

2016년 1학기 맞이~ MATLAB Workshop

2016. 2.22(월) ~ 2.23(화)

Matlab/Simulink 기본 사용법 및 응용

- 목 차 -

1. 매트랩 개요
2. 숫자, 셀과 구조 배열
3. 시뮬링크
4. 응용 예; 자동제어

경 북 대 학 교

IT대학 전자공학부

김 지 훈

MATLAB이란?

- ‘MATrix LABoratory’의 약어. 수치해석과 신호처리 그리고 편리한 그래픽 기능 등을 통합하여 고성능의 수치계산과 결과를 보여주는 프로그램.

- 특징

- 행렬(또는 배열) 기반의 수치 계산
 - 인터프리터(interpreter) 방식의 공학전용 언어 (M-file도 사용가능)
 - 그래픽 처리의 간편함 및 고급화
 - 다양한 응용분야별 라이브러리 제공(Toolbox)
 - C 프로그램과의 연계성
 - 실시간 하드웨어 제어 가능
 - Simulink
-
- 이전에는 주로 신호처리와 수치해석 분야에서 전문가들에 의해 사용
 - 최근에는 과학, 수학 및 대부분의 공학 분야에 널리 보급되어 활발히 사용
=> 많은 대학에서 저학년 교육과정으로 채택
-
- 이 강의의 목적은 “Matlab과 Simulink의 기본 사용법을 익힘으로써, 실험 및 이론 학습에 도움”을 주고자 함

1. 매트랩 개요

1.1 매트랩 : 상호대화식(interactive) 계산기

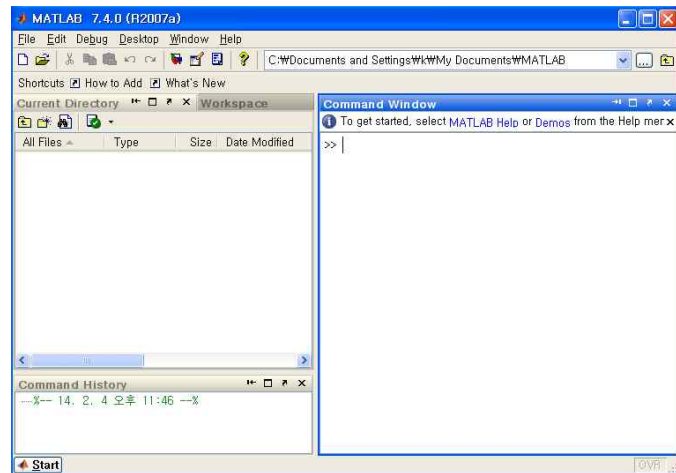
=> 매트랩 시작, 기본 계산법, 매트랩 종료



P 매트랩 시작 : 매트랩 아이콘을 더블클릭하면 작업화면(desktop)이 나타남

– 작업화면의 구성

- ① 명령창(command window)
- ② 명령이력창(command history)
- ③ 현재 디렉토리창
(current directory)
- ④ 워크스페이스(workspace)



– 명령창 : ‘명령어’, ‘함수’, ‘문장’ 등을 입력하여 ‘매트랩 프로그램’ 작성

=> 명령창의 프롬프트(>>)는 명령어를 받아들일 준비가 되었음을 나타냄

=> 명령어를 입력하기 전에 커서(cursor)가 프롬프트 바로 뒤에 위치시킴

– Current directory window : 파일관리자와 유사

=> 확장자 ‘.m’으로 된 파일을 더블 클릭하면 '매트랩 Editor'에서 열수 있음

– Workspace : current directory 창 위쪽의 Workspace 탭을 클릭

=> 명령창에서 만든 변수들을 나타냄

=> 변수 명을 더블 클릭하면 배열 편집창(Array editor)으로 확인 가능

– Command history window : 명령창에서 이전에 입력된 사항을 보여줌

=> 입력된 부분을 클릭하거나 편집기에 드래그하면 재사용 가능

– 작업화면 변경 : 각각의 창 윗부분에 있는 을 이용

※ 디폴트 환경을 복원하기 위해서는 ‘Desktop/Desktop Layout/Default’ 선택

▶ 명령어와 식의 입력 : 사용방법 간단, 예를 통해 확인

- 명령창의 프롬프트 뒤에 명령어를 입력

=> (Interactive) session : 작업자와 매트랩 사이의 상호작용

| 작업내용 | Command window | 설명 |
|-----------------------------|--|---|
| 8 나누기 10 | >> 8/10 Enter↵ ans = 0.8000 | 매트랩은 계산시 높은 정밀도를 가지지만, 결과가 정수일 경우를 제외하고 보통 소수점 이하 4자리 십진수로 표현 |
| 'ans'는 answer의 약어 | >> 5*ans Enter↵ ans = 4 | 매트랩에서 변수(variable)는 값을 가질 수 있는 기호 변수 ans는 이제 4가 됨 |
| 식에 사용하기 위해 변수를 사용 | >> r=8/10 Enter↵ r = 0.8000 | ans 대신 변수 r에 결과를 할당 |
| 변수의 값을 확인 | >> r Enter↵ r = 0.8000 | 변수 r의 값이 0.8임을 확인 |
| 이 변수는 다음 단계의 계산에 사용 가능 | >> s = 20 * r Enter↵ s = 16 | 만약 곱셈 기호 '*'를 생략하고, 수식을 's = 20r'로 입력하면 오류 메시지가 나타남 |
| 제곱근 함수 : sqrt (square root) | >> r=sqrt(9) r = 3 | r의 이전의 값이 3으로 바뀜 ※ 라인 끝에 세미콜론(:)을 입력하면 화면에 결과가 나타나지 않음 |

- 매트랩에서는 명령 파일에서 이전의 키 입력을 기억하고 있으므로 ↑, ↓키를 이용하여 사용되었던 명령들을 스크롤(scroll) 가능

=> 원하는 명령줄을 찾아 ←, →, Del, BackSpace를 이용하여 다시 편집 가능

- 스칼라 산술 연산과 연산의 우선순위

| 기호 | 연산 | 매트랩 형식 |
|----|-------------|--------|
| ^ | 지수 : a^b | a^b |
| + | 덧셈 : $a+b$ | $a+b$ |
| - | 뺄셈 : $a-b$ | $a-b$ |
| * | 곱셈 : ab | $a*b$ |
| / | 나눗셈 : a/b | a/b |

=> 연산 우선순위 : 지수 → 곱셈/나눗셈 → 덧셈/뺄셈, 괄호는 가장 안쪽부터

=> 우선순위가 확실하지 않은 곳에서는 오류를 피하기 위해 괄호를 삽입

[예제1] 매트랩을 사용하여 다음 식들을 계산하라.

$$\textcircled{1} \ 6\left(\frac{10}{13}\right) + \frac{18}{5(7)} + 5(9^2) \qquad \textcircled{2} \ 6(35^{1/4}) + 14^{0.35}$$

▶ **할당연산자(=)** : 좌측의 변수에 우측의 값을 할당, 등호보다 더 많은 것을 의미

| 식 | 의미 |
|-------------------------|-------------------------|
| $x = 3$ | 변수 x에 3을 할당하라 |
| $x = x + 3$ | 현재 x값에 3을 더한 새로운 값으로 대체 |
| $6 = x$ $x + 2 = 20$ | 매트랩에서 사용 불가 |
| $x = y + 5$ | 변수 y에 어떤 값이 할당되어 있다면 가능 |

[예제2] 원기둥의 부피 : $V = \pi r^2 h$

- ① 높이가 15m이고 반지름이 8m인 원기둥의 부피는?
- ② 부피가 ①보다 20% 더 크고 같은 높이를 갖는 또 다른 원기둥을 만들려 한다. 이 경우, 원기둥의 반지름은 얼마가 되어야 하는가?



▶ **변수이름** : 반드시 문자로 시작, 나머지는 문자, 숫자, '_'(underscore)의 조합
=> 매트랩에서는 대/소문자를 구별, 변수이름은 63자 보다 길지 않아야 됨

▶ **작업 세션 관리**

| 명령어 | 설명 |
|---------------|--|
| clc | 명령창을 깨끗이 함(변수는 남아 있음) |
| clear (all) | 메모리로부터 모든 변수를 지움 |
| clear var1 | 메모리로부터 변수 var1을 지움 |
| exist('name') | 'name'이라는 이름의 변수나 파일이 존재하는지 알림 cf)help |
| quit/exit | 매트랩을 종료 |
| who | 현재 메모리의 변수를 나열 |
| whos | 현재의 변수들과 크기, 0이 아닌 허수부 등을 나열 |
| : | 콜론; 일정한 간격의 원소를 갖는 배열을 생성 |
| , | 콤마; 배열의 원소들을 분리 |
| ; | 세미콜론; 화면에 출력이 나타나지 않게 함, 혹은 새로운 행 |
| ... | 생략부호(마침표 세 개); 라인이 계속이어짐 |

| 사용 예 | 설명 |
|---|---|
| <pre>>> x=2; y=6+x, x=y+7 y = 8 x = 15</pre> | <p>결과가 나타나지 않게 세미콜론 사용, 같은 줄에 여러 개의 명령어들을 사용하기 위해 콤마 사용</p> <p>[결과] x의 처음 값이 나타나지 않았으며, x의 값은 2에서 15로 변경됨</p> |

▶ 탭과 화살표 키 : 입력의 편의성 증대

- 스마트 불러오기 : 변수의 처음 몇 문자를 입력 후,  키 누름
- 탭 완성 : 이름의 처음 몇 문자를 입력 후,  키 누름
- 화살표 키 : 이전에 사용한 명령어를 scroll

▶ 삭제 및 지우기 : , , (전체 라인 삭제), (라인 끝까지 삭제)

※ 매트랩은 종료나 변수 값을 지우기 전에는 그 변수의 마지막 값을 기억함

※ 함수와 명령어 또는 문장(statement)의 차이

=> 함수는 괄호 속에 인수를 가지만 명령어나 문장은 인수를 갖지 않음

※  : 오래 걸리는 계산을 취소

▶ 미리 정의된 상수

| 명령어 | 설명 |
|------|--------------------------------|
| ans | 가장 최근의 답을 나타내는 임시 변수 |
| eps | 부동 소수점의 정밀도를 규정 |
| i, j | 허수 단위 $\sqrt{-1}$, 곱셈기호 없이 사용 |
| Inf | 무한대 |
| NaN | 정의되지 않은 수치 결과 |
| pi | 파이 |

| 사용 예 | 설명 |
|------------------|------------------------------|
| c1=1-2i | 복소수 $c_1 = 1 - 2i$ 를 표현 |
| c1=complex(1,-2) | 허수단위 i, j와 숫자 사이에는 곱셈기호가 불필요 |
| y=7/2*i | $y = (7/2)i = 3.5i$ |
| y=7/2i | $y = 7/(2i) = -3.5i$ |

[예제3] $x = -5 + 9i$, $y = 6 - 2i$ 일 때, 매트랩을 이용하여 $x+y$, xy , x/y 를 구하고 손으로 구한 결과와 비교하라. $1+7i$, $-12+64i$, $-1.2+1.1i$

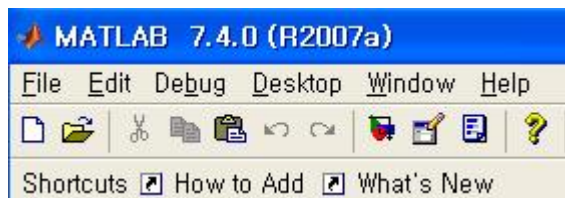
▶ 숫자 표시 형식 : format 명령어

- 매트랩에서 디폴트 포맷은 short 형식으로 소수점 이하 4자리를 사용

| 명령어 | 설명 |
|----------------|--------------------------------------|
| format short | 소수점 4자리(default) |
| format long | 16자리 |
| format short e | 5자리(소수점 4자리)와 지수 : $6.3792e+03$ |
| format long e | 16자리(소수점 15자리)와 지수, 여기서 e는 숫자 10을 의미 |
| format bank | 소수점 2자리(화폐 계산시) |
| format + | 결과의 양수, 음수, 0을 표시 |
| format rat | 유리수의 근사화 |
| format compact | blank line 억제 |
| format loose | 덜 간결한 형식으로 재설정 |

1.2 메뉴와 툴바

- 메뉴바는 창을 변경함에 따라 바뀜
- 메뉴들도 문맥에 따라 기능이 달라짐
- 툴바 아래의 버튼은 툴바에 단축키를 추가하기 위한 도움말 등임



1.3 배열, 파일 및 그래프 작성

▶ 배열 (혹은 행렬/벡터)

- 매트랩의 장점중의 하나는 배열(array)이라고 하는 수들의 모임을 하나의 변수처럼 다룰 수 있는 점

| 작업내용 | Command window | | 설명 |
|--|--|---|---|
| 0, 4, 3, 6을 순서대로 배열을 만들자 | <pre>>> x=[0, 4, 3, 6] Enter↵ x = 0 4 3 6</pre> | | 대괄호([])를 사용하여 배열로 묶음 콤마를 생략하고 space로 각각을 구분 할 수 있으나 콤마 사용이 바람직 |
| 배열은 순서를 가짐 | <pre>>> y=[6 3 4 0] Enter↵ y = 6 3 4 0</pre> | | 변수 y는 변수 x와 순서가 다르므로 서로 다른 배열 |
| 두 배열의 합 | <pre>>> z=x+y Enter↵ z = 6 7 7 6</pre> | | x와 y에 있는 모든 대응하는 숫자들을 더하여 z를 만듦 |
| 일정한 간격을 가지는 배열 만들기 예) 0, 0.1, 0.2,..., 10 | <pre>>> u=[0:0.1:10] Enter↵ u = 0 0.1000 0.2000 0.3000...</pre> | | u는 101개의 값을 가지는 배열 이런 경우 명령어의 맨 뒤에 ;을 쓰지 않는다면, |
| sine 함수 | <pre>>> w=5*sin(u); Enter↵</pre> | | 위에서 구한 u에 대하여, w=5sin(u)를 계산하기 위한 명령 실행결과 w는 101개의 값을 갖는 배열 |
| 배열의 참조 | <pre>>> u(7) ans = 0.6000</pre> | <pre>>> w(7) ans = 2.8232</pre> | u(7)을 입력하면, 배열 u의 7번째 값을 볼 수 있음, 숫자 7은 배열에서 특정한 원소를 가리키므로 배열 index라 함 |
| 배열(벡터)의 길이 | <pre>>> s=length(w) Enter↵ s = 101</pre> | | 배열에 얼마나 많은 값들이 있는지 알 수 있음 |

▶ 방정식의 근 (root)

- 매트랩에서 다항식의 표현
=> 다항식을 내림차순을 정리하여 계수들을 차례로 원소로 하는 배열로 표현
예) 다항식 $4x^3 - 8x^2 + 7x - 2$ 는 [4 -8 7 -2]와 같이 나타냄
- 다항식의 근을 구하기 위한 함수 : roots()

[예제4] ① 방정식 $x^3 - 7x^2 + 40x - 34 = 0$ 의 근을 매트랩을 이용하여 구하라.

$x=1, 3+5i, 3-5i$

② 매트랩을 이용하여 배열[cos(0):0.02:log10(100)]을 만들고 25번째 원소를 구하라. 그리고 배열에 얼마나 많은 원소들이 있는지 구하라. 1.48, 51

cf.) 예제4의 ①에서 명령어를 하나만 사용(한 줄로)하여 근을 구할 수 있을까?

▶ 내장함수

| 함수 | 매트랩 표현 | 함수 | 매트랩 표현 |
|---------------|----------|---------------|---------|
| e^x | exp(x) | $\sin x$ | sin(x) |
| \sqrt{x} | sqrt(x) | $\tan x$ | tan(x) |
| $\ln x$ | log(x) | $\cos^{-1} x$ | acos(x) |
| $\log_{10} x$ | log10(x) | $\sin^{-1} x$ | asin(x) |
| $\cos x$ | cos(x) | $\tan^{-1} x$ | atan(x) |

cf.) 위의 표에서 삼각함수는 라디안(radian)값을 사용. 단, sind(x)와 cosd(x)와 같은 경우에는 인수 x의 값이 각도(degree)임

▶ 파일작업 : M-파일/MAT-파일

- 매트랩에서 프로그램, 데이터와 세션 결과들을 저장시킬 수 있음
 - => 특히, 매트랩 함수들과 프로그램 파일들은 확장자 .m으로 저장(M-파일)
 - => 확장자 .mat로 매트랩 세션에서 생성되는 변수들의 이름과 값을 저장
- M-파일은 ASCII 파일이므로 어떤 문서 편집기로도 작성 가능
 - => but, MAT-파일은 2진법 파일이며 일반적으로 생성했던 s/w에서만 읽힘
- 데이터 파일(DAT-파일)은 아스키 형식에 따라 생성
 - => 워드 프로세서, 스프레드시트, 실험 데이터 획득 시스템 등에 사용 가능

▶ 작업공간 변수들의 저장과 복원 : save와 load 명령어

- 매트랩 사용을 중단하고 나중에 그 세션을 계속하려는 경우
 - => 명령창에 save를 입력하면, matlab.mat에 저장
 - => 불러오기 위해서 명령창에 load를 입력
- 선택적인 저장을 위해, "save filename var1 var2"와 같이 입력함
 - => 불러오기 위해서는 "load filename"을 입력함

cf.) 디렉터리와 경로(path) : 매트랩에 사용되는 파일들의 위치를 알 필요 있음
 명령어 pwd를 입력하면, 현재의 디렉터리를 알 수 있음
 => current directory window를 이용하면 편리

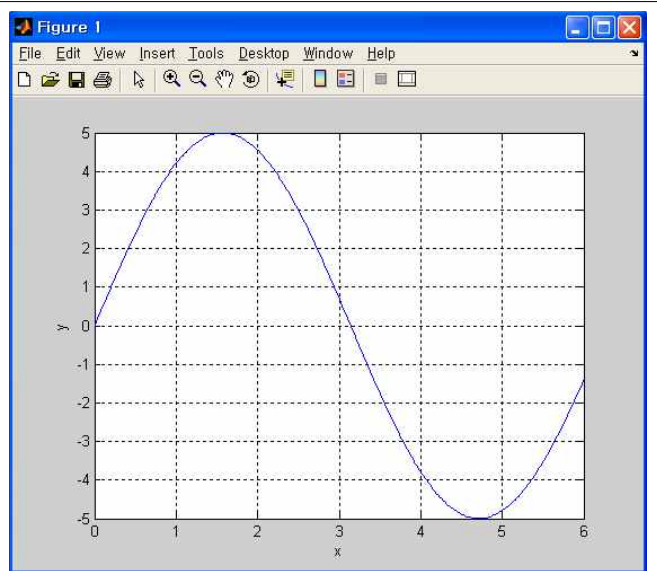
▶ 그래프 작성 plot()

- 직선, 대수, 표면, 등고선 그래프와 같은 여러 형태의 그래프 작성 가능

[Ex.1] $0 \leq x \leq 6$ 에서 $y = 5\sin x$ 의 그래프

```
>> x=[0:0.02:6];
>> y=5*sin(x);
>> plot(x,y),xlabel('x'),ylabel('y')
>> grid % 선을 나타냄
>> axis('equal')
>> title('y=5sin(x)')
```

xlabel : 작은따옴표 안에 있는
text를 수평축에 나타냄
ylabel : 수직축에 나타냄
title : 그래프 제목
gtext : 원마우스 클릭 위치에 글

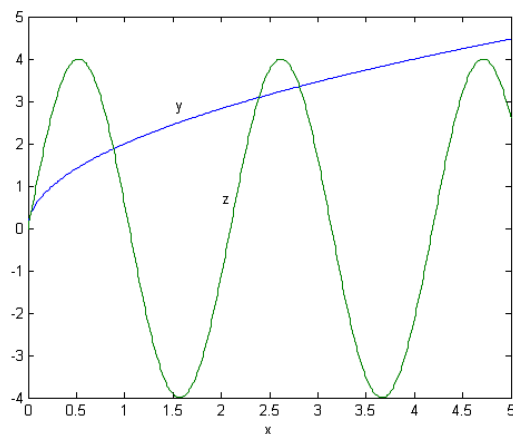


[Ex.2] $0 \leq x \leq 5$ 에서 $y = 2\sqrt{x}$ 와 $z = 4\sin 3x$ 의 그래프를 같이 그리자.

```
>> x=[0:0.01:5];
>> y=2*sqrt(x);
>> z=4*sin(3*x);
>> plot(x,y,x,z), xlabel('x')
>> gtext('y'), gtext('z')
```

이렇게 쓰면, 그래프는?

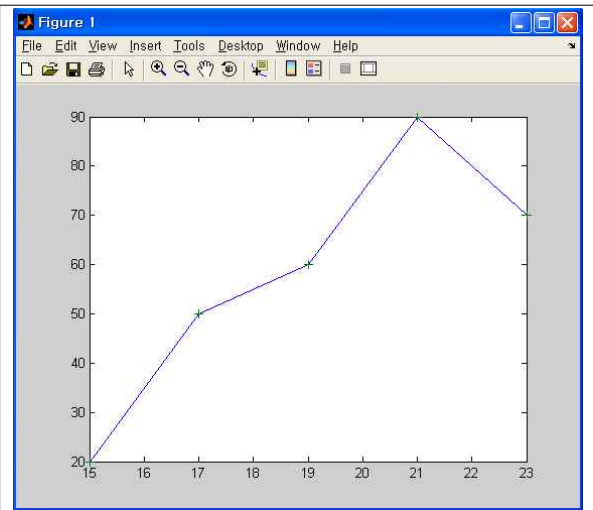
```
>> plot(x,y,x,z,'--')
```



- 그래프로부터 점의 좌표를 구해야 할 때, 함수 ginput()을 사용
=> 명령어 `[x,y]=ginput(n)` : n개의 점들을 구함; 길이가 n인 x와 y 벡터 구함
- 함수와는 별도로 데이터를 그래프에 나타낼 경우, 데이터 마커(+,*o)로 표시
=> `plot(x,y,'+')`
=> 필요한 경우, 데이터들을 선으로 연결 가능 : `plot(x,y,'+','x,y')`

[Ex.3] $x=[15:2:23]$, $y=[20\ 50\ 60\ 90\ 70]$ 인 경우

```
>> x=[15:2:23];  
>> y=[20 50 60 90 70];  
>> plot(x,y,x,y,'+')
```



[예제5]

① 매크랩을 이용하여 $0 \leq t \leq 5$ 에서 $s = 2\sin(3t+2) + \sqrt{5t+1}$ 의 그래프를 그려라. 그래프에 제목을 적고, 축에 라벨을 표시하라. 변수 s 는 1초당 피트의 속도 (ft/s)이고, 변수 t 는 초단위이다.

② 매크랩을 이용하여 $0 \leq x \leq 5$ 에서 $y = 4\sqrt{6x+1}$ 과 $z = 5e^{0.3x} - 2x$ 의 그래프를 그려라.

1.4 스크립트 파일과 편집기/디버거

- 매트랩은 두 가지 방법으로 연산을 수행 가능
 - ① 인터랙티브 모드 : 모든 명령을 명령창에 입력
 - ② 스크립트 파일(M-파일)로 저장된 매트랩 프로그램의 실행
 - => M-파일을 실행하는 것은 명령창에 명령어를 하나씩 입력하는 것과 동일
 - => 명령창 프롬프트에서 파일 이름을 입력하여 파일을 실행시킴
- 많은 명령어들을 수행시키거나 명령어를 반복 수행시킬 경우, 혹은 많은 원소를 갖는 배열들이 포함된 문제에서 인터랙티브 모드는 불편
 - => 이런 경우, 프로그램(M-파일)으로 작성할 수 있음
- M-파일의 두 가지 형태 : 스크립트 파일(script file)/함수 파일(function file)
 - => 편집기와 디버거를 사용하여 M-파일을 만들 수 있음
 - => 스크립트 파일은 명령파일(command file)로도 불림
- 함수 파일은 3장에서...

▶ 스크립트 파일 작성과 사용

| | |
|---|---|
| <pre>>> % This is a comment >> x=3+2 % So is this x = 5</pre> | <p>%는 주석을 나타내며, % 기호 오른쪽의 것은 모두 무시됨</p> |
|---|---|

[Ex.4] 간단한 스크립트 파일 만들기

- ① M-파일을 만들기 위해, “File/New/M-file”메뉴를 선택 => 편집기/디버거 창
- ② 아래와 같이 문서를 작성하고 ‘example1.m’으로 현재 디렉터리에 저장함
- ③ 프로그램을 실행하기 위해, 명령창에 ‘example1’을 입력(혹은 Debug/Run)

```
% Program example1.m
% This program computes the sine of
% the square root and displays the result.
x=sqrt([5:2:13]);
y=sin(x)
```

- 스크립트 파일의 이름은 변수의 이름이나 매트랩 명령어나 함수 이름과 같지 않도록 함, 확인을 위해 다음을 입력해 봄

=> `exist('example1')` : 변수 `example1`의 존재 여부 ; 없으면 0, 있으면 1

=> `exist('example1.m','file')` : 파일 `example1.m` 확인; 없으면 0, 있으면 2

=> `exist('example1','builtin')` : 내장함수 여부 확인 ; 없으면 0, 있으면 5

※ `exist('mean.m','file')`와 `exist('mean','builtin')`을 입력해 보면

=> 파일 '`mean.m`'은 존재하지만, 내장함수는 아님

=> 존재하는 M-파일의 주석문 보기 : >> `help filename`

▶ 스크립트 파일의 디버깅

- 프로그램을 디버깅하는 것은 bug 또는 에러를 찾아 제거하는 과정
- 일반적으로 에러는 구문(syntax) 에러나 런타임(runtime) 에러의 범주에 속함
- 매트랩은 상대적으로 프로그램이 간단하여 디버거를 사용할 필요가 없음

▶ 프로그램 형식

- 주석부/입력부/계산부/출력부로 구성

※ 주석문은 파일 어느 곳에나 놓을 수 있음. 그러나 첫 번째 주석문은 '`lookfor`'라는 명령어가 찾는 라인으로 그 파일을 설명하는 핵심어를 첫 번째 줄(H1)에 둠

=> 일반적으로 주석문은 다음의 사항을 포함하도록 함

- ① 첫 번째 줄에 프로그램 이름과 keyword를 씀
- ② 두 번째 줄에 작성 날짜와 작성한 사람의 이름을 씀
- ③ 모든 입·출력들에 대한 변수 이름을 정의. 이때 반드시 측정 단위를 명기
- ④ 프로그램이 호출하는 모든 사용자 정의 함수

▶ 입·출력 제어

| 명령어 | 설명 |
|------------------------------|---|
| <code>disp(A)</code> | 배열 A의 내용을 표시 |
| <code>disp('text')</code> | 작은따옴표 안의 text 문자열을 표시 |
| <code>x=input('text')</code> | 작은따옴표 안의 내용을 표시하고 사용자의 입력을 기다림 입력된 내용은 x에 할당 |

| 명령어 | 설명 |
|--|---|
| <code>k=menu('title','op1','op2',...)</code> | 문자열 변수 'title'을 제목으로 갖는 메뉴를 표시하며, 선택 옵션은 'op1', 'op2' 등이 있음 |

[Ex.5] 메뉴를 사용하여 그래프 마커 고르기

```
k=menu('Choose a data marker','o','*','x')
```

[예제6] 구의 표면적 A 는 $A = 4\pi r^2$ 과 같이 r 에 의해 결정된다. 사용자가 프롬프트 상에서 반지름을 입력하고 표면적을 계산한 후, 결과를 나타내는 스크립트 파일을 작성하라.

1.5 매트랩 도움말 시스템

- 여기서 다루지 않은 다른 기능들을 알기 위해서는 도움말을 사용할 필요 있음
 - ① Help 브라우저 : "Help/MATLAB help" 메뉴를 선택 혹은 툴바의 물음표, F1
 - ② Help 함수 : 함수 help, lookfor, doc를 사용하여 특정 함수의 정보를 검색
 - ③ 기타 자료들 : demo 프로그램 실행, Mathworks사에서 제공되는 문서 등
- 도움말 함수
 - ① help 함수 : 특정 함수의 구문법과 동작을 알 수 있는 가장 기본적인 방법
 - ② lookfor 함수 : 키워드를 기반으로 한 함수의 검색, H1 라인을 탐색
 - ③ doc 함수 : 도움말 브라우저에서 문서의 시작 페이지를 나타냄

[Ex.6] 명령창에 'help sine', 'lookfor sine', 'doc example1'을 입력해보라.

2. 숫자, 셀과 구조 배열

- 매트랩의 장점 중 하나는 배열을 하나의 변수로 처리할 수 있는 것
=> 프로그램을 간단하게 작성 가능

2.1 1차원 및 2차원 숫자 배열

- 1차원 배열 : 벡터; 오직 하나의 행이나 열로 구성
=> 행벡터 : 원소가 수평으로 정렬, 열벡터 : 원소가 수직으로 정렬

- ▶ 벡터의 생성**
- 행벡터 : 대괄호 안에 원소를 입력하고 공마로 원소를 분리
 - 열벡터 : 세미콜론을 이용하여 만듦. (혹은 빈칸)

| 사용 예 | | | 설명 |
|--|---|-------------------------------|---|
| <pre>>> g=[3;7;9] g = 3 7 9</pre> | <pre>>> g=[3 7 9]' g = 3 7 9</pre> | <pre>>> g=[3 7 9]</pre> | 열벡터의 생성(세 가지 방법) - 보통 세미콜론을 이용하여 만들 수 있으나, 행벡터를 만든 후 <u>전치</u> 를 이용하여 만들 수도 있음 (transpose) |
| <pre>>> r=[2 4 20]; w=[9 -6 3]; >> u=[r, w] u = 2 4 20 9 -6 3</pre> | | | 벡터 u는 크기가 1X6인 벡터 |
| <pre>>> x=[0:2:8] x = 0 2 4 6 8</pre> | | | 콜론(:)연산자를 이용하면 일정한 간격으로 원소를 갖는 큰 벡터를 생성 |
| <pre>>> x=[0:2:7] x = 0 2 4 6</pre> | | | x=[m;q:n]에서 첫 번째 값은 m, n-m이 q의 정수배이면 마지막 값은 n, 그렇지 않으면 n보다 작은 값이 됨 |
| <pre>>> y=[-3:2] y = -3 -2 -1 0 1 2</pre> | | | q 값이 생략되면 q를 1로 간주함 증분 q는 음수가 될 수도 있고 이 경우 m은 n보다 큰 값이어야 함 |
| <pre>>> linspace(5,8,31) ans = Columns 1 through 4 5.0000 5.1000 5.2000 5.3000...</pre> | | | 명령어 linspace(x1,x2,n)도 선형으로 증가되는 행벡터 생성, x1와 x2는 각각 상한과 하한을 나타내고 n은 원소의 수, 왼쪽 예는 [5:0.1:8]과 동일, |
| <pre>>> x= logspace(-1,1,4) x = 0.1000 0.4642 2.1544 10.0000</pre> | | | 원소가 로그 간격으로 된 배열, logspace(a,b,n) : n은 10^a 과 10^b 사이의 원소의 수, n 생략 시 원소 50개 |

▶ 2차원 배열 : 행렬

| | |
|--|--|
| <pre>>> A=[2 4 10;16 3 7]</pre> <pre>A =</pre> <pre> 2 4 10</pre> <pre> 16 3 7</pre> | <p>행렬의 생성</p> <p>2행 3열로 된 행렬(2X3행렬)</p> <p>행렬 A를 $[a_{ij}]$로 나타내며, i와 j는 각각 행과 열의 위치를 나타냄</p> |
|--|--|

| | |
|--|--|
| <pre>>> a=[1,3,5];b=[7,9,11];</pre> <pre>>> c=[a,b]</pre> <pre>c =</pre> <pre> 1 3 5 7 9 11</pre> | <pre>>> D=[a;b]</pre> <pre>D =</pre> <pre> 1 3 5</pre> <pre> 7 9 11</pre> |
|--|--|

▶ 행렬과 전치(transpose) 연산

- 전치연산으로 행렬의 행과 열을 바꿀 수 있음

| | |
|---|---|
| <pre>>> A=[1 2; 3 4]</pre> <pre>A =</pre> <pre> 1 2</pre> <pre> 3 4</pre> | <pre>>> A'</pre> <pre>ans =</pre> <pre> 1 3</pre> <pre> 2 4</pre> |
|---|---|

▶ 주소 지정

- 배열 인덱스를 사용하여 배열 원소의 행과 열 번호로 원소의 위치를 지정
 - => $v(5)$: 벡터 v 에 있는 다섯 번째 원소, $A(2,3)$: 행렬 A 의 2행, 3열의 원소
 - => 이를 이용하여 배열의 원하는 원소만을 다룰 수 있음
 - => $D(1,3)=6$: 행렬 D 의 1행 3열의 원소를 6으로 할당
- 콜론(:)연산자를 사용하여 배열을 다양하게 선택가능
 - => $v(:)$: 벡터 v 의 모든 행 또는 열의 원소를 나타냄
 - => $v(2:5)$: 벡터 v 의 두 번째 원소로부터 다섯 번째 원소까지를 나타냄
 - => $A(:,3)$: 행렬 A 의 세 번째 열에 있는 모든 원소를 나타냄 (열벡터)
 - => $A(3,:)$: 행렬 A 의 세 번째 행에 있는 모든 원소를 나타냄 (행벡터)
 - => $A(:,2:5)$: A 의 두 번째에서 다섯 번째 열에 있는 모든 원소를 나타냄(행렬)
 - => $A(2:3,1:3)$: 2X3행렬을 나타냄
 - => $v=A(:)$: 처음부터 끝 원소까지를 쌓아 올린 A 의 모든 열로 구성된 벡터
 - => $A(end,:)$, $A(:,end)$: 각각 A 의 마지막 행과 마지막 열을 나타냄

[Ex.1] 무작정 따라하며 관찰하기

| | |
|---------------------|-----------------|
| >> a=[1:9] | >> c=A(2,:) |
| >> A=reshape(a,3,3) | >> D=A(2:3,1:3) |
| >> A' | >> v=A(:) |
| >> b=A(:,3) | >> A(end,:) |

[참고]

| | |
|--|---|
| >> A(1,4)=10 A = 1 4 7 10 2 5 8 0 3 6 9 0 | 위의 행렬 A에 왼쪽과 같이 입력하면 아래의 결과가 나옴. => A에 4열이 없으므로 4열에 새로운 원소를 받아들이기 위해 자동으로 확장하고 나머지는 0으로 채움 |
| >> B=A(:,4:-1:1) B = 10 7 4 1 0 8 5 2 0 9 6 3 | 콜론연산자를 이용하여 인덱스를 감소시킬 수 있음. 왼쪽의 예는 콜론연산자를 이용하여 행렬 A의 열의 순서가 바뀜 |
| >> C=A([2,1,2],:) C = 2 5 8 0 1 4 7 10 2 5 8 0 | 행의 위치를 [2,1,2]과 같이 정해 줌으로써 행렬을 수정할 수 있음 |

P 유용한 배열 함수 - 매트랩에서는 배열을 다루기 위한 많은 함수가 있음

| 명령어 | 설명 |
|---------|---|
| find(x) | 배열 x의 0이 아닌 원소의 인덱스를 갖는 배열을 생성 |
| max(A) | A가 벡터이면, 대수적으로 가장 큰 원소의 값을 반환 A가 행렬이면, 각 열에서 가장 큰 원소를 갖는 행벡터를 반환 |
| min(A) | max(A)와 같은 기능이나 최소값을 돌려줌 |
| morm(x) | 벡터의 기하학적 길이를 계산 $\sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$ |
| size(A) | 행렬 A의 크기를 갖는 행벡터 [m,n]을 반환 |
| sort(A) | 배열 A의 각 열을 오름차순으로 정렬 |
| sum(A) | 배열 A의 각 열의 원소를 더하고 합으로 된 행벡터를 반환 |

2.2 다차원 숫자 배열

- 매트랩은 다차원 배열을 지원, 자세한 사항은 help datatypes를 이용.
- 함수 cat(n,A,B,C,...) : A, B, C 등을 연결시켜 n차원의 배열 생성

2.3 원소-원소 연산

| | | |
|---|--|---|
| <pre>>> A=[4 9; 5 -7]; 3*A ans = 12 27 15 -21</pre> | | 벡터의 스칼라 곱 - 행렬 A의 각 성분에 3을 곱함 매트랩에서의 곱셈의 정의 두 가지 - ① 배열 곱셈, ② 행렬 곱셈 |
| <pre>>> A=[6 -2;10 3];B=[9 8; -12 14]; >> A+B ans = 15 6 -2 17</pre> | | 두 행렬의 덧셈과 뺄셈(배열 덧셈) - 대응하는 원소들끼리의 덧셈 혹은 뺄셈 - 덧셈의 결합법칙과 교환법칙 성립 - 두 행렬의 사이즈가 같아야 연산 可 |
| <pre>>> [6, 3]+2 ans = 8 5</pre> | <pre>>> [8,3]-5 ans = 3 -2</pre> | 스칼라-행렬 덧셈과 뺄셈 |
| <pre>>> x=[2 4 -5]; y=[-7 3 8]; x.*y ans = -14 12 -40</pre> | | 배열 곱셈 - 각 대응하는 원소끼리의 곱셈 : x와 y가 행벡터이면 행벡터로, x와 y가 열벡터이면 열벡터가 됨 |
| <pre>>> x=[8,12,15];y=[-2,6,5];z=x./y z = -4 2 3</pre> | | 배열 나눗셈(원소-원소 나눗셈) 두 배열은 반드시 크기가 같아야 함 |
| <pre>>> x=[1 2 3];y=[2,3,4]; x.^2 ans = 1 4 9 >> y.^x ans = 2 9 64</pre> | | 배열의 거듭제곱 |
| <pre>z= exp(y).*(sin(x).*(cos(x))).^2</pre> | | $z = (e^y \sin x) \cos^2 x$ x와 y의 크기가 같아야 함 |

2.4 행렬 연산

- 행렬의 덧셈과 뺄셈은 원소-원소의 덧셈 및 뺄셈과 같은 방법임
=> But, 행렬의 곱셈과 나눗셈은 다름

▶ **벡터의 곱셈** : 두 벡터의 원소의 개수가 같으면 연산 가능

- 벡터 u 와 w 의 내적(dot product)은 스칼라이며, $u \cdot w$

$$u \cdot w = |u||w|\cos\theta = [u_1 \ u_2 \ u_3] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = u_1w_1 + u_2w_2 + u_3w_3$$

| | |
|---|------------------|
| <pre>>> x=[1 2 3]; y=[2; 3; 4]; x * y ans = 20</pre> | $1*2+2*3+3*4=20$ |
|---|------------------|

▶ **벡터-행렬 곱셈**

| | |
|---|--|
| <pre>>> A=[4 9; 5 -7]; x=[1; -1]; A * x ans = -5 12</pre> | $\begin{bmatrix} 4 & 9 \\ 5 & -7 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -5 \\ 12 \end{bmatrix}$ |
|---|--|

▶ **행렬-행렬 곱셈** : 행렬곱셈에서 결합과 분배법칙은 성립하나 교환법칙은 아님

- 두 행렬의 곱 AB 는 A 의 열의 수와 B 의 행의 수가 같아야 정의됨
=> $A(p \times q)$ 과 $B(q \times r)$ 의 행렬곱은 $AB(p \times r)$ 가 됨

| |
|--|
| <pre>>> A=[6 -2;10 3;4 7]; B=[9 8;-5 12]; A * B ans = 64 24 75 116 1 116</pre> |
|--|

[참고] 특수 행렬 - 영행렬/단위행렬

| | | | | | |
|--------------|----------|---------------|-----------|----------------|---------|
| eye(n) | nxn 단위행렬 | ones(n) | 원소 1, nxn | zeors(n) | nxn 영행렬 |
| eye(size(A)) | A와 같은 크기 | ones(m,n) | 원소 1, mxn | zeors(m,n) | mxn 영행렬 |
| | | ones(size(A)) | A와 동일 | zeros(size(A)) | A와 동일 |

▶ 선형 대수방정식과 역행렬

[Ex.2] 다음 연립방정식을 매트랩을 이용하여 풀자

| | |
|--|--|
| <pre>>> A=[6,12,4; 7,-2,3; 2,8,-9]; b=[70;5;64]; >> x=A\b % 혹은 y=inv(A)*b x = 3 5 -2</pre> | $\begin{aligned} 6x + 12y + 4z &= 70 \\ 7x - 2y + 3z &= 5 \\ 2x + 8y - 9z &= 64 \end{aligned}$ |
|--|--|

2.5 사용자 정의 함수

- M-파일의 다른 형식으로 함수파일(function file)이 있음
- 스크립트 파일과 달리 함수파일의 모든 변수는 지역변수로 함수 내에서만 사용 가능
- 함수파일은 여러 차례 반복되는 명령에서 유용
- 함수파일을 만들 때, 첫 라인은 입력과 출력 목록을 나타내는 **함수 정의 라인**
 => `function [output variables] = function_name (input variables)`
 => 출력변수는 대괄호로 둘러싸야 하고(출력이 하나만 있을 때는 선택사항) 입력변수는 소괄호로 둘러싸야 함
 => 함수파일의 이름은 파일 이름과 반드시 같아야 함
 예를 들어, 함수이름이 drop 인 경우, 파일이름은 drop.m으로 저장
 => 함수 정의 라인의 function은 반드시 소문자로 표시

[Ex.3] 간단한 함수의 예

| | |
|--|--|
| <pre>function z = fun (x,y) % fun.m u = 3 * x ; z = u + 6 * y.^2 ;</pre> | <pre>>> x=3; y=7; fun(x,y) ans = 303</pre> |
|--|--|

3. Simulink

- 시뮬링크는 매트랩을 기반으로 만들어졌음, 최근 인기가 급상승하고 있음.
- 그래픽 사용자 인터페이스(GUI)를 제공하여 블록으로 시뮬레이션 가능

3.1 시뮬레이션 선도(블록선도)

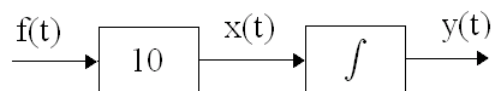
- 시뮬레이션 선도 : 해결해야 할 문제의 요소를 나타내는 선도

[Ex.1] 식 $\dot{y} = 10f(t)$ 를 고려

- 위 식의 해는 $y(t) = \int 10f(t)dt$ 이고, 중간 변수 x 를 이용하면

$$\Rightarrow x(t) = 10f(t), y(t) = \int x(t)dt \text{로 생각 가능}$$

\Rightarrow 이 해는 시뮬레이션 선도에 의해 그래픽으로 표현 가능



\Rightarrow 블록은 원인과 결과를 나타내고 화살표는 변수를 나타냄

\Rightarrow 숫자 10을 포함한 블록(이득 블록)은 $x(t) = 10f(t)$ 를 나타냄

\Rightarrow 적분 기호를 포함한 블록(적분기 블록)은 $y(t) = \int x(t)dt$ 를 나타냄

※ 시뮬레이션 선도에 사용되는 표기와 심볼은 약간씩 다를 수 있음

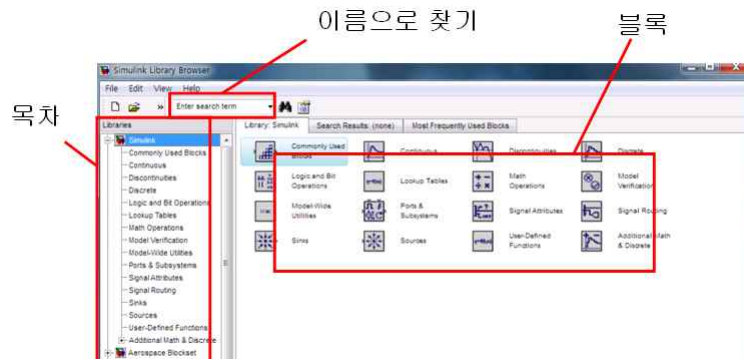
\Rightarrow 소자의 기호나 라플라스 변환에서 유래되기도 함

3.2 시뮬링크 소개

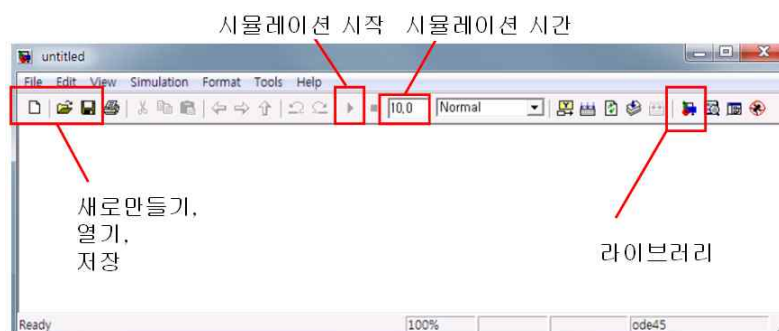
- 명령창에 'simulink'를 입력하면 시뮬링크 라이브러리 브라우저가 열림
 - 새로운 모델을 만들기 위해, 브라우저의 종이 모양의 아이콘을 클릭(File/New)
 - 시뮬레이션 선도를 만들기 위해, 라이브러리 안의 원하는 블록을 드래그 하여 새 모델 창으로 옮겨서 블록을 연결함. 적당한 블록 파라미터를 지정
 - 시뮬링크 모델 파일의 확장자는 .mdl
 - 모델 파일을 열고, 닫고, 저장하기 위해 모델 창의 File 메뉴를 사용
 - 모델을 복사, 절단, 붙이기 위해 Edit 메뉴를 사용
- \Rightarrow 물론 이 기능들은 마우스를 통해서도 사용가능

※ 예제를 통해 시뮬링크의 사용법을 배워보자

- 라이브러리 브라우저



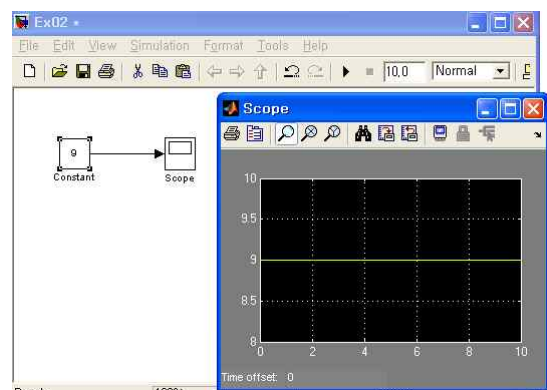
- 모델창(Simulink창)



[Ex.2] 시뮬레이션 실행

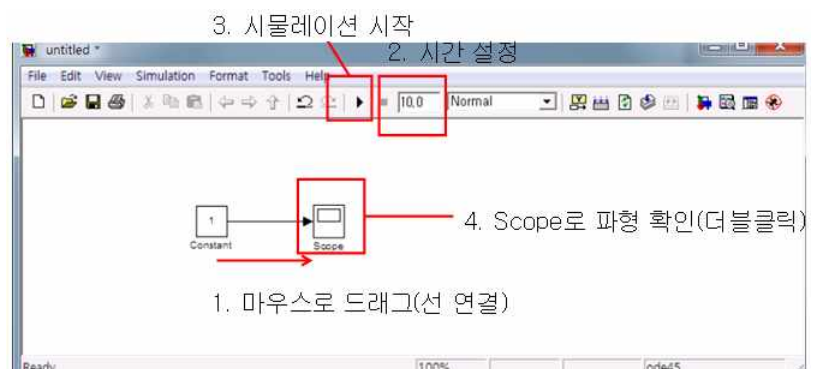
- ① Sources/Constant(상수)를 가져옴
- ② Sinks/Scope를 가져옴
- ③ 블록을 연결하고 시뮬레이션 시작
(파형보기 : Scope 창에서 쌍안경 이용)

※ 블록을 가져오기 위해, 검색창 이용



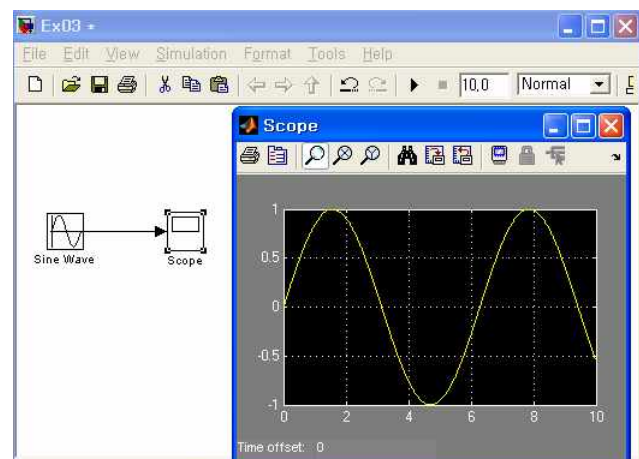
[확인문제]

- ① Constant 값 바꾸기
- ② 다양하게 시간 조절하기



[Ex.3] Step size

- ① 라이브러리에서 사인파(Sine Wave)를 찾아 가져옴. 10rad/sec.로 설정
- ② Scope와 연결하고 시뮬레이션
 - Simulation/Configuration Parameters를 선택
 - => Simulation time의 Max step size를 충분히 작은 값(0.01)으로 설정
 - => 또는 Fixed-step으로 설정하여 수치를 바꿔도 됨
 - => step size를 너무 작게 하면, 정확도는 높아지나 시뮬레이션 시간 길어짐

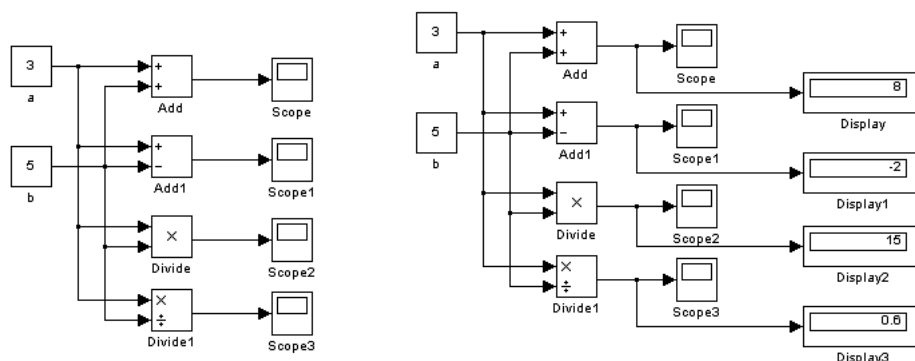


[확인문제]

- ① 다양한 주파수에 대해 실험
=> 1, 100 등
- ② 다양한 Step size에 대해 실험
=> 1, 0.1, 0.00001

[Ex.4] 4칙 연산

- ① Constant 블록 두 개를 가져와서 각각의 이름을 a와 b로 수정
- ② Math Operation 라이브러리에서 Add 블록을 두 개 가져옴
=> 그 중 하나를 '+'에서 '-'로 수정(빼기 연산)
- ③ 선을 연결, 선을 연결하고 중간에서 선을 추가하려면 'Ctrl'을 누르며 드래그
- ④ 곱셈과 나눗셈 블록은 'Divide'로 검색하여 두 개를 가져옴
=> 그 중 하나를 '*'에서 '**'로 수정(곱셈 연산)
- ⑤ Scope로 연결하여 확인
- ⑥ Display를 검색하여 연결
=> Scope는 파형을 보여주고, Display는 결과값을 출력

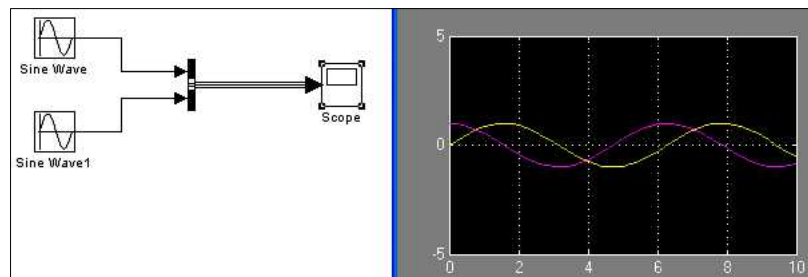


[확인문제] 변수 3개(a,b,c)를 만들어

$a+b+c$, $a*b*c$, $a-b+c$, $a/b*c$ 등의 다양한 계산식을 만들어 보자

[Ex.5] 여러 개의 파형을 한 번에 나타낼 때 버스(Bus)를 사용

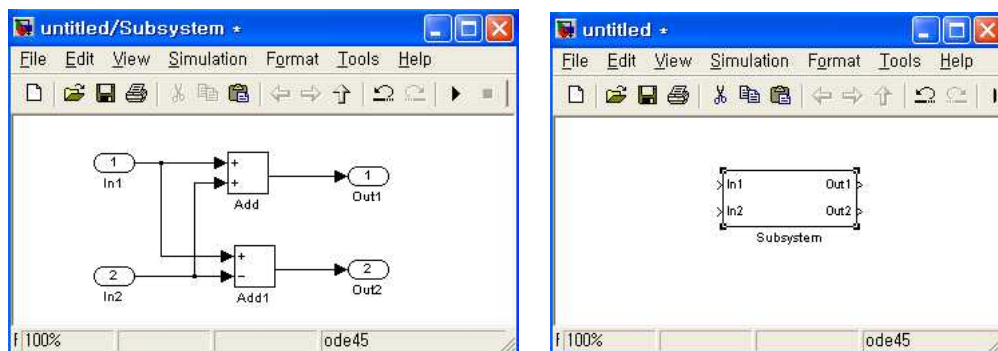
- ① 위상차가 다른 Sine 파를 각각 만들고
- ② Bus Creator를 검색하여 가져옴
- ③ Scope를 가져와서 연결하고 시뮬레이션 => 여러 개의 파형이 보임



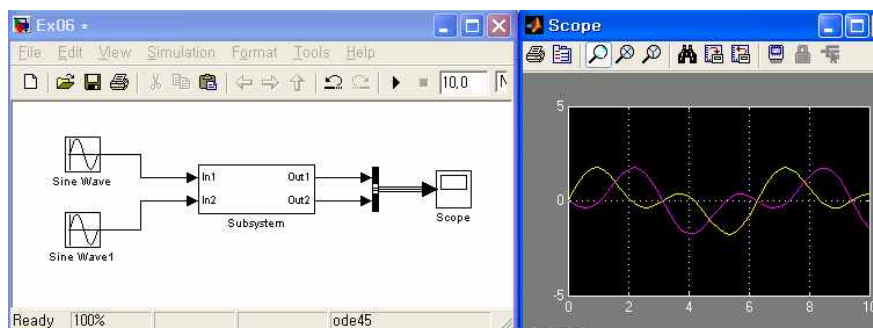
[Ex.6] 부시스템 : 여러 개의 복잡한 연산을 하나의 박스로 해결

예제로 2개의 변수를 받아 덧셈과 뺄셈을 출력하는 부시스템 만들어 보자

- ① Subsystem을 검색하여 가져옴. 블록을 더블클릭하면 부시스템 창이 생성



- ② 주파수가 서로 다른 사인파를 연결하여 Scope로 확인



※ 부시스템 생성하는 다른 방법 : 먼저 블록선도를 만든 후, 부시스템을 만들 부분을 지정하고 우클릭을 눌러 메뉴에서 'Create Subsystem' 클릭
=> 만든 후에는 더블클릭하여 수정 가능

[Ex.7] 전달함수 (transfer function)

다음 전달함수를 가지는 시스템의 스텝응답을 확인

$$H(s) = \frac{4s + 5}{s^2 + 2s + 3}$$

- ① Step 블록과 Transfer Fcn 블록을 검색하여 가져옴
- ② Step 블록과 Transfer Fcn 블록 수정

Transfer Fcn

The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of s.

Parameters

Numerator coefficients: $[a_n \ a_{n-1} \ \dots \ a_0] \Rightarrow a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s^0$

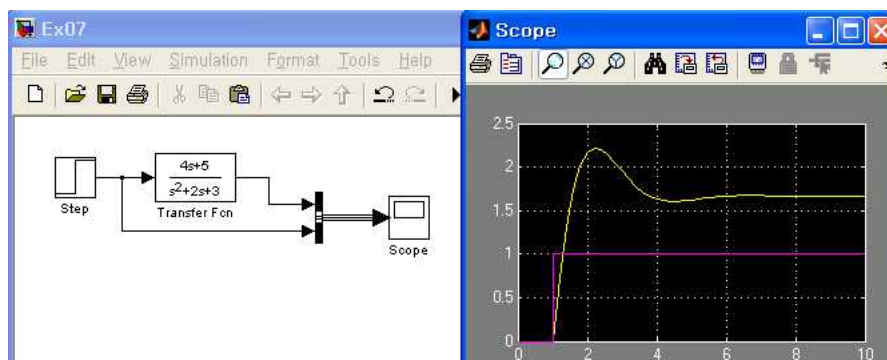
Denominator coefficients: 분자 계수

Absolute tolerance: auto

State Name: (e.g., 'position')

각 계수는 다음을 의미한다.

- ③ 시뮬레이션과 그 결과 파형

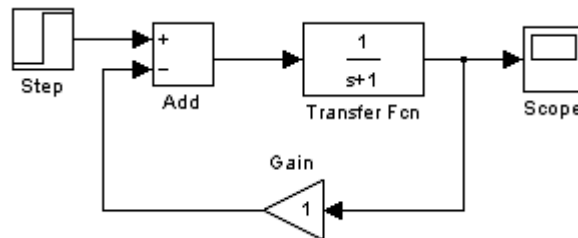


[확인문제] 어제 배운 매트랩의 결과와 비교

[Ex.8] 피드백 루프

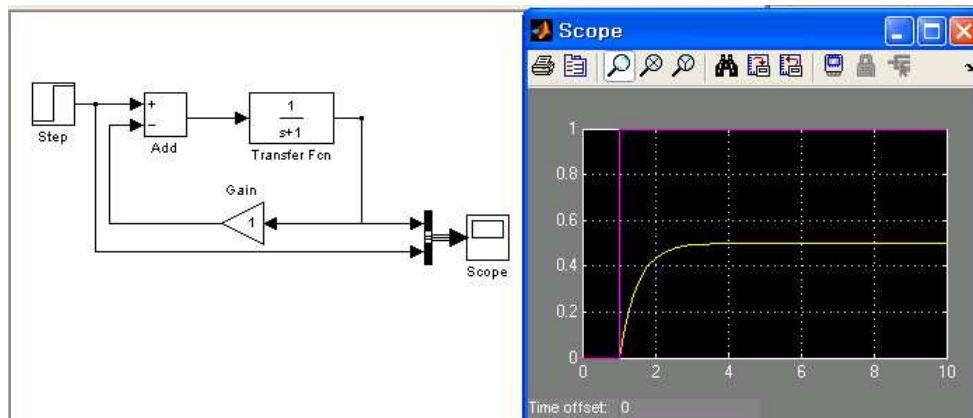
- 임의의 시스템에 대한 스텝응답을 구하는 모델에서 피드백 루프 구현

① Step, Transfer Fcn, Gain, Add, Scope 블록을 불러와서 아래와 같이 연결

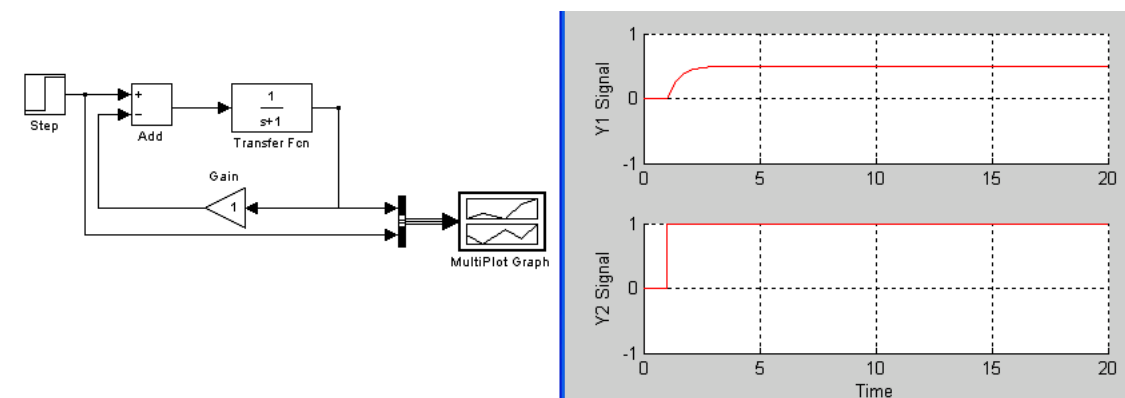


=> Gain의 방향 변경은 블록을 우클릭하여 'Format'을 변경

② 아래와 같이 Scope에 연결하면 입력과 출력 파형을 모두 관찰 가능



③ Multiplot graph 블록을 사용하면

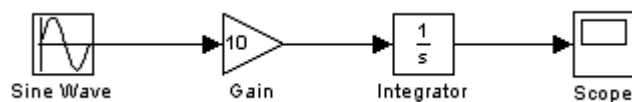


[Ex.9] 시뮬링크를 사용하여 $0 \leq t \leq 13$ 에서 다음 문제의 해를 구하라.

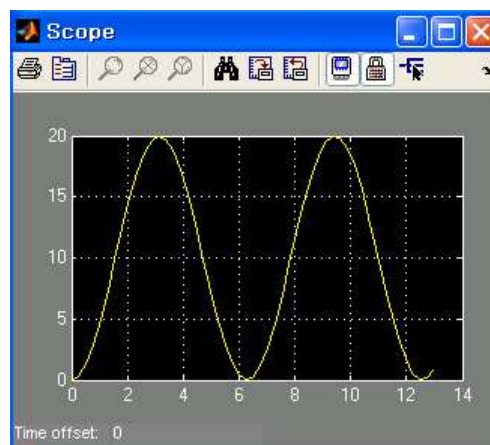
$$\frac{dy}{dt} = 10\sin t, \quad y(0) = 0, \quad \text{문제의 해는 } y(t) = 10(1 - \cos t).$$

- 해를 구하는 과정

- ① 시뮬링크 시작, 새로운 모델창 열기
- ② Source 라이브러리에서 Sine wave 블록을 선택하여 모델창으로 가져옴. 블록을 더블클릭하여 Parameters 창을 열고 Amplitude를 1로, Frequency를 1로 Phase는 0으로, Sample time은 0으로 하고 OK를 클릭
- ③ Math Operation 라이브러리에서 Gain 블록을 가져온 후, Parameters 창에서 Gain을 10으로 설정
- ④ Continuous 라이브러리에서 Integrator 블록을 가져온 후, Parameters 창에서 Initial condition을 0으로 설정. (초기조건이 0이므로)
- ⑤ Sink 라이브러리에서 Scope를 가져옴
- ⑥ 블록을 아래 그림과 같이 배치하고 연결. 연결을 위해 커서를 입력 포트나 출력 포트에 이동하면 커서가 +로 바뀌는데 이때 마우스 왼쪽 버튼을 누른 채 한 포트에서 다른 포트에 드래그



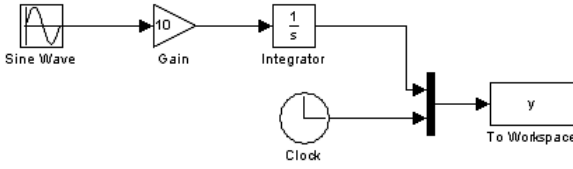
- ⑦ Simulation/Configuration Parameters를 선택한 후, Solver 탭을 클릭하고 Stop time을 13으로 설정. 이때 Start time은 0인지 확인
- ⑧ 시뮬레이션 시작을 위해, Simulation/Start를 선택. 또는 Start 아이콘 클릭
- ⑨ 새시뮬레이션이 끝나면 Scope 블록을 더블 클릭하고 쌍안경 아이콘을 클릭하여 자동 척도가 되도록 함. 진폭이 10이고 주기가 2π 인 사인함수.



※ 그림에 문구 넣거나 인쇄를 위해, Scope 블록 대신 Workspace 블록을 사용.

[Ex.10] 시뮬레이션 결과를 매트랩 Workspace로 출력하기.

=> 결과를 매트랩 함수를 이용하여 그림을 그리거나 해석 가능



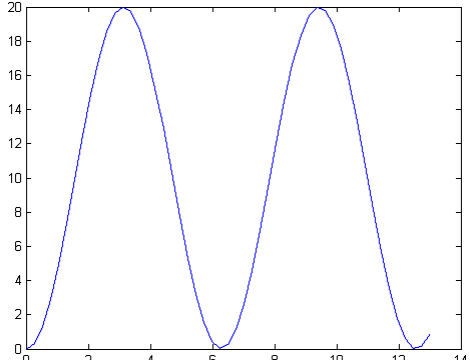
① Ex.2의 모델을 위 그림과 같이 변경

- Scope 블록과 연결하는 화살표를 클릭하여 Delete 키로 삭제
- Sinks 라이브러리의 To Workspace 블록과 Sources 라이브러리의 clock 블록, Signal Routing 라이브러리의 Mux 블록을 가져옴
- Mux 블록을 더블클릭하여 입력의 수를 2로 설정
- 블록을 연결하고 To Workspace 블록의 출력변수의 이름을 y로 변경함
- 출력 y의 '행'은 시뮬레이션 시간 구간 수이고, '열'은 블록의 입력 수임
=> Clock이 Mux의 두 번째 입력이므로 y의 두 번째 열은 시간임
- Clock의 Decimation=1로 함

② 시뮬레이션을 하면 workspace에 변수 y가 나타난 것을 확인할 수 있음

③ 명령창에 다음과 같이 입력

```
>> plot(y(:,2),y(:,1))
```



※ To Workspace 블록을 사용하면 시변수 tout를 매트랩 작업공간에서 자동적으로 설정함 (Simulation/Configuration Parameters 메뉴의 Data I/O에서 설정 可)