

매트랩 기초 1

1

매트랩 소개

● 매트랩(MATLAB)

- Matrix Laboratory의 앞쪽 세 글자의 합성: 배열을 만들고 조작
- 대화형 인터페이스(interface) 시스템
- 쉽고 효율적인 고급 프로그래밍 언어를 이용하여 신속하게 프로그램 개발
- 매트랩의 장점
 - 내장함수
최첨단 기술, 다양한 자료구조와 자료형을 이용하여 정확한 수치해석 가능
결과를 고급스럽고 선명한 시각적 자료로 제시
 - 심볼릭(Symbolic) 수학 도구상자
대수와 수식의 풀이 과정을 부호로 표시
C / C++, Java와 같은 다른 프로그래밍 언어와 확장, 호환해서 사용 가능
 - 시뮬링크
블록(block)을 이용하여 쉽게 시뮬레이션 설계 가능

2

도움 명령어

● 매트랩의 가장 큰 장점인 다양한 내장함수에 대한 정보를 찾는 데 유용

- **help, doc, lookfor**를 제공
- **sin**이라는 함수의 정보를 알고 싶으면 다음을 입력하고 Enter ↵키를 누른다.

```
>>help sin
```

- ✓ **sin**의 기본 설명이 명령창에 표시

- 더 자세한 **sin**에 대한 추가 설명과 간단한 예제를 알고 싶으면 다음을 입력

```
>>doc sin
```

- ✓ **sin**의 인터넷 자료가 담긴 참고 브라우저(browser)가 별도로 열림

- 필요한 함수명을 정확히 모르는 경우 비슷한 이름으로 찾고 싶으면 다음을 입력

```
>>lookfor sine
```

- ✓ 영문의 철자와 겹치는 다른 함수들도 나열, 와일드카드(wild-card) 기능

3

산술 연산자

● 매트랩의 가장 기본적인 기능은 공학용 계산기처럼 사칙연산을 수행한다.

- 덧셈과 뺄셈: +와 - 부호 사용
- 곱셈: × 부호 대신 * (별표) 사용
- 오른쪽 나눗셈: /
- 왼쪽 나눗셈: \

- ✓ 매트랩 명령창에 w 키로 입력

- ✓ 입력한 w는 사용중인 컴퓨터에 따라서 w 혹은 \로 명령창에 표시

4

산술 연산자

연습 1 산술 연산자: 지수승과 지수함수

지수승 1.23×10^{-10} 과 $1.23 \times 10^{+10}$, 지수함수 e^{-3} 을 매트랩에 입력하고 결과를 설명하라.

Tip !

- ✓ 수학적으로 밑수 2의 5지수승은 2^5 로 표현하지만, 매트랩에서는 2^5 로 표시
- ✓ 지수함수에 대한 매트랩 명령어는 **exp**

5

산술 연산자

- 소괄호는 매트랩에서 중요한 연산자 중 하나
 - ✓ 복잡한 수식을 간결하게 만들어 실제 코딩(coding)의 오류를 줄임.
- 산술 연산자의 우선순위
 - () → exp → *, / → +, -
 - ✓ 소괄호가 중첩되어 있는 경우는 가장 안쪽부터 바깥쪽으로 확장 해석
- 우선순위 없이 연산 실행하면 부정확한 결과 발생
 - ✓ 매트랩에서는 경고나 오류 메시지를 표시하지 않음
 - ✓ 전체 프로그래밍에 심각한 영향을 미침
 - ✓ 비슷자의 결과 생성: -Inf, Inf, NaN(Not-a-Number)

6

등호 지정 연산자

- 수학에서 등호 : 왼쪽과 오른쪽이 같음을 나타내는 부호
- 매트랩에서 등호 :
 - 수학에서 $x=3$ 은 변수 x 가 3과 같다는 의미
 - 매트랩에서 $x=3$ 은 변수 x 에 3이라는 수를 임시로 저장하는 의미
 - $x=3$ 일 때, 수학에서 $x=x+2$ 를 풀면
 - ✓ $0=2$ 라는 거짓 결과
 - $x=3$ 일 때, 매트랩에서 $x=x+2$ 를 풀면 ?

7

등호 지정 연산자

연습 2 등호 지정 연산자

수식 $x=x+2$ 만을 입력한 결과와 변수 x 에 3을 지정한 후 입력한 결과를 확인하라.

Tip !

- ✓ 매트랩에서 등호 지정 연산자 `=`의 의미를 확인
- ✓ 등호 지정 연산자 왼쪽에 있는 변수 이름은 영어 대문자, 영어 소문자, 숫자와 아래 밑줄을 혼합하여 31글자까지 사용 가능
- ✓ 등호 지정 연산자 오른쪽에는 상수 혹은 반드시 초깃값이 지정된 변수와 상수가 혼합된 형태로 입력

8

행렬 입력 서식

- 매트랩은 특정 자료의 형식이나 행렬의 차원에 대한 선언이 불필요함

➤ 행렬 $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ 의 입력

➤ 매트랩 명령창의 입력과 그 결과

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

```
A =
```

```
1 2 3
```

```
4 5 6
```

```
7 8 9
```

9

행렬 출력 서식

- 매트랩 입력 서식 명령어의 형태에 의해서 출력 서식 지정된다

➤ 출력 서식 명령어 없이 $\frac{4}{3}$ 와 1.2345×10^{-6} 입력

✓ 실수 표시는 **1.333333...**의 무한소수와 **0.0000012345**

➤ 매트랩 출력 결과

```
>>C = [4/3 1.2345e-6]
```

```
C =
```

```
1.3333 0.0000
```

✓ 매트랩의 디폴트(default, 기본 권장 사항)인 소수점 네 자리까지만 표시

➤ 출력 서식 명령어 **format short** 지정: 디폴트 결과와 같음

```
>>format short
```

```
>>C = [4/3 1.2345e-6]
```

```
C =
```

```
1.3333 0.0000
```

10

행렬 출력 서식

● 그 밖의 사용 가능한 출력 서식 명령어

- **format long** 명령어는 소수점 15자리까지 표시: 수치해석에서 자주 사용

```
>>format long
>>C = [4/3 1.2345e-6]

C =
    1.333333333333333    0.000001234500000
```

```
>>format short e
>>C = [4/3 1.2345e-6]

C =
    1.3333e+00    1.2345e-06
```

```
>>format long e
>>C = [4/3 1.2345e-6]

C =
    1.333333333333333e+00    1.234500000000000e-06
```

11

저장과 끝내기

● 중간에 작업을 멈추고 입력된 자료들을 안전하게 저장하고 명령창을 종료

- 이런 이유로 스크립트 파일 사용을 권장
- 명령창에 입력한 내용을 'data'의 이름으로 입력하는 방법

```
>>save data
```

- 매트랩 명령창 종료 방법

```
>>quit
```

- 혹은

```
>>exit
```

- 매트랩 명령창을 다시 열어 저장된 data 프로그램 불러오는 방법

```
>>load data
```

12

그래프 그리기 기초

● 매트랩 그래프

: 결과를 시각적으로 표현하고 실험이나 수식을 이용해서 얻은 복잡한 수치 자료를 명쾌하게 해석

- 보편적으로 선형 x - y 그래프가 많이 사용
 - ✓ 입력 자료를 x 축(가로축)에 놓는다
 - ✓ 출력 자료를 y 축(세로축)에 놓는다.
- 선형 x - y 그리기 명령어는 **plot**, **loglog**, **semilogx**, **semilogy**, **polar**
 - ✓ 매트랩에서 함수명 혹은 명령어는 영어의 소문자 사용이 원칙
 - ✓ 각 도표의 축의 크기와 그래프의 결과가 조금씩 다르게 표시
- 수치해석에서 가장 효과적으로 사용하는 **plot** 명령어만 설명

13

그래프 그리기 기초

● plot 명령어

- x 축에 자료를 입력하지 않고 y 축 원소 일곱 개에 대한 선 그리기
- 아래와 같이 명령어를 입력하고 Enter↵ 키를 누른다.

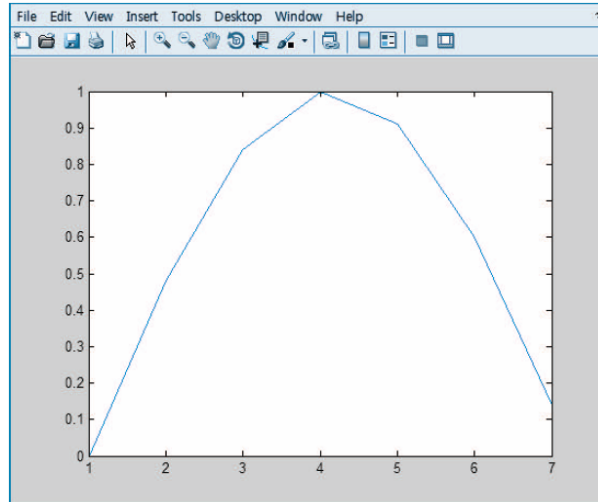
```
>>Y=[0.48.84 1.91.6.14];
>>plot(Y)
```

- 매트랩 그림창이 자동으로 생성
 - ✓ x 축의 1~7까지의 수는 입력된 자료가 아니라 원소 일곱 개에 상응하여 표시되는 색인

14

그래프 그리기 기초

[그림 1] Y의 원소 7개를 이용한 선 그리기



15

그래프 그리기 기초

- [그림 1]의 그래프는 어떤 자료의 무엇을 보여준다는 설명이 없다
- 문제점을 해결하고 그래프를 확실하게 이해하기 위하여 다음 명령어 추가

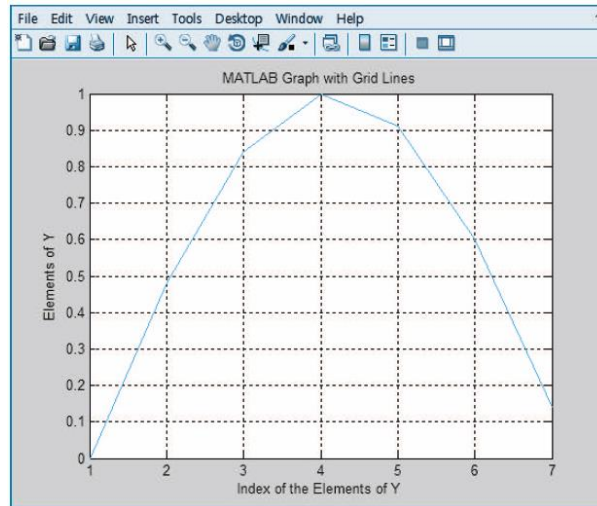
```
>>title('MATLAB Graph with Grid Lines')
>>xlabel('Index of the Elements of Y')
>>ylabel('Elements of Y')
>>grid
```

- [그림 2]에는 제목 및 가로축과 세로축 꼬리표(**label**)와 모눈(**grid**)을 붙임
 - ✓ [그림 1]과 비교하면 그래프를 좀 더 쉽게 이해할 수 있음

16

그래프 그리기 기초

[그림 2] 시각적으로 향상된 [그림 1]



17

그래프 그리기 기초

➤ 한 번 입력한 자료로 두 개의 그래프 비교하기

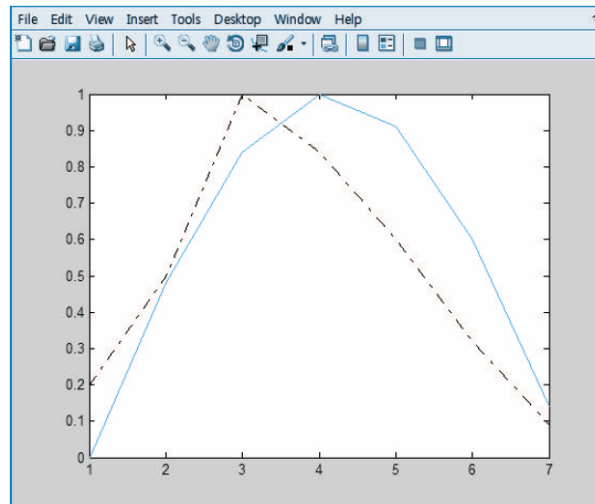
```
>>X=[1 2 3 4 5 6 7];
>>Y1=[0 .48 .84 1 .91 .6 .14];
>>Y2=[.2 .50 1 .84 .6 .32 .09];
>>plot(X,Y1, '- ', X,Y2, '-.')
```

- ✓ Y1과 Y2의 입력 순서는 바뀌어도 무관
- ✓ 겹치는 선의 구별: Y1에는 직선을 Y2에는 점선을 생성하는 표시 삽입

18

그래프 그리기 기초

[그림 3] 그림창에 선 두 개 그리기



19

그래프 그리기 기초

➤ 다양하게 매트랩에서 제공되는 자료 기호, 선과 색상을 [표 1]에 표시

[표 1] 그리기에 이용하는 다양한 선과 표시 종류

자료 기호	선 형태	색상
동그란 점 (.)	실선 —	검정색 k
별표 (*)	일점쇄선 -.	파란색 b
x 표 (x)	쇄선-점선 혼합 -.	청록색 c
원형 (o)	점선 :	초록색 g
플러스 부호 (+)		자홍색 m
정사각형 (□)		빨간색 r
다이아몬드 (◇)		흰색 w
다섯뿔쪽한 별 (★)		노란색 y

➤ [표 1]에서 세 가지 형태를 혼합해서 사용하면 복잡하게 그려지는 여러 선을 확실하게 구별할 수 있다.

20

그래프 그리기 기초

● hold 명령어

- 선을 하나 생성한 이후에 다음에 그려지는 그림을 연속으로 추가하여 그린다
- 중간마다 자료에 대한 확인과 수정 작업을 함께 진행
- **hold** 명령어를 사용하여 [그림 3] 다시 생성하는 방법

```
>>X = [1 2 3 4 5 6 7];
>>Y1 = [0 .48 .84 1 .91 .6 .14];
>>plot(X,Y1, '-.')
>>hold

Current plot held
>>Y2 = [.2 .50 1 .84 .6 .32 .09];
>>plot(X,Y2, '-.')

```

- **hold** 기능 중지: **hold** 명령어를 번갈아 입력하면 켜고/끄기 반복

```
>>hold

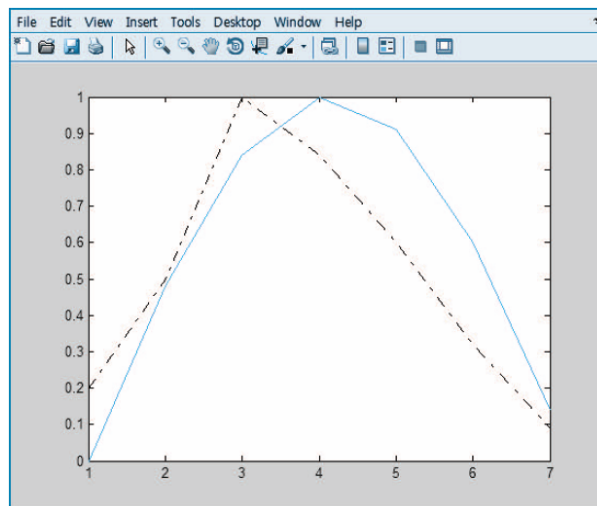
Current plot released

```

21

그래프 그리기 기초

[그림 4] hold 명령어로 선 두 개 그리기



22

그래프 그리기 기초

● colon 명령어

- 매트랩에서 세미콜론과 함께 콜론(colon)도 중요한 문장 부호
- 연산자와 와일드 카드의 기능을 동시에 가짐
- 콜론 연산자 없이 1~7까지 규칙적으로 1만큼 증가하는 자료를 변수 x 에 지정

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

✓ 1000개의 원소를 입력하면 몹시 지루하고 많은 시간을 낭비

- 콜론 연산자를 이용하여 다시 입력

```
>>x = [1:1:7];
```

✓ 첫 번째 1은 초깃값, 두 번째 1은 증가값, 마지막 7은 최종값

23

그래프 그리기 기초

- 중괄호 빼고 사용 가능

```
>>x = 1:1:7;
```

✓ 매트랩에서 원소를 벡터로 표시할 때에는 중괄호로 여담음

- 콜론 연산자를 이용하여 7~1까지 규칙적으로 하나씩 감소하는 자료를 z 에 지정

```
>>z = 7:-1:1;
```

✓ 반드시 초깃값이 최종값보다 큰 수로 지정

24

그래프 그리기 기초

연습 3 콜론 연산자

$0 \sim 2\pi$ 까지 $\frac{\pi}{4}$ 씩 규칙적으로 증가하는 벡터 **t**를 써라.

Tip !

✓ colon 연산자 이용

25

그래프 그리기 기초

● gtext, text, legend, subplot 명령어

- 복수로 그려진 그래프를 좀 더 분명하게 구별하기 위해서 사용
- **gtext** 명령어는 필요한 선의 수만큼 입력
 - ✓ 문자 입력을 위한 커서(위치를 나타내는 기호)가 표시
- **text** 명령어는 그래프를 생성하기 전에 미리 입력할 문자열의 좌표 위치 설정
- 범례 표시 명령어 **legend**는 여러 그래프를 구분하기 위해서 자주 사용
 - ✓ 반드시 **plot** 명령어 이후에 실행
- **subplot** 명령어는 여러 그래프를 한 화면에 나누어 그리는데 사용
- **subplot(mnp)**의 형태로 사용
 - ✓ m은 그림창 안에 분할시키는 행의 수, n은 열의 수, p는 분할된 칸 들의 위치
 - ✓ p의 자리수는 행과 열을 곱한 수 이상은 지정할 수 없다

26

그래프 그리기 기초

연습 4 gtext 명령어를 사용하여 그래프에 제목 달기

아래 명령어를 직접 입력하고 그 결과를 설명하라.

```
Command Window
>> t=0:pi/180:2*pi;
>> y=sin(t);
>> z=cos(t);
>> plot(y);
>> axis([0 360 -1 1]); ..... ①
>> hold;
Current plot held
>> plot(z,'-');
>> axis([0 360 -1 1]);
>> xlabel('Degree');
>> ylabel('Magnitude');
>> title('sine and cosine function');
>> grid;
>> gtext('sin(t)');
>> gtext('cos(t)');
```

27

그래프 그리기 기초

연습 5 text 명령어를 사용하여 그래프 생성하기

아래 명령어를 직접 입력하고 그 결과를 설명하라.

```
Command Window
>> x=0:pi/180:2*pi;
>> y=sin(x);
>> z=cos(x);
>> plot(x,y,x,z);
>> axis([0 2*pi -1 1]);
>> xlabel('Radian value');
>> ylabel('Magnitude');
>> title('Sine and cosine function');
>> gtext('sin(x)'); ..... ①
>> text(1.5,-0.6,'cos(x)'); ..... ②
```

28

그래프 그리기 기초

연습

6

legend 명령어를 사용하여 그래프 생성하기

아래 명령어를 직접 입력하고 그 결과를 설명하라.

```

Command Window
>> x=0:pi/180:2*pi;
>> y=sin(x);
>> z=cos(x);
>> plot(x,y,x,z, ':'); ..... ❶
>> axis([0 2*pi -1 1]);
>> xlabel('Radian value');
>> ylabel('Magnitude');
>> title('Sine and cosine');
>> legend('sin(x)', 'cos(x)', 3); ..... ❷

```

29

그래프 그리기 기초

연습

7

subplot 명령어를 그리기 위한 매트랩 프로그램

아래 명령어를 직접 입력하고 그 결과를 설명하라.

```

Command Window
>> t=0:pi/180:2*pi;
>> y1=sin(t);
>> y2=cos(t);
>> y3=y1+y2;
>> y4=y1-y2;
>> subplot(221);
>> plot(y1);
>> axis([0 360 -2 2]);
>> xlabel('Degree');
>> ylabel('Magnitude');
>> title('Sine function');
>> subplot(222);
>> plot(y2);
>> axis([0 360 -2 2]);
>> xlabel('Degree');
>> ylabel('Magnitude');
>> title('Cosine function');

```

30

그래프 그리기 기초

연습 7 (계속)

```
>> subplot(223);
>> plot(y3);
>> axis([0 360 -2 2]);
>> xlabel('Degree');
>> ylabel('Magnitude');
>> title('Addition function');
>> subplot(224);
>> plot(y4);
>> axis([0 360 -2 2]);
>> xlabel('Degree');
>> ylabel('Magnitude');
>> title('Subtraction function');
```

31

3차원 그리기

- 입체적으로 그래프를 생성할 수 있는 3차원 그리기 소개
- 3차원 공간 그리기
 - **plot3** 명령어는 직선들을 연결하여 3차원 공간에 차례로 그린다
 - ✓ 세 개의 직선이 서로 조화를 이루어 입체적으로 그려진다.

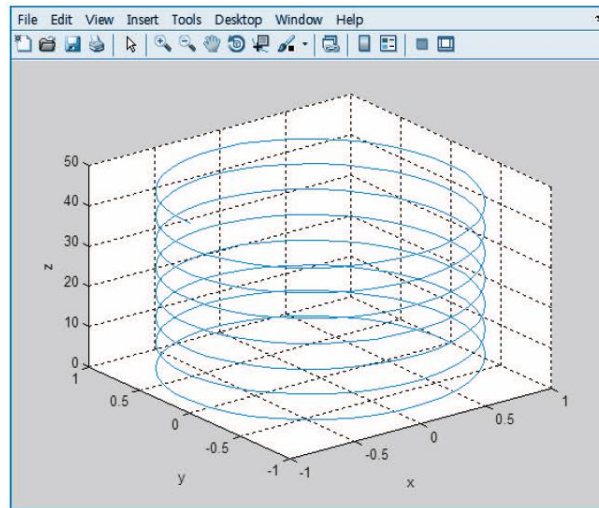
[그림 5] 나선형 모양을 생성시키는 프로그램의 입력 화면

```
Command Window
>> t=0:pi/20:15*pi;
>> x=cos(t);
>> y=sin(t);
>> plot3(x,y,t);
>> xlabel('x');
>> ylabel('y');
>> zlabel('z');
>> grid;
```

32

3차원 그리기

[그림 6] 실행 결과: `plot3` 명령어



33

3차원 그리기

● 3차원 표면 그리기

- `mesh`와 `surf` 함수를 사용하여 3차원 표면을 그린다
- `surf` 함수는 `mesh` 함수와 기능이 같지만, 컬러 표면을 그릴 때 사용
- `contour` 함수는 등고선을 그릴 때 사용
 - ✓ 3차원적인 표면을 2차원적으로 묘사

[표 2] 3차원 표면 그리기 함수 비교

함수 이름	차이점
<code>mesh</code>	격자 무늬만 그리기
<code>surf</code>	격자 무늬 사이에 컬러를 채워서 그리기
<code>contour</code>	2차원 등고선 그리기

34

3차원 그리기

● mesh 함수

- $-4 \leq x \leq 4$ 와 $-4 \leq y \leq 4$ 의 범위에 0.2 간격으로 놓여 있는 함수 $z = x^2 - 2y^2$ 의 표면을 **mesh** 함수를 이용해서 그려보는 프로그램의 입력 화면

[그림 7] 입력 화면

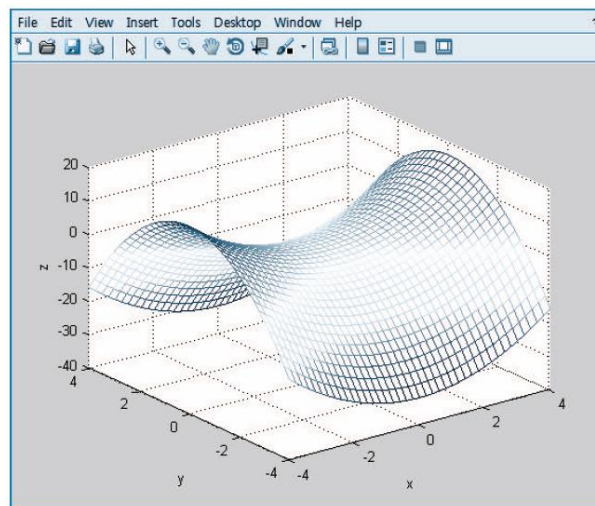
```
Command Window
>> [x, y] = meshgrid(-4:0.2:4); ..... ①
>> z = x.^2-2*y.^2; ..... ②
>> mesh(x, y, z);
>> xlabel('x');
>> ylabel('y');
>> zlabel('z');
```

- ①번은 x와 y의 격자무늬를 동시에 생성하는 입력 방법
- ②번에서 사용한 마침표는 1.5.3절에서 자세히 배울 예정

35

3차원 그리기

[그림 8] 실행 결과: mesh 함수



36

3차원 그리기

● mesh를 변형시킨 함수

➤ meshc, meshz, waterfall

- ✓ **meshc**: 표면 그림 밑으로 등고선을 동시에 표시
- ✓ **meshz**: 표면 그림 밑으로 x 축과 y 축 방향으로 연속적 수직선 표시
- ✓ **waterfall**: 표면 그림 밑으로 x 축 혹은 y 축 한 방향으로만 연속적 수직선 표시

[표 3] mesh 그리기 변형 함수 비교

함수 이름	차이점
meshc	표면 아래 등고선 그리기
meshz	표면 아래 x 축과 y 축 양방향으로 수직선 그리기
waterfall	표면 아래 x 축 혹은 y 축 한 방향으로 수직선 그리기

37

3차원 그리기

연습 8 mesh 그리기 변형 함수

함수 **meshc**, **meshz**, **waterfall**을 이용하여 함수 $z = x^2 - 2y^2$ 에 대한 3차원 그림을 그려라. 이때 범위는 $-4 \leq x \leq 4$ 와 $-4 \leq y \leq 4$ 이며, 각각의 간격은 0.2로 지정한다.

Tip !

- ✓ [그림 7]의 입력 프로그램에서 **mesh** 함수가 들어간 자리에 **meshc**, **meshz**, **waterfall**을 대입

38

3차원 그리기

● surf 함수

- mesh 함수와 비교하면 격자무늬 사이에 컬러가 더 뚜렷하게 채워짐

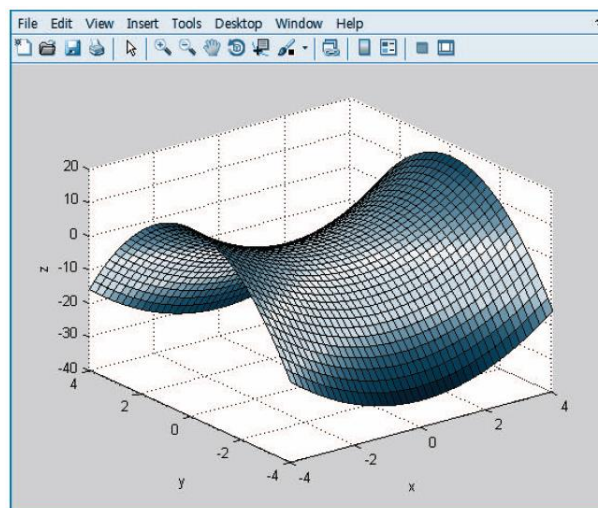
[그림 9] 입력 화면

```
Command Window
>> [x, y] = meshgrid(-4:0.2:4);
>> z = x.^2-2*y.^2;
>> surf(x, y, z);
>> xlabel('x');
>> ylabel('y');
>> zlabel('z');
```

39

3차원 그리기

[그림 10] 실행 결과: surf 함수



40

3차원 그리기

● contour 함수

[그림 11] 입력 화면

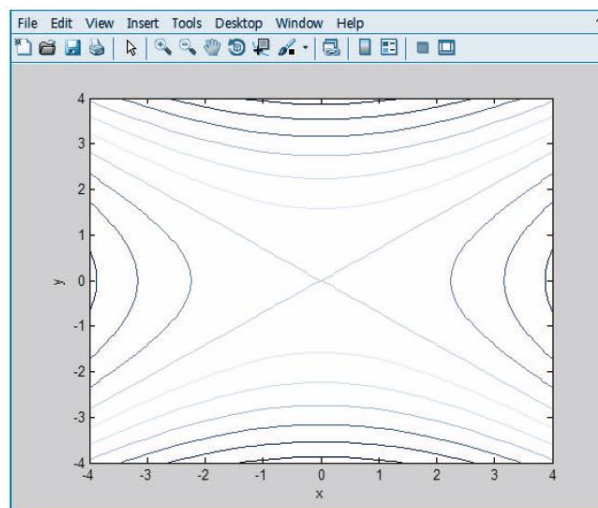
```
Command Window
>> [x, y] = meshgrid(-4:0.2:4);
>> z = x.^2-2*y.^2;
>> contour(x, y, z);
>> xlabel('x');
>> ylabel('y');
```

- [그림 7]에서 **mesh** 함수를 **contour** 함수로 바꿈
- **zlabel** 명령어는 불필요
 - ✓ 2차원적인 등고선 그림을 묘사

41

3차원 그리기

[그림 12] 실행 결과: contour 함수



42

저장과 출력

● ‘file’ 이라는 이름으로 완성된 그림 파일 저장하기

- 다음과 같은 순서로 클릭
[File] > [Save As]
- 선택 상자 안에 매트랩 그림 파일의 확장자 .fig를 사용하여 ‘file.fig’로 저장
 - ✓ .fig 반드시 매트랩이 설치되어 있고 매트랩창을 연 후에만 사용 가능
- 저장된 ‘file.fig’ 파일을 다시 사용하려면 다음과 같은 순서로 클릭
[File] > [Open] > file.fig
 - ✓ 스크립트 파일을 저장하는 동일한 디렉터리에 그림 파일도 저장
- 그림 파일을 프린터로 출력하려면 다음과 같은 순서로 클릭
[File] > [Print] > 프린터 이름 선택 > [OK]

43

프로그래밍이란?

● 프로그램 개발의 순서

- (1) 구조화된 프로그램을 위한 알고리즘 작성
 - 알고리즘은 명령어들의 순서를 정하고 연속적으로 명령어를 수행
 - 조건 작동 혹은 반복 작동과 같은 제어 알고리즘 구조 이용
 - ✓ 수행하는 중간에 일부 명령어 변경 가능
- (2) 프로그램을 종합된 동작을 하도록 모듈(module) 혹은 함수라고 부르는 작은 부분으로 분할
 - ✓ 모듈은 프로그램 설계를 위한 중요 기술 중 하나
 - ✓ 모듈은 입력 하나에 출력 하나를 얻는 구성
 - ✓ 모듈은 프로그램의 다른 부분 응용에도 쉽게 재사용
 - ✓ 매트랩에서는 내장함수 혹은 사용자정의함수가 모듈의 역할

44

프로그래밍이란?

- (3) 프로그램을 효과적으로 생성하기 위해 구조도 혹은 흐름도를 이용
 - 구조도는 전체 프로그램 상태를 직사각형을 연결시킨 그래프로 설명
 - 흐름도는 조건문을 이용하여 흐름을 제어하고 분기시키면서 결과에 도달
 - ✓ 화살표, 마름모, 직사각형을 이용
 - 프로그램 실행 이전에 기계코드 번역이 필요한 경우
 - ✓ 의사 코드(pseudocode)로 표시하는 것도 효과적

45

프로그래밍이란?

● 디버깅(debugging)

- 프로그램 제작할 때 발생하는 오류 혹은 버그(bug)를 찾고 제거하는 과정

● 매트랩에서 발견되는 오류의 종류

- (1) 구문상 오류
 - 괄호나 심표를 생략하거나 혹은 명령어 철자를 틀릴 때 발생
- (2) 실행 시간 오류
 - 프로그램 실행 중에 발생
 - 부정확한 수학적 표기 방법이 원인
 - ✓ 매번 발생하는 것이 아니라 특정한 경우를 입력했을 때만 발생

46

프로그래밍이란?

● 실행 시간 오류 미리 방지하기

- 명령줄 끝에 세미콜론의 생략
 - ✓ 연산 결과들을 즉시 확인하여 오류를 수정
- 작은 부분으로 구성된 모듈의 테스트
 - ✓ 작업 중에도 프로그램이 원하는 방향으로 진행되는지를 확인

● 디버깅 과정의 필요성

- 부분 프로그램의 호환성 문제로 새로운 오류가 종종 발생
 - ✓ 큰 프로그램 설계는 여러 명 프로그래머의 공동 작업
- 제작에 참여한 모든 내부 프로그래머와 전문가들의 오류 검사만으로는 부족
 - ✓ 프로그램 개발에 참여하지 않은 외부 전문가들의 오류 검사 필요
- 내부와 외부의 검사를 완벽하게 통과한 후에 프로그램을 출시

47

매트랩 변수와 함수

● 매트랩 변수명 생성하기

- 지정 연산자 등호 (=) 사용
 - ✓ 오른쪽에는 다양하게 수들의 혼합, 수학 연산자, 변수 지정
 - ✓ 왼쪽에는 새롭게 사용할 변수명 지정
- 새로운 변수명의 첫 글자는 영어의 소문자 혹은 대문자로 시작
 - ✓ 영문 이후에는 다른 영문자, 수, 밑줄(_)을 사용하여 최대 31자까지 가능
 - ✓ 반드시 소문자와 대문자는 구별하여 사용, 'Name'과 'name'은 다른 변수명
 - ✓ 매트랩에서 제공하는 내장변수명, 내장함수명, 혹은 이미 지정된 사용자정의함수명과 중복 사용 금지

48

매트랩 변수와 함수

● 내장 변수

[표 4] 매트랩 내장 변수 예

변수명	변수명의 의미
ans	수식을 연산할 때 특정 변수명을 지정하지 않더라도 수식의 결과에 자동 지정되는 내장변수명으로 사용
eps	부동 소수점 정밀도를 표시
pi	원주율 π (3.141592.....)를 표시
Inf	수학적 불능 연산 1/0의 결과를 나타내는 무한대 ∞
NaN	수학적 부정 연산 0/0의 결과를 나타내는 비 숫자 표시

- $\sqrt{-1}$ 로 정의된 복소수의 허수 단위인 i 혹은 j 도 매트랩의 내장 변수
- 허수 단위들은 일반 변수명으로 사용 가능

49

매트랩 변수와 함수

- 일반 변수명 a 는 ①번에 앞서서 지정
- ②번에서 내장변수 i 는 지정할 필요 없음
- ③번과 ④번에서 i 와 j 는 허수 단위 변수가 아닌 일반 변수로 취급
- 그러므로 ⑤번의 결과는 0이 아님

[그림 13] 예외적인 내장변수 보기

```

Command Window
>> x=1+2*a ..... ①
??? Undefined function or variable 'a'.

>> y=1+2i ..... ②

y =

    1.0000 + 2.0000i

>> i=5; ..... ③
>> j=8; ..... ④
>> u=sqrt(i-j) ..... ⑤

u =

    0 + 1.7321i

```

50

매트랩 변수와 함수

● 매트랩의 내장함수

: 항상 소문자로 표기하고 괄호를 이용하여 필요한 인자를 삽입

➤ 내장함수들은 도움 기능 명령어를 이용해서 찾음

➤ **sqrt**: 제곱근을 표시

```
>>sqrt(16)

ans =
     4
```

➤ **abs**: 절댓값을 표시

```
>>abs(-5)

ans =
     5
```

51

매트랩 변수와 함수

● ➤ **log**: 수학적 자연로그를 표시

```
>>log(128)

ans =
 4.8520
```

➤ **log10**: 밑수 10인 로그를 표시

```
>>log10(128)

ans =
 2.1072
```

52

매트랩 변수와 함수

[그림 14] 한 줄에 문장 여러 개를 표시

- 문장 여러 개를 한 줄에 실행하려면 문장과 문장 사이에 쉼표 사용
 - ✓ 장점: 진행 중인 작업 상황을 시각적으로 편하게 관리
 - ✓ 단점: 잘못 작성한 문장의 수정이 번거로움
- 문장 끝에 세미콜론을 사용하면 진행 과정의 결과를 숨기고 생략하면 결과 확인



```

Command Window
>> a=10;
>> b=sin(a)

b =

    -0.5440

>> c=cos(a)

c =

    -0.8391

>> a=10; b=sin(a), c=cos(a)

b =

    -0.5440

c =

    -0.8391
  
```

53

매트랩 변수와 함수

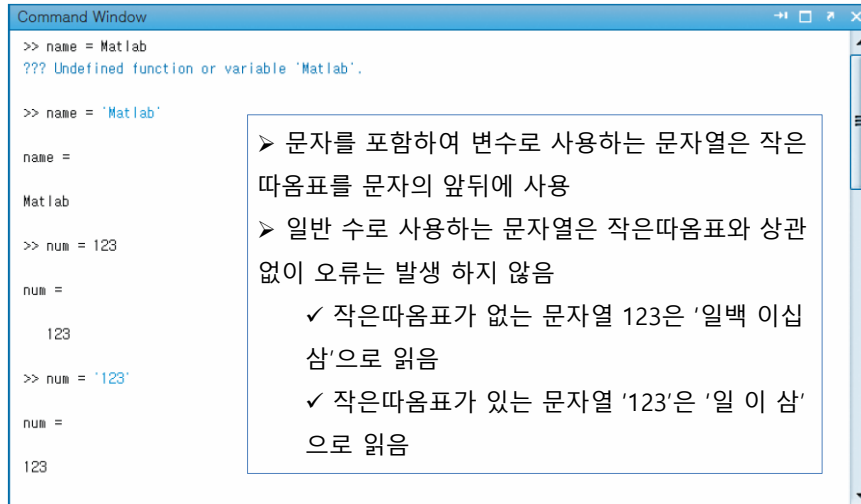
● 매트랩의 명령창 지우기 명령어

- **clc** 명령어는 명령줄을 지우고 명령창에 맨 처음 명령줄 위치로 돌아감
 - ✓ 이전에 입력한 내용은 계속 보관하기에 다시 사용 가능
- **clear** 명령어는 명령줄은 그대로 유지
 - ✓ 지금까지 입력한 내용은 더 불러낼 수 없음
 - ✓ 작업 중간마다 중복되어 잘못 지정되는 변수의 혼란 방지를 위해 사용

54

매트랩 변수와 함수

[그림 15] 문자열 변수 지정



The image shows a MATLAB Command Window with the following commands and outputs:

```
>> name = Matlab
??? Undefined function or variable 'Matlab'.

>> name = 'Matlab'

name =

Matlab

>> num = 123

num =

    123

>> num = '123'

num =

    123
```

Annotations in the image explain the behavior of strings in MATLAB:

- 문자를 포함하여 변수로 사용하는 문자열은 작은 따옴표를 문자의 앞뒤에 사용
- 일반 수로 사용하는 문자열은 작은따옴표와 상관 없이 오류는 발생 하지 않음
 - ✓ 작은따옴표가 없는 문자열 123은 '일백 이십 삼'으로 읽음
 - ✓ 작은따옴표가 있는 문자열 '123'은 '일 이 삼'으로 읽음

55