>> f4 = sqrt(x4).*log(x4);
>> subplot(224);
>> plot(x4, f4);
```



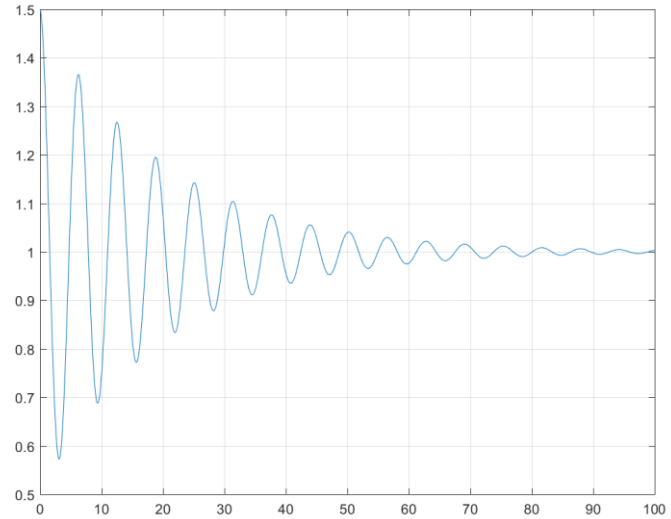
로그 축 및 기타 형식의 그래프

Function	Description
Plot(x1, y1, s1, x2, y2, s2, …)	Combines the plots defined by the (x, y, s) triples
loglog(x1, y1, s1, x2, y2, s2, …)	loglog(…) is the same as plot(…) except a logarithmic (base 10) scale is used for the X-axis and the Y-axis
semilogx(x1, y1, s1, x2, y2, s2, …)	semilogx(…) is the same as plot(…) except a logarithmic (base 10) scale is used for the X-axis
semilogy(x1, y1, s1, x2, y2, s2, …)	semilogy(…) is the same as plot(…) except a logarithmic (base 10) scale is used for the Y-axis
plotyy(x1, y1, x2, y2)	Plots Y1 versus X1 with y-axis labeling on the left and plots Y2 versus X2 with y-axis labeling on the right

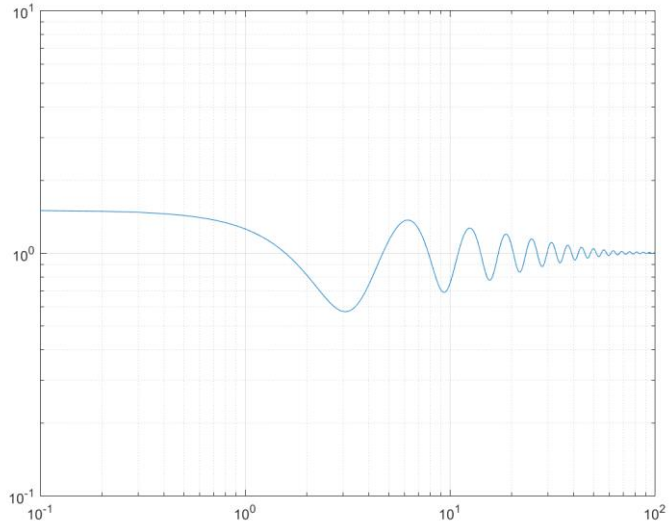
로그 축 및 기타 형식

- $x = 1 + \frac{1}{2}e^{-0.05t}\sin\left(t + \frac{\pi}{2}\right),\ 0 \le t \le 100,\ 0.1 \le x \le 10$

```
>> t=0:0.1:100;
>> x=1 + 0.5*exp(-0.05*t) .* sin(t + pi/2);
>> plot(t, x);
>> grid on
```



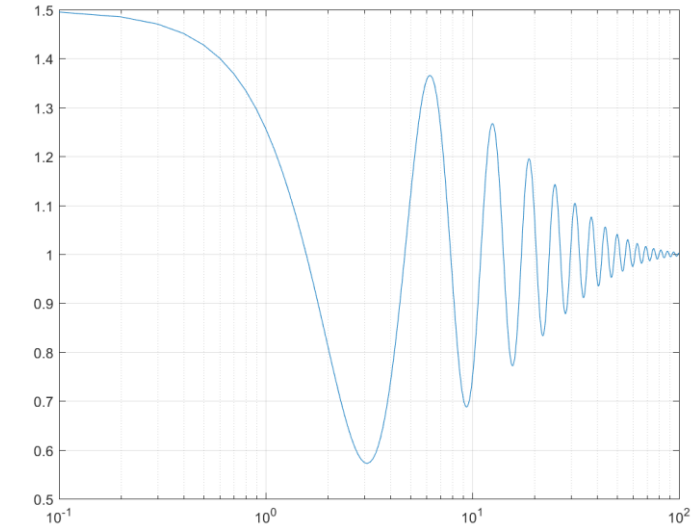

```
>> loglog(t, x);
>> axis([0, 100, 0.1, 10]);
>> grid on
```



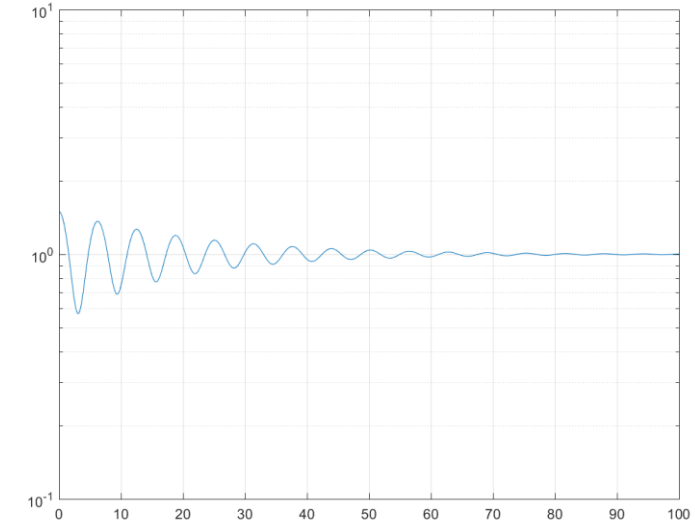
로그 축 및 기타 형식

- $x = 1 + \frac{1}{2}e^{-0.05t}\sin\left(t + \frac{\pi}{2}\right),\ 0 \le t \le 100,\ 0.1 \le x \le 10$

```
>> t=0:0.1:100;
>> x=1 + 0.5*exp(-0.05*t) .* sin(t + pi/2);
>> semilogx(t, x);
>> grid on
```



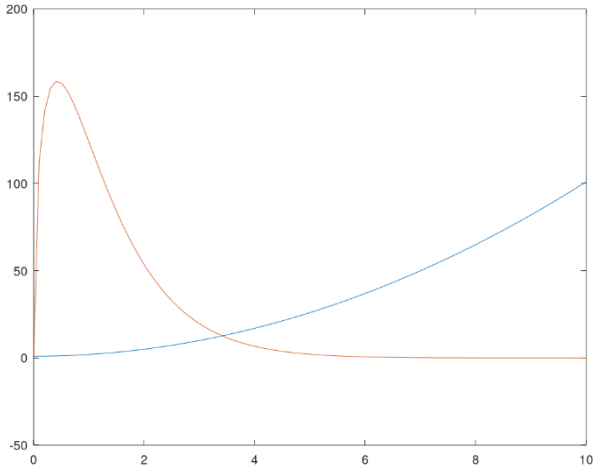

```
>> semilogy(t, x);
>> axis([0, 100, 0.1, 10]);
>> grid on
```



로그 축 및 기타 형식

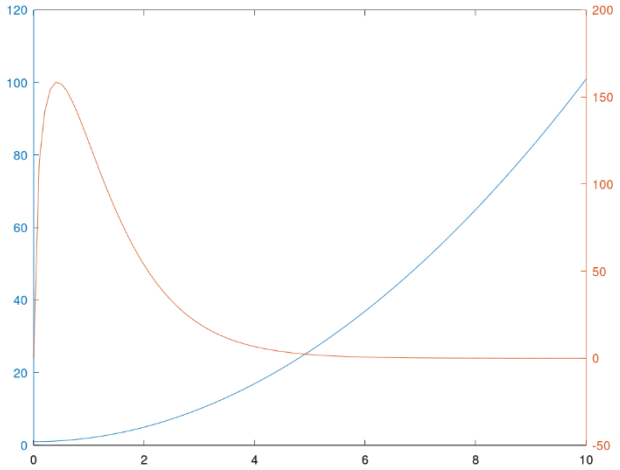
- $y = x^2 + 1, \quad y = 400\frac{\sin\sqrt{x}}{e^x}$, where $0 \le x \le 10$

```
>> x = 0:0.1:10;
>> y1 = x.^2 + 1;
>> y2 = 400 * sin(sqrt(x)) ./ exp(x);
>> plot(x, y1, x, y2);
```



- plotyy() : 두 개의 그래프, y축 별도

```
>> x = 0:0.1:10;
>> y1 = x.^2 + 1;
>> y2 = 400 * sin(sqrt(x)) ./ exp(x);
>> plotyy(x, y1, x, y2);
```



과제물

- 출석점검용 과제
  - 수업에 수행한 실습 화면 및 결과 화면을 캡쳐해서 제출하시오.

- 채점용 과제물
  - 2x2의 subplot을 만들어서, 함수 $x = 1 + \frac{1}{2}e^{-0.05t}\sin\left(t + \frac{\pi}{2}\right),\ 0 \le t \le 100,$ $0.1 \le x \le 10$의 그래프를 plot, semilogx, semilogy, loglog 함수를 사용하여 그리는 스크립트를 완성하라.
  - 그려진 그림은 print 함수를 이용하여 저장하여야 하며 (스크립트에 저장하는 부분이 포함되어야 함), 스크립트 파일 및 그래프의 그림 파일을 제출하도록 한다.

좌표 축 속성 설정하기

- set(gca, ‘property’, ‘value’, ‘property’, ‘value’, …)
  - set() 함수 : 주어진 개체(첫번째 인수)의 속성들을 설정
    - 개체 : 그래프 윈도우, 좌표축, 하나의 데이터 선(line) 등
    - F = gcf(); ⇒ 현재 그래프 윈도우 가져오기(get current figure)
    - A = gca(); ⇒ 현재 그래프 좌표축 가져오기 (get current axes) → 가장 빈번히 사용됨
    - L = plot(x, y); ⇒ 그래프에 선을 추가하고 그 개체를 가져오기
  - Get() 함수 : 현재의 속성을 가져 옴, 속성이 없는 경우 전체 속성 표시
    - 속성 참고
      - ✓ Axes 속성 : https://kr.mathworks.com/help/matlab/ref/matlab.graphics.axis.axes-properties.html
      - ✓ Figure 속성 : https://kr.mathworks.com/help/matlab/ref/matlab.ui.figure-properties.html
      - ✓ Line 속성 : https://kr.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.line-properties.html

좌표 축 속성 설정하기 (Matlab Axes 속성예)

```
>> get(gca)
                    ALim: [0 1]
                ALimMode: 'auto'
    ActivePositionProperty: 'outerposition'
         AmbientLightColor: [1 1 1]
              BeingDeleted: 'off'
                       Box: 'on'
                 BoxStyle: 'back'
               BusyAction: 'queue'
            ButtonDownFcn: ''
                     CLim: [0 1]
                CLimMode: 'auto'
           CameraPosition: [0.1638 -4.7007e-04 241.0890]
       CameraPositionMode: 'auto'
            CameraTarget: [0.1638 -4.7007e-04 0]
          CameraTargetMode: 'auto'
             CameraUpVector: [0 1 0]
         CameraUpVectorMode: 'auto'
            CameraViewAngle: 8.9714
       CameraViewAngleMode: 'auto'
                  Children: [1x1 Line]
                  Clipping: 'on'
             ClippingStyle: '3dbox'
                     Color: [1 1 1]
                ColorOrder: [7x3 double]
           ColorOrderIndex: 2
                 CreateFcn: ''
              CurrentPoint: [2x3 double]
          DataAspectRatio: [1 1 1]
       DataAspectRatioMode: 'manual'
                 DeleteFcn: ''
                 FontAngle: 'normal'
                  FontName: 'Helvetica'
                  FontSize: 10
              FontSmoothing: 'on'
                  FontUnits: 'points'
                 FontWeight: 'normal'
                  GridAlpha: 0.1500
              GridAlphaMode: 'auto'
                  GridColor: [0.1500 0.1500 0.1500]
              GridColorMode: 'auto'
              GridLineStyle: '-'
            HandleVisibility: 'on'
                    HitTest: 'on'
              Interruptible: 'on'
    LabelFontSizeMultiplier: 1.1000
                      Layer: 'bottom'
             LineStyleOrder: '-'
        LineStyleOrderIndex: 1
                  LineWidth: 0.5000
             MinorGridAlpha: 0.2500
         MinorGridAlphaMode: 'auto'
             MinorGridColor: [0.1000 0.1000 0.1000]
         MinorGridColorMode: 'auto'
         MinorGridLineStyle: ':'
                   NextPlot: 'replace'
              OuterPosition: [0 0 1 1]
                     Parent: [1x1 Figure]
             PickableParts: 'visible'
          PlotBoxAspectRatio: [434 342.3000 22.8200]
      PlotBoxAspectRatioMode: 'manual'
                   Position: [0.1300 0.1100 0.7750 0.8150]
                 Projection: 'orthographic'
                   Selected: 'off'
        SelectionHighlight: 'on'
                 SortMethod: 'childorder'
                        Tag: ''
                    TickDir: 'in'
                TickDirMode: 'auto'
       TickLabelInterpreter: 'tex'
                 TickLength: [0.0100 0.0250]
                 TightInset: [0.0435 0.0523 0 0.0555]
                      Title: [1x1 Text]
    TitleFontSizeMultiplier: 1.1000
           TitleFontWeight: 'normal'
                       Type: 'axes'
              UIContextMenu: [0x0 GraphicsPlaceholder]
                      Units: 'normalized'
                   UserData: []
                       View: [0 90]
                    Visible: 'on'
                      XAxis: [1x1 NumericRuler]
              XAxisLocation: 'bottom'
                     XColor: [0.1500 0.1500 0.1500]
                XColorMode: 'auto'
                       XDir: 'normal'
                      XGrid: 'on'
                     XLabel: [1x1 Text]
                      XLim: [-18.7498 19.0773]
                  XLimMode: 'auto'
                 XMinorGrid: 'off'
                 XMinorTick: 'off'
                     XScale: 'linear'
                      XTick: [-15 -10 -5 0 5 10 15]
                 XTickLabel: {7x1 cell}
             XTickLabelMode: 'auto'
          XTickLabelRotation: 0
                 XTickMode: 'auto'
                     YAxis: [1x1 NumericRuler]
            YAxisLocation: 'left'
                     YColor: [0.1500 0.1500 0.1500]
                 YColorMode: 'auto'
                       YDir: 'normal'
                      YGrid: 'on'
                     YLabel: [1x1 Text]
                      YLim: [-14.9178 14.9169]
                  YLimMode: 'auto'
                 YMinorGrid: 'off'
                 YMinorTick: 'off'
                     YScale: 'linear'
                      YTick: [-10 -5 0 5 10]
                 YTickLabel: {5x1 cell}
             YTickLabelMode: 'auto'
          YTickLabelRotation: 0
                 YTickMode: 'auto'
                     ZAxis: [1x1 NumericRuler]
                     ZColor: [0.1500 0.1500 0.1500]
                 ZColorMode: 'auto'
                       ZDir: 'normal'
                      ZGrid: 'on'
                     ZLabel: [1x1 Text]
                      ZLim: [-0.9945 0.9945]
                  ZLimMode: 'auto'
                 ZMinorGrid: 'off'
                 ZMinorTick: 'off'
                     ZScale: 'linear'
                      ZTick: [-0.5000 0 0.5000]
                 ZTickLabel: ''
             ZTickLabelMode: 'auto'
          ZTickLabelRotation: 0
                 ZTickMode: 'auto'
>>
```

좌표 축 속성 설정하기 (대표적인 속성)

```
           CurrentPoint: [2x3 double]
        DataAspectRatio: [1 1 1]
    DataAspectRatioMode: 'manual'
              FontAngle: 'normal'
               FontName: 'Helvetica'
               FontSize: 10
          FontSmoothing: 'on'
              FontUnits: 'points'
             FontWeight: 'normal'
              GridAlpha: 0.1500
          GridAlphaMode: 'auto'
              GridColor: [0.1500 0.1500 0.1500]
          GridColorMode: 'auto'
          GridLineStyle: '-'
         LineStyleOrder: '-'
    LineStyleOrderIndex: 1
              LineWidth: 0.5000
         MinorGridAlpha: 0.2500
     MinorGridAlphaMode: 'auto'
         MinorGridColor: [0.1000 0.1000 0.1000]
                 Parent: [1x1 Figure]
               Position: [0.1300 0.1100 0.7750 0.8150]
```

대부분의 속성들은 설정할 필요가 없으며, 세밀한 설정이 필요한 경우에만 사용함. 자주 사용되는 속성들은 다양한 그래픽 함수들(grid, axis, xlabel, …)을 통해서 설정됨.

```
                TickDir: 'in'
            TickDirMode: 'auto'
   TickLabelInterpreter: 'tex'
             TickLength: [0.0100 0.0250]
                  Title: [1x1 Text]
TitleFontSizeMultiplier: 1.1000
        TitleFontWeight: 'normal'
                   Type: 'axes'
                  Units: 'normalized'
                   View: [0 90]
          XAxisLocation: 'bottom'
                 XColor: [0.1500 0.1500 0.1500]
             XColorMode: 'auto'
                   XDir: 'normal'
                  XGrid: 'on'
                 XLabel: [1x1 Text]
                   XLim: [-18.7498 19.0773]
              XLimMode: 'auto'
             XMinorGrid: 'off'
             XMinorTick: 'off'
                 XScale: 'linear'
                  XTick: [-15 -10 -5 0 5 10 15]
             XTickLabel: {7x1 cell}
         XTickLabelMode: 'auto'
     XTickLabelRotation: 0
              XTickMode: 'auto'
```

좌표 축 속성 설정하기

- 좌표 축 속성 변경의 예

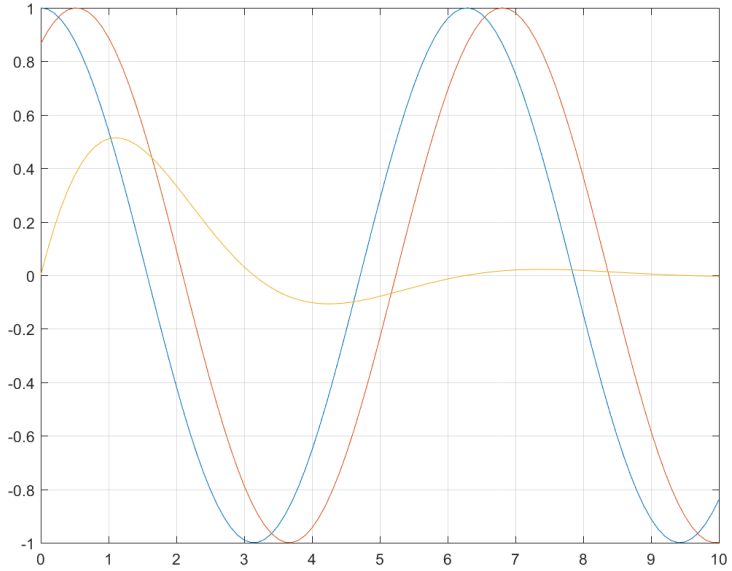
```
>> set(gca, 'FontSize', 20);
>> set(gca, 'GridLineStyle', '--');
```

- 선의 속성 변경의 예

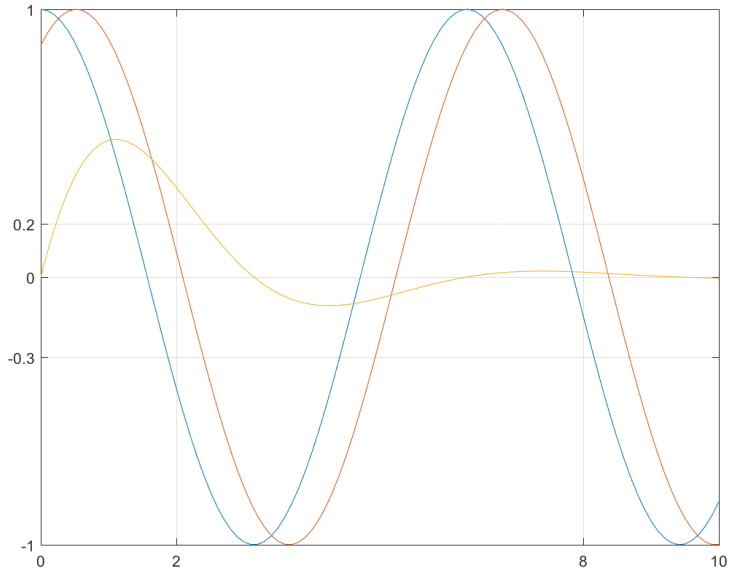
```
>> set(plot(x, y), 'LineStyle', '--');
```

좌표 축 속성 설정하기

```
>> x = 0:0.1:10;
>> y1 = cos(x);
>> y2 = cos(x - pi / 6);
>> y3 = exp(-0.5*x) .* sin(x);
>> plot(x, y1, x, y2, x, y3);
>> grid on
```




```
>> xt = [0, 2, 4, 6, 8, 10];
>> yt = [-1, -0.5, 0, 0.5, 1];
>> set(gca, 'xtick', xt);
>> set(gca, 'ytick', yt);
```



극좌표계 그래프

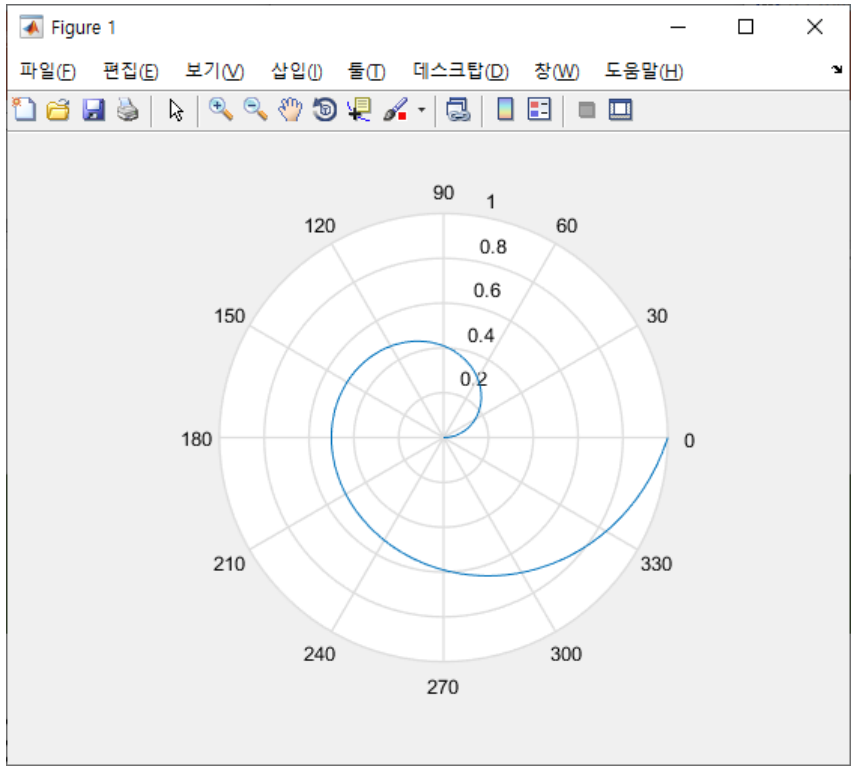
Function	Description
polar(theta, r, s)	Makes a plot using polar coordinates of the angle ‘theta’, in radians, versus the radius ‘r’
[theta, r] = cart2pol(x, y)	Transforms the corresponding elements of data stored in Cartesian coordinates x, y to polar coordinates angle ‘theta’, radius ‘r’
[x, y] = pol2cart(theta, r)	Transforms the corresponding elements of data stored in polar coordinates ‘theta’, radius ‘r’ to Cartesian coordinates x, y

극 좌표계 그래프

- $r = \frac{\sin\theta + \theta}{2\pi}, \quad 0 \le \theta \le 2\pi$

```
>> theta = 0:(2*pi)/100:2*pi;
>> r = (sin(theta) + theta) / (2*pi);
>> polar(theta, r);
```

```
>> [x, y] = pol2cart(theta, r);
>> plot(x, y)
>> grid
>> axis equal
>> axis([-1 1 -1 1]);
```



과제

- 출석점검용 과제
  - 수업에 수행한 실습 화면 및 결과 화면을 캡쳐해서 제출하시오.

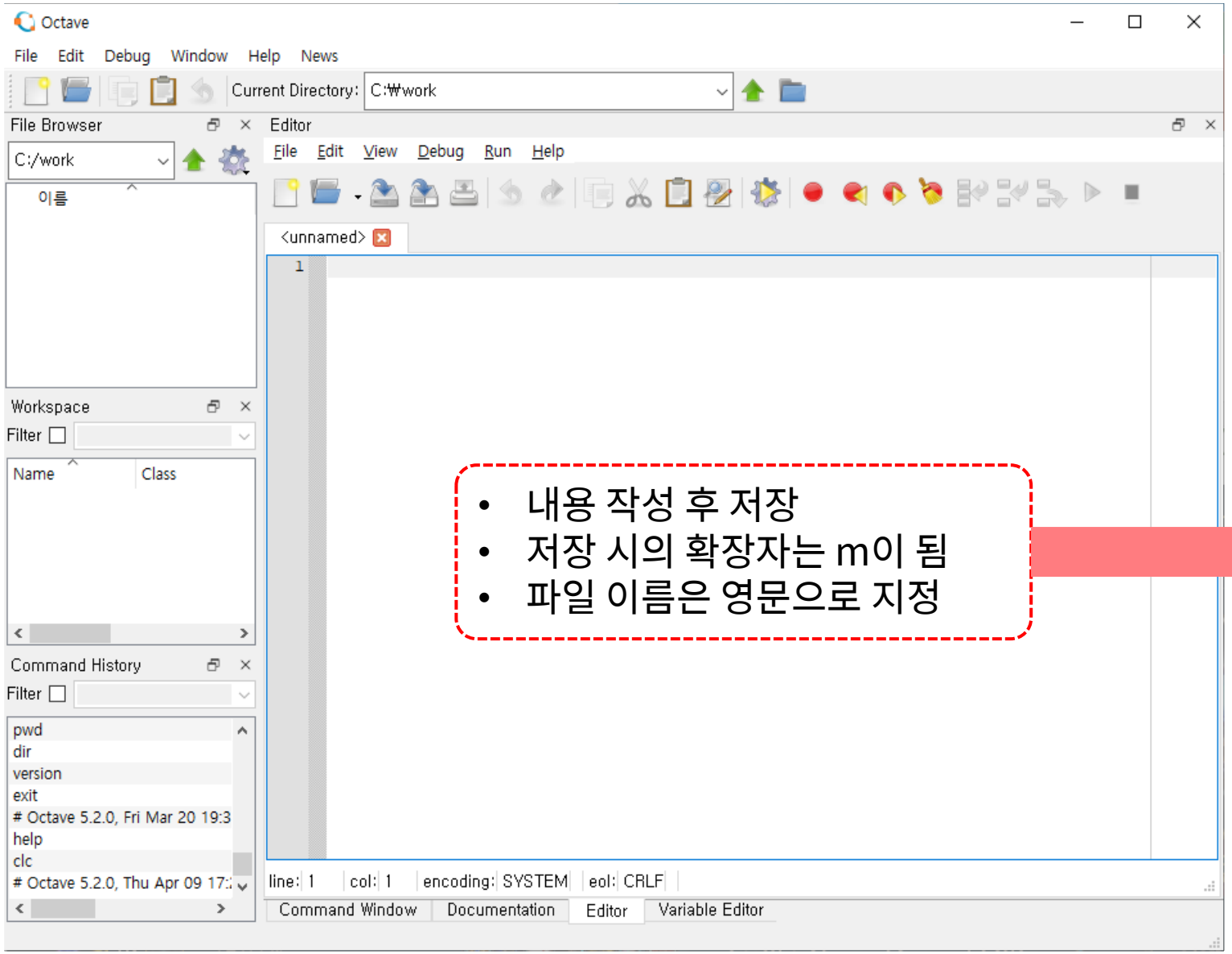
- 채점용 과제
  - 중심이 $x = 3, y = 2$ 이며 주축이 $a = 8, b = 6$인 타원의 그래프를 그려라.
  - 다음의 매개변수 방정식의 그래프를 극좌표계로 그려라.
    - $r(t) = 25 + 30[1 - e^{0.07t}], \quad \theta(t) = 2\pi(1 - e^{-0.2t}), \quad 0 \le t \le 20$

M-file 프로그래밍, 조건 연산자

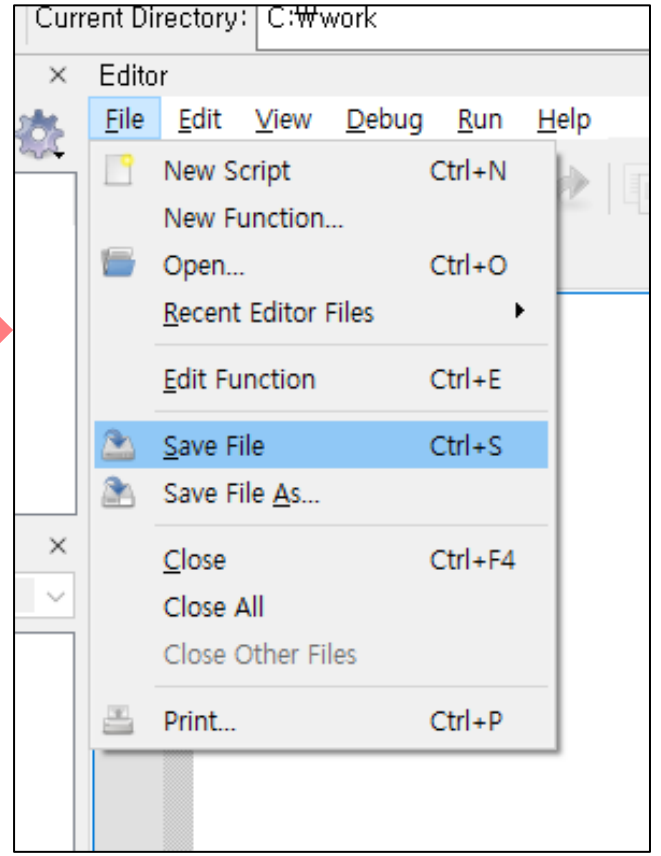
M 파일의 작성

- M파일 : Matlab에서 사용하는 프로그램 저장용 파일
  - 확장자가 'm'으로 되어 있음
  - 스크립트(script) 파일과 함수(function) 파일로 나누어짐
- 스크립트 파일
  - 명령어들을 미리 모아놓은 파일
  - 명령창(command window)에서 호출 → 저장된 명령어들을 순차적으로 실행
- 함수 파일
  - 함수의 형식으로 저장 → 함수를 사용하는 방법으로 호출 (입력변수 → 출력값)
- 새파일 작성
  - 메뉴에서 [File → New → New Script]를 선택하여 실행

M 파일의 작성



- 내용 작성 후 저장
- 저장 시의 확장자는 m이 됨
- 파일 이름은 영문으로 지정



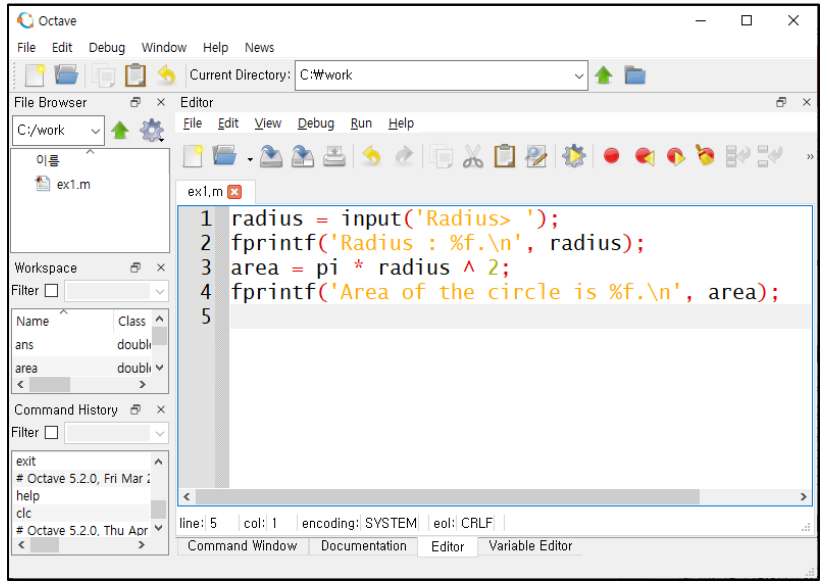
M 파일의 작성

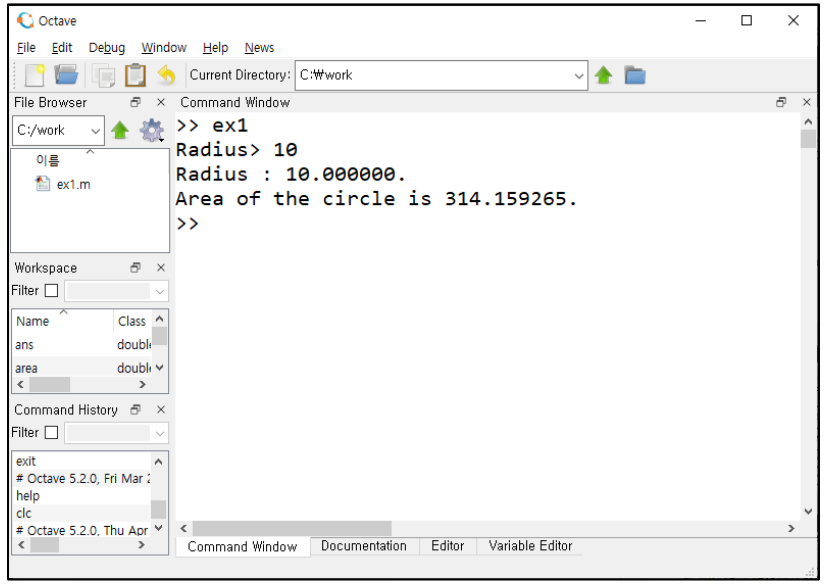
Review

- M-file 작성 (스크립트 파일)
  - 현재 폴더에 저장 (파일 이름은 영문으로)
    명령창에서 한글입력 불가능
  - 파일이름
- 수행 (스크립트)
  - 명령창에서 파일 이름 (확장자 제외) 입력
    >> ex1

ex1.m

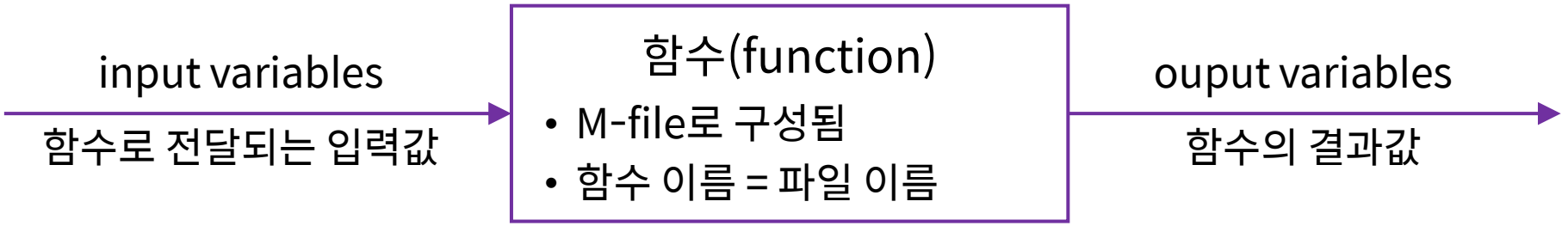
```
radius = input('Radius> ');
fprintf('Radius : %f.\n', radius);
area = pi * radius ^ 2;
fprintf('Area of the circle is %f.\n', area);
```





사용자 정의 함수

- 함수(function)
  - 일반적인 프로그래밍 언어의 함수와 거의 같은 기능을 수행
    - C언어 : 함수(function), Basic & FORTRAN : subroutine, Pascal : procedure
  - 프로그램의 모듈 역할을 하는 기능으로 자주 사용하는 기능을 구현한 뒤 함수로 만들어놓고 간단하게 불러서 사용하는 것



사용자 정의 함수

- 함수(function)의 구조

```
% ------------------------------------------------------------
% Contents of 'help' command
% ------------------------------------------------------------

function [output_variables] = function_name(input_variables)
    % do MATLAB commands
end
```

Item	Description
function	M 파일이 함수를 정의한다는 것을 의미하는 키워드
output_variables	함수의 결과를 저장할 변수명 (생략 가능하며, 두 개 이상인 경우 [] 사용)
function_name	함수 이름 – 파일 이름과 반드시 같게 한다
input_variables	해당 함수의 입력 변수 (단일 혹은 복수 개)

사용자 정의 함수

- 예제

```
%-------------------------------------------------------------
% function sum = intsum(a, b)
% return sum of two input variables
%-------------------------------------------------------------

function sum = intsum(a, b)
    sum = a + b
end
```

```
>> intsum(2, 3)
sum =  5
ans =  5
```

사용자 정의 함수

- 도움말 (help)

```
%-----------------------------------------------------------------
% function sum = intsum(a, b)
% return sum of two input variables
%-----------------------------------------------------------------

%

function sum = intsum(a, b)
    sum = a + b
end
```

첫번째 주석문 그룹을
도움말(help)에 표시

```
>> help intsum
'intsum' is a function from the file c:\work\intsum.m

-----------------------------------------------------------------
 function sum = intsum(a, b)
 return sum of two input variables
-----------------------------------------------------------------
...
```

사용자 정의 함수

- 스크립트 파일 vs. 함수 파일
  - 두 경우 모두 확장자가 *.m
  - 스크립트/함수의 구분
    - 첫번째 실행문이 `function ...` 으로 시작 → 함수 파일
    - 아니면 → 스크립트 파일
  - 파일의 내부 변수
    - 함수 파일의 내부 변수들 → 지역 변수 (함수 내부에서만 사용 가능한 변수)
    - 스크립트 파일의 내부 변수들 → 전역 변수 (command window에서 수행하는 것과 동일)
  - Workspace의 변수 사용
    - 함수 파일 내부 → 사용이 불가능 (필요한 경우 인수/입력 변수로 전달하여야 함)
    - 스크립트 파일 내부 → 사용 가능 (command window에서 수행하는 것과 동일)

사용자 정의 함수

- 예제 : radian → degree

```
%-------------------------------------------------------------
% ret = rad2deg(r)
% Converts r (in radians) value to ret (in degree) value
%-------------------------------------------------------------

function ret = rad2deg(r)
    ret = r * 180 / pi;
end
```

```
>> rad2deg(pi / 2)
ans =  90
>> rad2deg(pi / 6)
ans =  30.000
>> deg = rad2deg(pi / 4)
deg =  45
>>
```

Matlab 프로그래밍

- 관계연산자 (relational operator)
  - 두 피 연산자의 관계를 판단하는 연산자로 조건문에서 비교문(compare statement)에 주로 사용

Operator	Meaning	Example	C언어
<	Less than	A < B	<
<=	Less or equal than	A <= B	<=
>	Greater than	A > B	>
>=	Greater or equal than	A >= B	>=
==	Equal	A == B	==
~=	Not equal	A ~= B	!=

  - 관계연산자의 결과값 : 참 → 1, 거짓 → 0

Matlab 프로그래밍

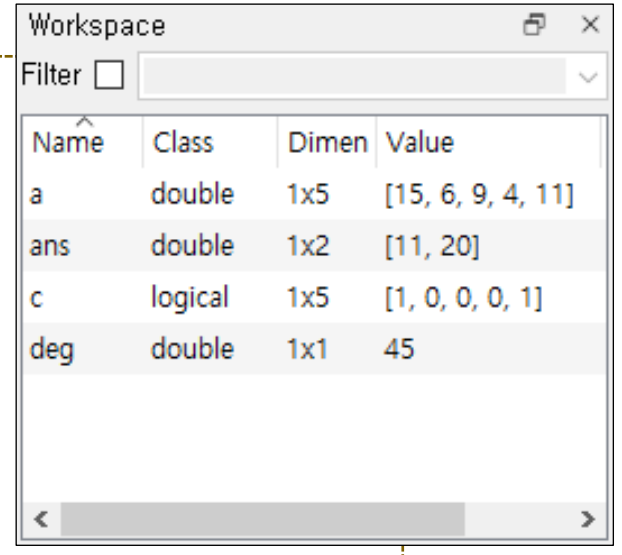
- 관계연산자
  - 벡터/행렬의 경우 원소 단위로 비교하여 벡터/행렬로 반환

```
>> 5 > 8
ans = 0
>> a = 5 < 10
a = 1
>> x = (6 < 10) + (7 > 8) + (5 * 3 == 60 / 4)
x =  2
>> a = [15 6 9 4 11];
>> b = [8 20 9 2 19];
>> b >= a
ans =

  0  1  1  0  1

>>
```

```
>> a == b
ans =

  0  0  1  0  0

>> a ~= b
ans =

  1  1  0  1  1

>> a > 10
ans =

  1  0  0  0  1
```

- 관계연산자
  - 벡터/행렬의 경우 관계연산자의 결과는 논리 벡터/행렬로 반환 → 주소지정에 사용되어 참(1)인 위치의 원소만 추출 가능

```
>> a = [15 6 9 4 11];
>> c = a > 10
c =

  1   0   0   0   1

>> a(c)
ans =

   15   11

>>
```



벡터 a에서 10보다 큰 값만 추출하기

- 논리연산자 (logical operator)
  - 관계연산자의 결과값(참/거짓)과 같은 논리값의 연산으로 두 개 이상의 복합 조건을 판단할 때 사용

Operator	Meaning	Example	C언어
&	Logical AND	A & B	&&
|	Logical OR	A | B	||
~	Logical NOT	~A	!

Matlab 프로그래밍

- 논리연산자 (logical operator)
  - 벡터/행렬의 경우 각 요소별로 계산됨

```
>> 3 & 7
ans = 1
>> a = 5 | 0
a = 1
>> ~25
ans = 0
>> a = [9 2  0   1 0 10];
>> b = [2 0 12 -10 0  2];
>> a & b
ans =
  1  0  0  1  0  1
>> z = a | b
z =
  1  1  1  1  0  1
>> ~(a + b)
ans =
  0  0  0  0  1  0
```

- 연산자 우선순위

우선순위	연산자	결합방향
1	( )	→
2	.' .^ ' ^	→
3	(단항)+ (단항)- ~	→
4	.* ./ .\ * / \	→
5	+ -	→
6	:	→
7	< <= > >= == ~=	→
8	&	→
9	|	→
10	&&	→
11	||	→

응용전산 및 실습 (01) - 12주차

M-file 프로그래밍, 제어문

Matlab 프로그래밍 : if

- if 문

```
if 조건문
    문장;   % 조건문이 만족하는 경우 수행하는 문장들...
end         % Octave 편집기에서는 endif가 자동으로 나타나나
            % Matlab과의 호환성을 위해서 end로 수정할 것
```

- Example

```
k = input('Enter k : ');
if k > 0
    disp('You entered positive number.');
end
```

Matlab 프로그래밍 : if

- if ~ else 문

```
if 조건문
    문장;   % 조건이 만족하는 경우 수행하는 문장
else
    문장;   % 만족하지 않은 경우 수행하는 문장
end
```

- Example

```
k = input('Enter k : ');
if k > 0
    disp('You entered positive number.');
else
    disp('You entered negative number.');
end
```

Matlab 프로그래밍 : if

- if ~ else 문 예제
  - 수학, 영어 과목의 점수들을 입력받아 두 과목 모두 60점 이상인 경우 통과(pass) 아니면 탈락(fail)을 계산하는 스크립트 파일 작성.
  - 관계 연산자 및 논리 연산자 사용
  - 통과 조건을 검사하는 경우 vs. 탈락 조건을 검사하는 경우 (2가지 조건문 가능)

Matlab 프로그래밍 : if

- if ~ elseif ~ else 문

```
if 조건문1
    문장;   % 조건 1이 만족하는 경우 수행하는 문장
elseif 조건문2
    문장;   % 조건 1이 만족하지 않고 조건 2가 만족하는 경우
else
    문장;   % 조건 1, 2가 모두 만족하지 않는 경우
end
```

```
k = input('Enter number : ');
if k > 0
    disp('You entered positive number.');
elseif k < 0
    disp('You entered negative number.');
else
    disp('You entered 0.');
end
```

Matlab 프로그래밍 : if

- if ~ elseif ~ else 문 예제
  - 점수(score)을 입력받아 90점 이상이면 'A', 80점 이상 90점 미만이면 'B', … , 60점 미만이면 'F'학점을 출력하는 스크립트 파일 작성.
  - 알고리즘]
    - 만일 점수가 90 이상이면
      'A' 학점 출력
      그렇지 않고 만일 점수가 80 이상이면
      'B' 학점 출력
      그렇지 않고 만일 점수가 70 이상이면
      'C' 학점 출력
      …
      그렇지 않으면
      'F' 학점 출력

Matlab 프로그래밍 : if

- 중첩 조건문 (nested if)
  - 하나의 if문에 따른 조건문 속에 다른 조건문이 포함되는 경우

```
if 조건문1
    if 조건문2  % 조건 1이 만족하는 경우, 조건 2를 검사
        문장;   % 조건 1이 만족하면서 조건 2도 만족하는 경우 수행
    else
        문장;   % 조건 1이 만족하면서 조건 2는 만족하지 않는 경우
    end
else
    if 조건문3  % 조건 1이 만족하지 않으면서, 조건 3을 검사
        문장;   % 조건 1이 만족하지 않으면서 조건 2를 만족하는 경우 수행
    else
        문장;   % 조건 1이 만족하지 않으면서 조건 2도 만족하지 않은 경우
    end
end
```

Matlab 프로그래밍 : if

- 중첩 조건문 예제
  - 1[in] = 2.54[cm], 1[ft]=12[in] 인 관계를 이용하여 cm, ft, in 를 서로 변환시키는 스크립트 파일 작성.
    - 실행 예] (파란색은 입력)

```
Enter value : 10
Enter unit : in          → 현재 단위
Enter new unit : cm      → 변환할 단위
10[in] is 25.4[cm]
```

- 응용
  - 1[in] = 2.54[cm], 1[ft]=12[in] 인 관계를 이용하여 m, cm, in, ft 를 서로 변환시키는 스크립트 파일 작성.

Matlab 프로그래밍 : for

- for 문

```
for 변수=초기값:최종값
    문장;   % 변수가 초기값에서 최종값까지 1씩 증가하면서 반복됨
end         % Octave 편집기에서는 endfor가 자동으로 나타나나
            % Matlab과의 호환성을 위해서 end로 수정할 것
```

- Example : sumall.m

```
n = input('Enter number : ');
sum = 0;
for k=1:n
    sum = sum + k;
end
fprintf('Sum from 1 to %d is %d\n', n, sum);
```

Matlab 프로그래밍 : for

- for 문

```
for 변수=초기값:증가값:최종값
    문장;   % 변수가 초기값에서 최종값까지 증가값씩 증가하면서 반복됨
end         % Octave 편집기에서는 endfor가 자동으로 나타나나
            % Matlab과의 호환성을 위해서 end로 수정할 것
```

- Example : sumodd.m

```
n = input('Enter number : ');
sum = 0;
for k=1:2:n
    sum = sum + k;
end
fprintf('Sum of odd numbers from 1 to %d is %d\n', n, sum);
```

Matlab 프로그래밍 : for

- for 문 예제
  - 1부터 주어진 인수(입력변수) n까지의 합을 구하는 함수 AddTo()

```
%-------------------------------------------------------------
% function sum = AddTo(n)
% Calculate sum of the numbers from 1 to n
%-------------------------------------------------------------

function sum = AddTo(n)
  sum = 0;
  for k = 1:n
    sum = sum + k;
  end
end
```

Matlab 프로그래밍 : for

- for문의 반복 범위
  - `1:4 == [1, 2, 3, 4]` ⇒ for문에서도 벡터값을 이용하여 반복

```
for k = 1:2:6
    sum = sum + k;
end
```

```
for k = [1 3 5]
    sum = sum + k;
end
```

Matlab 프로그래밍 : nargin, nargout

- 함수의 입/출력 변수의 개수
  - nargin : 입력 변수의 개수 (number of input arguments)
  - nargout : 출력 변수의 개수 (number of output arguments)
  - 호출할 때 사용된 입/출력 변수의 개수에 알맞은 기능을 제공하는 함수 작성이 가능 : 예] plot 함수

Matlab 프로그래밍 : nargin, nargout

```
% function [res1, res2, res3] = argtest(a, b, c, d)
% Test function for nargin, nargout

function [res1, res2, res3] = argtest(a, b, c, d)
    fprintf('nargin = %d\n', nargin);
    fprintf('nargout = %d\n', nargout);

    res1 = nargin;
    if nargout > 1
      res2 = nargout;
    end
    if nargout > 2
      res3 = 0;
    end
End
```

```
>> argtest
nargin = 0
nargout = 0
ans = 0
>> a = argtest(1, 2)
nargin = 2
nargout = 1
a =  2
>> [a, b] = argtest(1, 2)
nargin = 2
nargout = 2
a =  2
b =  2
>> [a, b] = argtest(1, 2, 3)
nargin = 3
nargout = 2
a =  3
b =  2
```

Matlab 프로그래밍 : for

- for 문 예제 : AddTo()함수의 확장 ⇒ AddToEx()
  - AddToEx(n) : 1~n 까지의 합계
  - AddToEx(a, b) : a~b 까지의 합계

```
%-------------------------------------------------------------
% function sum = AddToEx(n)
% Calculate sum of the numbers from 1 to n
% function sum = AddToEx(a, b)
% Calculate sum of the numbers from a to b
%-------------------------------------------------------------

function sum = AddToEx(a, b)
    if nargin == 1
        b = a;
        a = 1;
    end
    sum = 0;
    for k = a:b
        sum = sum + k;
    end
end
```

Ex : 원주율 (π) 계산하기

- for문을 사용하여 π계산하기 #1 : 라이프니츠(Leibniz)의 원주율공식

$$\pi = 4\left(\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} \cdots\right)$$

➢ From Gregory Series : $\tan^{-1} x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$, where $x = 1$

```
n = 10000;     % iteration number
pisum = 0;     % result

for i=1:n
  pisum = pisum + (-1)^(i+1) / (2*i - 1);
  if mod(i, 1000) == 0
    fprintf('Iter [%d], pi = %.12f\n', i, pisum * 4);
  end
end

fprintf('pi = %.12f\n', pisum * 4);
```

Ex : 원주율 (π) 계산하기

- for문을 사용하여 π계산하기 #1 : 라이프니츠(Leibniz)의 원주율공식

$$\pi = 4\left(\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15}\cdots\right) = 4\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{2k-1}$$

- 함수로 구현

```
%-------------------------------------------------
% function ret = pi1f(n)
% Calculate pi using Leibniz's equation
% n : iteration number
%-------------------------------------------------

function ret = pi1f(n)
  pisum = 0; % result
  for i=1:n
    pisum = pisum + (-1)^(i+1) / (2*i - 1);
  end
  ret = pisum * 4;
end
```

```
>> pi1f(1)
ans =  4
>> pi1f(5)
ans =  3.3397
>> pi1f(10)
ans =  3.0418
>> pi1f(100)
ans =  3.1316
>> pi1f(1000)
ans =  3.1406
>> pi1f(10000)
ans =  3.1415
>> pi1f(100000)
ans =  3.1416
```

Ex : 원주율 (π) 계산하기

- for문을 사용하여 π계산하기 #2 : 월리스(Wallis)의 공식

$$\frac{\pi}{2} = \frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\frac{8}{7}\cdot\frac{8}{9}\cdot\frac{10}{9}\cdots = \left(\frac{2}{1}\cdot\frac{2}{3}\right)\cdot\left(\frac{4}{3}\cdot\frac{4}{5}\right)\cdot\left(\frac{6}{5}\cdot\frac{6}{7}\right)\cdots = \prod_{k=1}^{\infty}\left(\frac{4\cdot k^2}{4\cdot k^2-1}\right)$$

```
n = 10000;     % iteration number
pival = 1;     % result

for k=1:n
  pival = pival * 4 * k^2 / (4 * k^2 - 1);
  if mod(k, 1000) == 0
    fprintf('Iter [%d], pi = %.12f\n', k, pival * 2);
  end
end

fprintf('pi = %.12f\n', pival * 2);
```

Ex : 원주율 (π) 계산하기

- π계산하기 #3 : 오일러(Euler)의 곱셈 공식

$$\frac{\pi^2}{6} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \frac{1}{6^2} + \cdots$$

- π계산하기 #4 : 프랑수아 비에트(François Viète)의 무한급수

$$\frac{2}{\pi} = \frac{\sqrt{2}}{2} \cdot \frac{\sqrt{2+\sqrt{2}}}{2} \cdot \frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2} \cdots$$

Ex : Taylor Series

- 테일러 급수
  - 임의의 해석함수(analytic function)를 도함수의 함 점에서의 값으로 계산된 항의 무한합으로 나타내는 방식 (유도과정 및 자세한 내용 생략)
  - a 근처에서는 몇 개의 항으로도 근사값 계산

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \frac{1}{3!}f'''(a)(x-a)^3 + \cdots$$

  - $a = 0$인 경우 ⇒ Maclaurin series
- 테일러 급수의 응용?
  - 모델의 단순화 : 복잡한 함수가 있을 때 1~3차의 다항 함수로 근사화
  - 정적분의 계산 : 부정 적분의 계산이 힘든 경우 급수를 활용하여 근사화 후 계산
  - 컴퓨터에서 초월 함수(sin, cos, tan, exp, log, ...) 계산
    - ➢ 초월 함수의 H/W 계산 회로 구성은 불가능 ⇒ 근사 다항식을 이용하여 계산

Ex : Taylor Series

- 테일러 급수의 예
  - a 근처에서

$$f(x)=\sum_{n=0}^{\infty}\frac{f^{(n)}(a)}{n!}(x-a)^n=f(a)+f'(a)(x-a)+\frac{1}{2}f''(a)(x-a)^2+\frac{1}{3!}f'''(a)(x-a)^3+\cdots$$

  - 예] $a=0$인 경우 ⇒ 0 근처에서 근사값

$$\sin x=\sin 0+x\sin' 0+\frac{x^2}{2!}\sin'' 0+\frac{x^3}{3!}\sin''' 0+\cdots$$

$$\sin x=0+x\cdot 1+\frac{x^2}{2!}\cdot 0+\frac{x^3}{3!}\cdot(-1)+\frac{x^4}{4!}\cdot 0+\cdots$$

$$\sin x=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\cdots=\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1)!}x^{2n+1}$$

$$\cos x=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-\cdots=\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n)!}x^{2n},\qquad e^x=\sum_{n=0}^{\infty}\frac{x^n}{n!}=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots$$

Ex : Taylor Series

- 테일러 급수를 이용한 sin 함수

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

```
%-------------------------------------------------------
% function ret = sint(x)
% Calculate sine value using Taylor series expansion
%-------------------------------------------------------

function ret = sint(x)
  ret = 0;
  for n = 0:5
    ret = ret + (-1)^n / factorial(2*n + 1) * x^(2*n + 1);
  end
end
```

```
>> sin(pi / 4)
ans =  0.70711
>> sint(pi / 4)
ans =  0.70711
>> sin(pi)
ans =    1.2246e-16
>> sint(pi)
ans = -0.00044516
```

Ex : Taylor Series

- 테일러 급수를 이용한 sin 함수
  - 다항식의 항의 수를 인수로 전달할 수 있도록 확장

```
%-------------------------------------------------------
% function ret = sinto(x, order)
% Calculate sine value using Taylor series expansion
% x : angle [rad], order : order of series expansion
%-------------------------------------------------------

function ret = sinto(x, order)
    if nargin < 2
        order = 5;
    end
    ret = 0;
    for n = 0:order
        ret = ret + (-1)^n / factorial(2*n+1)*x^(2*n+1);
    end
end
```

```
>> sin(pi)
ans =    1.2246e-16
>> sinto(pi, 3)
ans = -0.075221
>> sinto(pi) ⇒ sinto(pi,5)
ans = -0.00044516
>> sinto(pi, 7)
ans = -0.00000077279
>> sinto(pi, 10)
ans =  0.000000000010348
>>
```

Ex : Taylor Series

- 테일러 급수를 이용한 sin 함수
  - 인수로 벡터/행렬도 처리할 수 있도록 확장

```
%-------------------------------------------------------
% function ret = sintv(x, order)
% Calculate sine value using Taylor series expansion
% x can be either vector or matrix
% x : angle [rad], order : order of series expansion
%-------------------------------------------------------

function ret = sintv(x, order)
  if nargin < 2
    order = 5;
  end
  ret = zeros(size(x));
  for n = 0:5
    ret = ret + (-1)^n / factorial(2*n + 1)*x .^ (2*n + 1);
  end
end
```

Matlab 프로그래밍 : while

- while 문

```
while 조건식
    문장;   % 조건식이 만족하는 동안 문장을 반복
end         % Octave 편집기에서는 endwhile이 자동으로 생성되나
            % Matlab과의 호환성을 위해서 end로 수정할 것
```

- Example : sumall.m

```
n = input('Enter number : ');
sum = 0;
j = 1;
while(j <= n)
    sum = sum + j;
    j = j + 1;
end
fprintf('Sum from 1 to %d is %d\n', n, sum);
```

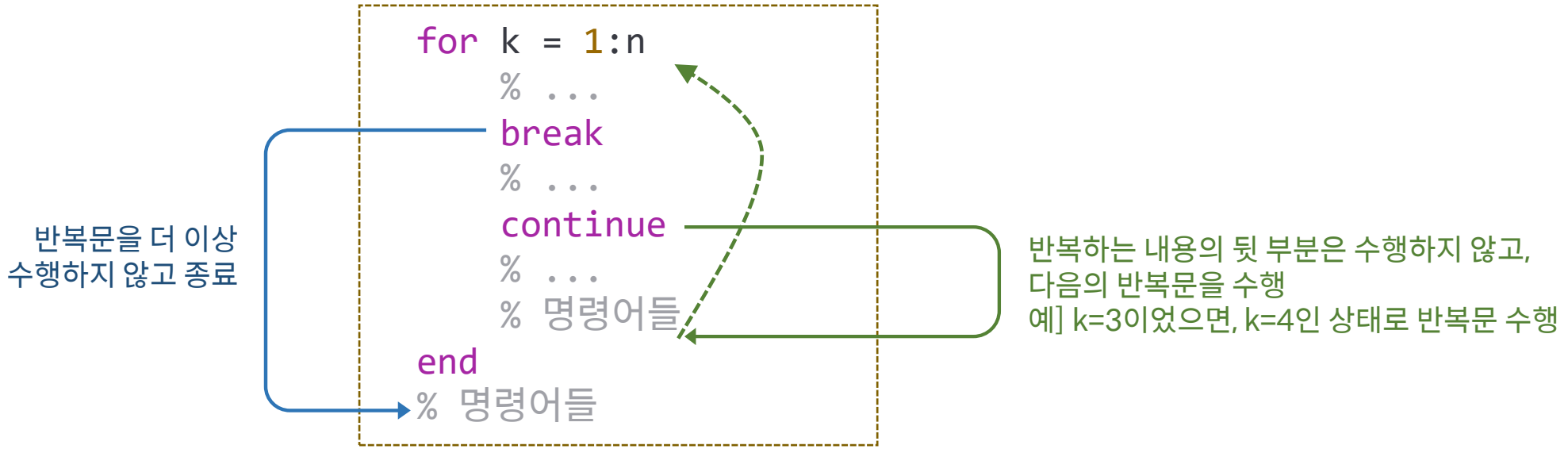
Matlab 프로그래밍 : while

- while 문 예제
  - 1부터 주어진 인수(입력변수) n까지의 합을 구하는 함수 AddTo()

```
%-------------------------------------------------------------
% function sum = AddTo(n)
% Calculate sum of the numbers from 1 to n
%-------------------------------------------------------------

function AddTo(n)
    sum = 0;
    j = 1;
    while(j <= n)
        sum = sum + j;
        j = j + 1;
    end
end
```

Matlab 프로그래밍 : break, continue

- break 문
  - for 또는 while문의 반복문을 종료
- continue 문
  - for 또는 while문의 나머지 반복문을 수행하지 않고 다음 반복문을 수행



Matlab 프로그래밍 : break, continue

- Break, continue 문 예제

```
for k = 1:6
  if k == 4
    break;
  end
  k
end
```

```
>> break1
k =  1
k =  2
k =  3
>>
>>
```

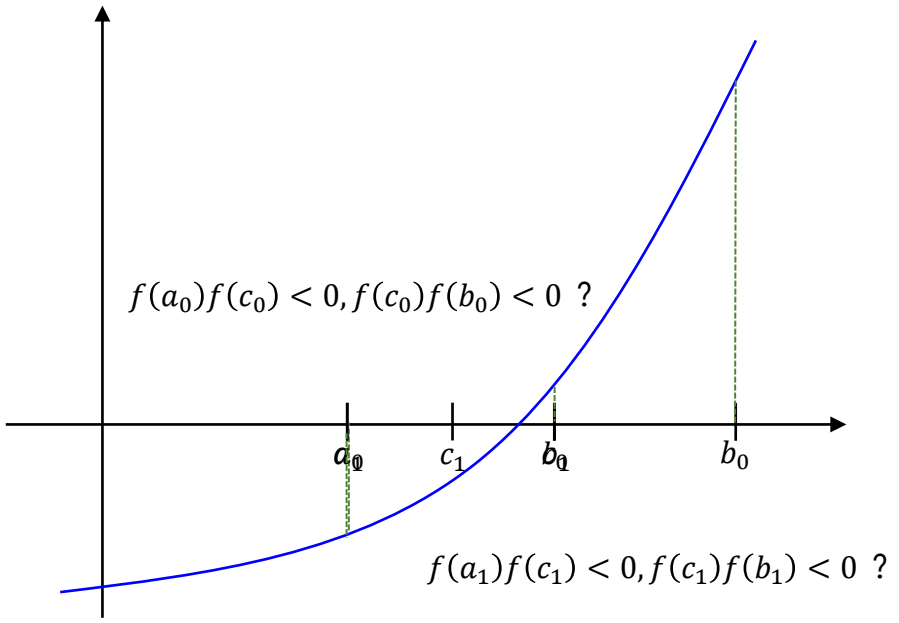
```
for k = 1:5
  if k == 2 | k == 3
    continue;
  end
  k
end
```

```
>> continue1
k =  1
warning: Matlab-style short-circuit…
k =  4
k =  5
>>
```

Matlab에서는 나타나지 않음

Ex: Root-Finding

- 함수의 근 찾기 : Bisection method
  - $f(x) = 0$이 되는 x (roots) 찾기, 초기값 : 근이 존재하는 범위 설정
  - 구간을 절반씩 나누어서 검색$[f(x) \approx 0]$하는 것을 반복함
- 수행 순서 (algorithm)
  - $k = 0$에서 시작 : 초기 구간 $[a_0, b_0]$
    - ➢ 근이 존재하는 조건 : $f(a_0)f(b_0) < 0$
  - k를 증가시키면서 다음을 반복
    - ➢ 구간의 중간값 계산 : $c_k = \frac{a_k + b_k}{c_k}$
    - ➢ $f(a_k)f(c_k)$, $f(c_k)f(b_k)$의 부호 계산
    - ➢ $f(a_k)f(c_k) \Rightarrow a_{k+1} = a_k\ , b_{k+1} = c_k$
    - ➢ $f(c_k)f(b_k) \Rightarrow a_{k+1} = c_k\ , b_{k+1} = b_k$
    - ➢ 오차 : $|a_k - b_k| < \epsilon$ 보다 작으면 중단, 근 $\Rightarrow c_k$
      - ✓ ϵ : 오차 범위, 예] 소수점 5째자리 $\Rightarrow \epsilon = 10^{-6}$

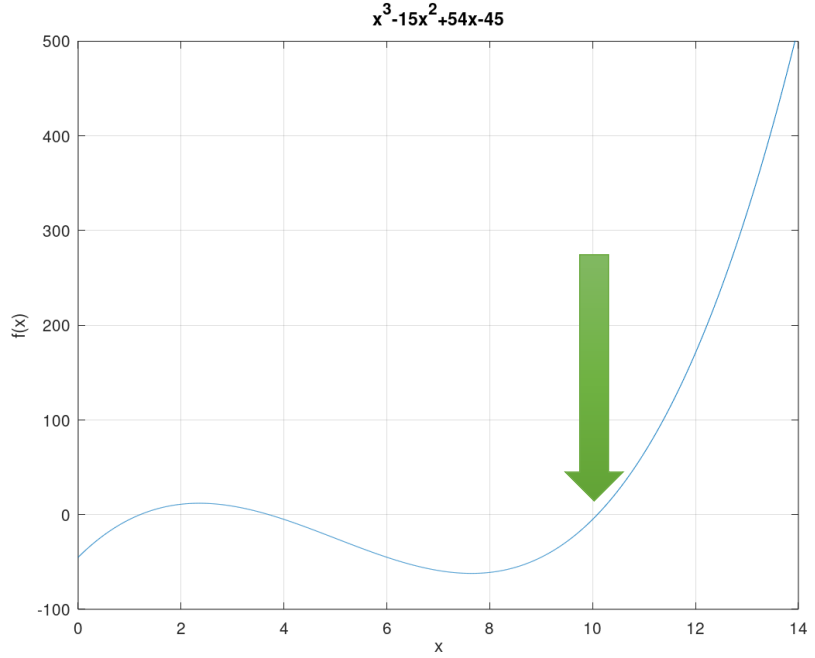


Ex: Root-Finding

- Bisection method example

  - $f(x) = x^3 - 15x^2 + 54x - 45$, 초기 구간 : $[8, 13]$

```
% set initial interval [8, 13]
a = 8;
b = 13;

eps = 1e-6; % epsilon
iter = 1;   % iteration number

fprintf('   k        a           b         f(c)\n');
fprintf('-----------------------------------------\n');
while abs(a - b) > eps
  c = (a + b) / 2;
  if func(a) * func(c) < 0
    b = c;
  else
    a = c;
  end
  fprintf('%5d %10.7f %11.7f  %11.7f\n', iter, a, b, func(c));
  iter = iter + 1;
end
```



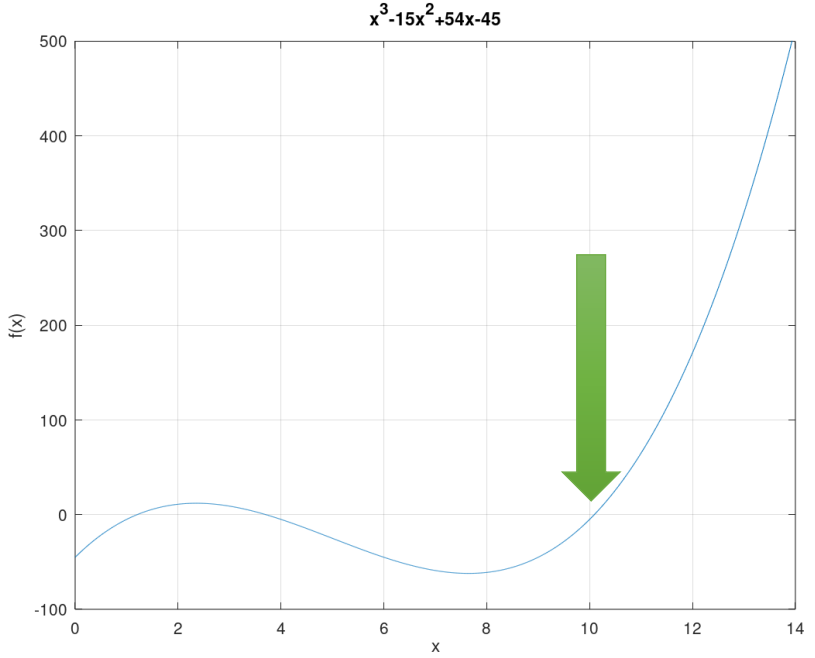

```
function y = func(x)
  y = x.^3 - 15 * x.^2 + 54 * x - 45;
end
```

Ex: Root-Finding

- Bisection method example

  - $f(x) = x^3 - 15x^2 + 54x - 45$, 초기 구간 : $[8, 13]$

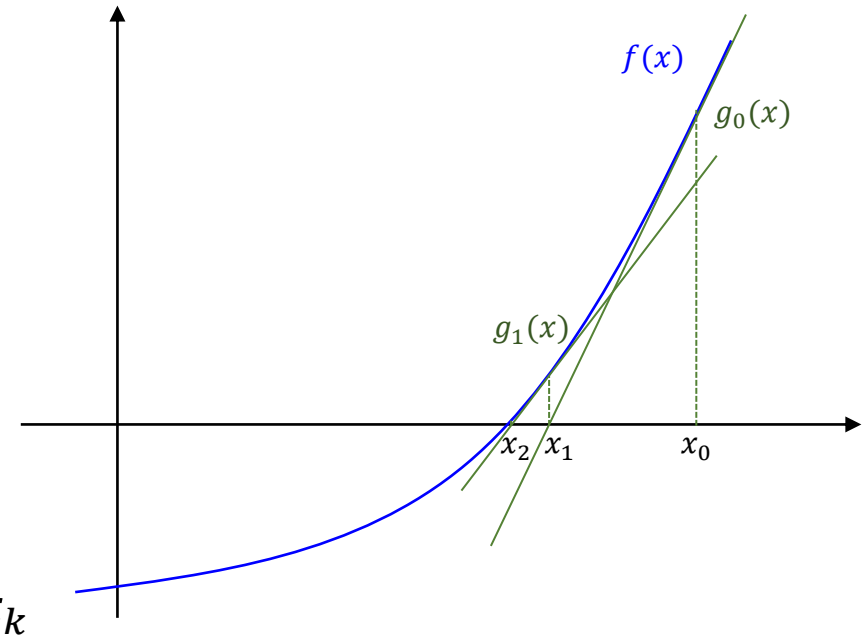
```
% set initial interval [8, 13]
a = 8;
b = 13;

eps = 1e-6; % epsilon
iter = 1;   % iteration number

fprintf('   k        a           b         f(c)\n');
fprintf('-----------------------------------------\n');
while 1
  c = (a + b) / 2;
  if func(a) * func(c) < 0
    b = c;
  else
    a = c;
  end
  fprintf('%5d %10.7f %11.7f  %11.7f\n', iter, a, b, func(c));
  if abs(a - b) < eps
    break
  end
  iter = iter + 1;
end
```



Ex: Root-Finding

- 함수의 근 찾기 : Newton-Raphson method
  - $f(x) = 0$이 되는 근 근처에서 초기값 x_0 설정
  - $x = x_k$에서 $f(x)$의 선형 근사 함수 $g(x)$의 해를 구하여 x_{k+1}로 설정한 후 반복

- 수행 순서 (algorithm)
  - $k = 0$에서 시작 : 초기 해 x_0 설정
  - k를 증가시키면서 다음을 반복
    - ➢ $x = x_k$에서 선형 근사 함수 $g_k(x)$계산
      - ✓ $g_k(x) = f(x_k) + f'(x_k)(x - x_k)$
    - ➢ $g_k(x) = 0$이 되는 x계산
      - ✓ $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$: $f'(x)$가 필요함 [단점]
    - ➢ 오차 : $|x_k - x_{k+1}| < \epsilon$ 보다 작으면 중단, 근 $\Rightarrow x_k$
      - ✓ ϵ : 오차 범위, 예] 소수점 5째자리 $\Rightarrow \epsilon = 10^{-6}$

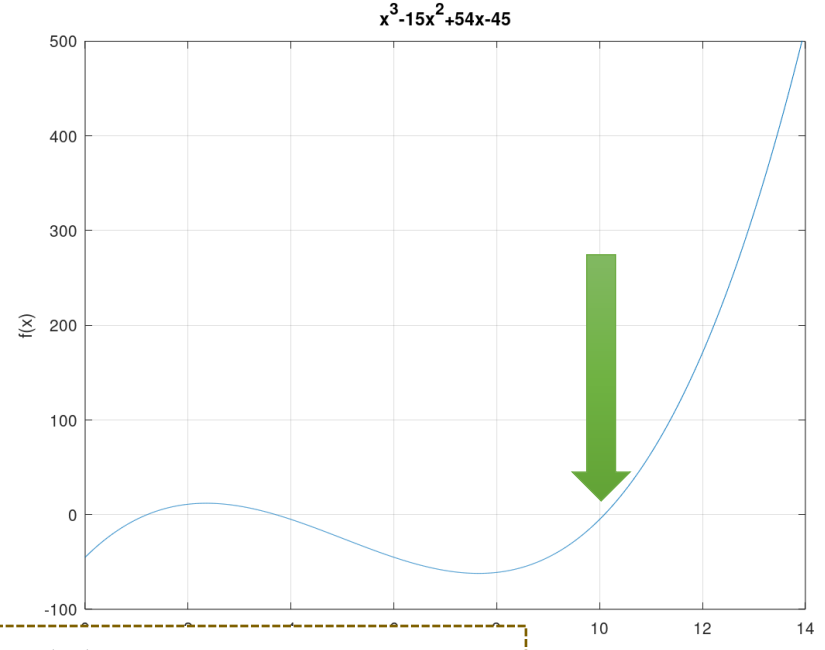


Ex: Root-Finding

- Newton-Raphson method example

  - $f(x) = x^3 - 15x^2 + 54x - 45$, 초기값 : $f_0 = 12$

```
x_cur = 8;  % initial root
eps = 1e-6; % epsilon
iter = 1;   % iteration number
max_iter = 30; % maximum iteration number

fprintf('   k        x         f(c)\n');
fprintf('-----------------------------\n');
while 1
  x_next = x_cur - func(x_cur) / funcd(x_cur);
  fprintf('%5d %11.7f  %11.7f\n', iter, x_next, func(x_next));
  if abs(x_next - x_cur) < eps
    break
  end
  x_cur = x_next;
  iter = iter + 1;
  if iter > max_iter
    break
  end
end
```




```
function y = func(x)
  y = x.^3 - 15 * x.^2 + 54 * x - 45;
end
```

```
function y = funcd(x)
  y = 3 * x.^2 - 30 * x + 54;
end
```

Root-Finding : Comparison

```
>> bisect1
   k        a           b          f(c)
-----------------------------------------
    1  8.0000000  10.5000000   25.8750000
    2  9.2500000  10.5000000  -37.4843750
    3  9.8750000  10.5000000  -11.5175781
    4  9.8750000  10.1875000    5.6589355
    5 10.0312500  10.1875000   -3.2978210
    6 10.0312500  10.1093750    1.0870018
    7 10.0703125  10.1093750   -1.1286197
    8 10.0898438  10.1093750   -0.0266338
    9 10.0898438  10.0996094    0.5287250
   10 10.0898438  10.0947266    0.2506812
   11 10.0898438  10.0922852    0.1119326
   12 10.0898438  10.0910645    0.0426267
   13 10.0898438  10.0904541    0.0079907
   14 10.0901489  10.0904541   -0.0093230
   15 10.0903015  10.0904541   -0.0006665
   16 10.0903015  10.0903778    0.0036621
   17 10.0903015  10.0903397    0.0014978
   18 10.0903015  10.0903206    0.0004157
   19 10.0903111  10.0903206   -0.0001254
   20 10.0903111  10.0903158    0.0001451
   21 10.0903111  10.0903134    0.0000099
   22 10.0903122  10.0903134   -0.0000578
   23 10.0903128  10.0903134   -0.0000240
```

```
>> nr
   k        x          f(c)
------------------------------
    1  18.1666667  1981.0879630
    2  14.1972134  559.8391617
    3  11.7920568  145.6982188
    4  10.5509747   29.4734893
    5  10.1384116    2.7642441
    6  10.0909240    0.0346554
    7  10.0903134    0.0000057
    8  10.0903133    0.0000000
```

inline 함수

- inline() 함수
  - 입력 변수를 함수 형태의 문자열을 받아 인라인 함수로 변환
- Bisection method example

```
a = input('Enter function of x : ', 's');
func = inline(a);
a = input('Enter first point of interval : ');
b = input('Enter second point of interval : ');

eps = 1e-6; % epsilon
iter = 1;   % iteration number

% 나머지는 동일
% ...
```

```
>> bisect2
Enter function of x : cos(x) + 2*sin(x) + x^2
Enter first point of interval : -1
Enter second point of interval : 1
   k         a           b          f(c)
-------------------------------------------
    1 -1.0000000   0.0000000    1.0000000
    2 -1.0000000  -0.5000000    0.1687315
    3 -0.7500000  -0.5000000   -0.0690887
…
   21 -0.6592665  -0.6592655   -0.0000004
```

find 함수

- find()
  - 입력 배열 중에서 조건을 충족하는 배열의 요소들만 반환

```
>> nn = 1:10
nn =
    1    2    3    4    5    6    7    8    9   10

>> find(x > 6)
ans =
    7    8    9   10

>> find(mod(nn, 2) == 0)
ans =
    2    4    6    8   10

>> find(mod(nn, 2))
ans =
   1   3   5   7   9
```

Ex: 소수 구하기

- 소수(prime number) 구하기 알고리즘 : 에라토스테네스의 체
  - 2~100사이의 모든 소수를 구하는 경우
    - ➢ 배열에 2:100 = [2, 3, 4, 5, ⋯ , 100]을 저장
    - ➢ 배열의 처음 요소 2는 소수에 포함하고 배열 요소 중에서 2의 배수를 모두 제거한다.
      - ✓ [2, 3, 4, 5, ⋯ , 100] ⇒ [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, ⋯ , 99], 소수 : [2]
    - ➢ 배열의 처음 요소 3은 소수에 포함하고 배열 요소 중에서 3의 배수를 모두 제거한다.
      - ✓ [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, ⋯ , 99] ⇒ [5, 7, 11, 13, 17, 19, 23, ⋯ , 97], 소수 : [2, 3]
    - ➢ 배열의 처음 요소 5는 소수에 포함하고 배열 요소 중에서 5의 배수를 모두 제거한다.
      - ✓ [5, 7, 11, 13, 17, 19, 23, ⋯ , 97] ⇒ [7, 11, 13, 17, 19, 23, ⋯ , 97], 소수 : [2, 3, 5]
    - ➢ 배열의 남은 요소가 없을 때까지 반복한다.
      - ✓ ⋯
      - ✓ [97] ⇒ [], 소수 : [2, 3, 5, 7, 11, 13, 17, 19, 23, ⋯ , 89, 97]

Ex: 소수 구하기

- 소수(prime number) 구하기 알고리즘 : 에라토스테네스의 체

```
%-------------------------------------------------------------
% function primes = getprimes(n)
% Get prime numbers between 1 to n
%-------------------------------------------------------------

function primes = getprimes(n)
  nos = 2:n;   % 전체 숫자 배열 만들기
  primes = []; % 소수저장용 배열
  while length(nos) > 0
    % 숫자 배열의 처음 요소를 소수에 추가
    primes = [primes nos(1)];
    % 숫자 배열의 처음 요소로 나누어지는 모든 수 삭제
    nos = nos(find(mod(nos, nos(1))));
  end
end
```

```
>> getprimes(10)
ans =  2   3   5   7
>> getprimes(50)
ans =  2    3    5    7   11   13   17   19   23   29   31   37   41   43   47
```