

신호 처리를 위한 MATLAB 프로그래밍

**전북대학교 공과대학 전자공학부
백 흥 기 교수**

목 차

1. MATLAB

- 1.1 MATLAB 이란 무엇인가?
- 1.2 MATLAB Desktop
- 1.3 MATLAB 의 도움말
- 1.4 명령 창(command window)
- 1.5 연습 문제

2. 행렬 연산(Matrix Operation)

- 2.1 행렬의 생성
- 2.2 행렬의 원소 참조
- 2.3 행렬의 정보
- 2.4 행렬의 변형
- 2.5 행렬의 행과 열 삭제
- 2.6 행렬의 함수
- 2.7 행렬의 연산
- 2.8 문자열
- 2.9 셀형(cell) 배열
- 2.10 구조체형(struct) 배열
- 2.11 관계 연산자
- 2.12 논리 연산자
- 2.13 다항식
- 2.14 연습 문제

3. M-File 프로그래밍

- 3.1 M-file 이란 무엇인가?
- 3.2 새 m-File 만들기
- 3.3 M-File 저장하기
- 3.4 M-File 불러오기
- 3.5 M-File 실행하기

- 3.6 제어 문장
- 3.7 M-file 의 실행 속도 비교
- 3.8 MATLAB 함수(function)
- 3.9 함수의 입출력 인자
- 3.10 프로그램 디버깅(debugging)
- 3.11 M-file 에 유용한 함수
- 3.12 개인적으로 사용하는 유용한 함수
- 3.13 연습 문제

4. 그래픽스(Graphics)

- 4.1 MATLAB 그래프
- 4.2 2 차원 그림
- 4.3 3 차원 그림
- 4.4 그림 분할
- 4.5 좌표축의 종류
- 4.6 시스템의 주파수 특성
- 4.7 그림 저장 및 읽기
- 4.8 그림 편집
- 4.9 그림 붙여 넣기
- 4.10 연습 문제

5. 파일 입출력(File I/O)

- 5.1 High-level 파일 입출력
- 5.2 Low-level 파일 입출력
- 5.3 표준 파일 포맷(standard file format)
- 5.4 연습 문제

6. 신호와 스펙트럼

- 6.1 신호
- 6.2 신호의 스펙트럼
- 6.3 연습 문제

7. 필터 설계와 신호의 필터링

7.1 필터의 종류

7.2 필터 설계

7.3 신호의 필터링

7.4 연습문제

8. Symbolic Math

8.1 Symbolic Math 란 무엇인가?

8.2 Symbolic Math 관련 함수

8.3 Symbolic Math 의 도움말

8.4 Symbolic 객체 생성

8.5 Symbolic 과 수치 데이터의 변환

8.6 Symbolic 연산

8.7 Symbolic Math 와 그래프

8.8 연습 문제

9. Simulink

9.1 Simulink 란 무엇인가?

9.2 Simulink 의 시작

9.3 Simulink 의 구성

9.4 Simulink 블록 세트

9.5 블록 모델

9.6 신호의 파형

9.7 신호의 스펙트럼

9.8 필터 설계와 신호의 필터링

9.9 Subsystem

9.10 연습 문제

10. 디지털 신호 처리 문제

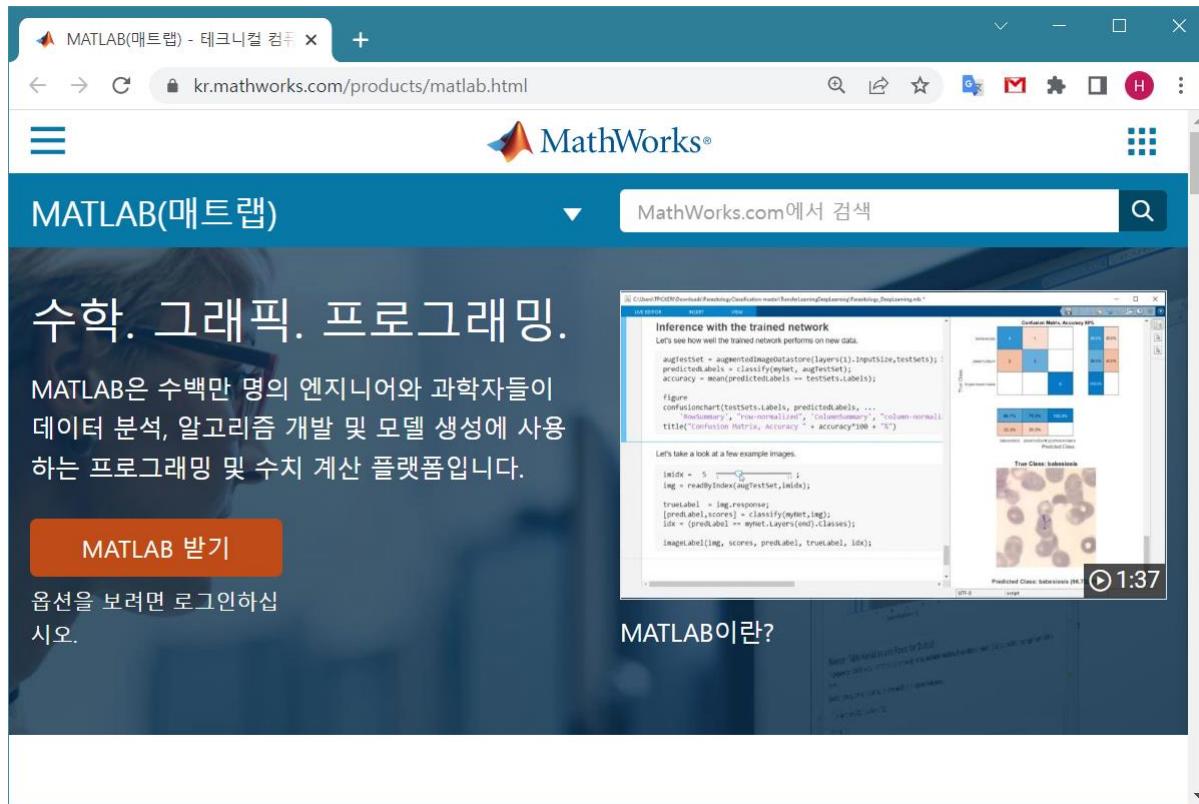
10.1 신호와 그래프

10.2 연속 시간 시스템의 주파수 응답

- 10.3 이산 시간 시스템의 주파수 응답
- 10.4 라플라스 변환
- 10.5 z 변환
- 10.6 디지털 시스템의 구조
- 10.7 아날로그 필터의 설계
- 10.8 디지털 필터의 설계
- 10.9 스펙트럼 추정
- 10.10 Decode of DTMF Signal

1. MATLAB

1.1 MATLAB 이란 무엇인가?



- ✚ <https://kr.mathworks.com/products/matlab.html>
- ✚ The Language of Technical Computing
- ✚ 전세계적으로 6500 개 이상의 대학에서 사용
- ✚ MATLAB 응용 분야
 - ✓ 제어 시스템
 - ✓ 딥 러닝
 - ✓ 영상 처리 및 컴퓨터 비전
 - ✓ 머신 러닝
 - ✓ 예측 정비
 - ✓ 로봇 공학
 - ✓ 신호 처리
 - ✓ 테스트 및 측정
 - ✓ 무선 통신
 - ✓ <https://kr.mathworks.com/products/matlab.html>

1.1.1 MATLAB 주요 기능

- 수치 연산, 시각화 및 응용 프로그램 개발을 위한 높은 수준의 언어
- 반복적인 데이터 분석, 설계 및 문제 해결을 위한 쉬운 사용자 환경 제공
- 선형 대수, 통계, 푸리에 해석, 필터링, 최적화, 수치 적분 및 상미분 방정식 풀이
- 데이터 시각화를 위한 내장 그래프와 사용자 전용 플롯 작성을 위한 툴 제공
- 코드 품질 및 유지보수성을 향상시키고 성능을 극대화하는 개발 툴
- 사용자 전용 그래픽 인터페이스가 있는 응용 프로그램 작성용 툴 지원
MATLAB 기반 알고리즘을 외부 응용 프로그램과 C, Java, .NET 및 Microsoft excel 같은 언어로 통합하는 기능
- 수치 계산
 - ✓ MATLAB은 데이터 분석, 알고리즘 개발, 모델 작성을 위한 다양한 수치 연산 메소드를 제공합니다. MATLAB 언어에는 일반적인 엔지니어링 및 과학 연산을 지원하는 수학 함수가 포함됩니다. 핵심 수학 함수는 벡터와 행렬 연산을 빠르게 실행할 수 있도록 프로세서 최적화 라이브러리를 사용합니다.
- 데이터 분석 및 시각화
 - ✓ MATLAB은 데이터를 수집, 분석 및 시각화함으로써 스프레드시트나 기존의 프로그래밍 언어를 사용한 경우 보다 시간을 단축하실 수 있습니다. 또한 플롯과 자동 리포트생성 기능을 통해, 또는 자동화된 MATLAB 코드로 결과를 공유할 수 있습니다.
- 프로그래밍 및 알고리즘 개발
 - ✓ MATLAB은 알고리즘과 응용 프로그램을 빠르게 개발 및 분석할 수 있는 높은 수준의 언어와 개발 도구를 제공합니다.
- 응용 프로그램 개발 및 배포
 - ✓ MATLAB 도구와 추가 기능 제품은 응용 프로그램 개발과 구현을 위한 다양한 옵션을 제공합니다. 개별 알고리즘과 응용 프로그램을 다른 MATLAB 사용자와 함께 공유하거나 MATLAB이 없는 다른 사람에게 로열티 없이 배포할 수 있습니다.

1.1.2 MATLAB 제품 군

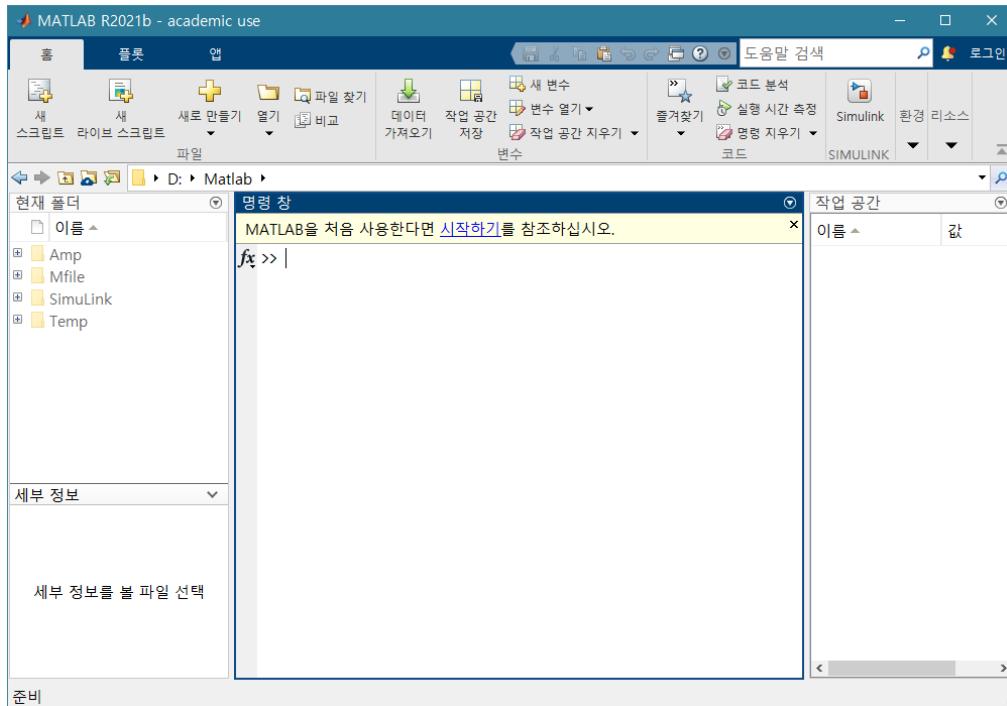
- MATLAB 제품 군
- SIMULINK 제품 군
- Toolbox
 - 전공 분야에 맞게 MATLAB 설치할 때 Toolbox 도 설치해야 함.
 - <https://kr.mathworks.com/products.html>
 - https://kr.mathworks.com/products.html?s_tid=gn_ps
 - 본 교재 사용에 필요한 tool box
 - ✓ Signal processing Toolbox
 - ✓ DSP System Toolbox
 - ✓ Symbolic math toolbox

1.1.3 MATLAB 과 C/C++의 비교

MATLAB	C, C++
전용 언어	범용 언어
사용이 쉽다. 사용 분야가 좁다. (과학 분야) 이식성이 없다.	사용이 어렵다. 사용 분야가 넓다. 이식성이 좋다.
인터프리터 방식 변수 선언 불필요 다차원 배열 가능	컴파일 방식 변수 선언 필요 다차원 배열 불가능
반복문의 벡터화 가능 다양한 함수	

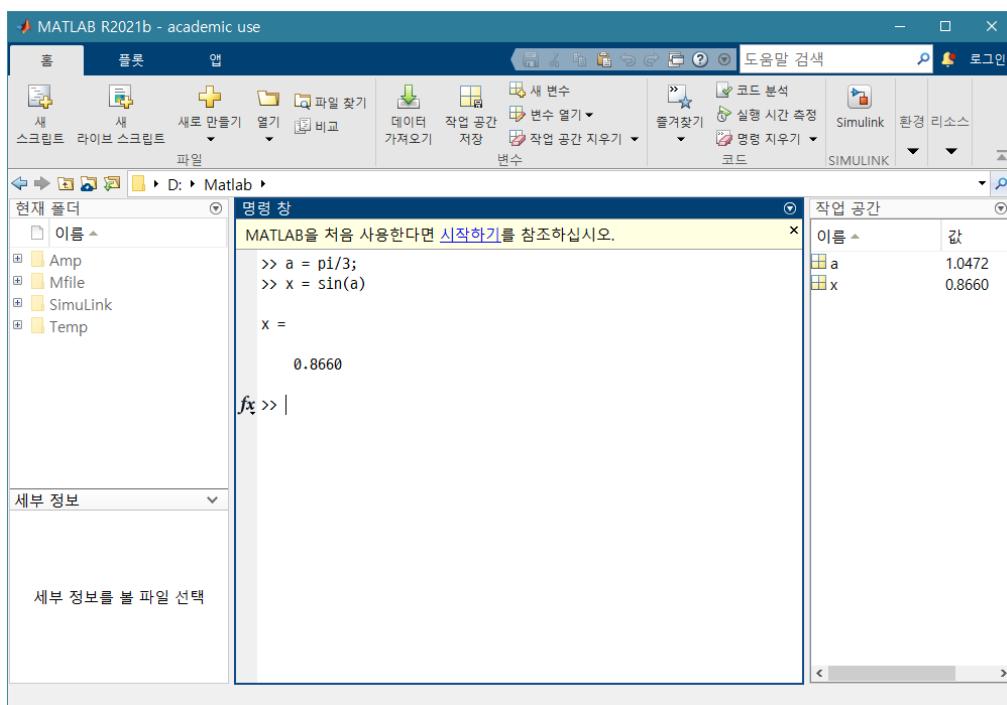
1.2 MATLAB Desktop

MATLAB Desktop (Default)

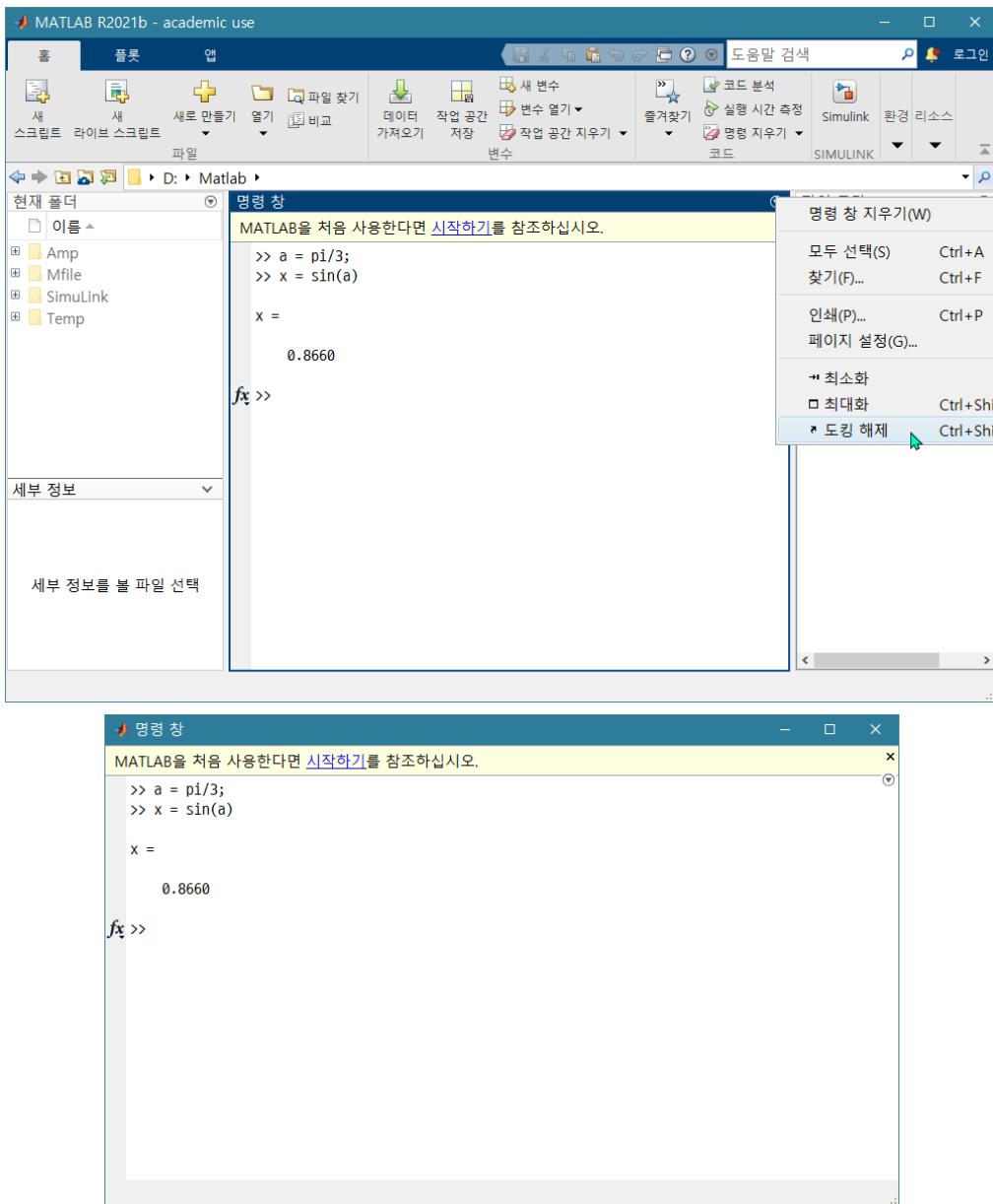


명령 창(command window)

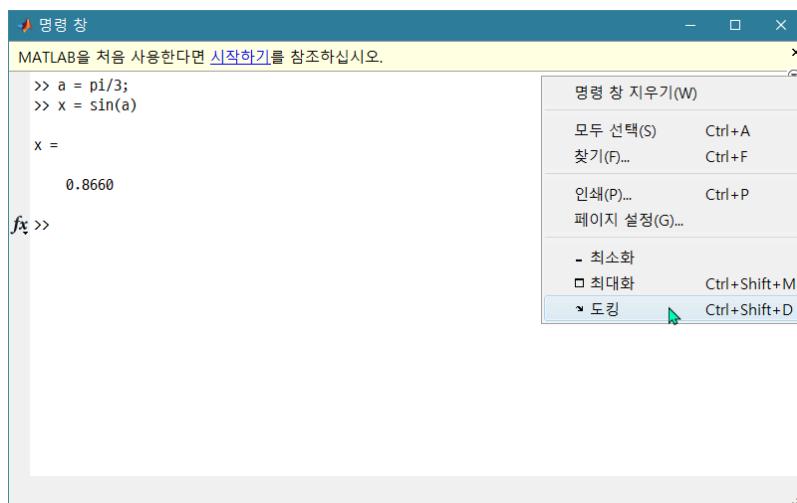
- ✓ 명령을 적을 수 있고 실행 결과를 보여주는 창

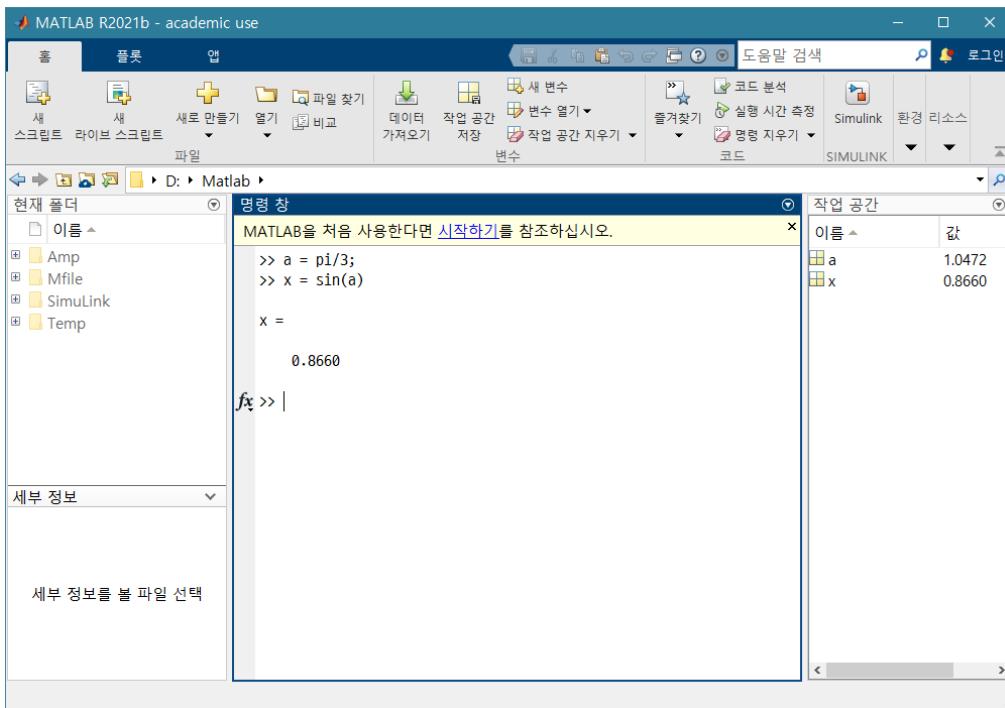


- ✓ 명령 창은 별도의 창으로 분리할 수 있다. (도킹 해제)



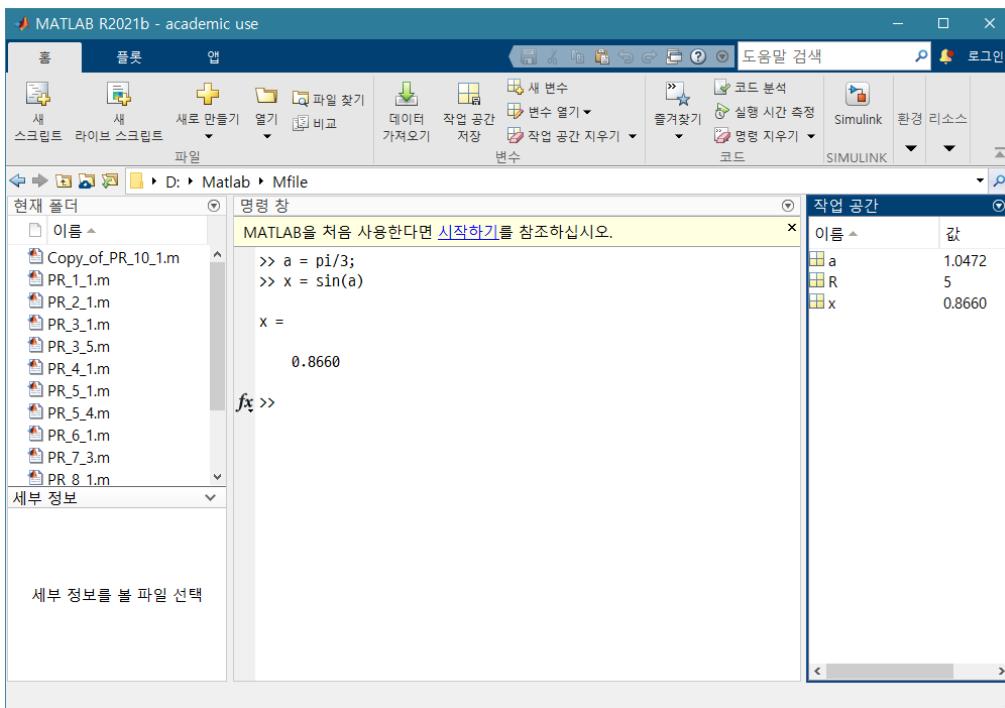
- ✓ 별도의 명령 창을 MATLAB desktop 안으로 합체시킬 수 있다. (도킹)





작업 공간(workspace)

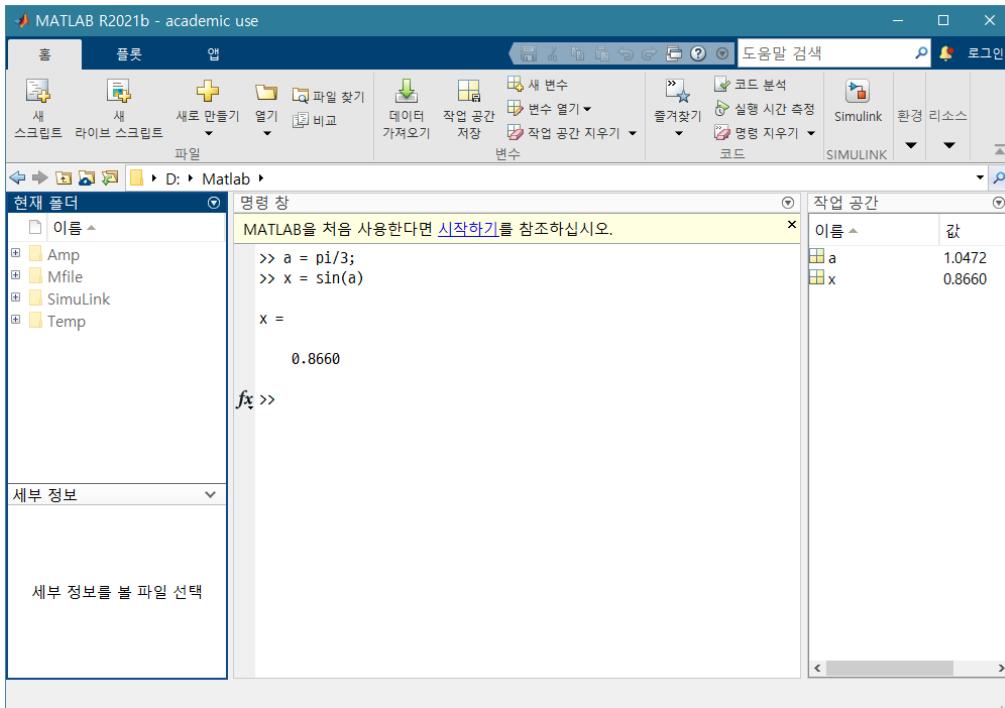
- ✓ MATLAB 내의 모든 데이터를 나타내는 곳



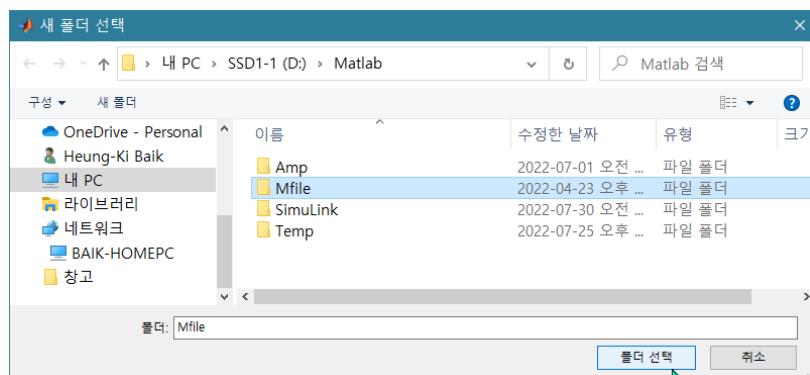
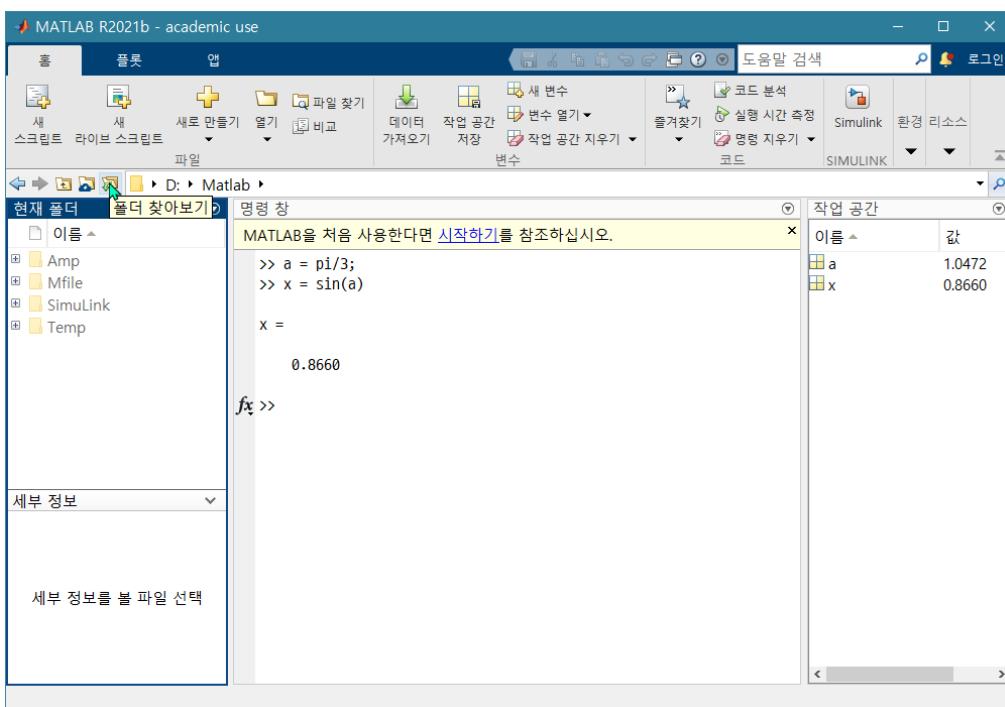
- ✓ 명령 창과 마찬가지로 별도의 창으로 분리할 수 있다.

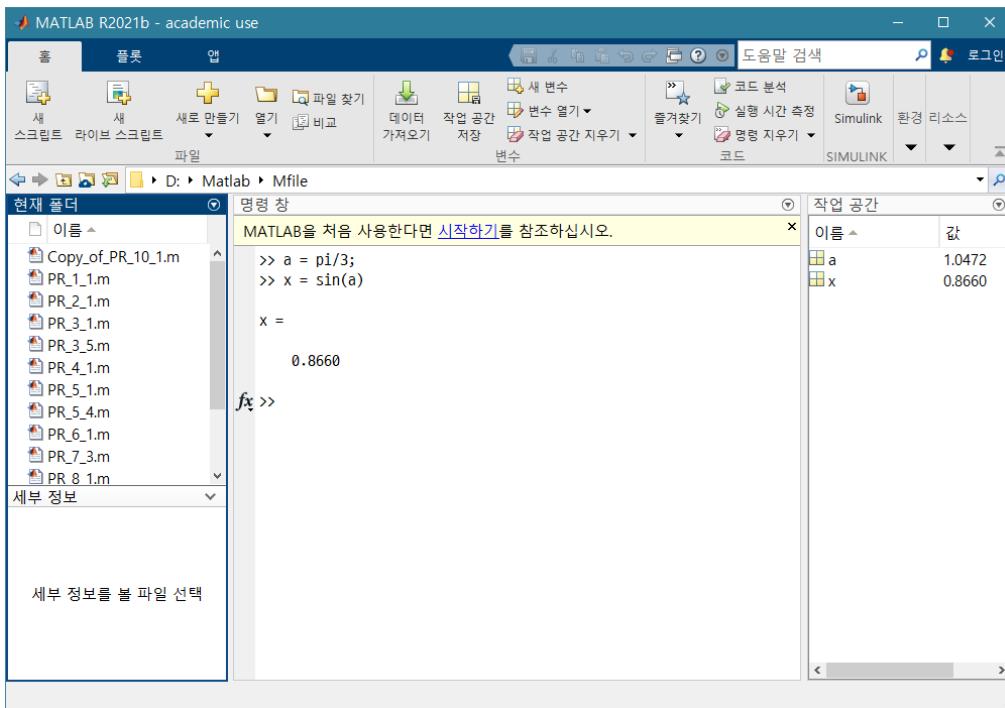
현재 폴더(current folder)

- ✓ 파일 (프로그램, 데이터 등)을 만들면 저장되는 폴더, 작업하는 폴더

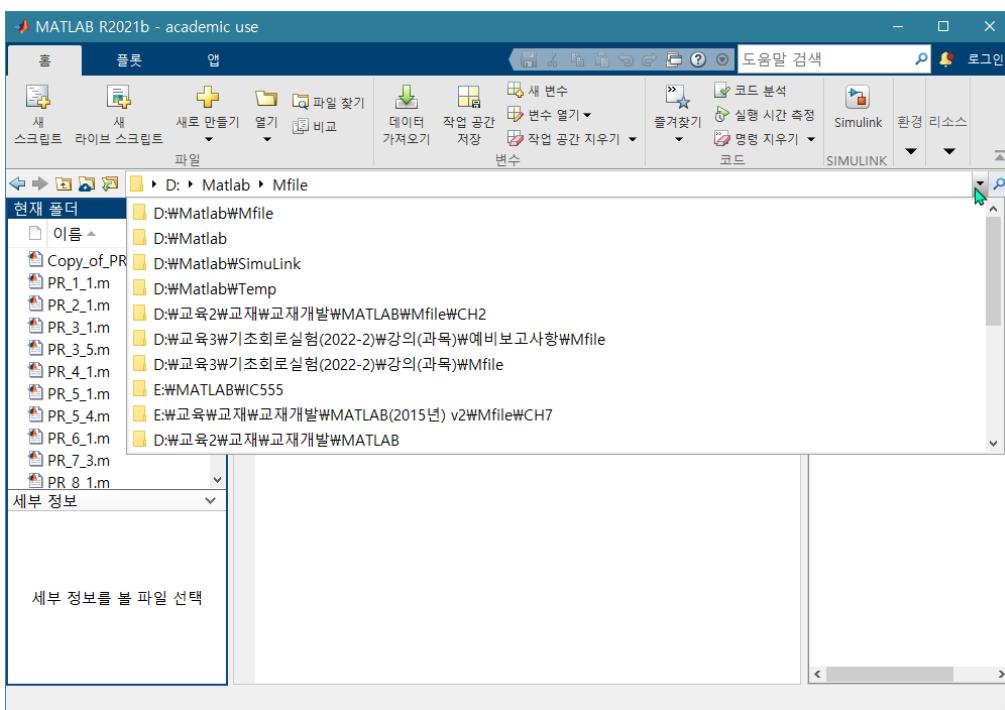


- ✓ 원하는 폴더에서 작업을 하려면 현재 폴더를 변경해야 한다.



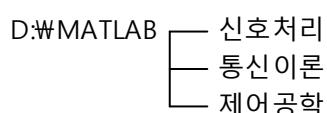


- ✓ 한 번 선택을 하면 다음부터는 쉽게 현재 폴더를 선택할 수 있다.



- ✓ MATLAB 폴더 관리

- ✗ 중앙 관리

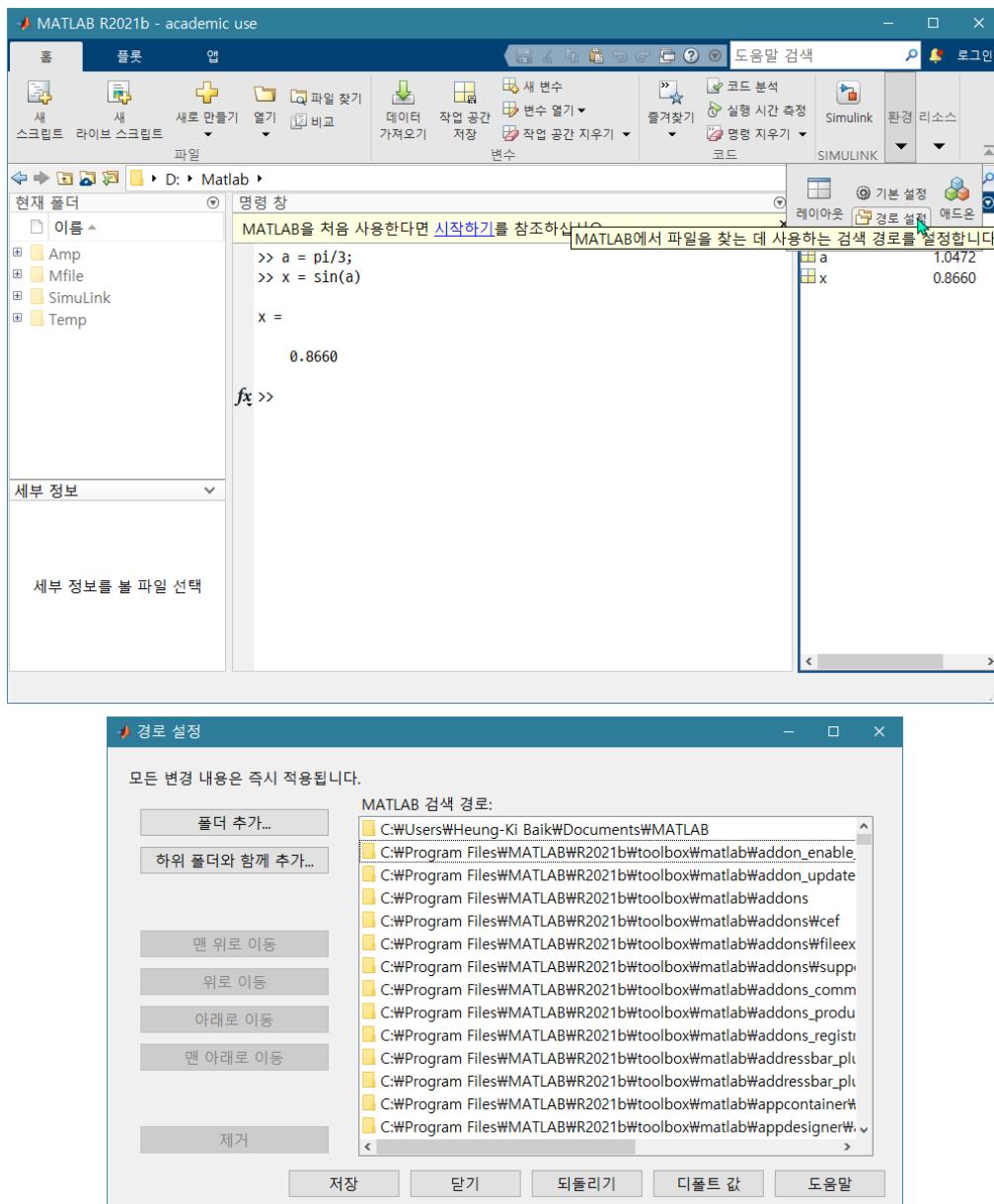


- ✗ 지방 관리



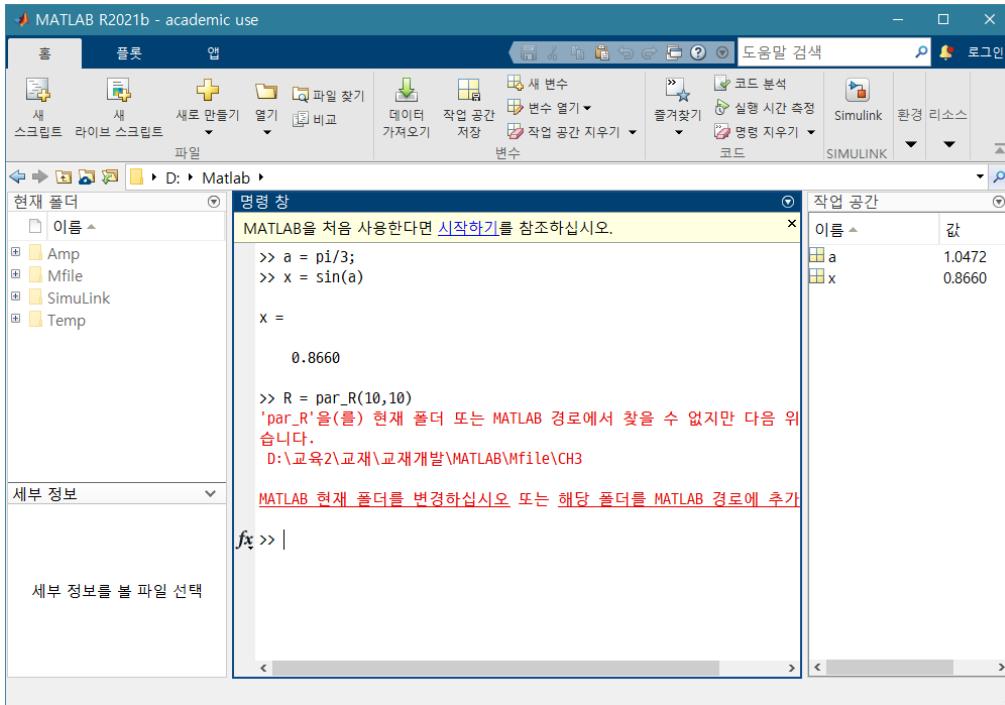
▶ 경로 설정(set path)

✓ 환경 – 경로 설정

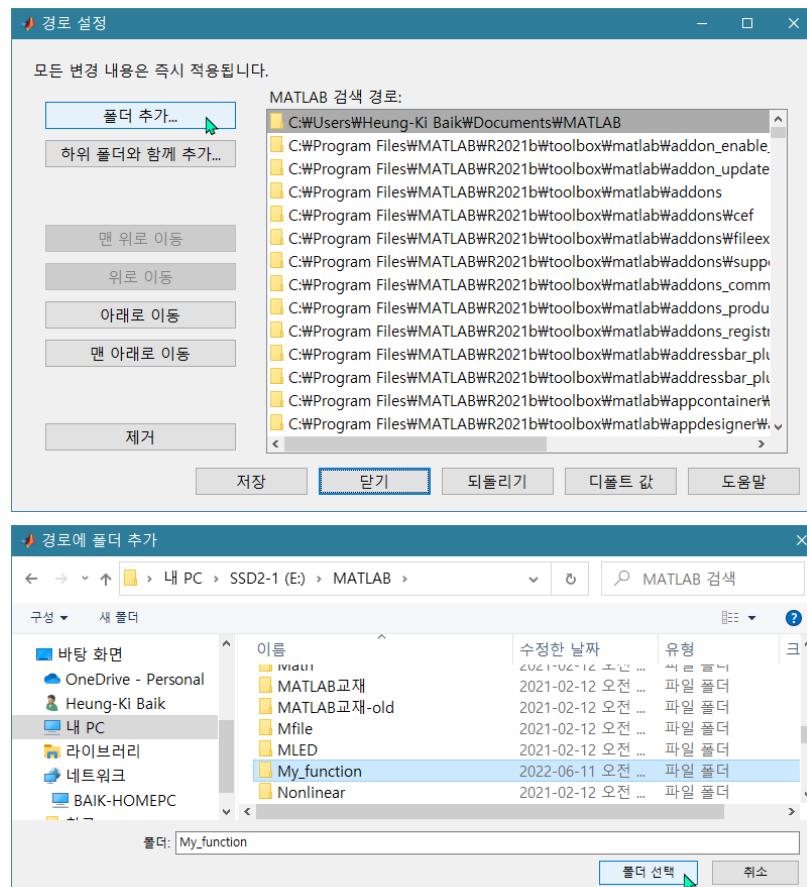


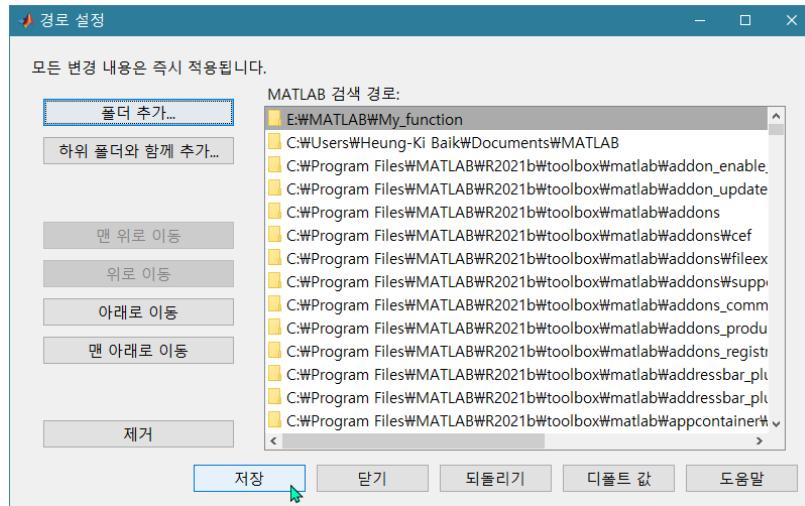
- ✓ MATLAB에서 함수를 사용하면 작업 폴더에서 함수를 찾고 없으면 경로 설정에 포함된 모든 폴더에서 함수를 찾는다. 그래도 못 찾으면 에러 문장이 표시된다.
- ✓ 경로 추가

- ✖ 나만의 함수를 만들었을 때 그 함수가 현재 폴더에 없으면 실행 시 에러가 발생한다. (예 : 병렬 저항값 구하는 함수)

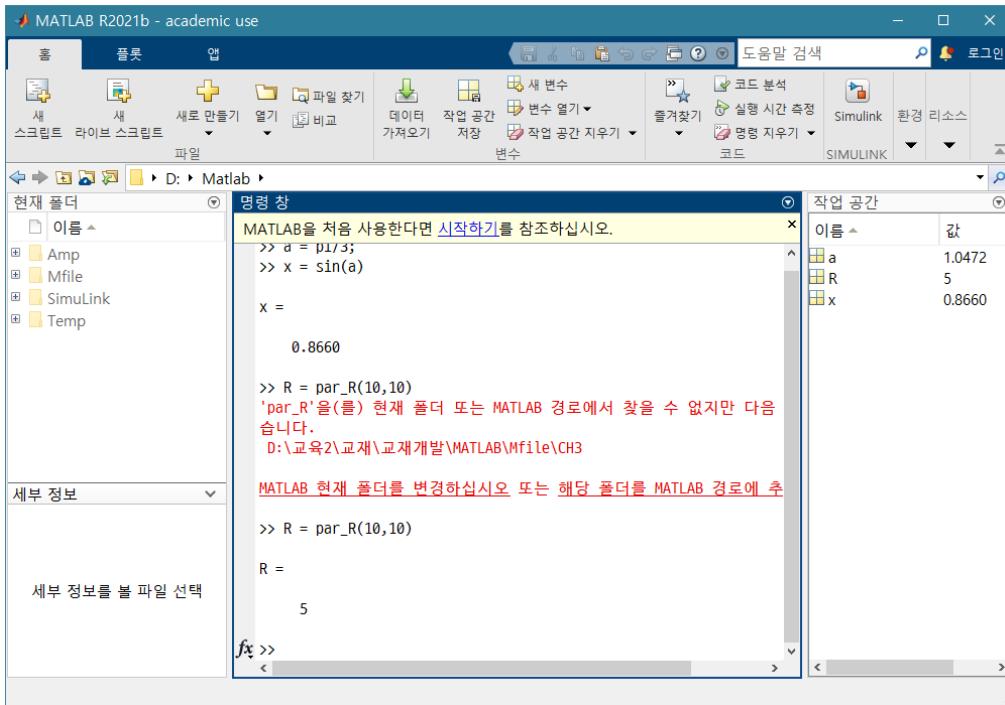


- ✖ 나만의 함수를 별도의 폴더에 두고 그 폴더를 등록시키면 에러없이 실행시킬 수 있다.



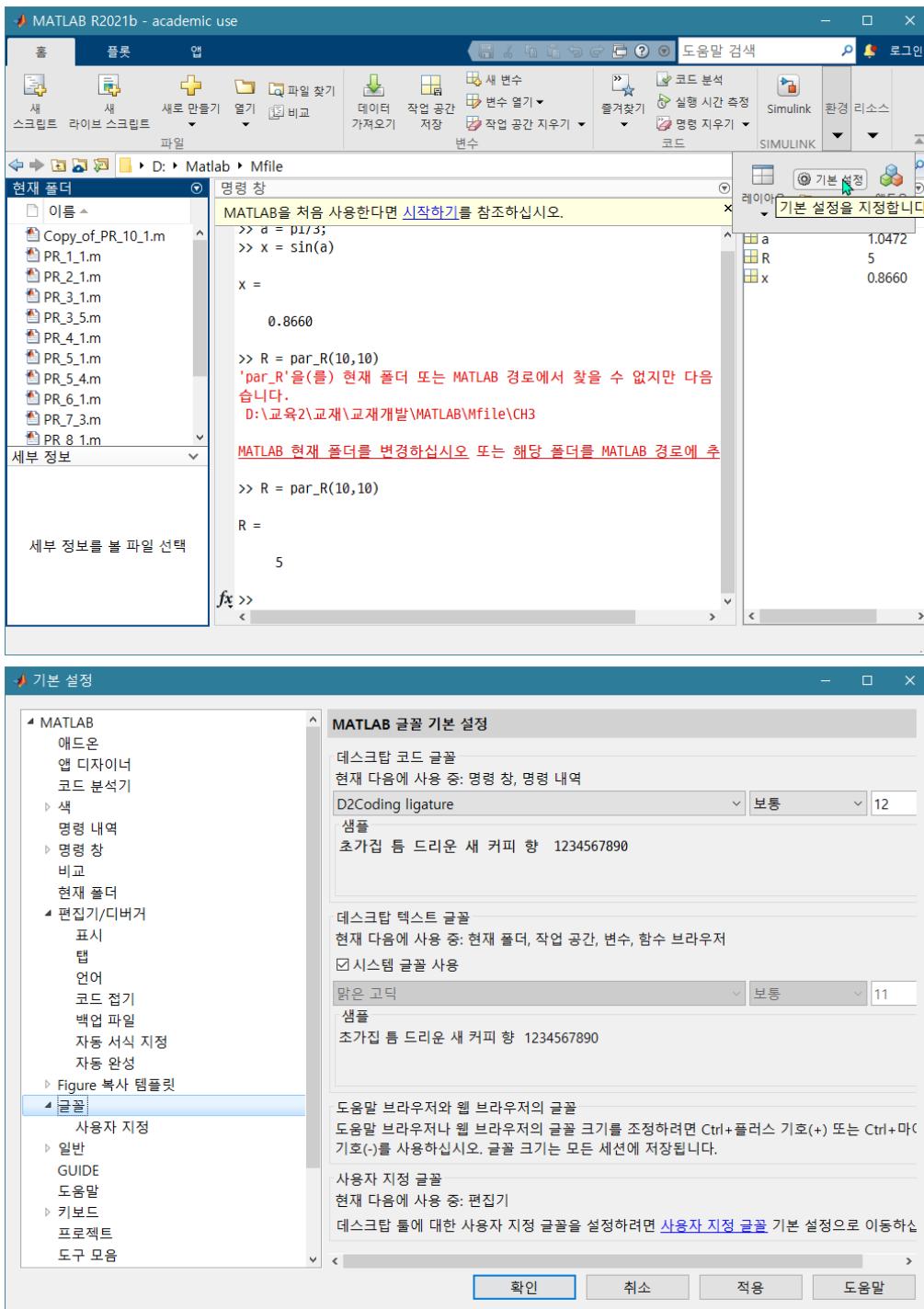


- ✖ 병렬 저항값 구하는 함수 재 실행 (현재 폴더에 함수가 없어도 에러가 발생하지 않는다.)

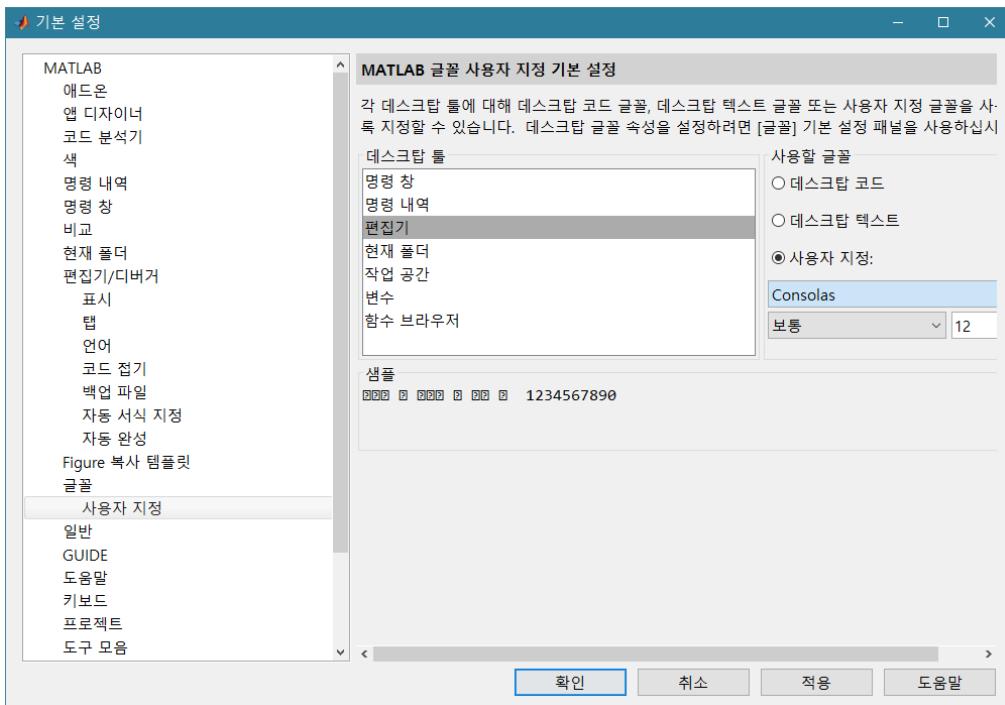


➊ 기본 설정(preference)

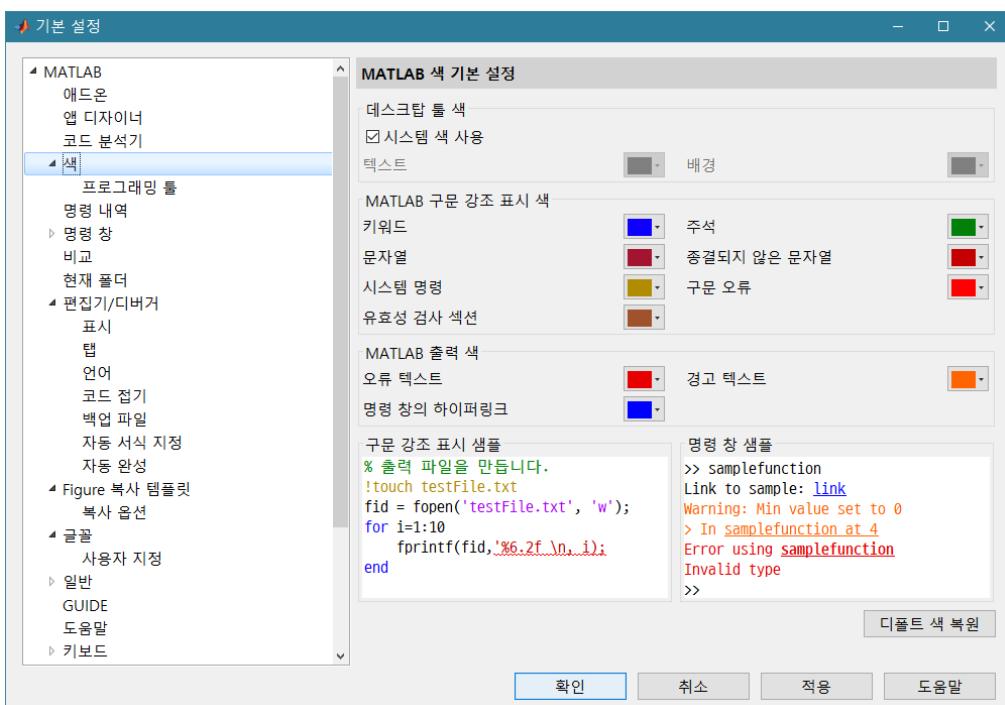
- ✓ MATLAB의 여러 가지 환경 설정을 바꿀 수 있다.
- ✓ 환경 – 기본 설정



- ✓ 글꼴 선택 시 주의 사항
 - ✗ 글꼴은 전체 선택 후 각 부분 창 별로 선택할 수 있다.
 - ✗ 명령 창의 여러 문장은 한글로 표시되므로 한글 표기 가능한 글꼴을 사용한다.
 - ✗ 편집기 글꼴은 문자 폰이 똑 같은 글꼴을 선택하되 한글 사용하려면 한글이 표시되는 글꼴을 사용한다.



문장 색 변경

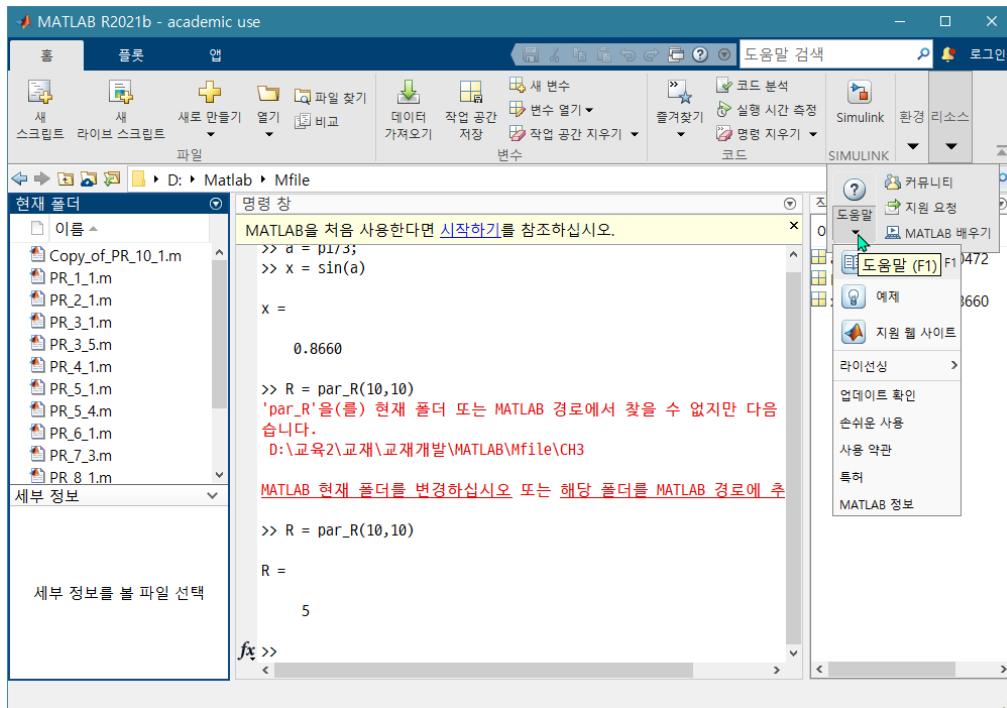


탭 크기 4로 변경

1.3 MATLAB 의 도움말

Desktop 에서의 도움말

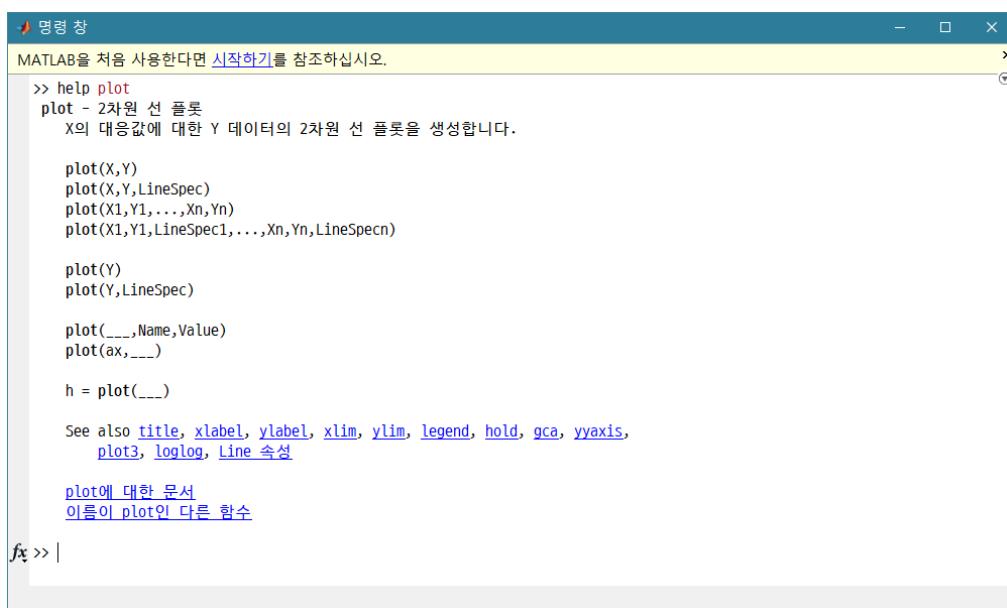
- ✓ 리소스 – 도움말



명령 줄(command line)에서의 도움말

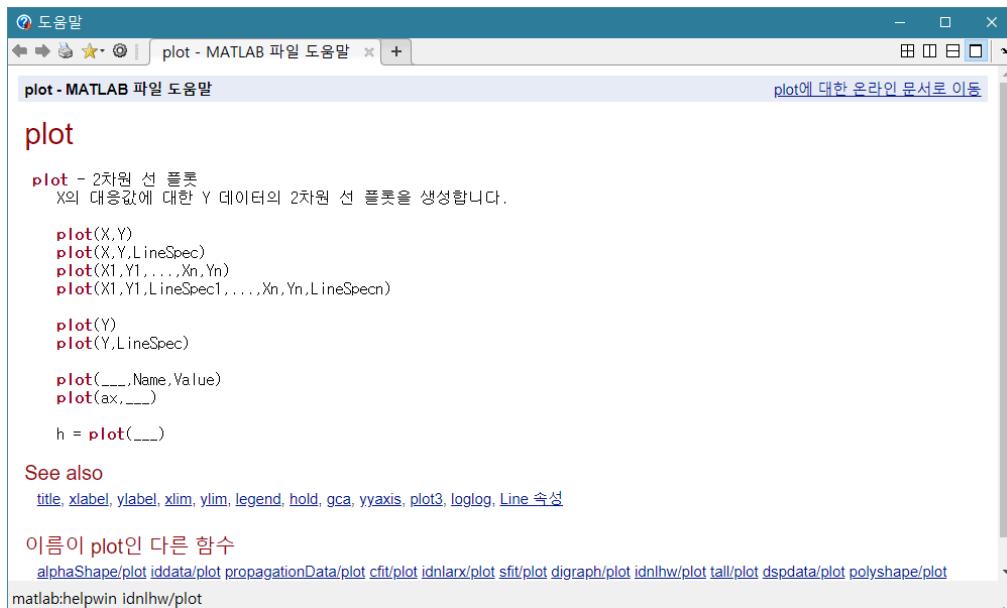
- ✓ 명령 창에 입력한다.
- ✓ help function_name

✗ 명령 창에 함수에 대한 정보를 보여준다.

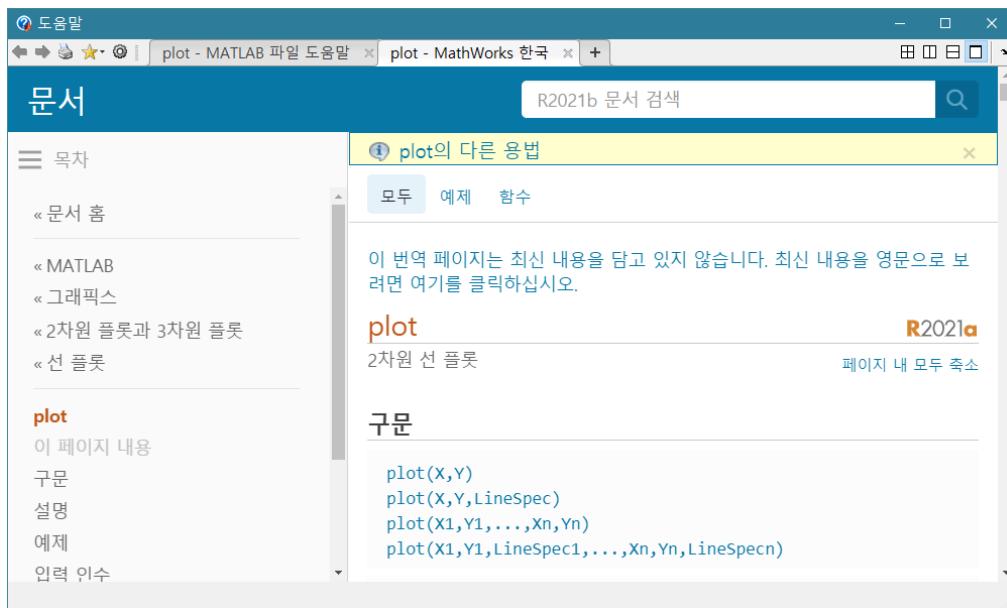


- ✓ helpwin function_name

- ✖ 별도의 창에 함수에 대한 정보를 보여준다.

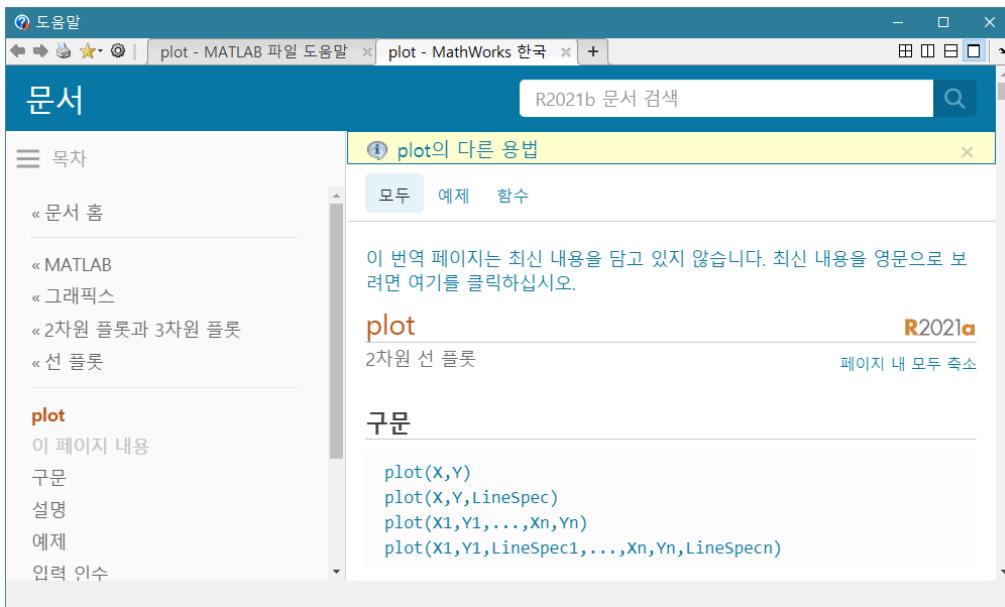


- ✖ 온라인 문서로 이동



✓ doc function_name

- ✖ 온라인 문서로 바로 이동한다.



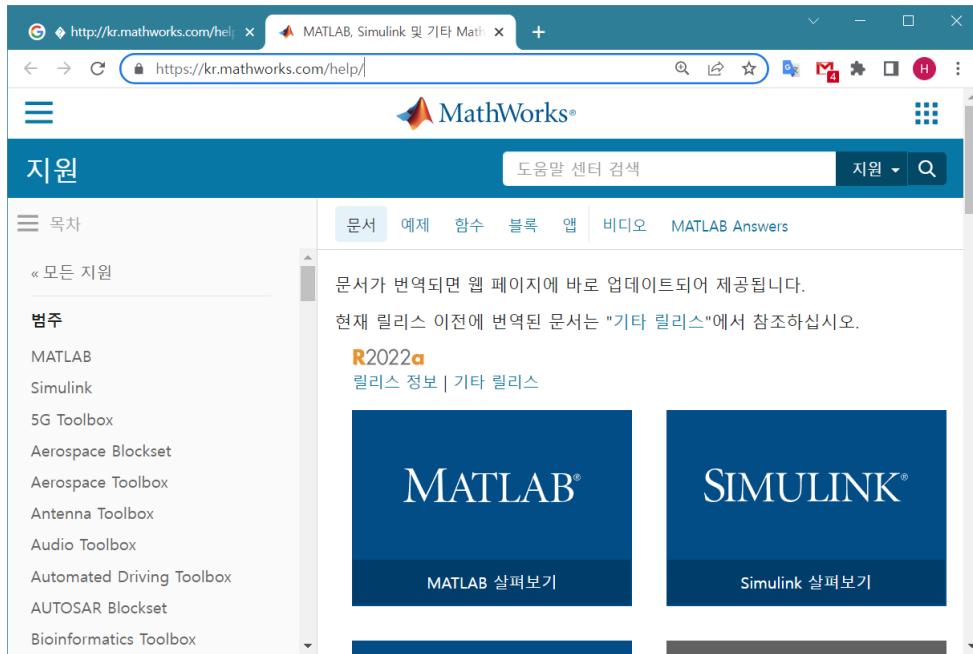
✓ Lookfor

- ✗ 명령 창에 찾고자 하는 단어를 포함하는 함수를 보여준다.
- ✗ 중지시키려면 Cont-C를 누른다.



✚ MATLAB 도움말 문서 지원 사이트

- ✓ <https://kr.mathworks.com/help/>



1.4 명령 창(command window)

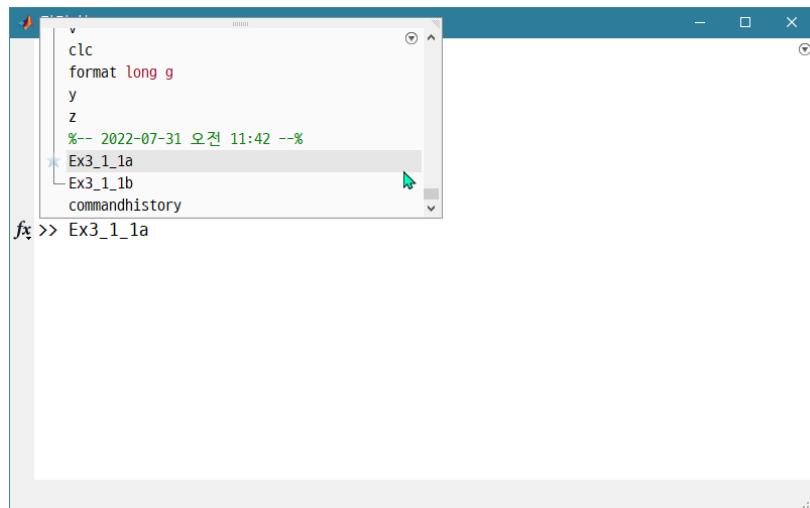
- ▶ 명령 줄(command line)에서의 연산
- ✓ 계산기 모드

```
>> -5/(4.8+5.2)^2
ans =
-0.0500
>> (3+4i)*(3-4i)
ans =
25
>> cos(pi/2)
ans =
6.1232e-017
```

- ✓ 변수 할당 모드

```
>> a = 2; % 세미 콜론은 변수의 출력을 억제한다.
>> b = 5;
>> a^b % 출력 변수가 없으면 ans에 결과를
ans = % 할당한다.
32
>> x = pi/2;
>> y = sin(x)
y =
1
```

- ✓ 모든 값은 double precision floating-point format으로 저장된다
- ▶ 기존에 입력했던 명령어를 다시 실행할 수 있다.
 - ✓ 명령 창에서 ↑ 키를 누르거나 commandhistory를 실행하면 전에 사용했던 명령어 목록을 볼 수 있다.
 - ✓ 명령어 목록에서 마우스를 이용하여 선택한 후 마우스나 화살표를 이용하여 원하는 곳에서 명령어를 편집할 수 있다.
 - ✓ 명령어를 선택한 후에도 ESC를 누르면 명령어가 clear 된다.



▣ 긴 문장의 입력 : Ellipsis (3 period) 사용

```
>> a = 1+2+3+4+5+6+7 ...  
+8+9+10  
a =  
55
```

▣ 한 줄에 여러 명령어 입력 : 세미콜론 (;) 사용

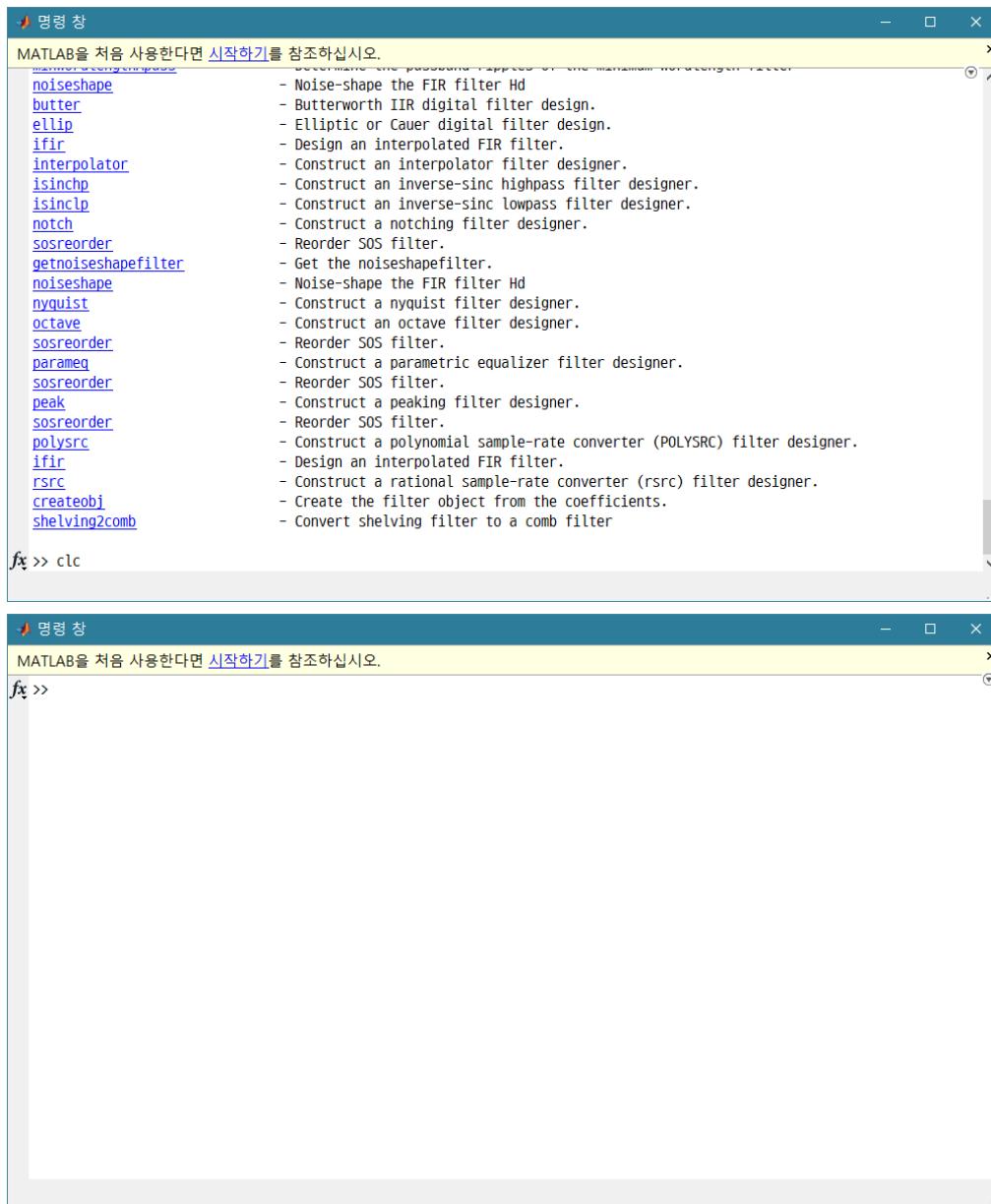
```
>> a = 1; b = 2; c = 3;  
>> d = a+b+c  
d =  
6
```

▣ 기본적인 함수

- ✓ clc
- ✓ clear
- ✓ close all
- ✓ format

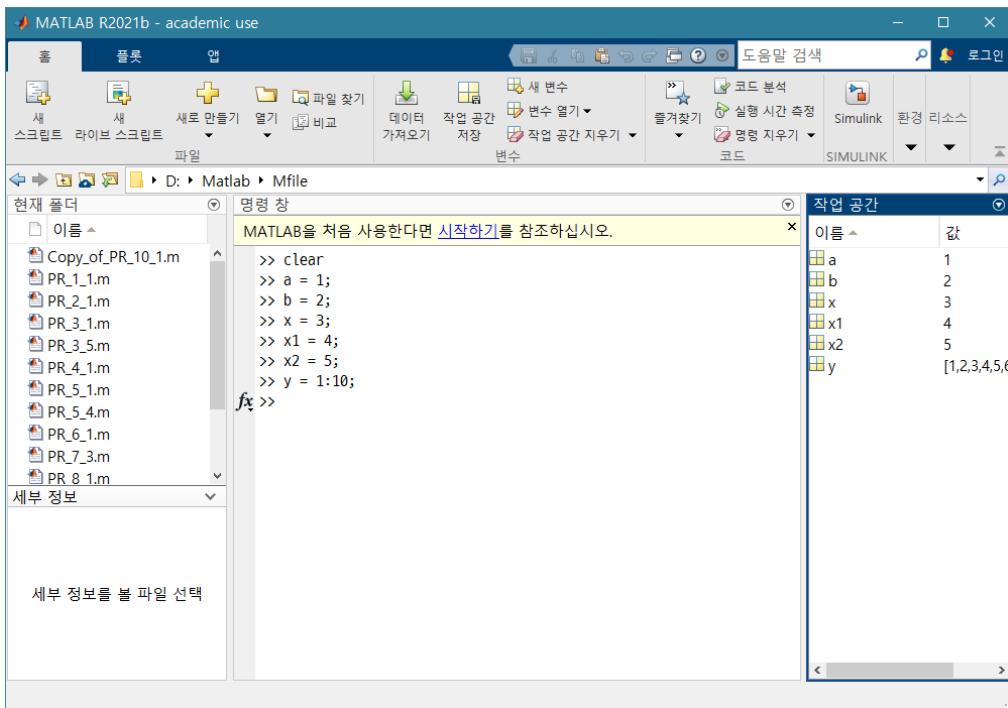
▣ clc (clear command window)

- ✓ 명령 창을 모두 지운다.



▶ clear

- ✓ `clear x` : 변수 `x` 를 지운다.
- ✓ `clear x*` : `x` 로 시작하는 모든 변수를 지운다.
- ✓ `clear` : 작업 공간의 모든 변수를 지운다.



✓ 연습 (작업 공간에서 결과 확인)

- ✗ clear y
- ✗ clear a b
- ✗ clear x*
- ✗ clear

➊ format

✓ 명령 창에 표시되는 형식을 지정한다.

명령어	설명
format short	0.001 ~ 1000인 수를 소수점 이하 4자리수의 고정 소수점으로 표시, 그 외의 수는 short e 형식으로 표시
format long	0.001 ~ 1000인 수를 소수점 이하 14자리수의 고정 소수점으로 표시, 그 외의 수는 long e 형식으로 표시
format short e	소수점 이하 4자리수의 부동 소수점으로 표시
format long e	소수점 이하 15자리수의 부동 소수점으로 표시
format short g	고정 소수점과 부동 소수점 표시 중에서 편리한 방법으로 표시 (유효 숫자는 5개)
format long g	고정 소수점과 부동 소수점 표시 중에서 편리한 방법으로 표시 (유효 숫자는 15개)

명령어	설명
format bank	소수점 이하 2자리까지만 표시
format compact	화면에 많은 정보가 표시되도록 빈 줄을 제거
format loose	format compact와 반대로 빈 줄 삽입
format	default로 format short 와 동일

format 예제

```
>> x = 1/3; y = 10^4/3; z = 10^-3/3; v = [10^3 10^-3];  
>> format  
>> format long  
>> format short  
>> format long e  
>> format short e  
>> format long g  
>> format short g  
>> format rat  
>> format compact  
>> format loose  
>> format
```

<pre> >> x = 1/3; y = 1/3*10^4; z = 1/3*10^-3; >> v = [10^3 10^-3]; >> format >> x x = 0.3333 >> y y = 3.3333e+03 >> z z = 3.3333e-04 >> v v = 1.0e+03 * 1.0000 0.0000 </pre>	<pre> >> format rat >> x x = 1/3 >> y y = 10000/3 >> z z = 1/3000 >> v v = 1000 1/1000 </pre>
<pre> >> format long >> x x = 0.3333333333333333 >> y y = 3.333333333333333e+03 >> z z = 3.333333333333333e-04 >> v v = 1.0e+03 * 1.0000000000000000 0.0000010000000000 </pre>	<pre> >> format short >> x x = 0.3333 >> y y = 3.333e+03 >> z z = 3.333e-04 >> v v = 1.0e+03 * 1.0000 0.0000 </pre>

The image displays four MATLAB command windows arranged in a 2x2 grid, illustrating the use of `format` commands to control numerical output.

Top Left Window (format long e):

```
>> format long e  
>> x  
  
x =  
  
3.33333333333333e-01  
  
>> y  
  
y =  
  
3.33333333333333e+03  
  
>> z  
  
z =  
  
3.33333333333333e-04  
  
>> v  
  
v =  
  
1번 열  
  
1.00000000000000e+03  
  
2번 열  
  
1.00000000000000e-03
```

Top Right Window (format short e):

```
>> format short e  
>> x  
  
x =  
  
3.3333e-01  
  
>> y  
  
y =  
  
3.3333e+03  
  
>> z  
  
z =  
  
3.3333e-04  
  
>> v  
  
v =  
  
1.0000e+03 1.0000e-03
```

Bottom Left Window (format long g):

```
>> format long g  
>> x  
  
x =  
  
0.33333333333333  
  
>> y  
  
y =  
  
3333.3333333333  
  
>> z  
  
z =  
  
0.00033333333333  
  
>> v  
  
v =  
  
1번 열  
  
1000  
  
2번 열  
  
0.001
```

Bottom Right Window (format short g):

```
>> format short g  
>> x  
  
x =  
  
0.33333  
  
>> y  
  
y =  
  
3333.3  
  
>> z  
  
z =  
  
0.00033333  
  
>> v  
  
v =  
  
1000 0.001
```

- ✓ 참고 : format long g 에서 v는 명령 창 폭이 좁아 위와 같음 (명령 창 폭을 키우고 다시 실행해 보기 바람)

The image shows two MATLAB command windows side-by-side. Both windows have a title bar labeled '명령 창' (Command Window) and a close button 'x'. The left window is titled '명령 창' and contains the following MATLAB session:

```
>> format compact
>> x
x =
    0.33333
>> y
y =
    3333.3
>> z
z =
    0.00033333
>> v
v =
    1000      0.001
>> format
>> format compact
>> x
x =
    0.3333
>> y
y =
    3.3333e+03
>> z
z =
    3.3333e-04
>> v
v =
    1.0e+03 *
    1.0000      0.0000
fx >>
```

The right window is also titled '명령 창' and contains the same session, but with different output widths:

```
>> format loose
>> x
x =
    0.3333
>> y
y =
    3.3333e+03
>> z
z =
    3.3333e-04
>> v
v =
    1.0e+03 *
    1.0000      0.0000
fx >>
```

In the right window, the output for variables x, y, and z is displayed with more digits than in the left window, while the output for v is identical.

1.5 연습 문제

- ▣ 문제 1. 나만의 함수를 만들고 이 함수가 항상 동작하도록 하는 방법에 대해 설명하라.
- ▣ 문제 2. 적절한 폴더를 만들고 이 폴더에서 모든 작업이 이루어지도록 하는 방법에 대해 설명하라.
- ▣ 문제 3. 명령 창을 별도의 창으로 만드는 방법에 대해 설명하라.
- ▣ 문제 4. MATLAB 의 특징을 조사하라.
- ▣ 문제 5. MATLAB 에서 제공하고 있는 내용 중에서 현재 배우고 있는 전공 분야 (전공 과목)와 관련된 내용이 무엇이 있는지 조사하라.

2. 행렬 연산(Matrix Operation)

✚ MATLAB에서 행렬을 배열을 의미하며 넓은 의미로 벡터도 행렬로 간주한다.

2.1 행렬의 생성

✚ 행 구분자, 열 구분자를 이용한 행렬의 생성

- ✓ 행 구분자 : 세미콜론(;)
- ✓ 열 구분자 : 빈 칸, 컴마(,)

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

✚ 콜론 연산자(:)를 이용한 벡터의 생성

```
>> n = 1:5
n =
    1     2     3     4     5
>> a = 1.5:4.5
```

```
a =
1.5000 2.5000 3.5000 4.5000
>> x = 0:0.2:1
x =
0 0.2000 0.4000 0.6000 0.8000 1.0000
```

 유ти리티 함수를 이용한 행렬의 생성

- ✓ MATLAB은 다양한 함수를 이용하여 원하는 행렬을 쉽게 생성할 수 있다.
-

```
>> A = ones(2,3) % 모든 원소가 1
>> B = zeros(3) % 모든 원소가 0 (3x3)
>> C = eye(3) % 단위 행렬
>> D = pascal(3) % Pascal 행렬
>> E = rand(2,3) % uniform 난수
>> F = randn(3) % Gaussian 난수 (3x3)
>> G = randi(10,2,3) % uniform 난수(정수)
>> H = magic(3) % 마방진
>> x = linspace(0,1,6) % 선형 간격 벡터
>> y = logspace(0,5,6) % 로그 간격 벡터
```

 유ти리티 함수를 이용한 행렬의 생성

```
>> A = ones(2,3)
A =
1 1 1
1 1 1
>> B = zeros(3)
B =
0 0 0
0 0 0
0 0 0
```

```
>> x = linspace(0,1,6)
    0    0.2000    0.4000    0.6000    0.8000    1.0000
>> y = logspace(0,5,6)
    1    10    100    1000    10000    100000
```

▶ 원소값 대입에 의한 행렬의 생성

```
>> P(2,3) = 6
P =
    0    0    0
    0    0    6
>> Q = [1 2 3]
Q =
    1    2    3
>> Q(2,4) = 8
    1    2    3    0
    0    0    0    8
```

2.2 행렬의 원소 참조

▣ 행렬의 원소

- ✓ 행렬은 내부적으로 벡터로 저장된다.
- ✓ MATLAB은 열 우선이다.

$$\mathbf{A} = \begin{bmatrix} 1 & 6 & 11 & 16 & 21 \\ 2 & 7 & 12 & 17 & 22 \\ 3 & 8 & 13 & 18 & 23 \\ 4 & 9 & 14 & 19 & 24 \\ 5 & 10 & 15 & 20 & 25 \end{bmatrix}$$

▣ 행렬의 원소 참조

```
>> A(3,1)
```

```
ans =
```

```
3
```

```
>> A(17)
```

```
ans =
```

```
17
```

▣ 행렬의 부분 참조

```
>> A(2,1:5)
```

```
ans =
```

```
2 7 12 17 12
```

```
>> A(2,:end)
```

```
ans =
```

```
2 7 12 17 12
```

```
>> A(2,:) % 2 행
```

```
ans =
```

```
2 7 12 17 12
```

```
>> A(:,end) % 마지막 열
```

```
ans =
```

```
21  
22  
23  
24  
25  
>> A(4:5,2:3)          % 부분 행렬  
ans =  
    9    14  
   10    15
```

2.3 행렬의 정보

✚ 행렬의 크기, 길이, 원소 갯수, 차원

```
>> A = randn(2,3) % Gaussian 난수
A =
-0.4326    0.1253   -1.1465
-1.6656    0.2877   1.1909

>> k = size(A) % 배열의 크기
k =
2      3

>> [n,m] = size(A) % 배열의 크기
n =
2
m =
3

>> a = length(A) % 배열의 길이
a =
3

>> b = numel(A) % 배열의 원소 갯수
b =
6

>> d = ndims(A) % 배열의 차원
d =
2
```

✚ 행렬의 자료 구조

```
>> A = randn(2,3); % Gaussian 난수
>> x = [1 2 3] % 행 벡터
>> y = [1; 2; 3] % 열 벡터
```

```
>> isscalar(A)
ans =
0                                     % 거짓

>> isscalar(x)
ans =
0                                     % 거짓

>> isvector(A)
ans =
0                                     % 거짓

>> isvector(x)
ans =
1                                     % 참

>> ismatrix(A)
ans =
1                                     % 참

>> ismatrix(x)
ans =
1                                     % 참 (1 x 3 인 행렬로 본다.)

>> isrow(A)
ans =
0                                     % 거짓

>> isrow(x)
ans =
1                                     % 참

>> iscolumn(x)
ans =
0                                     % 거짓

>> isvector(y)
ans =
```

```
1          % 참  
>> isrow(y)  
ans =  
0          % 거짓  
>> iscolumn(y)  
ans =  
1          % 참
```

2.4 행렬의 변형

행렬의 벡터화

```
>> A = [1 2; 3 4]; % 2x2 행렬
>> x = A(:); % 열 벡터
x =
1
3
2
4
>> y = A(:)'; % 행 벡터
y =
1     3     2     4
```

벡터, 행렬의 행렬화

```
>> x = [1 2 3 4 5 6]; % 행 벡터
>> X = reshape(x, 2, 3); % 2x3 행렬
X =
1     3     5
2     4     6
>> Y = [1 2; 3 4; 5 6]; % 3x2 행렬
Y =
1     2
3     4
5     6
>> Z = reshape(Y, 2, 3); % 2x3 행렬
Z =
1     5     4
3     2     6
```

 행렬의 회전과 대칭화

```
>> A = [1 2; 3 4; 5 6]
A =
1 2
3 4
5 6
>> B = rot90(A) % 90 도 회전
B =
2 4 6
1 3 5
>> C = fliplr(A) % 좌우 대칭
C =
2 1
4 3
6 5
>> D = flipud(A) % 상하 대칭
D =
5 6
3 4
1 2
```

 행렬의 치환

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

```

    7     8     9
>> A(2,:) = b

A =
    1     2     3
    7     8     9

>> C = randn(2,3)

C =
    0.3376    -1.6642    -0.2781
    1.0001    -0.5900     0.4227

>> C(find(C<0)) = 0          % 0 보다 작은 원소는 0으로 치환

C =
    0.3376         0         0
    1.0001         0     0.4227

```

벡터의 정렬

```

>> x = [30 20 40 60 10 70];
>> y = sort(x,'ascend')        % 오름 차순 정렬

y =
    10    20    30    40    60    70

>> z = sort(x,'descend')      % 내림 차순 정렬

z =
    70    60    40    30    20    10

>> [z,p] = sort(x,'descend')

z =
    70    60    40    30    20    10

p =
    6     4     3     1     2     5

```

 행렬의 정렬

```
>> A = [3 2 4; 1 7 5]
A =
    3    2    4
    1    7    5
>> B = sort(A, 'ascend')          % 모든 열을 오름 차순 정렬
B =
    1    2    4
    3    7    5
> C = sortrows(A)                % 1 열 값을 기준으로 행 오름 차순 정렬
C =
    2    3    4
    1    5    7
> D = sortrows(A,1)              % 1 열 값을 기준으로 행 오름 차순 정렬
D =
    2    3    4
    1    5    7
> E = sortrows(A,2)              % 2 열 값을 기준으로 행 오름 차순 정렬
E =
    2    3    4
    1    5    7
```

2.5 행렬의 행과 열 삭제

행과 열의 삭제

```
>> A = pascal(3)
```

```
A =
```

```
1 1 1  
1 2 3  
1 3 6
```

```
>> A(3, :) = [] % 3 행 삭제
```

```
A =
```

```
1 1 1  
1 2 3
```

```
>> A = pascal(3)
```

```
A =
```

```
1 1 1  
1 2 3  
1 3 6
```

```
>> A(:, 2) = [] % 2 열 삭제
```

```
A =
```

```
1 1  
1 3  
1 6
```

2.6 행렬의 함수

Matrix function	
inv	matrix inverse
det	determinant
rank	matrix rank
eig	eigenvector & eigenvalue
svd	singular value decomposition
norm	matrix / vector norm

▶ 역 행렬

```
>> A = [1 2 3;4 5 2;3 5 4];
>> B = inv(A) % 역 행렬
B =
2.0000 1.4000 -2.2000
-2.0000 -1.0000 2.0000
1.0000 0.2000 -0.6000
```

▶ 행렬값(determinant)

```
>> A = [1 2 3;4 5 2;3 5 4];
>> b = det(A) % 행렬식
b =
5.0000
```

▶ 고유값(eigenvalue)과 고유 벡터(eigenvector)

```
>> A = [1 0.5; 0.5 1];
>> [V,D] = eig(A)
V = % 고유 벡터
-0.7071 0.7071
0.7071 0.7071
D = % 고유값 (0.5, 1.5)
```

```
0.5000    0  
0         1.5000
```

▶ 유용한 행렬 함수

- ✓ MATLAB 행렬 연산은 열 우선 연산이다.

```
>> 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 = size(A)
```

```
a =
```

```
3   4
```

```
>> b = length(A)
```

```
b =
```

```
4
```

```
>> c = sum(A)
```

```
c =
```

```
15   18   21   24
```

```
>> d = prod(A)
```

```
d =
```

```
45   120  231  384
```

```
>> e = mean(A)
```

```
e =
```

```
5   6   7   8
```

```
>> f = max(A)
```

```
f =
```

```
9   10  11  12
```

2.7 행렬의 연산

▶ 벡터와 행렬과 배열 연산자

	Matrix Operators	Array Operators
()	parentheses	
+	addition	
-	subtraction	
*	multiplication	. *
/	division	. /
\	left division	
'	transpose	. '
^	power	. ^

▶ 행렬과 행렬의 덧셈과 뺄셈

- ✓ 행렬의 크기가 같아야 한다.

```
>> A = [1 2 3; 4 5 6];
>> B = [1 3 2; 4 6 5];
>> C = A+B
C =
    2     5     5
    8    11    11
>> D = A-B
D =
    0    -1     1
    0    -1     1
```

▶ 행렬과 스칼라의 덧셈과 뺄셈

- ✓ 행렬과 스칼라의 덧셈과 뺄셈이 가능하다.

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

B =

4	5	6
7	8	9

>> C = A-2**C =**

-1	0	1
2	3	4

>> D = 4-A**D =**

3	2	1
0	-1	-2

 행렬과 행렬의 곱셈

```
>> A = [1 2 3; 4 5 6];
>> B = [2 5; 3 4; 1 6];
>> C = A*B
C =

```

11	31
29	76

 행렬과 행렬의 원소 대 원소 곱셈 (배열 곱셈)

- ✓ 행렬의 크기가 같아야 한다.

```
>> A = [1 2 3; 4 5 6];
>> B = [2 3 1; 5 4 6];
>> C = A.*B
C =

```

2	6	3
20	20	36

▶ 행렬과 스칼라의 곱셈

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

```
>> B = 3*A
```

```
B =
```

3	6	9
12	15	18

▶ 행렬과 행렬의 나눗셈

- ✓ 행렬 **B**의 역 행렬이 존재하고 크기가 같아야 한다.
-

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

```
>> B = [4 5; 3 4];
```

```
>> C = A/B % A/B = A*inv(B)
```

```
C =
```

-7	10
-5	7

▶ 행렬과 행렬의 원소 대 원소 나눗셈 (배열 나눗셈)

- ✓ 행렬의 크기가 같아야 한다.
-

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

```
>> B = [4 5; 3 4];
```

```
>> C = A./B
```

```
C =
```

0.5000	1.0000
0.3333	0.7500

▶ 행렬과 행렬의 좌 나눗셈(left division)

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

```
>> B = [4 5; 3 4];
```

```
>> C = A\b % A\b = inv(A)*B
C =
-3 -5
2 3
```

▣ 좌 나눗셈을 이용한 선형 연립방정식 풀기

$$\begin{aligned}-x_1 + x_2 + 2x_3 &= 2 \\ 3x_1 - x_2 + x_3 &= 6 \\ -x_1 + 3x_2 + 4x_3 &= 4\end{aligned}$$

```
>> A = [-1 1 2; 3 -1 1; -1 3 4];
>> b = [2; 6; 4];
>> x = A\b % A\b = inv(A)*b
x =
1.0000
-1.0000
2.0000
```

▣ 행렬의 누승

- ✓ 행렬 A는 정방 행렬이어야 한다.
-

```
>> A = [1 2; 3 4];
>> B = A^2
B =
7 10
15 22
```

▣ 행렬의 원소 대 원소 누승 (배열 누승)

```
>> A = [1 2; 3 4];
>> B = A.^2
B =
```

1	4
9	16

▶ 실수 행렬의 전치행렬(transpose)

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

▶ 복소 행렬의 전치행렬(transpose)

```
>> A = [1 2; 3 4];
>> C = (1+j)*A
C =
1.0000 + 1.0000i 2.0000 + 2.0000i
3.0000 + 3.0000i 4.0000 + 4.0000i
>> D = C' % Hermitian matrix
D =
1.0000 - 1.0000i 3.0000 - 3.0000i
2.0000 - 2.0000i 4.0000 - 4.0000i
>> E = C.' % Transpose matrix
E =
1.0000 + 1.0000i 3.0000 + 3.0000i
2.0000 + 2.0000i 4.0000 + 4.0000i
```

2.8 문자열

▣ 문자열의 생성

- ✓ 흔 따옴표(')를 이용하여 생성한다.

```
>> str1 = 'Hello,'           % 문자열은 빨강색으로 표시된다.

str1 =
Hello,

>> str2 = 'Everyone'

str2 =
Everyone
```

▣ 문자열은 원소가 문자인 벡터나 행렬이다.

- ✓ 문자열의 각 문자는 행렬의 원소에 해당한다.

H	e	l	l	o	,
---	---	---	---	---	---

▣ 문자열의 연결

- ✓ 문자열의 연결은 행렬의 연결과 같다.
- ✓ 문자열을 이용하여 행렬을 만들 때 각 행은 크기가 같아야 한다.

```
>> str1 = 'Hello,';
>> str2 = 'Everyone';
>> new_str1 = [str1, ' ', str2]
new_str1 =
Hello, Everyone

>> new_str2 = [str1;str2]
```

다음 사용 중 오류가 발생함: **vertcat**

연결 (Concatenate) 된 행렬의 차원이 일치하지 않습니다.

▣ 문자열의 크기가 다를 때의 행렬 만드는 방법

- ✓ 문자열 연결 함수를 이용한다.

```

>> str1 = 'Hello,' ;
>> str2 = 'Everyone' ;
>> new_str3 = strvcat(str1,str2)
new_str3 =
Hello,
Everyone
>> new_str4 =str2mat(str1,str2)
new_str4 =
Hello,
Everyone

```

▣ 문자열 비교 함수

Function	Usage
strcmp	Compare whole strings
strncmp	Compare first ‘n’ characters
findstr	Find a sub-string with a larger string
strrep	Replace string with another

▣ 문자열 변환 함수

Function	Usage
num2str	Convert from numeric to string array
str2num	Convert from string to numeric array
mat2str	Convert matrix to EVAL’able string
str2mat	Form blank padded character matrix from strings

```

>> str1 = 'Hello, Everyone';
>> str2 = 'Hello, Student';
>> a = strcmp(str1,str2)
a =
0
>> b = strncmp(str1,str2,5)
b =

```

```
1  
>> c = findstr(str1,'Every')  
c =  
8  
>> new_str = strrep(str1,'Hello','Hi')  
New_str =  
Hi, Everyone
```

▣ 문자의 문자열 변환

```
>> str1 = num2str(pi)  
str1 =  
3.1416  
>> a = str2num(str1)  
a =  
3.1416  
>> M = 10;  
>> str = ['M = ',num2str(M)];  
disp(str)  
M = 10
```

▣ 벡터나 행렬의 문자열 변환

```
>> A = [1 2 3; 4 5 6; 7 8 9];  
>> str = mat2str(A)  
str =  
[1 2 3;4 5 6;7 8 9]
```

2.9 셀형(cell) 배열

■ 셀형 배열

- ✓ 셀이라는 인덱싱된 데이터 컨테이너를 사용하는 데이터 형이다.
- ✓ 각 셀은 서로 다른 데이터형을 포함할 수 있다.
- ✓ 셀형 배열은 주로 텍스트 목록, 텍스트와 숫자 조합, 각기 크기가 다른 숫자형 배열 중 하나를 포함한다.

■ 셀형 배열의 생성

- ✓ {}를 사용하여 빈 0×0 셀형 배열을 생성할 수 있다.

```
>> A = {}
```

```
A =
```

0×0 비어 있는 cell 배열

- ✓ cell 함수를 사용하여 생성할 수 있다.

```
C = cell(n)
C = cell(m,n)
```

✗ $n \times n, m \times n$ 셀형 배열을 만든다.

```
>> B = cell(2)
```

```
B =
```

2×2 cell 배열

```
{0×0 double} {0×0 double}
{0×0 double} {0×0 double}
```

- ✓ 셀형 배열 생성 연산자 {}를 사용하여 생성한다.

```
>> C = {'one', 'two', 'three'; 1,2,3}
```

```
C =
```

2×3 cell 배열

```
{'one'} {'two'} {'three'}
```

```
{ [ 1] } { [ 2] } { [ 3] }
```

셀형 배열의 데이터에 접근하는 방법

- ✓ 셀형 배열 부분 셀 참조 (인덱스 이용)
-

```
>> C1 = C(1:2,1:2)
C1 =
2×2 cell 배열

{ 'one' } { 'two' }
{ [ 1] } { [ 2] }

>> C2 = C(1,:)
C2 =
1×3 cell 배열

{ 'one' } { 'two' } { 'three' }

>> C3 = C(2,:)
C3 =
1×3 cell 배열

{ [1] } { [2] } { [3] }
```

- ✓ 셀형 배열을 일반 행렬로 변환
-

```
>> a1 = cell2mat(C3)
a1 =
1 2 3
```

- ✓ 셀형 배열 데이터 접근 ({}) 사용
-

```
>> b1 = C{1,1}
b1 =
'onе'

>> b2 = C{2,2}
```

```
b2 =
```

```
2
```

☞ 셀형 배열에 데이터에 넣기.

```
>> B{1,2} = 'MATLAT'
```

```
B =
```

```
2x2 cell 배열
```

```
{0x0 double} { 'MATLAT' }
```

```
{0x0 double} {0x0 double}
```

2.10 구조체형(struct) 배열

▣ 구조체형 배열은 필드라는 데이터 컨테이너를 사용하여 관련 데이터를 그룹화하는 데이터형이다.

- ✓ 각 필드에는 모든 데이터형이 포함될 수 있다.
- ✓ 필드의 데이터에 접근하려면 `strucName.fieldName` 형식의 점을 사용한다.

▣ 구조체형 배열의 생성

- ✓ `struct` 함수

```
s = struct
s = struct([])
s = struct(fld1,val1, ..., fldN,valN)
```

- ✗ 필드가 없는 스칼라(1x1) 구조체를 생성한다.
- ✗ 필드가 없는 빈(0 x 0) 구조체를 생성한다.
- ✗ 주어진 필드와 값을 갖는 구조체를 생성한다.

```
>> s1 = struct
s1 =
    struct에 필드가 없습니다.

>> s2 = struct([])
s2 =
    0x0 비어 있는 struct 배열에 필드가 없습니다.

>> s3 = struct('name','MATLAB 교육','num',[])
s3 =
    다음 필드를 포함한 struct:
        name: 'MATLAB 교육'
        num: []
>> s4.name = 'MATLAB 교육';
>> s4.num = 29;
```

```
>> s4
```

```
s4 =
```

다음 필드를 포함한 struct:

```
name: 'MATLAB 교육'
```

```
num: 29
```

```
>> s4.teacher = '백흥기'
```

```
s4 =
```

다음 필드를 포함한 struct:

```
name: 'MATLAB 교육'
```

```
num: 29
```

```
teacher: '백흥기'
```

▣ 구조체 배열의 데이터에 접근하기.

```
>> n = s4.num
```

```
n =
```

```
29
```

▣ 구조체 배열에 데이터 넣기.

```
>> s3.num = 29
```

```
s3 =
```

다음 필드를 포함한 struct:

```
name: 'MATLAB 교육'
```

```
num: 29
```

2.11 관계 연산자

<code>==</code>	<code>eq</code>	equal
<code>~=</code>	<code>ne</code>	not equal
<code>></code>	<code>gt</code>	greater than
<code><</code>	<code>lt</code>	less than
<code>>=</code>	<code>ge</code>	greater than or equal
<code><=</code>	<code>le</code>	less than or equal

▣ 스칼라와 스칼라의 비교

```

>> a = 3; b = 4;
>> c = a==b                                % c = eq(a,b)
c =
0                                         % 거짓
>> d = a~=b                                % d = ne(a,b)
d =
1                                         % 참
>> e = a<b                                % e = lt(a,b)
e =
1                                         % 참

```

▣ 복소수와 복소수의 비교

- ✓ 복소수의 대소 비교는 실수부만 비교한다.

```

>> a = 3+4i; b = 3-2i;
>> c = a==b
c =
0                                         % 거짓 (실수부, 허수부 비교)
>> d = a~=b
d =
1                                         % 참 (실수부, 허수부 비교)

```

```
>> e = a>b  
e =  
0 % 거짓 (실수부만 비교)
```

▶ 스칼라와 벡터(행렬)의 비교

```
>> a = 3; b = [3 4 1];  
>> c = a==b  
c =  
1 0 0 % 참, 거짓, 거짓  
>> d = a~=b  
d =  
0 1 1 % 거짓, 참, 참  
>> e = a>=b  
e =  
1 0 1 % 참, 거짓, 참
```

▶ 벡터(행렬)와 벡터(행렬)의 비교

```
>> A = [1 2; 2 1];  
>> B = [1 3; 1 1];  
>> C = A==B  
C =  
1 0  
0 1  
>> D = A>B  
D =  
0 0  
1 0
```

2.12 논리 연산자

▣ 논리 연산자

<code>~</code>	<code>not</code>	Logical NOT
<code>&</code>	<code>and</code>	Logical AND
<code> </code>	<code>or</code>	Logical OR
	<code>xor</code>	Logical Exclusive OR
	<code>any</code>	True if any element of vector is nonzero
	<code>all</code>	True if all elements of vector is nonzero

▣ 변수의 논리 연산

```

>> a = 0; b = 1; c = 2;

>> ~c

ans =

    0                      % 거짓

>> d = a&b           % d = and(a,b)

d =

    0                      % 거짓

>> e = a|b           % e = or(a,b)

e =

    1                      % 참

>> f = xor(a,b)

f =

    1                      % 참

```

▣ 벡터(행렬) 변수의 논리 연산

```

>> a = [0 1 2 3];

>> any(a)

ans =

    1                      % 참

```

```
>> all(a)  
ans =  
0 % 거짓  
>> b = ~a  
b =  
1 0 0 0 % 참, 거짓, 거짓, 거짓
```

2.13 다항식

▣ 다항식 표기법

- ✓ 다항식을 내림차순으로 정리한 후 계수를 벡터로 나타낸다.

$$y = f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \Rightarrow f = [a_n, a_{n-1}, \dots, a_1, a_0]$$

$$f(x) = 2x^3 + 3x^2 + x + 5$$

$$g(x) = x^2 + 2x + 3$$

```
>> f = [2 3 1 5]; % f(x)
```

```
>> g = [1 2 3]; % g(x)
```

▣ 다항식의 값

```
>> fv = polyval(f, 2)
```

```
fv =
35 % f(2)=35
```

▣ 다항식의 곱셈과 나눗셈

$$f(x)g(x) = 2x^5 + 7x^4 + 13x^3 + 16x^2 + 13x + 15$$

$$\frac{f(x)}{g(x)} = 2x - 1 + \frac{-3x + 8}{g(x)}$$

```
>> p = conv(f, g) % p(x)=f(x)*g(x)
```

```
p =
2 7 13 16 13 15
```

```
>> [q, r] = deconv(f, g) % f(x)/g(x)
```

```
q =
2 -1 % 몫
```

```
r =
0 0 -3 8 % 나머지
```

▣ 다항식과 근

$$f(x) = 2x^3 + 3x^2 + x + 5 = 0$$

$$f(x) = x^3 + \frac{3}{2}x^2 + \frac{1}{2}x + \frac{5}{2} = 0$$

```
>> r = roots(f) % f(x)=0
```

r =

-1.9186
0.2093 + 1.1222i
0.2093 - 1.1222i

```
>> p = poly(r)
```

p =

1.0000 1.5000 0.5000 2.5000

다항식의 미분

✓ 다항식의 미분

$$f'(x) = 6x^2 + 6x + 1$$

✓ 두 다항식의 곱의 미분

$$[f(x)g(x)]' = 10x^4 + 28x^3 + 39x^2 + 32x + 13$$

✓ 두 다항식의 나눗셈의 미분

$$\left[\frac{f(x)}{g(x)} \right]' = \frac{2x^4 + 8x^3 + 23x^2 + 8x - 7}{x^4 + 4x^3 + 10x^2 + 12x + 9}$$

```
>> fd = polyder(f) % 미분
```

fd =

6 6 1

```
>> p = polyder(f,g) % 곱의 미분
```

p =

10 28 39 32 13

```
>> [q,d] = polyder(f,g) % 나눗셈의 미분
```

q =

2 8 23 8 -7 % 문자 다항식

d =

1 4 10 12 9 % 분모 다항식

▣ 다항식의 적분

- ✓ 적분 상수를 임의로 줄 수 있다.

$$\int f(x)dx = 0.5x^4 + x^3 + 0.5x^2 + 5x$$

$$\int f(x)dx = 0.5x^4 + x^3 + 0.5x^2 + 5x + 3$$

>> fi = polyint(f) % 적분 상수=0

fi =

0.5000 1.0000 0.5000 0.5000 0

>> fi = polyint(f, 3) % 적분 상수=3

fi =

0.5000 1.0000 0.5000 0.5000 3.000

▣ 다항식의 curve fitting

- ✓ (1,2), (2,3), (3,6), (4,9) 를 지나는 다항식
- ✓ 3 차 다항식

$$p_1(x) = -0.3333x^3 + 3x^2 - 5.6667x + 5$$

- ✓ 2 차 다항식 (최소 제곱 해를 구한다.)

$$p_2(x) = 0.5x^2 - 0.1x + 1.5$$

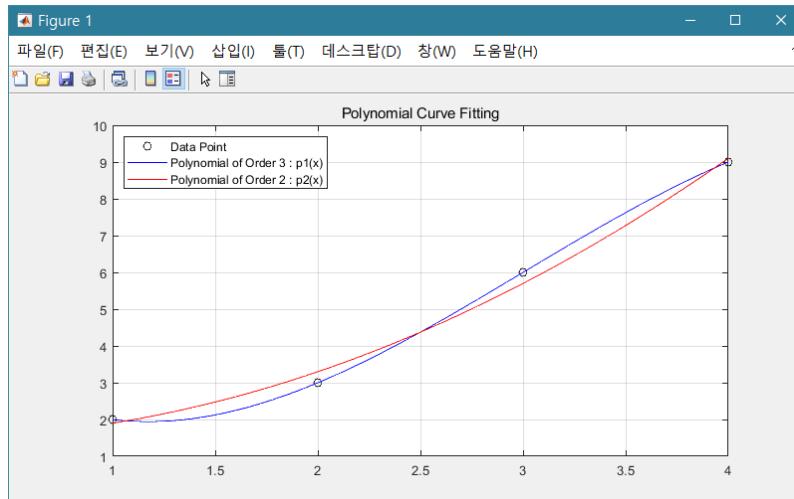
>> x = [1 2 3 4]; y = [2 3 6 9];

>> p1 = polyfit(x,y,3) % 3 차 다항식

-0.3333 3.0000 -5.6667 5.0000

>> p2 = polyfit(x,y,2) % 2 차 다항식

0.5000 -0.1000 1.5000



▣ 다항식의 부분 전개

✓ 라플라스 역 변환

$$H(s) = \frac{s^3 + 2s^2 + 3s + 4}{s^2 + 5s + 6} = s - 3 + \frac{-2}{s+2} + \frac{14}{s+3}$$

```
>> b = [1 2 3 4];
>> a = [1 5 6];
>> [r,p,k] = residue(b,a)
r = % 유수
    14.0000
   -2.0000
p = % 극점
   -3.0000
   -2.0000
k =
    1    -3
```

✓ z 역 변환

$$H(z) = \frac{1+z^{-1}+z^{-2}+z^{-3}}{1-\frac{5}{2}z^{-1}+z^{-2}} = \frac{7}{2} + z^{-1} + \frac{-5}{1-\frac{1}{2}z^{-1}} + \frac{\frac{5}{2}}{1-2z^{-1}}$$

```
>> b = [1 1 1 1];
```

```
>> a = [1 -5/2 1];  
>> [r,p,k] = residue(b,a)  
r = % 유수  
    2.5000  
    -5.0000  
p = % 극점  
    2.0000  
    0.5000  
k =  
    3.5000    1.0000
```

2.14 연습 문제

- ▶ 문제 1. $a = 1+2+\cdots+10$ 을 구하라
- ▶ 문제 2. $b = 1^2 + 2^2 + \cdots + 10^2$ 을 구하라.
- ▶ 문제 3. 다음 두 벡터의 내적을 구하라.

$$\mathbf{a} = (1, 2, 3, 4, 5), \quad \mathbf{b} = (2, 3, 4, 5, 6)$$

- ▶ 문제 4. 크기가 10×10 인 가우시안 백색 잡음 행렬을 생성하고 3 번째 행과 5 번째 열의 평균을 각각 구하라.
- ▶ 문제 5. 평균이 1이고, 분산이 5인 가우시안 백색 잡음 신호 1000개를 생성하라.
- ▶ 문제 6. +1과 -1을 랜덤하게 갖는 신호 1000개를 생성하라.
- ▶ 문제 7. 계산된 저항값이 $R = 10\text{k}\Omega$ 일 때 다음과 같이 나타내게 프로그램을 작성하라.

>> R = 10 [kV/A]

- ▶ 문제 8. 다음 선형 연립 방정식을 풀어라.

$$\begin{aligned}x_1 + 3x_2 - 2x_3 &= 1 \\2x_1 - 3x_2 + x_3 &= -1 \\2x_1 + x_2 - 3x_3 &= -5\end{aligned}$$

- ▶ 문제 9. 다음 식을 라플라스 역 변환하라.

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

- ▶ 문제 10. 다음 식을 z 역 변환하라.

$$X(z) = \frac{2 - \frac{1}{4}z^{-1}}{1 - \frac{1}{4}z^{-1} - \frac{1}{8}z^{-2}}$$

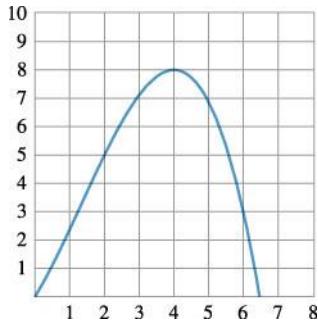
- ▶ 문제 11. $\mathbf{x} = [0, 0.01, 0.02, \dots, 10]$ 일 때 다음 식을 만족하는 벡터 \mathbf{y} 를 구하라.

$$y = \frac{\sin(x^2)}{(x+1)^2}$$

- ▶ 문제 12. Curve fitting 을 이용하여 다음 적분값을 구하라.

$$\int_0^1 \sin\left(\frac{\pi x^2}{2}\right) dx$$

▣ 문제 13. 다음과 같이 주어지는 그래프와 x 축으로 둘러싸인 면적을 구하라.



3. M-file 프로그래밍

3.1 M-file 이란 무엇인가?

✚ M-file 이란?

- ✓ MATLAB 언어로 쓰여진 파일들

✚ M-file 의 종류

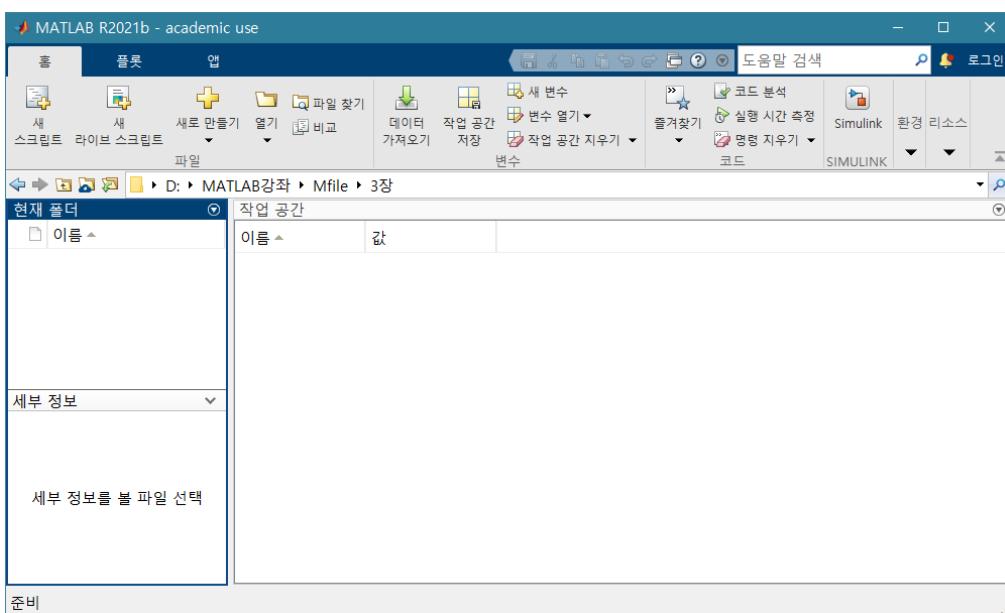
- ✓ Script m-file
- ✓ 함수(function) m-file

Script mode m-file	Function mode m-file
연속적인 MATLAB 명령어로 이루어진 m-file	입력과 출력 매개 변수가 있는 m-file
파일 이름은 어떠한 이름으로도 할 수 있다.	파일 이름은 함수 이름과 가능한 같게 한다.
Base workspace에 있는 데이터에 대해서 연산	Function workspace에 있는 데이터에 대해서 연산

✚ M-file 의 저장 폴더 구조



- ✓ 작업 폴더 : D:\MATLAB 강좌\ Mfile\ 3장



✚ Script m-file

```
% Ex3_1_1a.m
% 평균과 분산 구하기
x = randn(1,1000);
clc; clear;
mx = mean(x);
vx = var(x);
disp(['x의 평균 : ', num2str(mx)])
disp(['x의 분산 : ', num2str(vx)])
```

✓ Script m-file 의 실행

- ✗ 명령 창에 파일 이름(Ex3_1_1a)을 치거나 편집기 창에서 실행 아이콘을 클릭한다. (단축키 : F5)
- ✗ 주의 : 파일 이름에 – 기호가 포함되면 제대로 실행되지 않는다. (예 : 파일 이름이 Ex1-1.m일 때 이 프로그램을 실행하면 Ex1 – 1을 실행하고 Ex1.m 파일을 찾기 때문에 제대로 동작하지 않는다.)

```
>> Ex3_1_1a
x의 평균 : -0.032632
x의 분산 : 0.99793
```

✚ 함수(function) m-file

```
function [mx,vx] = mean_var(x)
% mean_var.m
% 평균과 분산을 구하고 나타내는 함수
mx = mean(x);
vx = var(x);
disp(['x의 평균 : ', num2str(mx)])
disp(['x의 분산 : ', num2str(vx)])
```

- ✓ 함수 m 파일은 저장하면 함수 이름으로 자동 저장된다.

✓ 함수(function) m-file 의 실행

- ✗ 함수는 독립적으로 실행할 수 없다.

```
>> mean_var
```

입력 인수가 부족합니다.

오류 발생: mean_var (4 번 라인)

```
mx = mean(x);
```

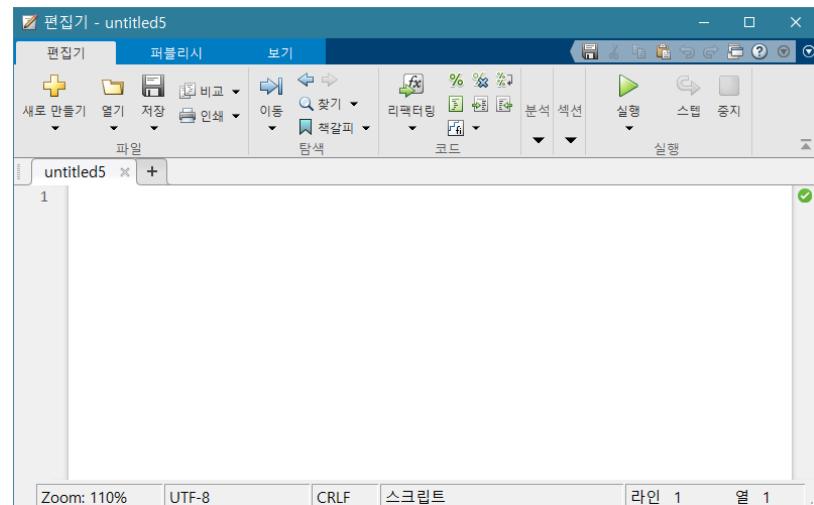
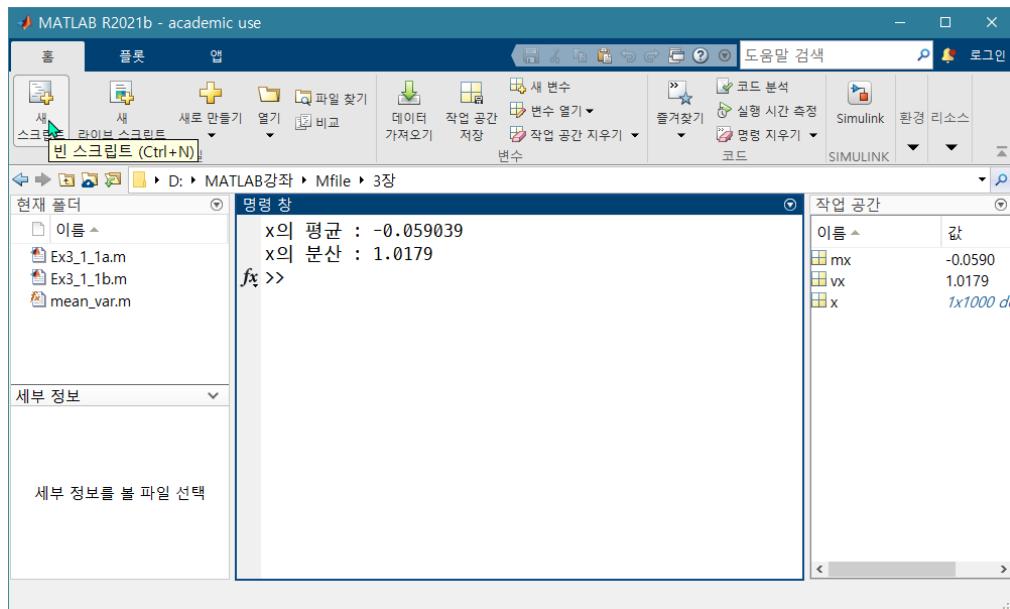
* 함수는 별도의 m 파일에서 불러 주어야 한다.

```
% Ex3_1-1b.m  
% 평균과 분산 구하기  
clc; clear;  
x = randn(1,1000);  
[mx,vx] = mean_var(x);
```

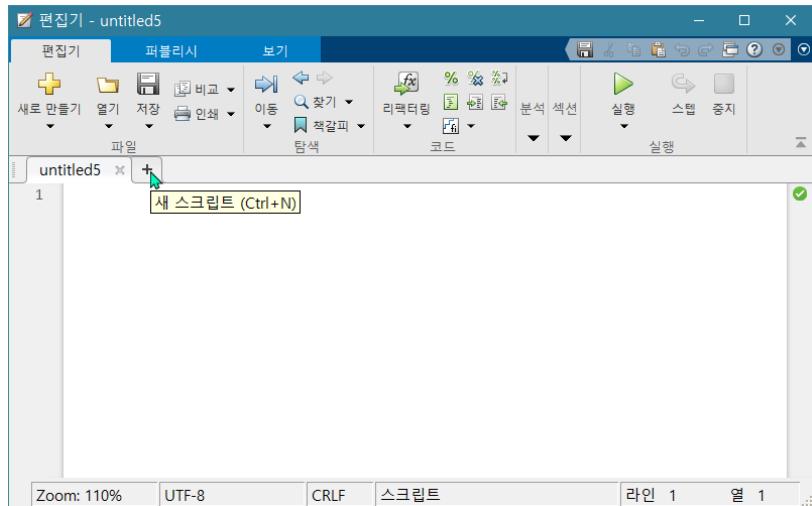
3.2 새 m-file 만들기

내장된 m-file 편집기를 이용하는 방법

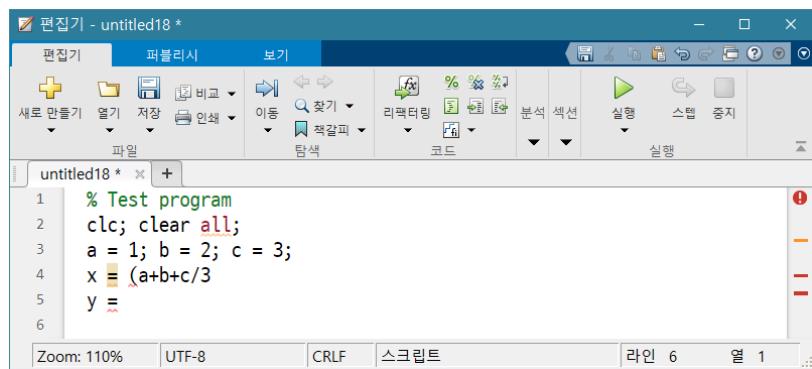
- ✓ 새 스크립트 아이콘을 클릭하거나 새로 만들기 아이콘을 클릭한 후 스크립트를 선택한다. (단축 키 Ctrl-N)



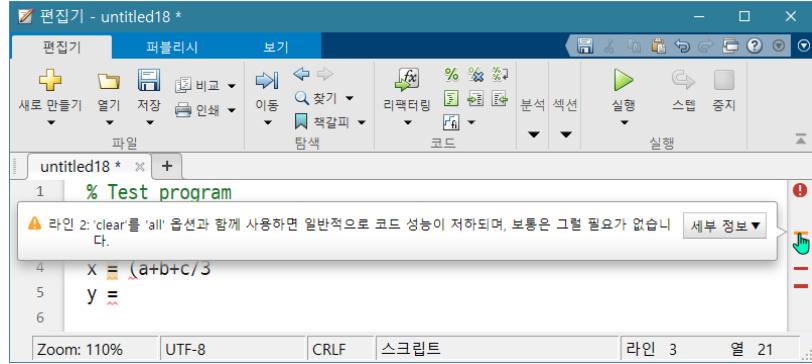
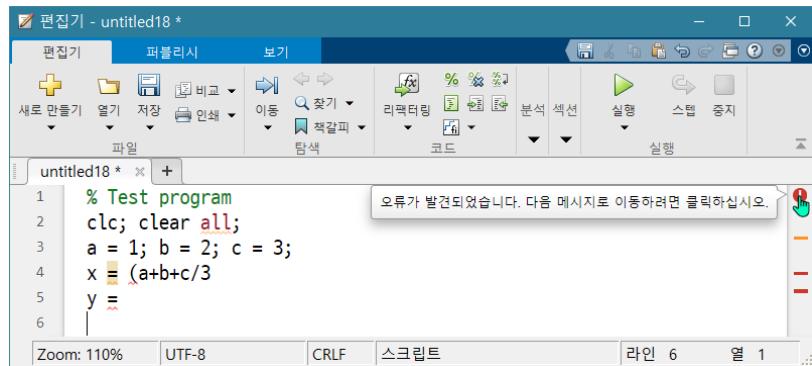
- ✓ 편집기에서 새 m 파일을 추가하려면 + 탭을 클릭한다.

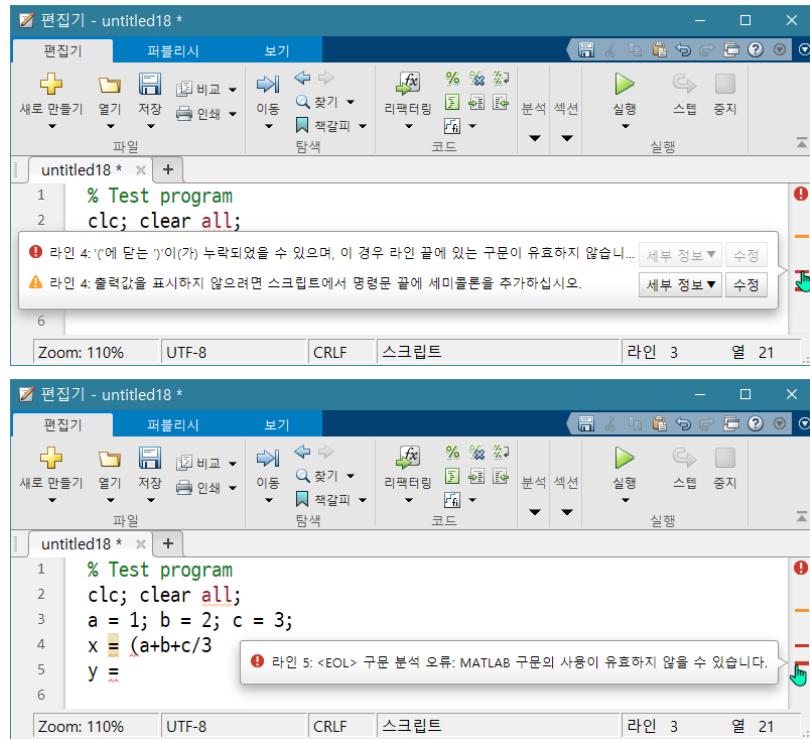


▶ 프로그램 작성 중 경고, 에러 표시 기능



- ✓ 오른쪽에 에러 사인이 나타난다.





The image shows two side-by-side windows of the MATLAB Editor. Both windows have the title '편집기 - untitled18 *'. The top window contains the following code:

```
% Test program
clc; clear all;
```

It displays two error messages in a tooltip:

- 라인 4: "("에 닫는 ")"이(가) 누락되었을 수 있으며, 이 경우 라인 끝에 있는 구문이 유효하지 않습니다... (Line 4: A closing ')' is missing or incorrect. The text after the line end is invalid.)
- 라인 4: 출력값을 표시하지 않으려면 스크립트에서 명령문 끝에 세미콜론을 추가하십시오. (Line 4: If you do not want to display the output, add a semicolon at the end of the command line in the script.)

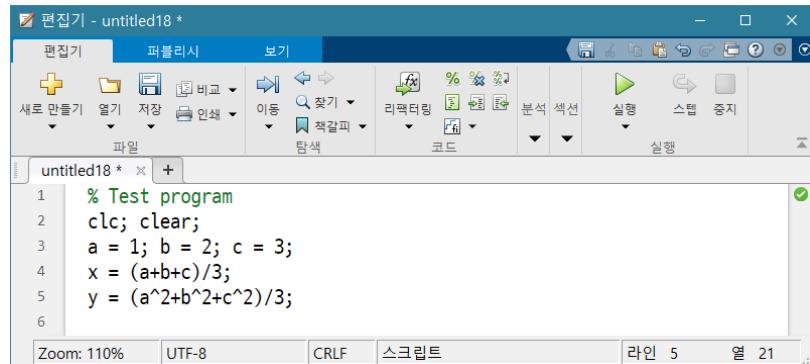
The bottom window contains the same code but with additional lines 3 and 4:

```
% Test program
clc; clear all;
a = 1; b = 2; c = 3;
x = (a+b+c)/3
y =
```

It also displays an error message in a tooltip:

라인 5: <EOL> 구문 분석 오류: MATLAB 구문의 사용이 유효하지 않을 수 있습니다. (Line 5: MATLAB syntax error: The use of the syntax is invalid.)

✓ 에러가 없는 프로그램



The image shows a single window of the MATLAB Editor with the title '편집기 - untitled18 *'. It contains the following corrected code:

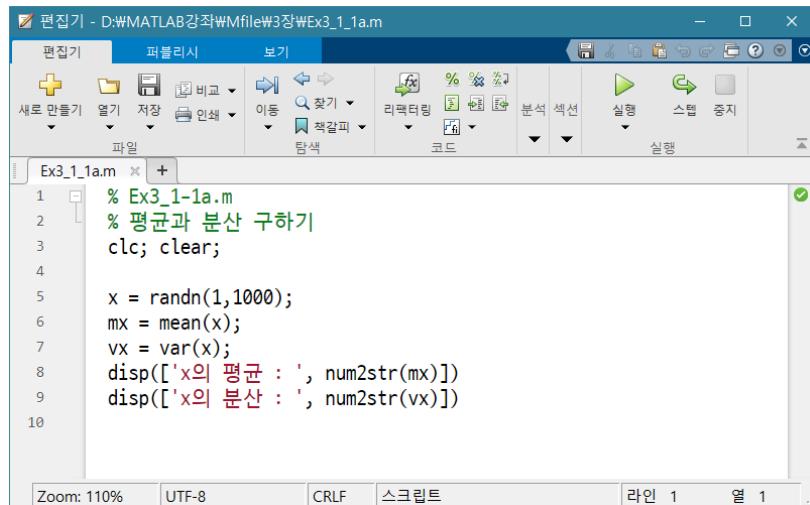
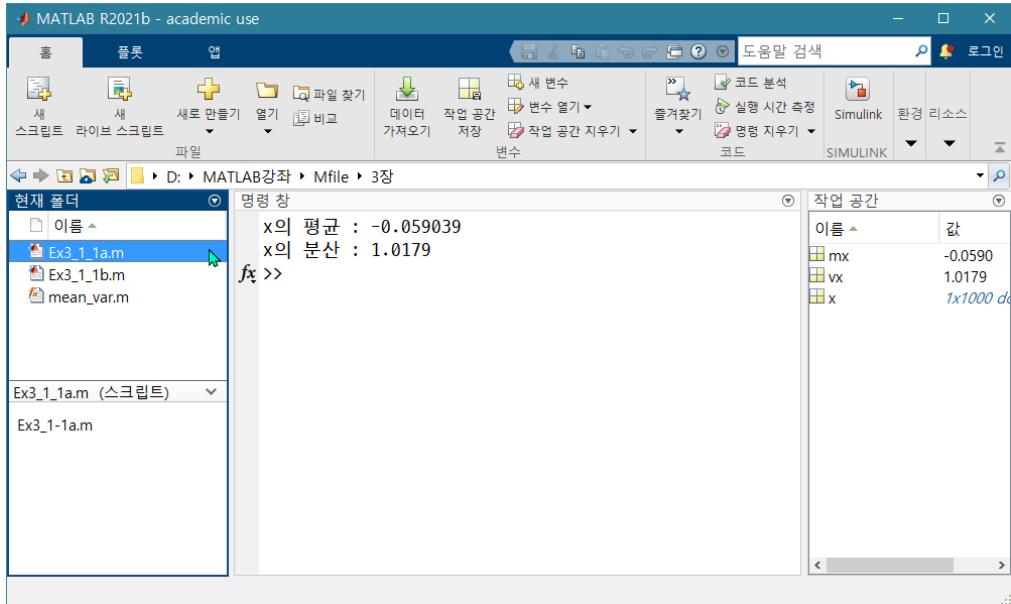
```
% Test program
clc; clear;
a = 1; b = 2; c = 3;
x = (a+b+c)/3;
y = (a^2+b^2+c^2)/3;
```

A green checkmark icon is located in the status bar at the bottom right of the editor window.

3.3 M-file 불러오기

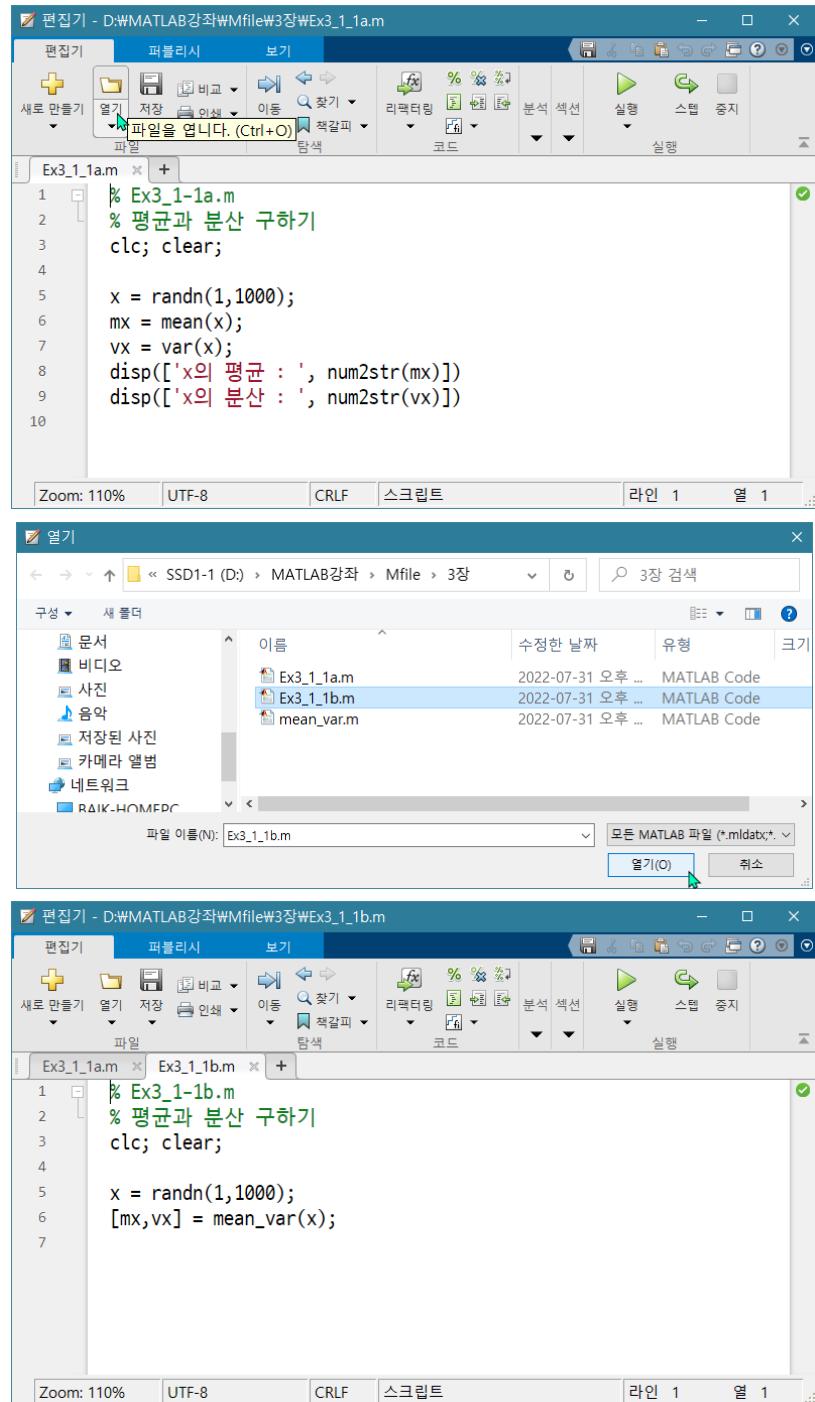
Desktop에서 불러오기

- ✓ 현재 폴더에서 파일을 선택한 후 더블 클릭한다.



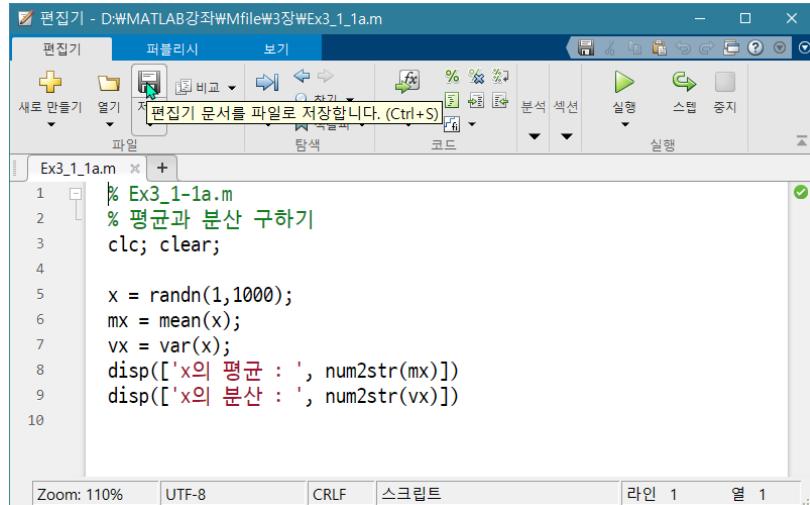
M-file 편집기에서 불러오기

- ✓ 열기 아이콘을 클릭한 후 파일을 선택한다.

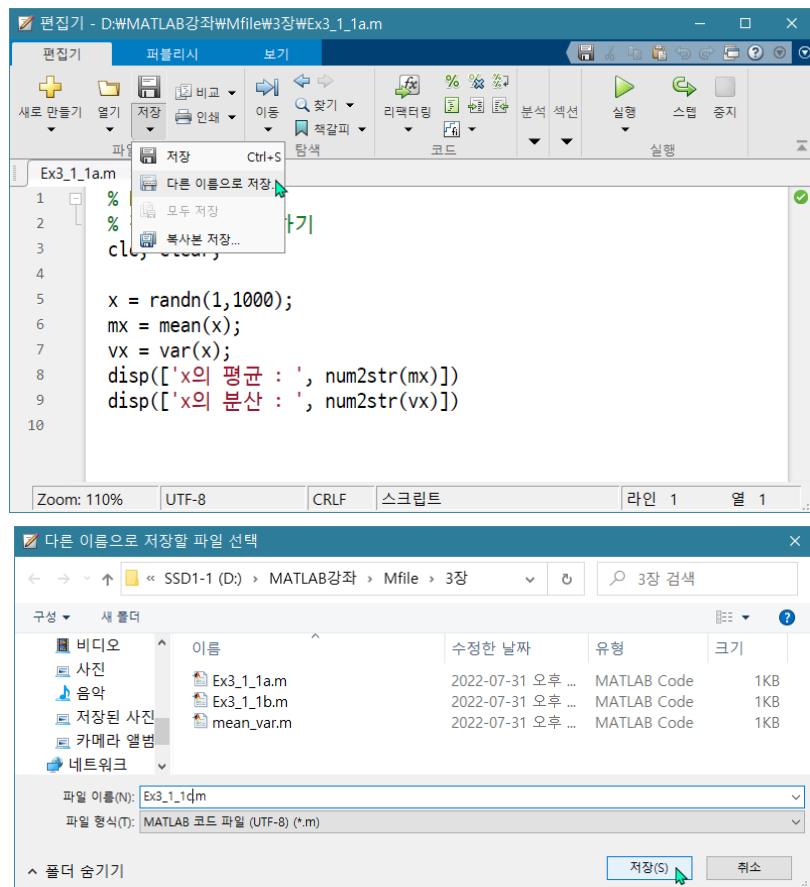


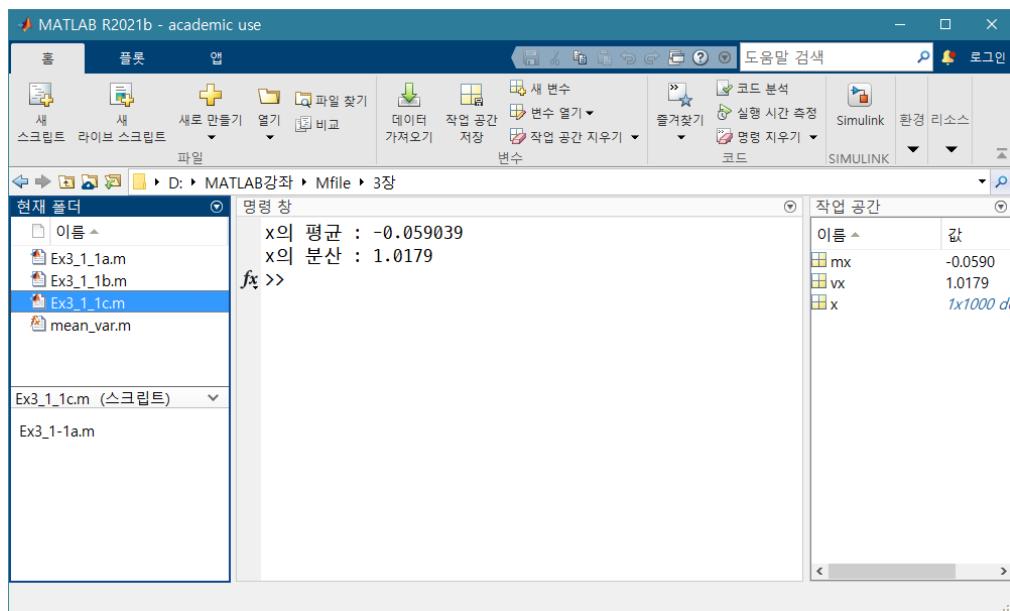
3.4 M-file 저장하기

- ✚ 저장 아이콘을 클릭한 후 파일 이름을 입력한다.
- ✓ 파일 이름에 -를 포함하면 안된다. (- 대신 _를 사용한다.)



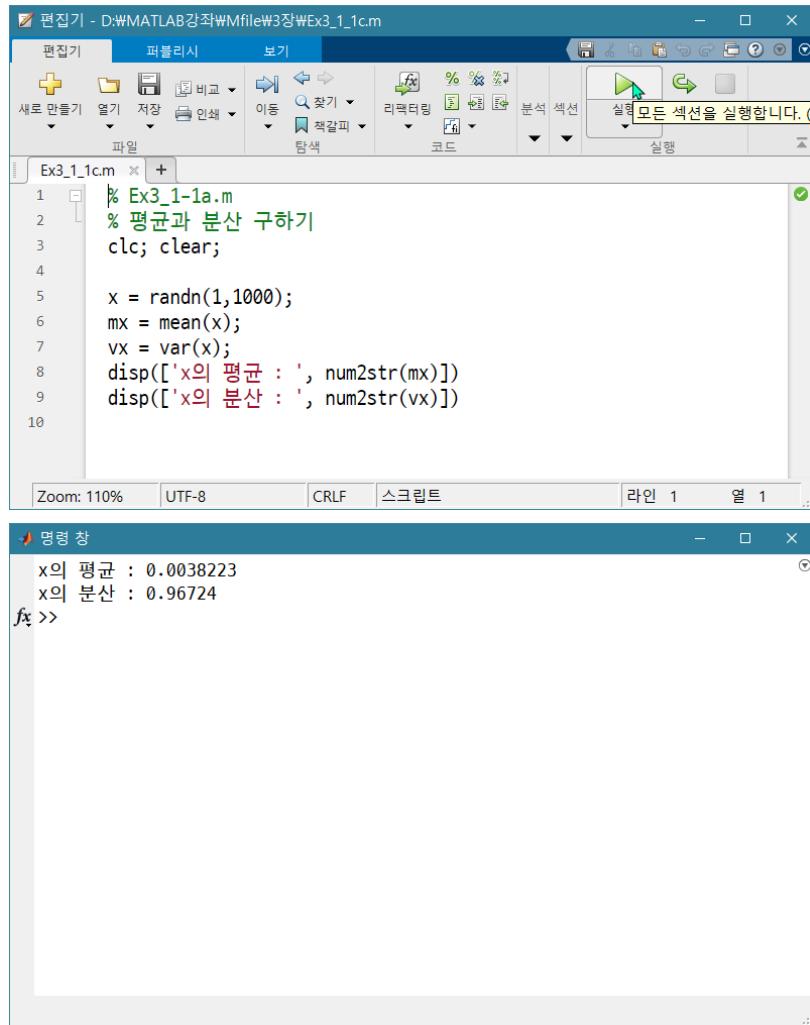
- ✚ 기존의 파일을 변경하여 다른 이름으로 저장할 수 있다.





3.5 M-file 실행하기

- M-file 편집기에서 실행 아이콘을 클릭한다. (단축 키 F5)



- 명령 창에서 실행할 파일의 이름을 입력한 후 Enter 키를 누른다.



3.6. 제어 문장

✚ if – elseif 문

- ✓ 조건이 참일 때 실행한다.

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

✚ if - elseif 문 예제

```
x = 10;
if x == 0
    xi = 0;
elseif x > 0
    xi = 1;
else
    xi = -1;
end
xi
```

✚ switch – case 문

- ✓ 여러 그룹 중의 하나를 실행한다.
- ✓ 문자열을 취급할 경우 switch-case 문이 if-else 문보다 더 효율적이다.

```
switch switch expression
    case case expression
    case case expression
    ...
    otherwise
        case expression
end
```

 switch –case 문 예제

```
item = 'green';
switch item
case 'green'
    type = 'color';
case {'apple', 'banana'}
    type = 'fruit';
otherwise
    type = 'etc';
end
type
```

 for loop 문

- ✓ 주어진 횟수만큼 반복한다.
- ✓ MATLAB에서는 가능하면 for loop 문을 사용하지 않는 것이 좋다.

```
for index = values
    statements
end
```

 for loop 문 예제

```
N = 100;
sum = 0;
for i = 1:N
    sum = sum+i;
end
sum
```

```
for x = 1:-0.2:0
    disp(x)
end
```

 while loop 문

- ✓ while 문장의 조건을 만족하는 한 계속 루프 안의 문장을 실행한다.

```
while expression  
    statements  
end
```

 while loop 문 예제

```
M = 100;  
sum = 0;  
j = 1;  
while j <= M  
    sum = sum+j;  
    j = j+1;  
end  
sum
```

 break 문

- ✓ for 나 while 루프 안에서 빠져 나올 때 사용한다.

```
M = 100;  
sum = 0;  
j = 1;  
while true  
    sum = sum+j;  
    if j>=M  
        break  
    else  
        j = j+1;  
    end  
end  
sum
```

 continue 문

- ✓ for 나 while 루프의 다음 반복으로 이동한다. (루프 문장의 마지막으로 이동한다. 다만 마지막 문장은 실행하지 않는다.)
-

```
M = 1000;  
x = rand(1,M);  
count = 0;  
for i = 1:M  
    if x(i)<0.5  
        continue  
    end  
    count = count+1;  
end  
count
```

3.7 M-file 의 실행 속도 비교

- ✚ For 문보다 배열 연산자를 이용하면 더 효율적이다.
- ✚ For 문을 이용하여 작성한 함수 예
 - ✓ ptest1.m

```
function D = ptest1(M,L,W,H)
[rows, cols] = size(M);
for I = 1:rows
  for J = 1:cols
    D(I,J) = M(I,J) / (L(I,J)*W(I,J)*H(I,J));
  end
end
```

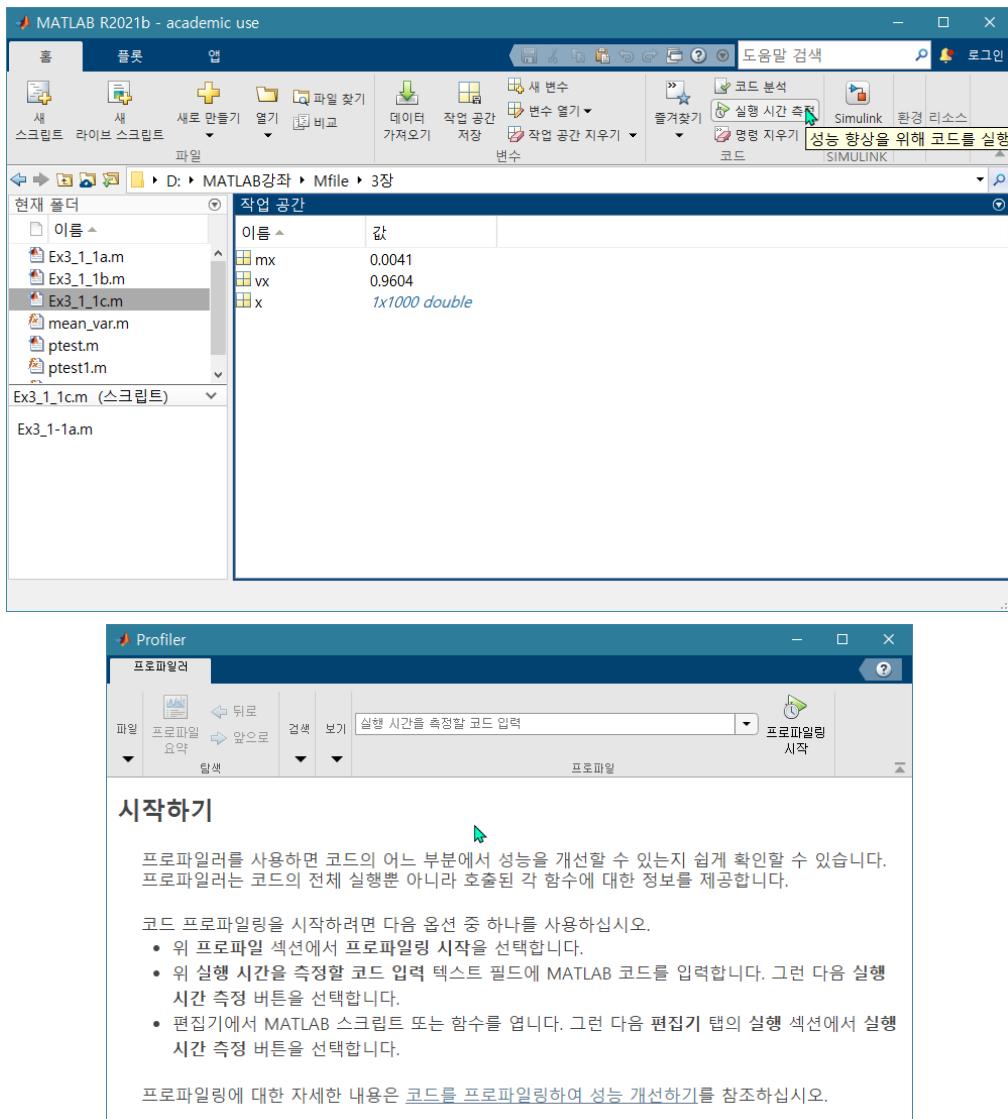
- ✚ 배열 연산자를 이용하여 작성한 함수 예
 - ✓ ptest2.m

```
function D = ptest2(M,L,W,H)
D = M ./ (L.*W.*H);
```

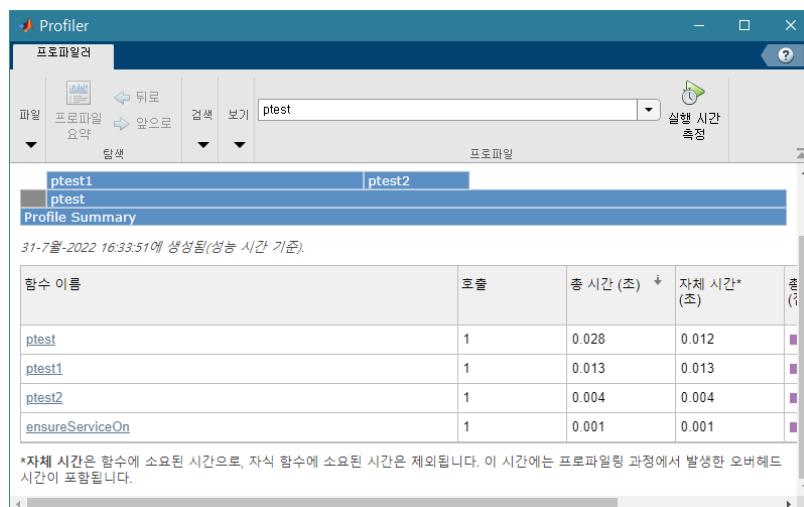
- ✚ 두 함수의 실행 속도를 비교하기 위한 메인 프로그램
 - ✓ 두 함수를 이용하여 같은 작업을 수행한다.
 - ✓ ptest.m

```
M = rand(5,10000); L = rand(5,10000);
W = rand(5,10000); H = rand(5,10000);
D1 = ptest1(M,L,W,H);
D2 = ptest2(M,L,W,H);
```

- ✚ 실행 시간의 비교
 - ✓ 실행 시간 측정 아이콘을 클릭한다.



- ✓ 파일 이름(ptest)을 입력한 후 프로파일링 시작(P)를 클릭한다.



- ✓ 함수 ptest1 을 실행하는데 걸린 시간이 함수 ptest2 를 실행하는데 걸린 시간보다 훨씬 적다.

- ✓ for 문장보다는 원소 연산자 (도트 연산자)를 사용하는 것이 프로그램도 간단하고 실행 시간도 짧다.

3.8 MATLAB 함수(function)

3.8.1 함수의 종류

- MATLAB 이 제공하는 함수
 - ✓ Core 함수 : sin, cos, abs, exp, ...
 - ✓ MATLAB 이 만든 함수 : mean, var, plot, ...
- 사용자가 만든 함수 : par_R, ptest1, ptest2, ...

3.8.2 지역 함수(local 함수)

- 같은 m-file 내에 1 개 이상의 함수를 포함할 수 있다.
- 주 함수(main 함수)
 - ✓ 파일 처음에 나오는 함수
 - ✓ 다른 파일에서 인식할 수 있고 명령 창에서 호출할 수 있다.
 - ✓ 주 함수의 예 : mean_var.m
- 지역 함수(local 함수)
 - ✓ 파일에 부가적으로 붙은 함수
 - ✓ 명령 창에서 호출할 수 없다.
 - ✓ 함수가 있는 파일에서만 호출할 수 있다.
- 지역 함수를 포함한 함수의 예

```

function [mx,vx] = mystats(x) % 주 함수
n = length(x);
mx = mymean(x,n);
vx = myvar(x,n);
end

function mxx = mymean(xx,n) % 지역 함수 1
mxx = sum(xx)/n;
end
function vxx = myvar(xx,n) % 지역 함수 2
vxx = sum(xx.^2)/n - (sum(xx)/n)^2;
end

```

 명령 창에서 실행

```
>> x = 1:10;
>> [mx,vx] = mystats(x)
mx =
5.5000
vx =
8.2500
>> mx = mymean(x,10)
'mymean'은(는) 인식할 수 없는 함수 또는 변수입니다
```

3.8.3 중첩(nested) 함수

 함수 안에 함수를 둘 수 있다.

 주의 사항

- ✓ 중첩 함수를 사용할 경우 반드시 함수 끝에 end 를 넣어야 한다.
- ✓ 제어 문장 내에는 중첩 함수를 사용할 수 없다.
- ✓ 중첩 함수 내의 모든 변수는 명확하게 정의되어야 한다.
- ✓ 부모 함수와 중첩 함수는 변수를 공유할 수 있다. (중첩 함수에 변수를 전달하지 않고 변수를 수정할 수 있다.)

 중첩 함수를 포함한 함수의 예

```
function [mx,vx] = mystats2(x) % 부모 함수
n = length(x);
meanvar_x
    function meanvar_x % 중첩 함수
        mx = sum(x)/n;
        vx = sum(x.^2)/n - (sum(x)/n)^2;
    end
end
```

 명령 창에서 실행

```
>> x = 1:10;
```

```
>> [mx,vx] = mystats2(x)  
mx =  
    5.5000  
vx =  
    8.2500  
>> meanvar_x  
'meanvar_x'은(는) 인식할 수 없는 함수 또는 변수입니다
```

3.9 함수의 입출력 인자

- ✚ 함수의 입출력 인자 수를 가변으로 만들 수 있다.
- ✚ 함수 입력 인자
 - ✓ nargin : 함수의 입력에 사용되는 인자의 수
 - ✓ nargout : 함수의 출력에 사용되는 인자의 수
- ✚ 가변 입력 인자 함수
 - ✓ 3 개까지 병렬 저항 값을 구하는 함수

```
function R = par_R(R1,R2,R3)
% 병렬 저항 값
if nargin == 1
    R = R1;
elseif nargin == 2
    R = 1/(1/R1+1/R2);
elseif nargin == 3
    R = 1/(1/R1+1/R2+1/R3);
end
```

- ✓ 실행 예

```
>> R1 = par_R(20)
R1 =
    20
>> R2 = par_R(10,10)
R2 =
    5
>> R3 = par_R(30,30,30)
R3 =
    10
>> R4 = par_R(20,20,20,20)
다음 사용 중 오류가 발생함: par_R
입력 인수가 너무 많습니다.
```

- ✓ 입력 인자가 정해진 개수를 초과하면 에러가 발생한다.

✚ 가변 출력 인자 함수

- ✓ 삼각 함수 값 구하는 함수

```
function [y1,y2,y3] = trim_val(x)
% 삼각 함수 값 구하는 함수
if nargout == 1
    y1 = sin(x);
elseif nargout == 2
    y1 = sin(x);
    y2 = cos(x);
else
    y1 = sin(x);
    y2 = cos(x);
    y3 = tan(x);
end
```

- ✓ 실행 예

```
>> trim_val(pi/2)
ans =
    1
>> y1 = trim_val(pi/2)
y1 =
    1
>> [y1,y2] = trim_val(pi/2)
y1 =
    1
y2 =
    6.1232e-017          % y2 = 0
>> [y1,y2,y3] = trim_val(pi/2)
y1 =
    1
y2 =
    6.1232e-17          % y2 = 0
y3 =
```

```
1.6331e+16          % 무한대
>> [y1,y2,y3,y4] = trim_val(pi/2)
다음 사용 중 오류가 발생함: trim_val
출력 인수가 너무 많습니다.
```

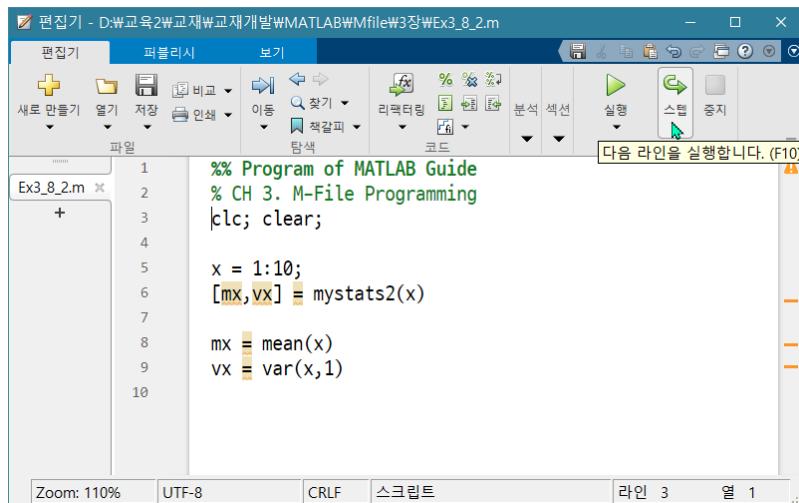
- ✓ 출력 인자가 정해진 개수를 초과하면 에러가 발생한다.

3.10 프로그램 디버깅(debugging)

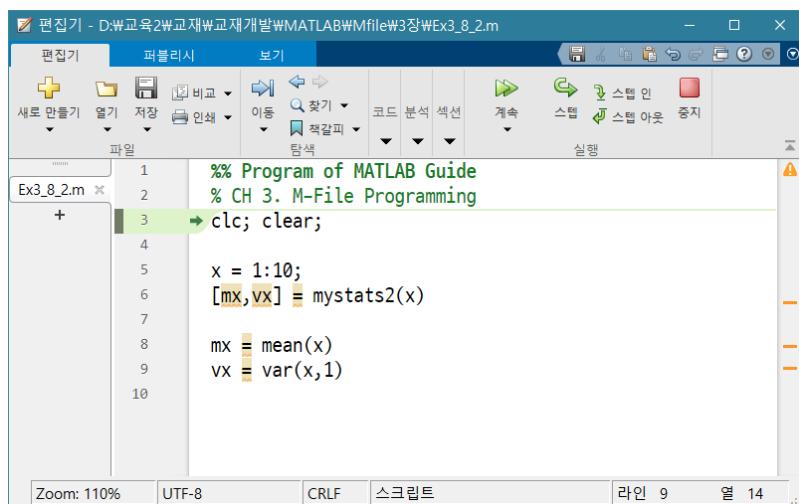
- ❖ 디버깅 - 오류(bug)를 찾아 제거하는 과정
 - ✓ 프로그램에 논리적인 오류가 있으면 디버깅을 하여 논리적 오류를 찾아야 한다.
 - ✓ 디버깅을 하려면 프로그램을 한 줄씩 실행시키면서 변수 값을 확인하면 된다.

❖ 처음부터 한 줄씩 실행하는 방법

- ✓ 스텝 아이콘(단축키 F10)을 클릭한다.



✖ 화살표가 첫번째 실행 문장을 가리킨다.



❖ 변수값을 확인하는 방법

- ✓ 해당 줄을 F10 을 눌러 실행시킨 후 마우스를 해당 변수 위에 위치시키면 팝업 창으로 해당 변수의 값을 알 수 있다.

```
%> 편집기 - D:\교육2\교재\교재개발\ MATLAB\Mfile\3장\Ex3_8_2.m
%> 파일 1: Ex3_8_2.m
1 %% Program of MATLAB Guide
2 % CH 3. M-File Programming
3 clc; clear;
4
5 x = 1:10;
6 [mx,vx] = mystats2(x)
7
8 mx = mean(x)
9 vx = var(x,1)
10

x: 1x10 double =
1 ~ 8번 열
1 2 3 4 5 6 7 8
9 ~ 10번 열
9 10

%> Zoom: 110% UTF-8 CRLF 스크립트 라인 6 열 1
```

함수를 만났을 때 디버깅

- ✓ 호출 함수 내부로 들어가지 않고 호출 함수 실행 결과를 보려면 스텝 아이콘(단축키 F10)을 클릭한다.

```
%> 편집기 - D:\교육2\교재\교재개발\ MATLAB\Mfile\3장\Ex3_8_2.m
%> 파일 1: Ex3_8_2.m
1 %% Program of MATLAB Guide
2 % CH 3. M-File Programming
3 clc; clear;
4
5 x = 1:10;
6 [mx,vx] = mystats2(x)
7
8 mx = mean(x)
9 vx = var(x,1)
10

%> Zoom: 110% UTF-8 CRLF 스크립트 라인 9 열 14
```

✗ mystats2 함수 통과

```
%> 편집기 - D:\교육2\교재\교재개발\ MATLAB\Mfile\3장\Ex3_8_2.m
%> 파일 1: Ex3_8_2.m
1 %% Program of MATLAB Guide
2 % CH 3. M-File Programming
3 clc; clear;
4
5 x = 1:10;
6 [mx,vx] = mystats2(x)
7
8 mx = mean(x)
9 vx = var(x,1)
10

%> Zoom: 110% UTF-8 CRLF 스크립트 라인 8 열 1
```

- ✓ 호출 함수 안으로 이동하려면 스텝 인 아이콘(단축키 F11)을 클릭한다.

```
%> 편집기 - D:\교육2\교재\교재개발\ MATLAB\Mfile\3장\Ex3_8_2.m
1 %% Program of MATLAB Guide
2 % CH 3. M-File Programming
3 clc; clear;
4
5 x = 1:10;
6 [mx,vx] => mystats2(x)
7
8 mx = mean(x)
9 vx = var(x,1)
10
```

Zoom: 110% UTF-8 CRLF 스크립트 라인 6 열 1

✗ mystats2 함수 내부로 이동

```
%> 편집기 - D:\교육2\교재\교재개발\ MATLAB\Mfile\3장\mystats2.m
1 편집기 파일 비교 이동 찾기 코드 분석 색션 계속 스텝
2 새 만들기 열기 저장 인쇄 책갈피 탐색 실행
3 파일 Ex3_8_2.m mystats2.m
4
5 function [mx,vx] = mystats2(x) % 부모 함수
6 n = length(x);
7 meanvar_x
8
9 function meanvar_x % 중첩 함수
10 mx = sum(x)/n;
11 vx = sum(x.^2)/n - (sum(x)/n)^2;
12 end
13 end
```

Zoom: 110% UTF-8 CRLF mystats2 라인 2 열 5

- ✓ 호출 함수의 내부에서 함수 밖으로 나오려면 스텝 아웃 아이콘(단축키 Shift+F11)을 클릭한다.

```
%> 편집기 - D:\교육2\교재\교재개발\ MATLAB\Mfile\3장\mystats2.m
1 편집기 파일 비교 이동 찾기 코드 분석 색션 계속 스텝
2 새 만들기 열기 저장 인쇄 책갈피 탐색 실행
3 파일 Ex3_8_2.m mystats2.m
4
5 function [mx,vx] = mystats2(x) % 부모 함수
6 n = length(x);
7 meanvar_x
8
9 function meanvar_x % 중첩 함수
10 mx = sum(x)/n;
11 vx = sum(x.^2)/n - (sum(x)/n)^2;
12 end
13 end
```

Zoom: 110% UTF-8 CRLF mystats2 라인 3 열 5

✗ 메인 프로그램으로 이동

The screenshot shows the MATLAB Editor window with two files open: 'Ex3_8_2.m' and 'mystats2.m'. The code in 'Ex3_8_2.m' is as follows:

```
% Program of MATLAB Guide
% CH 3. M-File Programming
clc; clear;
x = 1:10;
[mx,vx] = mystats2(x);
mx = mean(x)
vx = var(x,1)
```

The line `[mx,vx] = mystats2(x);` is highlighted in green, indicating it is currently executing. The status bar at the bottom right shows '라인 6 열 1', which corresponds to the current line and column of the highlighted code.

➊ 디버깅 중지

- ✓ 중지 아이콘(Shift+F5)을 클릭한다.

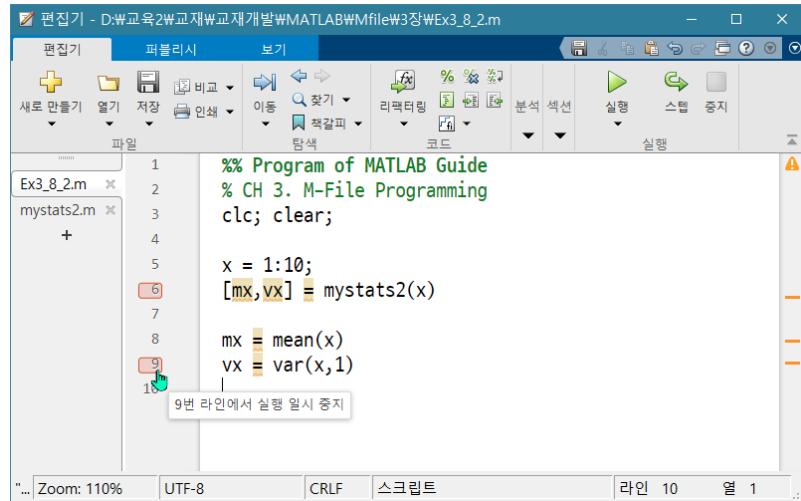
This screenshot is similar to the previous one, but the 'Stop' button in the toolbar has been clicked. The status bar now displays '실행을 중지합니다. (Shift+F5)', confirming that the program execution has been stopped.

This screenshot shows the MATLAB Editor with the 'Break' button in the toolbar highlighted. This allows for stopping the execution of the program at the current line.

➋ 프로그램 중간까지 바로 실행하는 방법

- ✓ 라인 번호를 마우스로 클릭하여 중단 지점을 만든다.
- ✗ 중단 지점은 여러 개 설정할 수 있다.

- ✖ 중단 지점을 해제하려면 마우스로 다시 클릭한다.
- ✖ 주석 문장에는 중단 지점을 설정할 수 없다.

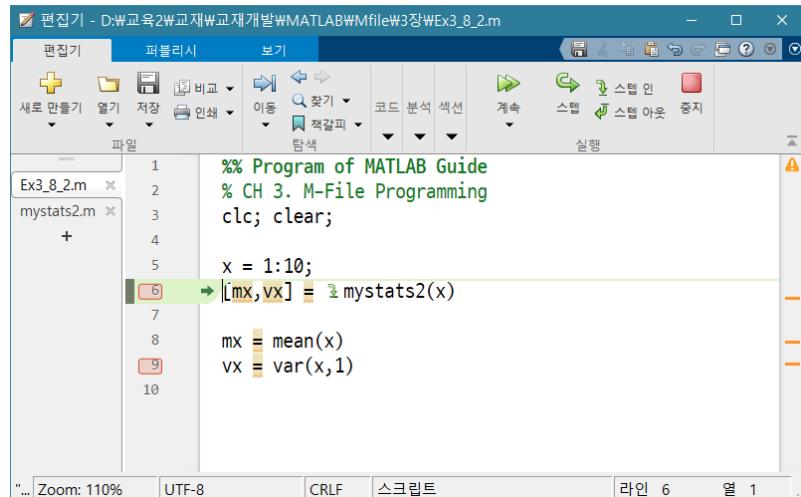


The screenshot shows the MATLAB Editor window with two files open: 'Ex3_8_2.m' and 'mystats2.m'. The code in 'Ex3_8_2.m' is:

```
%> Program of MATLAB Guide
% CH 3. M-File Programming
clc; clear;
x = 1:10;
[mx,vx] = mystats2(x);
mx = mean(x)
vx = var(x,1)
```

A red box highlights line 9, which contains the assignment statement 'mx = mean(x)'. A green arrow points to this line, indicating it is the current line of execution. A tooltip '9번 라인에서 실행 일시 중지' (Breakpoint at line 9) is displayed over the line. The status bar at the bottom right shows '라인 10 열 1'.

- ✓ 실행 아이콘(단축키 F5)을 클릭하면 중단 지점까지 바로 실행된다.



The screenshot shows the MATLAB Editor window with the same two files open. The code in 'Ex3_8_2.m' is identical to the previous screenshot. The line 6 statement '[mx,vx] = mystats2(x);' is highlighted with a green arrow, indicating it is the current line of execution. The status bar at the bottom right shows '라인 6 열 1'.

- ✓ 다음 중단 지점까지 실행하려면 계속 아이콘(F5)를 클릭한다. (실행과 단축키가 같다.)

3.11 M-file 에 유용한 함수

✚ Input 함수

```
R = input(prompt)
```

- ✓ prompt 문장을 출력하고 값(숫자)이 입력되기를 기다린다.

```
R = input(prompt, 's')
```

- ✓ prompt 문장을 출력하고 값(문자)이 입력되기를 기다린다.

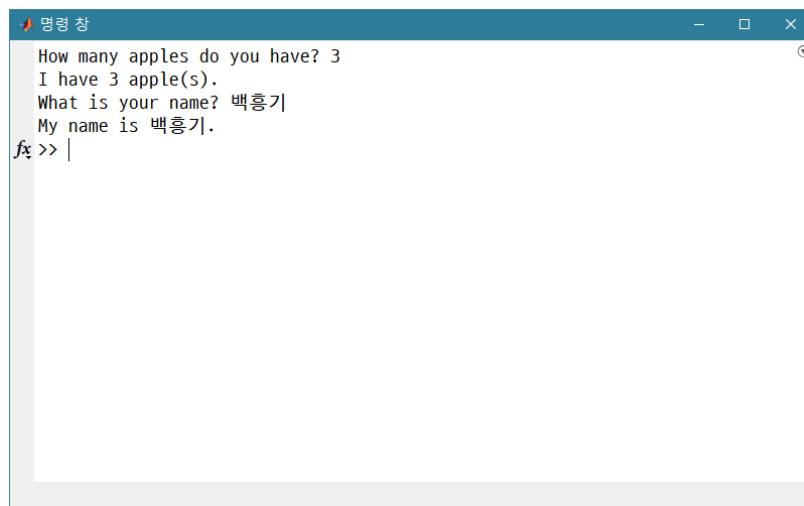
✚ Disp 함수

```
disp(X)
```

- ✓ 문자열 X 를 출력한다.

✚ Input 함수, Disp 함수의 예제

```
R = input('How many apples do you have? ');
str1 = ['I have ',num2str(R),' apple(s). '];
disp(str1)
name = input('What is your name? ','s');
str2 = ['My name is ',name,'.'];
disp(str2)
```



✚ Menu 함수

```
choice = menu>Title,menu1,menu2,...)
```

- ✓ Title 이 적혀 있고 menu1, menu2 등이 표시되는 버튼이 있는 창을 생성하고 버튼이 눌려지기를 기다린다.

■ Menu 함수의 예제

```
K = menu('Choose a color','Red','Blue','Green');
switch(K)
    case 1
        str1 = 'red';
    case 2
        str1 = 'blue';
    case 3
        str1 = 'green';
end
str2 = ['I like ',str1,' color.'];
disp(str2)
```



■ MsgBox 함수

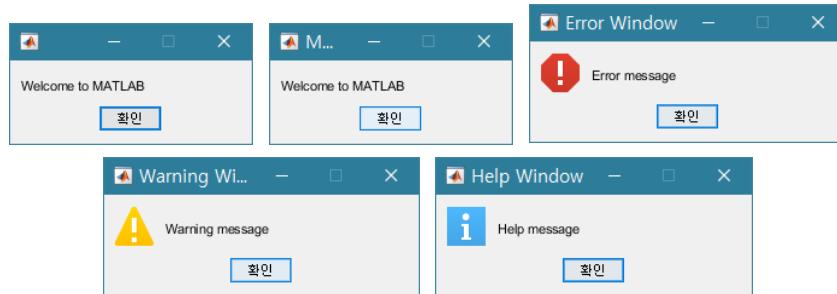
```
msgbox(message)
msgbox(message,title)
msgbox(message,title,icon)
```

- ✓ 다양한 메시지 창을 만들고 입력을 기다린다.

■ MsgBox 함수의 예제

```
msgbox('Welcome to MATLAB');
```

```
msgbox('Welcome to MATLAB','MATLAB Window');
msgbox('Error message','Error Window','error');
msgbox('Help message','Help Window','help');
msgbox('Warning message','Warning Window','warn');
```



- ✓ 모두 확인 버튼을 클릭하면 창이 없어진다.

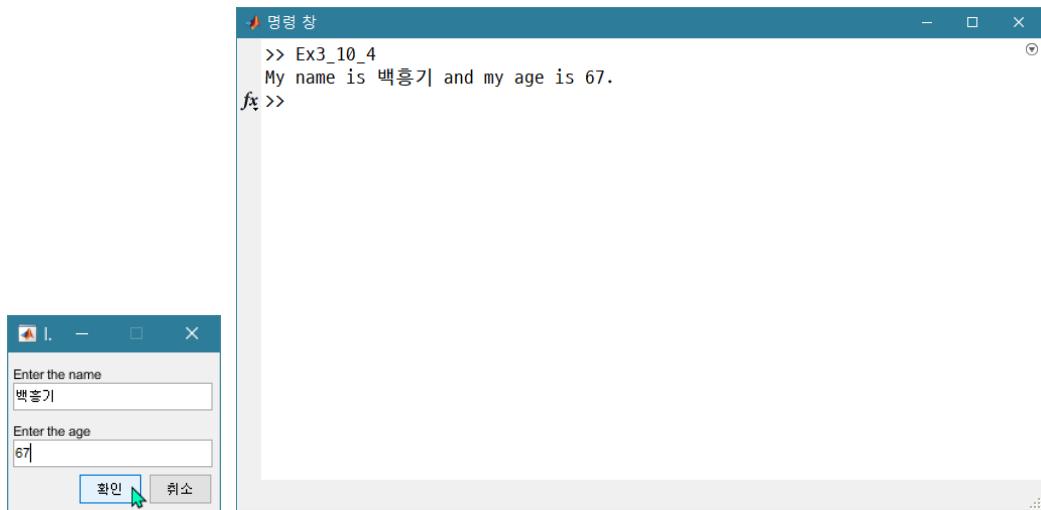
Inputdlg 함수

```
answer = inputdlg(prompt)
answer = inputdlg(prompt,title)
```

- ✓ prompt 문자열을 표시하는 창을 만들고, 값이 입력되기를 기다린다.
- ✓ prompt 문자열을 표시하고 제목이 title 인 창을 만들고, 값이 입력되기를 기다린다.

Inputdlg 함수의 예제

```
str1 = 'Enter the name';
str2 = 'Enter the age';
prompt = {str1,str2};
R = inputdlg(prompt,'Input window');
str3 = ['My name is ',R{1}, ' and my age is ',R{2},'.'];
disp(str3)
```



■ Questdlg 함수

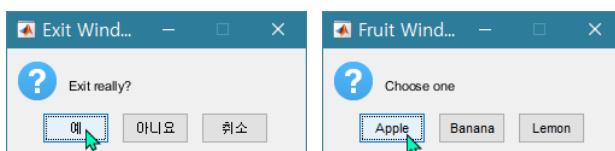
```
btn = questdlg(prompt,title)
btn = questdlg(prompt,title,btn1,btn2,...,default)
```

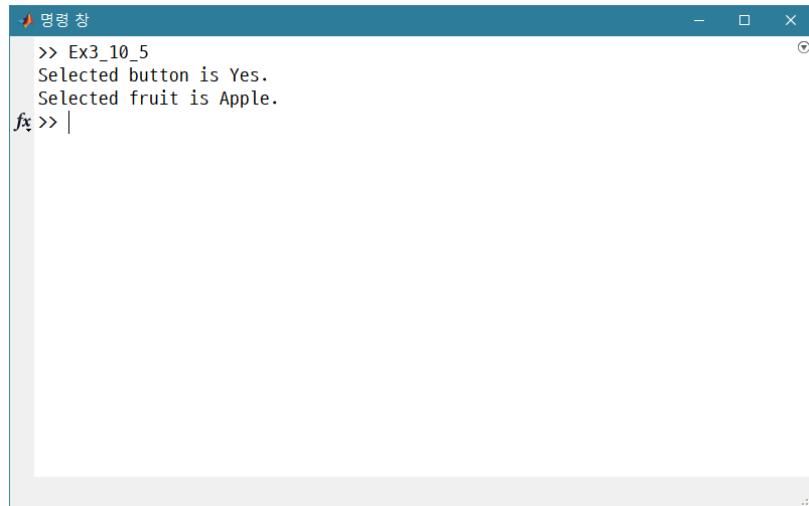
■ Questdlg 함수의 예제

```
button = questdlg('Exit really?','Exit Window');
str = ['Selected button is ',button,'.'];
disp(str)

btn1 = 'Apple';
btn2 = 'Banana';
btn3 = 'Lemon';

button = questdlg('Choose one','Fruit Window',
btn1,btn2,btn3,btn1);
str2 = ['Selected fruit is ',button,'.'];
disp(str2)
```



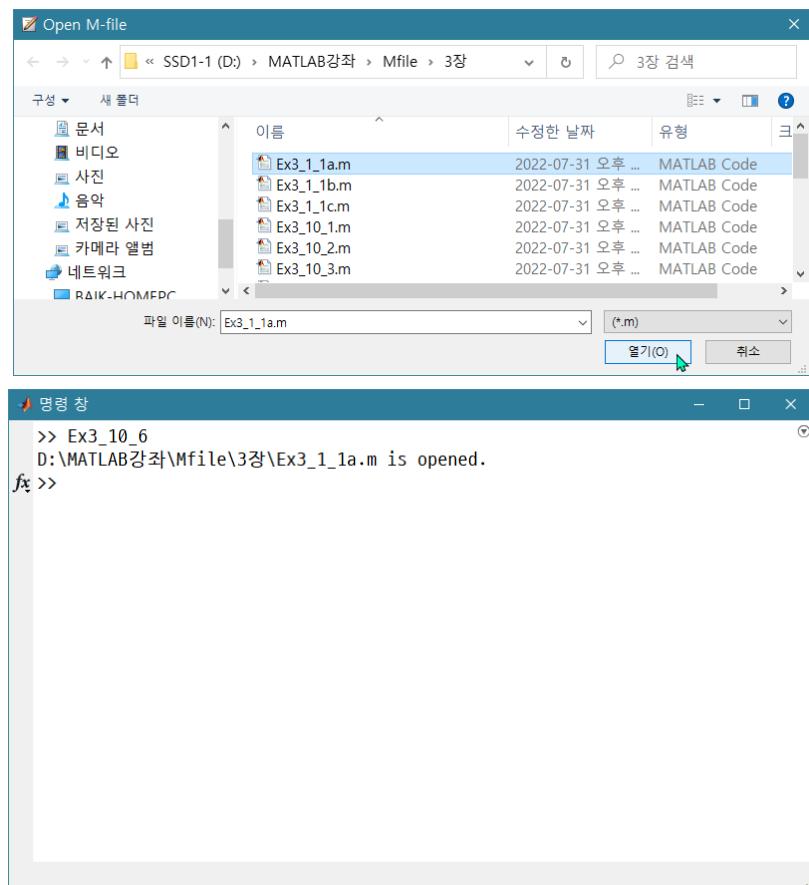


■ Uigetdlg 함수

```
[filename, pathname] = uigetfile(filterspec, title)
```

■ Uigetdlg 함수의 예제

```
[filename, pathname] = uigetfile('*.m', 'Open M-file');
str = [pathname, filename, ' is opened.'];
disp(str)
```

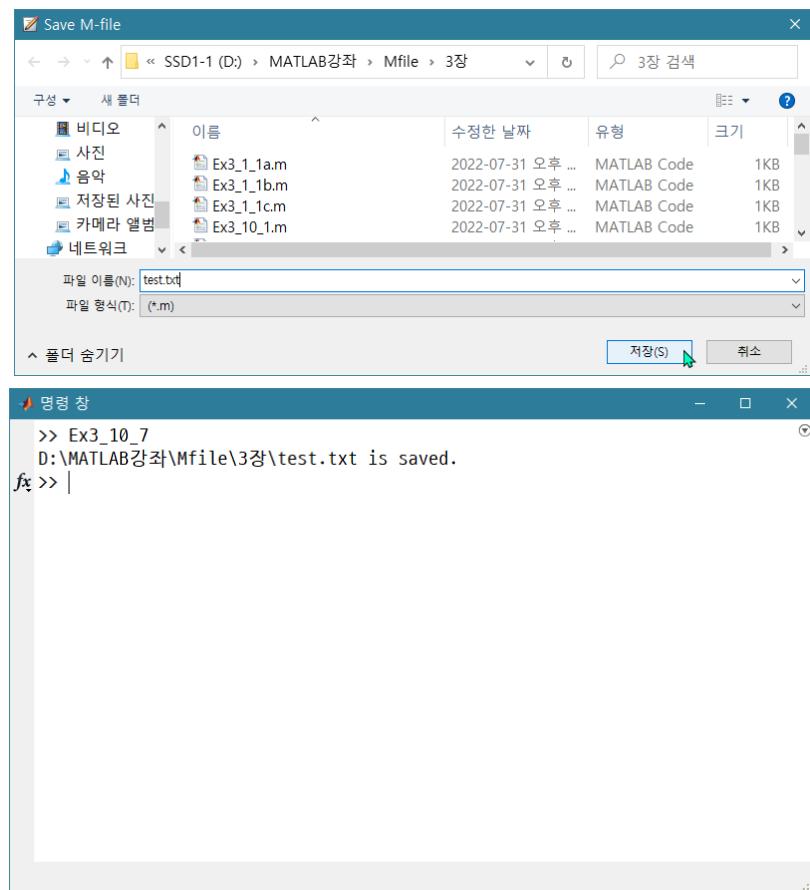


✚ Uiputdlg 함수

```
[filename, pathname] = uiputfile(filterspec, title)
```

✚ Uiputdlg 함수의 예제

```
[filename, pathname] = uiputfile('*.m', 'Save M-file');  
str = [pathname, filename, ' is saved.'];  
disp(str)
```



3.12 개인적으로 사용하는 유용한 함수

▶ 변수 값을 표현하는 방법

- ✓ 단위(unit) 사용
- ✓ 변수 값과 단위를 같이 나타내는 함수 사용

▶ 기존의 프로그램

```
clc; clear;

% Case 1

V = 10; R1 = 10*10^3; R2 = 30*10^6; % 저항 10kohm, 30Mohm
I1 = V/R1
I2 = V/R2

% Case 2

V = 10; I = 0.1*10^-3; % 전류 0.1mA
R = V/I

% Case 3

IC = 1*10^-3; VT = 26*10^-3;
gm = IC/VT
```

- ✓ 결과 (명령 창)

```
I1 =
1.0000e-03
I2 =
3.3333e-07
R =
100000
gm =
0.0385
```

▶ 변수 값과 단위를 보여주는 함수

```
function display_value(str1,x,str2)
str3 = [ ' ',str1,num2str(x), ' ',str2];
disp(str3)
```

 수정한 프로그램

```

clc; clear;

% Unit

k = 10^3; M = 10^6; G = 10^9; T = 10^12; P = 10^15;
m = 10^-3; u = 10^-6; n = 10^-9; p = 10^-12; f = 10^-15;

% Case 1

V = 10; R1 = 10*k; R2 = 30*M;
I1 = V/R1;
I2 = V/R2;
display_value('I1 = ', I1/m, '[mA]')
display_value('I2 = ', I2/u, '[uA]')

% Case 2

V = 10; I = 0.1*10^-3;
R = V/I;
display_value('R = ', R/k, '[kV/A]')

% Case 3

IC = 1*10^-3; VT = 26*10^-3;
gm = IC/VT;
display_value('gm = ', gm/m, '[mA/V]')

```

 결과 (명령 창)

```

I1 = 1 [mA]
I2 = 0.33333 [uA]
R = 100 [kV/A]
gm = 38.4615 [mA/V]

```

-  프로그램 변동은 없고 값을 직접 보여주는 대신 함수를 통해 보여준다.

3.13 연습 문제

- ▶ 문제 1. 평균과 분산이 각각 mx , vx 이고 nx 개의 가우시안 백색 잡음을 생성하는 함수 `wgauss.m` 을 작성하고, 이를 이용하여 평균이 1이고 분산이 5인 가우시안 백색 잡음 1000개를 생성하라.
- ▶ 문제 2. 주어진 입력 변수의 평균을 구하는 함수 `aver.m` 을 작성하라. 단 입력 변수는 4개 이하이며 4개를 초과하면 `error` 문을 표시하도록 하라.
- ▶ 문제 3. 배열이 주어질 때 배열의 최대값, 최소값을 구하는 함수 `min_max.m` 을 작성하고, 이를 이용하여 적절히 만든 배열의 최소값과 최대값을 구하라.
- ▶ 문제 4. 4개 이내의 저항의 병렬값을 구하는 함수 `par_R.m` 을 작성하고 이 함수를 테스트하라.
- ▶ 문제 5. 두 개의 저항 R_1 , R_2 의 병렬값을 R 이라 할 때 R_1 과 R 이 주어질 때 R_2 를 구하는 함수 $R_2 = \text{depar_R}(R, R_1)$ 을 작성하라.
- ▶ 문제 6. 배열로 주어지는 다항식을 4개까지 더하는 함수 `polyadd.m` 을 작성하라.
- ▶ 문제 7. 주어진 배열을 내림 차순으로 배열하고 번호도 돌려주는 함수 `sort_score.m` 을 작성하라. 이를 이용하여 10명의 성적을 내림 차순으로 배열하라.
- ▶ 문제 8. 카이저 창을 이용하여 디지털 필터를 설계할 때 감쇄 A 가 주어질 때 β 는 다음과 같이 구할 수 있다. A 가 주어질 때 β 를 구하는 함수 `kaiser_beta` 함수를 작성하라.

$$\beta = \begin{cases} 0, & A < 21 \\ 0.5842(A - 21)^{0.4} + 0.07886(A - 21), & 21 \leq A \leq 50 \\ 0.1102(A - 8.7), & A > 50 \end{cases}$$

4. 그래픽스(Graphics)

4.1 MATLAB 그래프

4.1.1 2 차원 그림

Line Graphs	Bar Graphs	Area Graphs
<code>plot</code> 	<code>bar (grouped)</code> 	<code>area</code>
<code>plotyy</code> 	<code>barh (grouped)</code> 	<code>pie</code>
<code>loglog</code> 	<code>barh (stacked)</code> 	<code>fill</code>
<code>semilogx</code> 	<code>barh (stacked)</code> 	<code>contourf</code>
<code>semilogy</code> 	<code>histogram</code> 	<code>image</code>
<code>stairs</code> 	<code>pareto</code> 	<code>pcolor</code>
<code>contour</code> 	<code>errorbar</code> 	<code>ezcontourf</code>
<code>ezplot</code> 	<code>stem</code> 	
<code>ezcontour</code> 		

Direction Graphs	Radial Graphs	Scatter Graphs
<code>feather</code> 	<code>polar</code> 	<code>scatter</code>
<code>quiver</code> 	<code>rose</code> 	<code>spy</code>
<code>comet</code> 	<code>compass</code> 	<code>plotmatrix</code>
	<code>ezpolar</code> 	

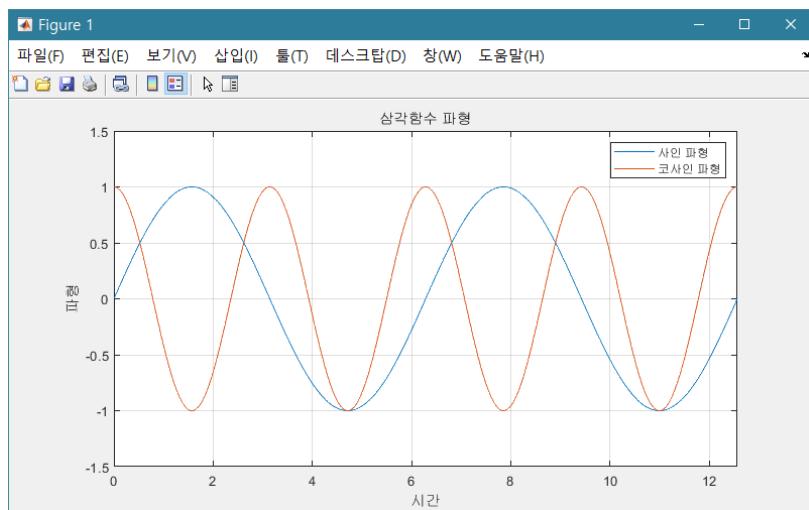
4.1.2 3 차원 그림

Line Graphs	Mesh Graphs and Bar Graphs	Area Graphs and Constructive Objects
plot3 	mesh 	pie3 
contour3 	meshc 	fill3 
contourslice 	meshz 	patch 
ezplot3 	ezmesh 	cylinder 
waterfall 	stem3 	ellipsoid 
	bar3 	sphere 
	bar3h 	

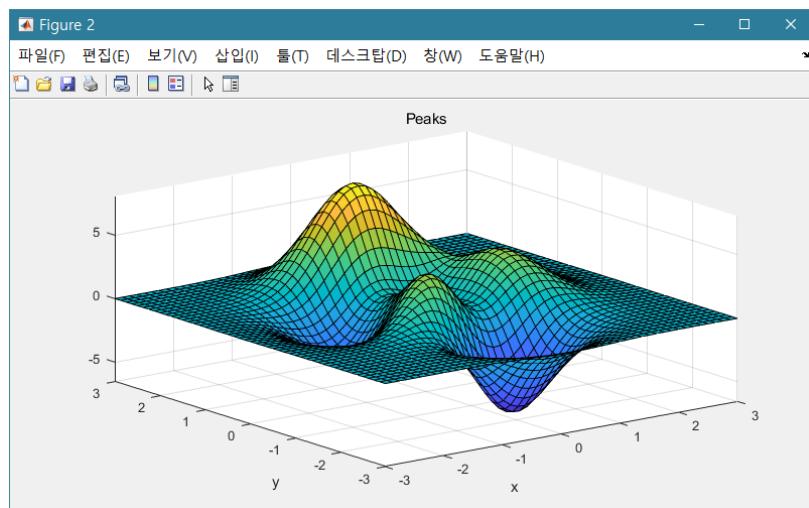
Surface Graphs	Direction Graphs	Volumetric Graphs
surf 	quiver3 	scatter3 
surfl 	comet3 	coneplot 
surf c 	streamslice 	streamline 
ezsurf 		streamribbon 
ezsurf c 		streamtube 

4.1.3 그림 예

➊ 2 차원 그림 예



3 차원 그림 예



4.1.4 그림 창

그림 창 함수

```
figure
```

- ✓ 새 그림 창을 연다.

```
figure (n)
```

- ✓ n 번 그림 창을 연다. 없으면 만들고 연다.

```
close
```

- ✓ 최근의 그림 창을 닫는다.

```
close (n)
```

- ✓ n 번 그림 창을 닫는다.

```
close all
```

- ✓ 모든 그림 창을 닫는다.

4.2 2 차원 그림

▣ 2-D plotting 함수

- ✓ plot : 점을 선으로 이어 그린다.
- ✓ stem : 점에서 가로축까지 수직선을 그린다.
- ✓ polar : 극 좌표계 그림을 그린다.
- ✓ fplot : 주어진 함수 그림을 그린다.

▣ 2-D plotting 관련 함수

- ✓ grid on/off : 눈금을 나타내거나 감춘다.
- ✓ axis : x 축, y 축의 범위를 설정한다.
- ✓ xlim : x 축의 범위를 설정한다.
- ✓ ylim : y 축의 범위를 설정한다.
- ✓ title : 그래프의 제목을 쓴다.
- ✓ xlabel : x 축의 이름을 쓴다.
- ✓ ylabel : y 축의 이름을 쓴다.
- ✓ legend : 파형의 속성을 쓴다.
- ✓ text/gtext : 주어진 좌표에 문자열을 쓴다.
- ✓ hold on/off : 그림 판을 유지하거나 유지하지 않는다.
- ✓ linspace : 주어진 구간에 주어진 개수 만큼 선형 간격의 값을 돌려준다.
- ✓ logspace : 주어진 구간에 주어진 개수 만큼 로그 간격의 값을 돌려준다.

4.2.1 plot 함수

▣ 일반적인 경우

```
plot(ydata)
```

- ✓ x 축은 ydata 의 지수, y 축은 ydata 의 값을 나타낸다.

```
plot(xdata,ydata)
```

- ✓ x 축은 xdata 의 값, y 축은 ydata 의 값을 나타낸다.

```
plot(xdata,ydata,string)
```

- ✓ x 축은 xdata의 값, y 축은 ydata의 값, string은 선의 색, 선의 심볼, 선의 종류를 나타낸다.

```
plot(x1,y1,str1, x2,y2,str2, x3,y3,str3, ...)
```

- ✓ 여러 개의 데이터를 그린다.

▣ 특수한 경우

```
plot(ydata)
```

- ✓ ydata가 복소수 벡터인 경우 : x 축은 실수 성분, y 축은 허수 성분을 나타낸다.
- ✓ ydata가 행렬인 경우 : 열 벡터를 열의 수만큼 선으로 나타낸다.

```
plot(xdata,ydata)
```

- ✓ xdata가 행 벡터이고, ydata가 행렬인 경우 : ydata의 행 벡터를 행의 수만큼 선으로 나타낸다.
- ✓ xdata가 열 벡터이고, ydata가 행렬인 경우 : ydata의 열 벡터를 열의 수만큼 선으로 나타낸다.

▣ String (선의 색, 선의 심볼, 선의 종류)

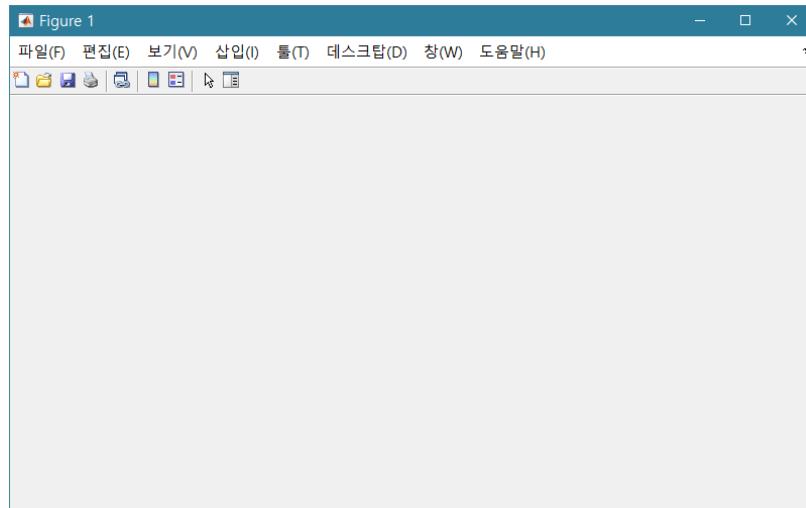
Line color		Line symbol		Line type	
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star		
y	yellow	s	square		
k	black	d	diamond		
		v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

4.2.2 단일 그래프 그리기

$$y = \sin(x), \quad 0 \leq x \leq 4\pi$$

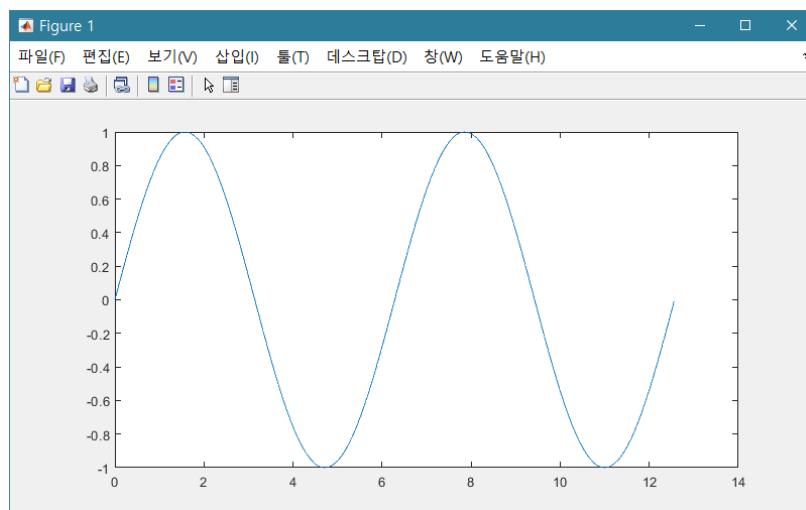
➊ 모든 그림 창 닫고 새 그림 창 열기

```
>> close all; % 모든 그림 창 닫기
>> figure % 새로운 그림 창 준비
```



➋ x, y 값 설정과 그래프 그리기

```
>> x = 0:0.01:4*pi; % x 값 설정
>> y = sin(x); % y 값 계산
>> plot(x,y) % 그래프 그리기
```



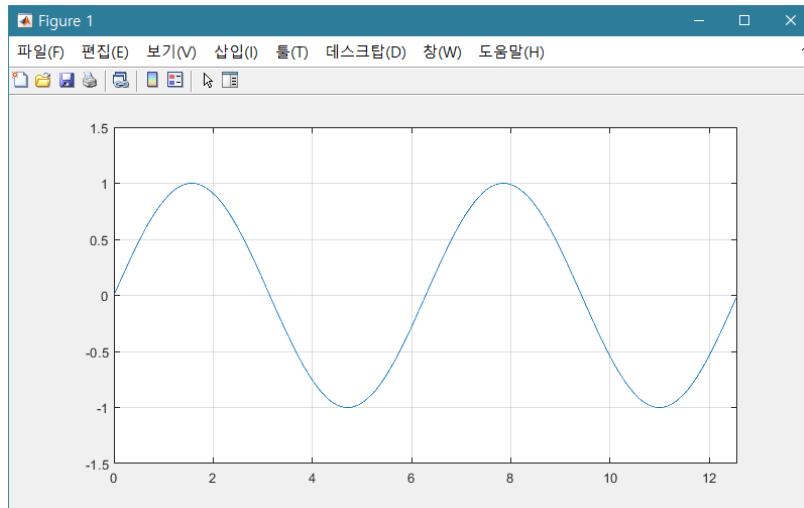
✓ 참고 함수

✗ linspace

✗ logspace

➌ 좌표축 범위 설정과 눈금 표시

```
>> axis([0,4*pi,-1.5,1.5]) % 좌표 축 범위 설정
>> grid on % 눈금 표시
```



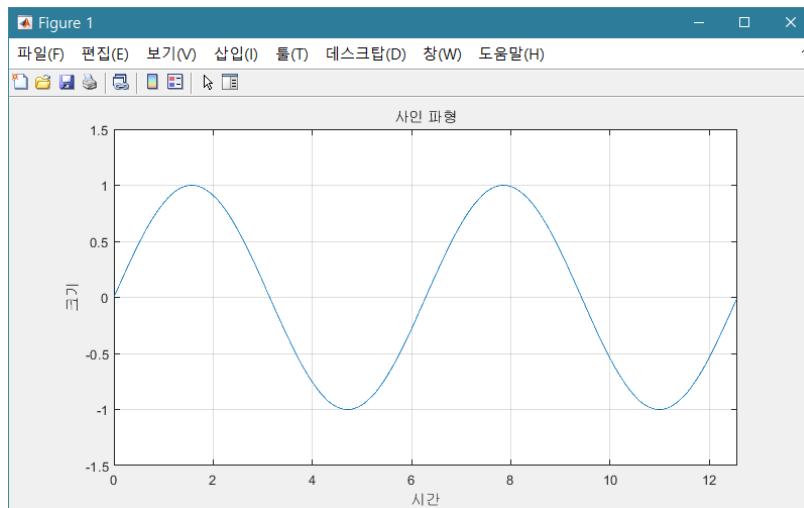
✓ 참고 함수

✗ xlim

✗ ylim

✚ 타이틀, x 축 레이블, y 축 레이블 추가

```
>> title('사인 파형') % 그래프 타이틀 설정
>> xlabel('시간') % x축 레이블 설정
>> ylabel('크기') % y축 레이블 설정
```



✚ 전체 프로그램

```
clc; close all;
```

```
figure
```

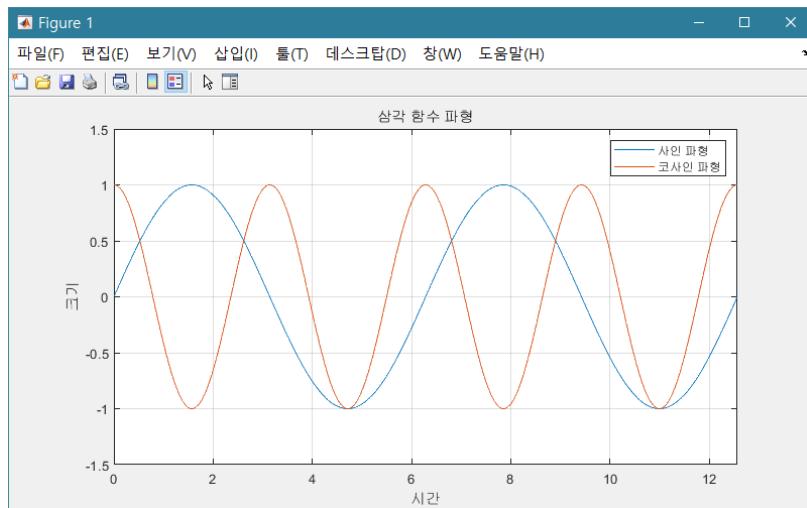
```
x = 0:0.01:4*pi;
y = sin(x);
plot(x,y)
axis([0,4*pi,-1.5,1.5])
grid on
title('사인 파형')
xlabel('시간')
ylabel('크기')
```

4.2.3 다중 그래프 그리기

$$y_1 = \sin(x), \quad y_2 = \cos(2x), \quad 0 \leq x \leq 4\pi$$

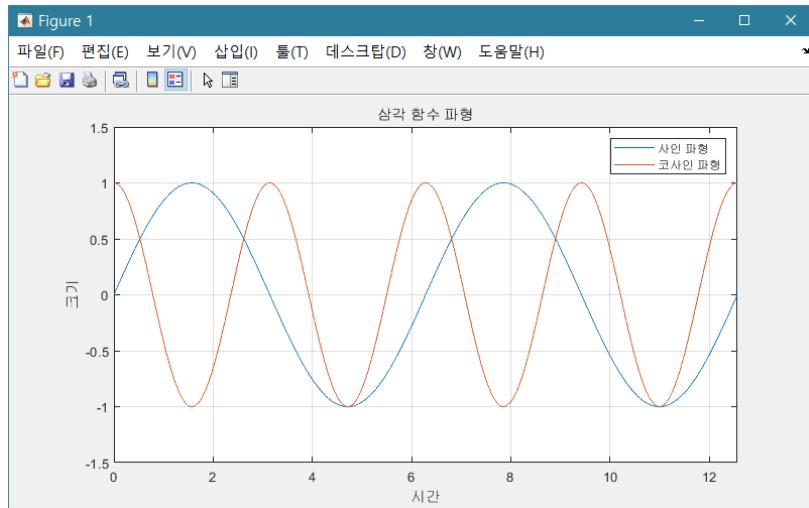
➊ 다중 plot 함수 이용하는 방법

```
clc; close all;
figure
x = linspace(0,4*pi,1001);
y1 = sin(x);
y2 = cos(2*x);
plot(x,y1,x,y2) % 다중 그래프 그리기
axis([0,4*pi,-1.5,1.5])
grid on
title('삼각 함수 파형')
xlabel('시간')
ylabel('크기')
legend('사인 파형','코사인 파형')
```



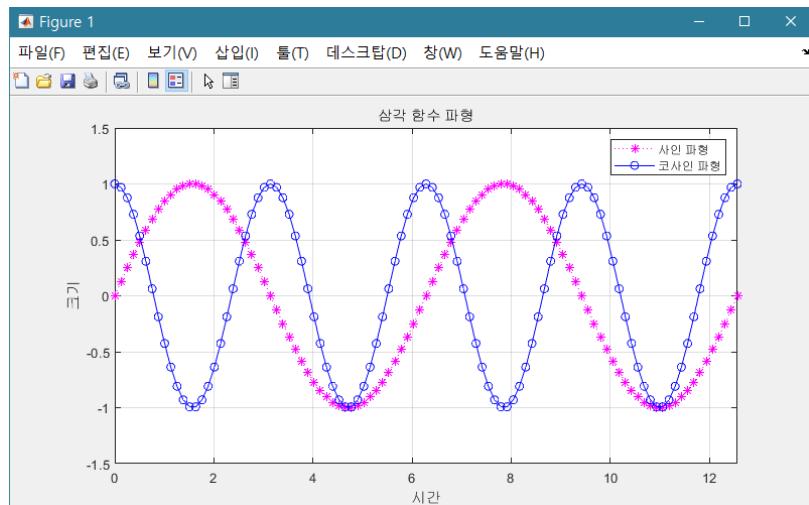
➊ hold on 함수 이용하는 방법

```
clc; close all;
figure
x = linspace(0,4*pi,1001);
y1 = sin(x);
plot(x,y1)
hold on
y2 = cos(2*x);
plot(x,y2)
axis([0,4*pi,-1.5,1.5])
grid on
title('삼각 함수 파형')
xlabel('시간')
ylabel('크기')
legend('사인 파형','코사인 파형')
hold off
```



graph 속성 변경

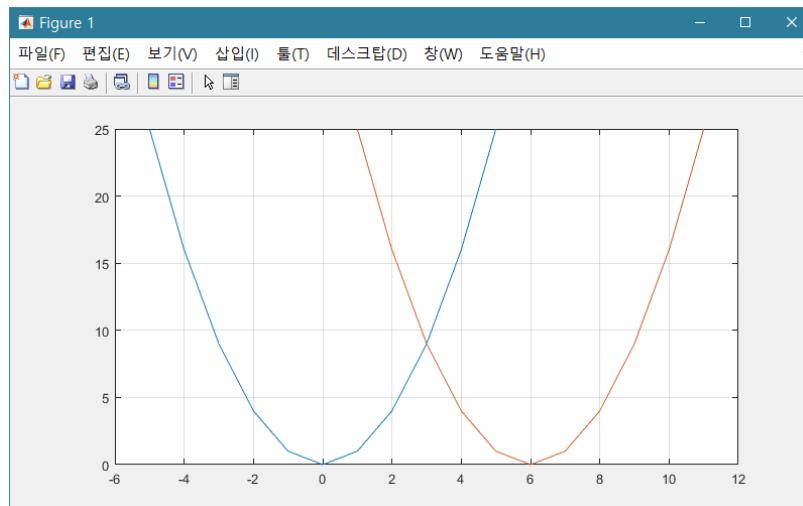
```
clc; close all;
figure
x = linspace(0,4*pi,101);      % 포인트 수 축소
y1 = sin(x);
y2 = cos(2*x);
plot(x,y1,'-*:',x,y2,'bo-')  % 속성 부여
axis([0,4*pi,-1.5,1.5])
grid on
title('삼각 함수 파형')
xlabel('시간')
ylabel('크기')
legend('사인 파형','코사인 파형')
```



4.2.4 특수한 경우

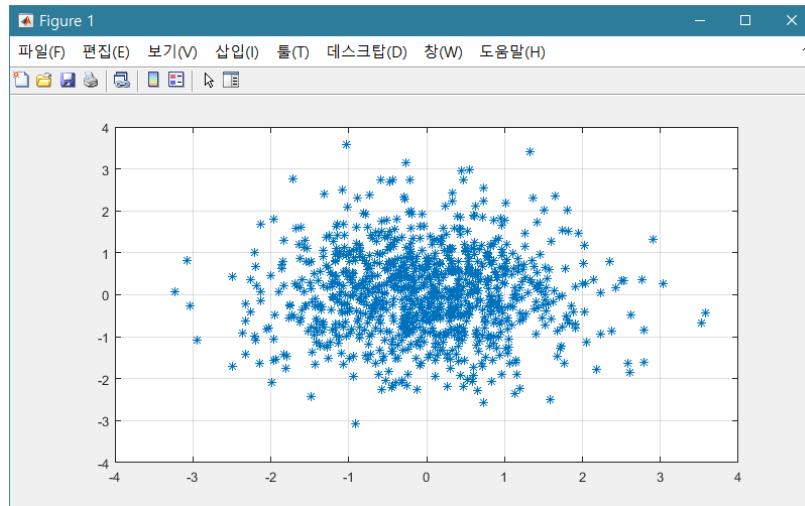
■ x 데이터가 없는 경우

```
clc; close all;  
figure  
x = -5:5;  
y = x.^2;  
plot(x,y)  
hold on  
plot(y)  
grid on  
hold off
```



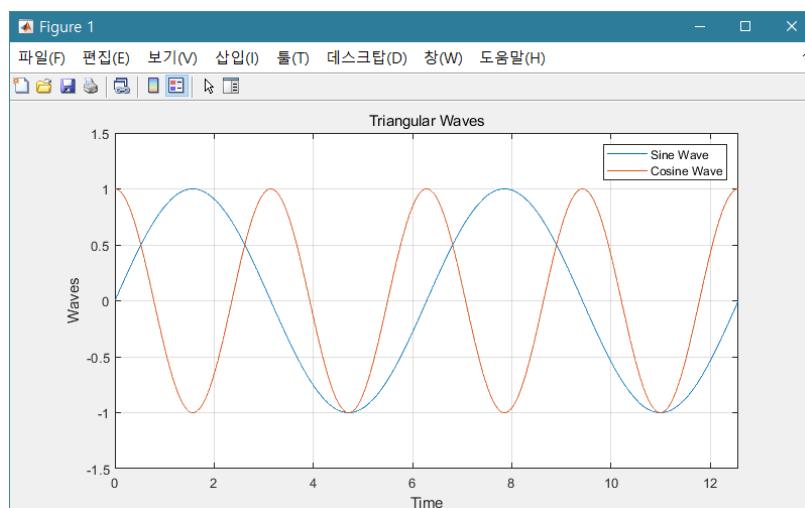
■ y 데이터가 복소수인 경우

```
clc; close all;  
figure  
xr = randn(1,1000);  
xi = randn(1,1000);  
x = xr + j*xi; % 복소수  
plot(x,'*')  
axis([-4,4,-4,4])  
grid on
```



➊ y 데이터가 행렬인 경우

```
clc; close all;
figure
x = linspace(0,4*pi,1000);
y1 = sin(x);
y2 = cos(2*x);
y = [y1;y2]; % 행렬
plot(x,y)
axis([0,4*pi,-1.5,1.5])
grid on
title('Triangular Waves')
xlabel('Time')
ylabel('Waves')
legend('Sine Wave','Cosine Wave')
```



4.2.5 stem 함수

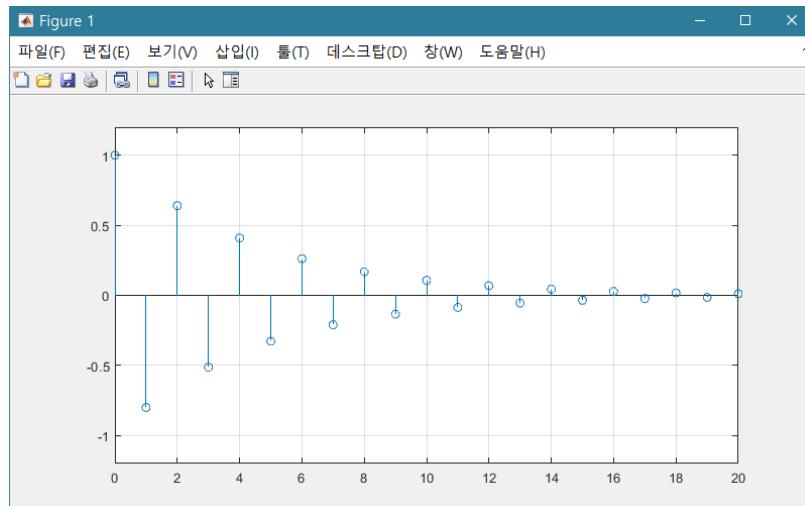
▣ 이산 시간 신호를 그릴 때 사용한다.

stem(xdata, ydata)

- ✓ x 축은 xdata의 값, y 축은 ydata의 값을 나타낸다.

$$x[n] = \left(-\frac{4}{5}\right)^n, \quad n = 0, 1, \dots, 20$$

```
clc; close all;
figure
n = 0:20;
x = (-4/5).^n;
stem(n,x)
axis([0,20,-1.2,1.2])
grid on
```



4.2.6 polar 함수

▣ 극 좌표 그래프를 그릴 때 사용한다.

polar(theta, rho)

- ✓ $z = \rho \exp[j\theta]$

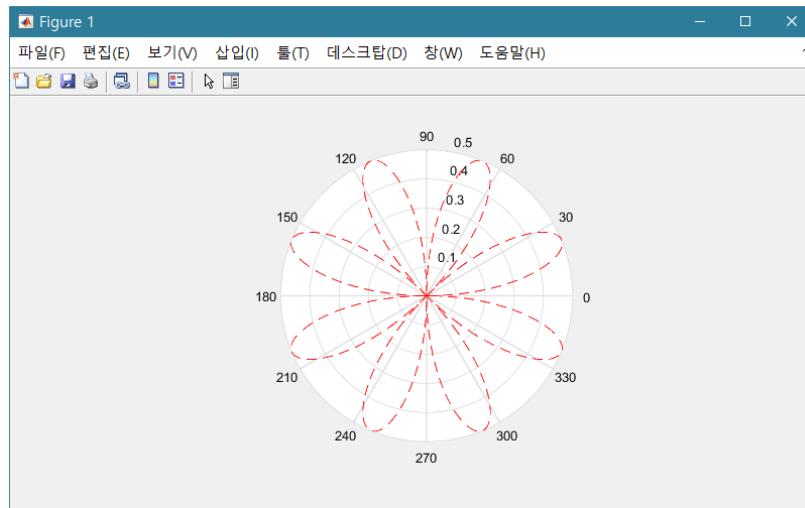
polar(theta, rho)

- ✓ string 은 선의 색, 선의 심볼, 선의 종류를 나타낸다.

- ✓ polar 함수는 다중 그래프를 그릴 수 없다. (행렬로 그리는 것은 가능하다.)

▣ 극 좌표 그래프

```
clc; close all;
figure
t = 0:0.01:2*pi;
x = sin(2*t).*cos(2*t);
polar(t,x,'--r')
grid on
```



4.2.7 fplot 함수

▣ fplot 함수

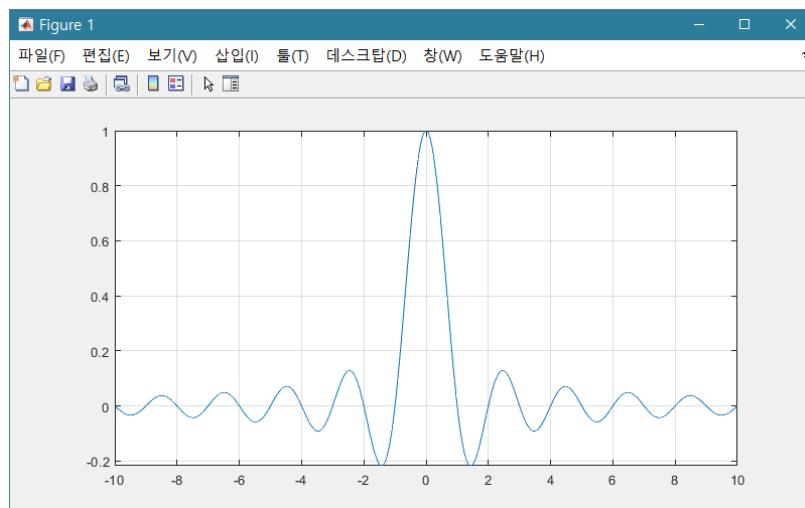
```
fplot(@fun,limits)
```

- ✓ fun : fun=f(x) 형태의 함수 명
- ✓ limits : x 축의 범위

▣ 함수 그리기

$$y = \text{sinc}(x), -10 \leq x \leq 10$$

```
clc; close all;
figure
fplot(@sinc,[-10,10])
grid on
```



4.3 3 차원 그림

■ 3-D plotting 함수

- ✓ plot3
- ✓ stem3
- ✓ mesh, meshc, meshz
- ✓ surf, surfc, surfz
- ✓ contour, contourf

■ 3-D plotting 관련 함수

- ✓ meshgrid
- ✓ colorbar
- ✓ clabel

4.3.1 plot3, stem3 함수

■ 일반적인 경우 (x, y, z 가 모두 벡터인 경우)

plot3(x, y, z)

- ✓ x 축은 x 의 값, y 축은 y 의 값, z 축은 z 의 값을 나타낸다.

plot3(x, y, z, string)

- ✓ x 축은 x 의 값, y 축은 y 의 값, z 축은 z 의 값, string 은 선의 색, 선의 심볼, 선의 종류를 나타낸다

plot3(x1, y1, z1, str1, x2, y2, z2, str2, ...)

- ✓ 여러 개의 데이터를 그린다.

■ 특수한 경우 (x, y, z 가 모두 행렬인 경우)

plot3(x, y, z)

- ✓ x 축은 x 의 값, y 축은 y 의 값, z 축은 z 의 값을 나타낸다.
- ✓ x, y, z 의 열 벡터를 열의 수만큼 선으로 나타낸다.

■ stem3 함수

```
stem3(x,y,z)
```

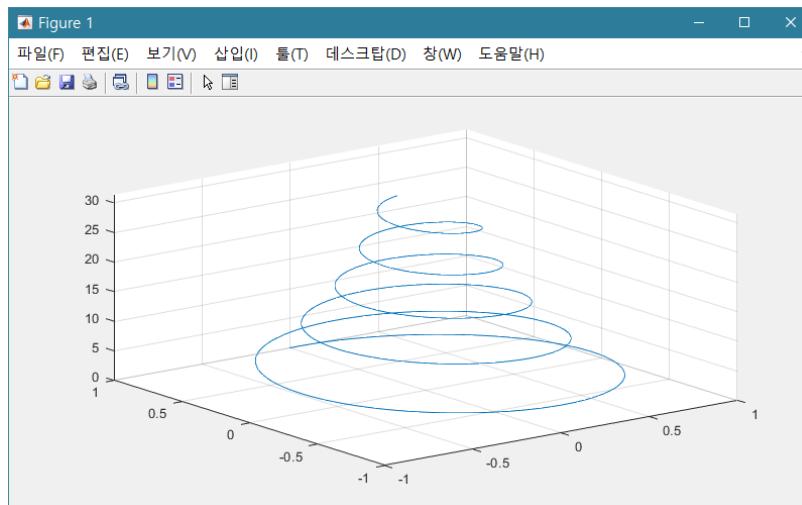
- ✓ 3 차원 이산 시간 신호를 그린다.

4.3.2 3 차원 선 그리기

plot3 함수

$$x = \exp\left[-\frac{t}{20}\right] \sin t, \quad y = \exp\left[-\frac{t}{20}\right] \cos t, \quad z = t, \quad 0 \leq t \leq 10\pi$$

```
clc; close all;
figure
t = 0:pi/50:10*pi;
x = exp(-t/20).*sin(t);
y = exp(-t/20).*cos(t);
z = t;
plot3(x,y,z)
grid on
```

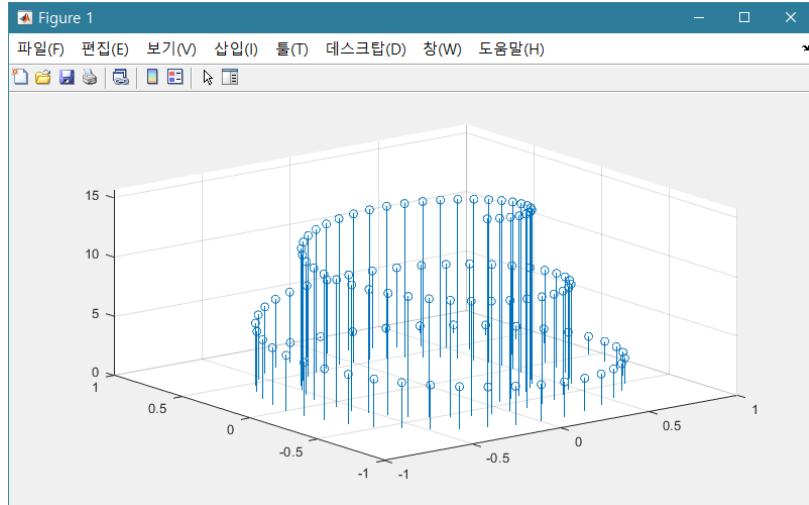


stem3 함수

$$x = \exp\left[-\frac{t}{20}\right] \sin t, \quad y = \exp\left[-\frac{t}{20}\right] \cos t, \quad z = t, \quad t = 0, \frac{\pi}{20}, \frac{2\pi}{20}, \frac{3\pi}{20}, \dots, 5\pi$$

```
clc; close all;
figure
t = 0:pi/20:5*pi;
x = exp(-t/20).*sin(t);
```

```
y = exp(-t/20).*cos(t);
z = t;
stem3(x,y,z)
grid on
```



4.3.3 mesh, surf, contour 함수

meshgrid 함수

```
[X,Y] = meshgrid(x,y)
```

- ✓ x, y : x 축, y 축의 범위
- ✓ X, Y : 면 그래프에 사용되는 행렬

```
>> x = -1:1;
>> y = 0:1;
>> [X,Y] = meshgrid(x,y)
X =
    -1     0     1
    -1     0     1
Y =
    0     0     0
    1     1     1
```

3-D 그물 격자 그래프 함수

```
mesh(X,Y,Z)
```

- ✓ X,Y : meshgrid 에서 얻은 행렬 변수
- ✓ Z : 3-D 함수

❖ 3-D 표면 그래프 함수

```
surf(X,Y,Z)
```

❖ 3-D 등고선 그래프 함수

```
contour(X,Y,Z)
```

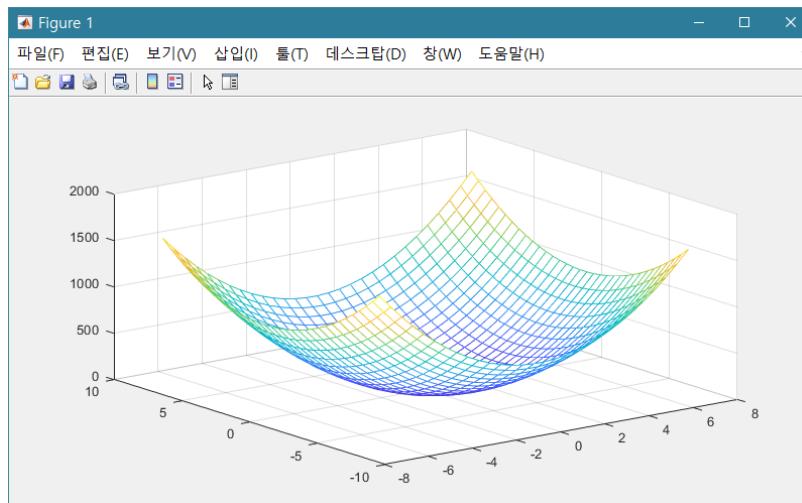
```
[CH,S] = contour(X,Y,Z)
```

4.3.4 3 차원 면 그래프 그리기

❖ 3 차원 그물 격자 그래프

$$z = f(x, y) = 20x^2 + 10y^2 + xy + 5$$

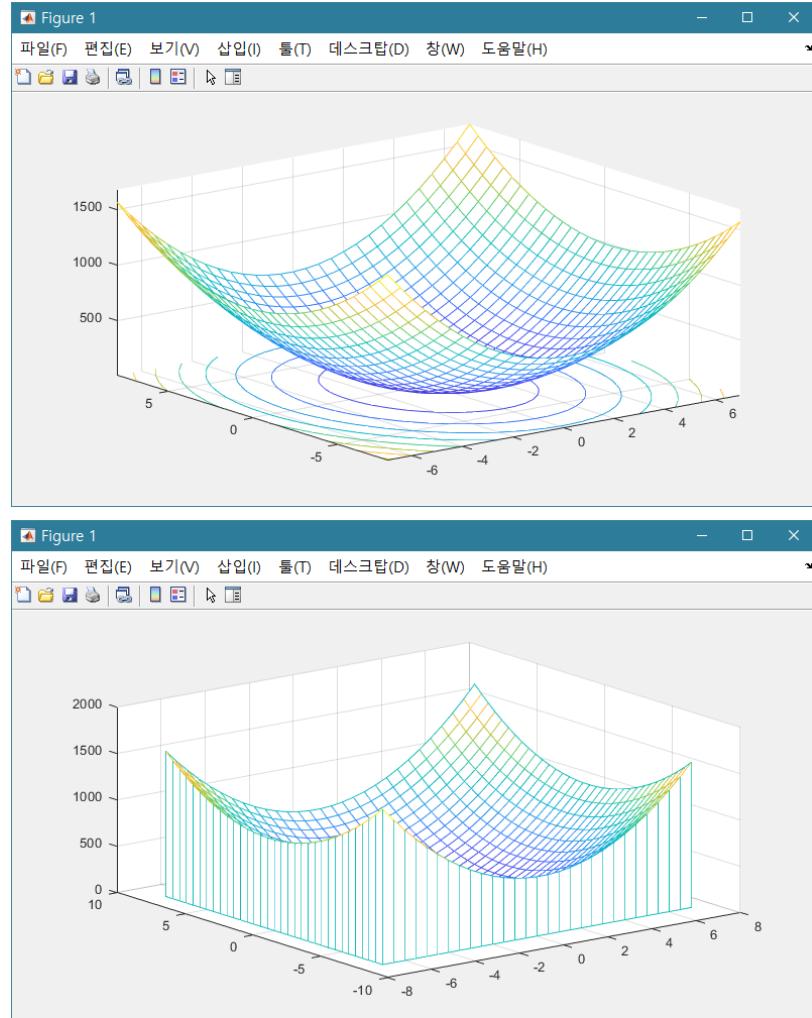
```
clc; close all;
figure
x = -7:0.5:7;
y = -8:0.5:8;
[X,Y] = meshgrid(x,y);
z = 20*x.^2+10*y.^2+x.*y+5;
mesh(X,Y,Z)
grid on
```



- ✓ 관련 함수

✗ meshc

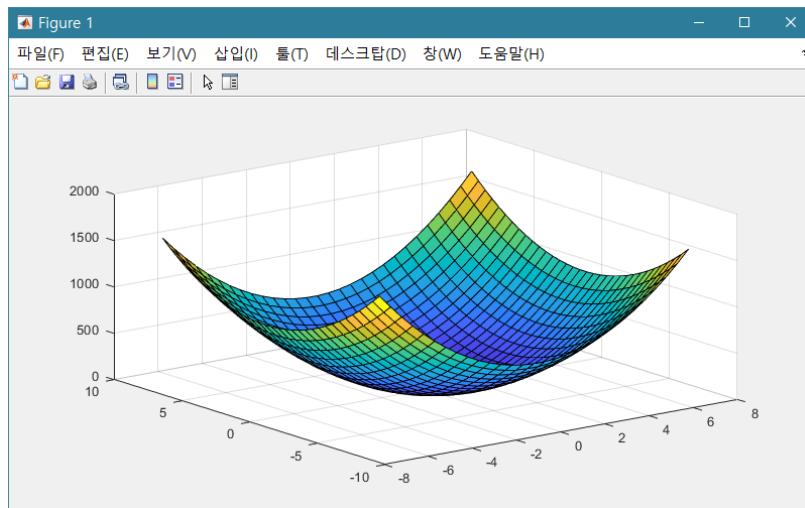
✗ meshz



➊ 3 차원 그물 표면 그래프

$$z = f(x, y) = 20x^2 + 10y^2 + xy + 5$$

```
clc; close all;
figure
x = -7:0.5:7;
y = -8:0.5:8;
[X,Y] = meshgrid(x,y);
z = 20*x.^2+10*y.^2+x.*y+5;
surf(X,Y,z)
grid on
```



✓ 관련 함수

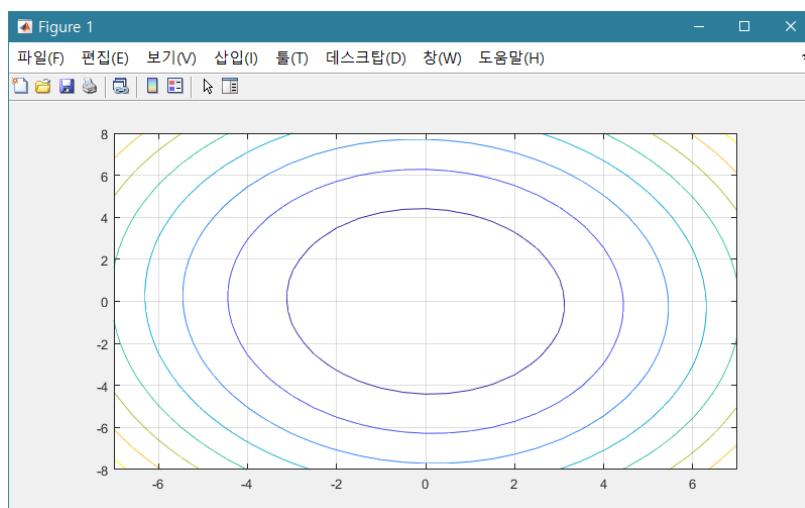
✗ surf

✗ surfz

✚ 3 차원 등고선 그래프

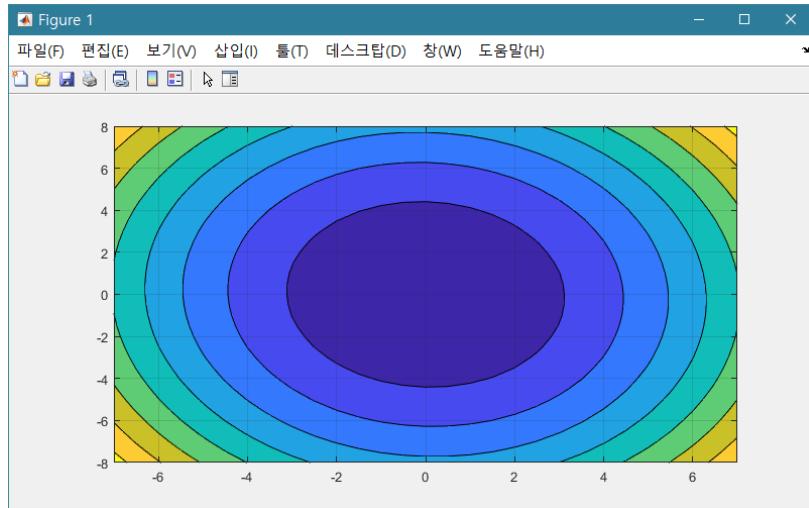
$$z = f(x, y) = 20x^2 + 10y^2 + xy + 5$$

```
clc; close all;
figure
x = -7:0.5:7;
y = -8:0.5:8;
[X,Y] = meshgrid(x,y);
Z = 20*x.^2+10*y.^2+x.*y+5;
contour(X,Y,Z)
grid on
```

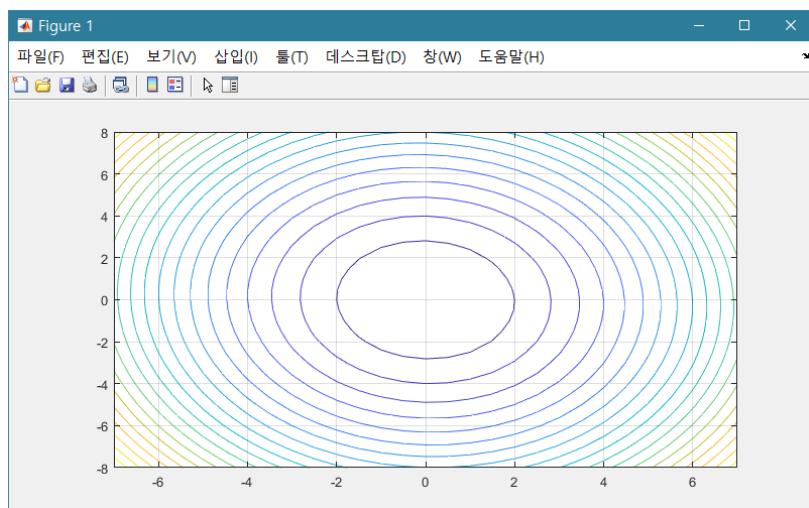


✓ 관련 함수

✗ contourf



contour(X, Y, Z, 20)

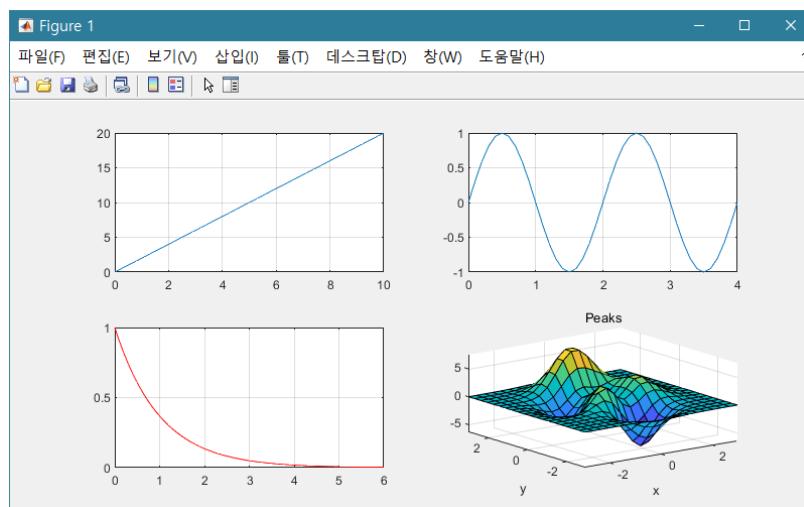


4.4 그림 분할

▶ 한 개의 창에 여러 개의 그림 그리기

```
subplot(#rows,#cols,index)
```

```
clc; close all;
figure
subplot(2,2,1) % 2 사분면
x = 0:0.1:10;
plot(x,2*x)
grid on
subplot(2,2,2) % 1 사분면
x = 0:0.1:4;
plot(x,sin(pi*x))
grid on
subplot(2,2,3) % 3 사분면
x = 0:0.1:6;
plot(x,exp(-x), 'r')
grid on
subplot(2,2,4) % 4 사분면
peaks(20)
grid on
```



4.5 좌표 축의 종류

- x 축 : 선형 눈금, y 축 : 선형 눈금

```
plot(...)
```

- x 축 : 로그 눈금, y 축 : 선형 눈금

```
semilogx(...)
```

- x 축 : 선형 눈금, y 축 : 로그 눈금

```
semilogy(...)
```

- x 축 : 로그 눈금, y 축 : 로그 눈금

```
loglog(...)
```

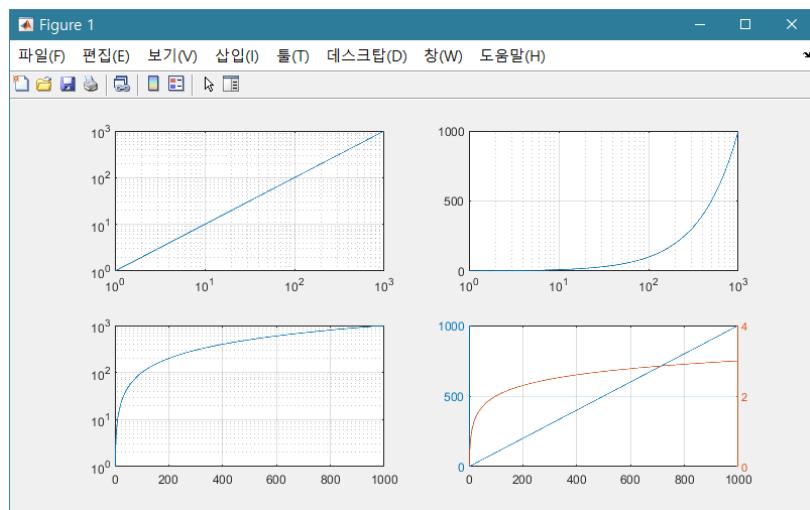
- x 축 : 선형 눈금, y1 축 : 선형 눈금, y2 축 : 선형 눈금 (이중 눈금)

```
plotyy(x1,y1,x2,y2)
```

```

clc; close all;
figure
x = 1:1000;
y = x;
subplot(2,2,1)
loglog(x,y) % x 축, y 축 : 로그 눈금
grid on
subplot(2,2,2)
semilogx(x,y) % x 축 : 로그 눈금
grid on
subplot(2,2,3)
semilogy(x,y) % y 축 : 로그 눈금
grid on
subplot(2,2,4)
plotyy(x,y, x,log10(x)) % y 축 : 이중 눈금
grid on

```



4.6 시스템의 주파수 특성

4.6.1 아날로그 시스템의 주파수 특성

✚ 아날로그 시스템의 전달 함수

$$H(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0}$$

- ✓ s 에 대해 내림 차순으로 정렬한다.

✚ 아날로그 시스템의 주파수 특성

$$H(j\Omega) = \frac{b_m (j\Omega)^m + b_{m-1} (j\Omega)^{m-1} + \cdots + b_1 (j\Omega) + b_0}{a_n (j\Omega)^n + a_{n-1} (j\Omega)^{n-1} + \cdots + a_1 (j\Omega) + a_0}$$

- ✓ 아날로그 시스템의 주파수는 로그 눈금이다.

✚ 아날로그 시스템의 주파수 특성 함수

- ✓ freqs

$$\mathbf{b} = [b_m, b_{m-1}, \dots, b_1, b_0]$$

$$\mathbf{a} = [a_n, a_{n-1}, \dots, a_1, a_0]$$

$$H(s) = \frac{1}{s^3 + 2s^2 + 2s + 1} \Rightarrow \mathbf{b} = [1, 2, 2, 1], \mathbf{a} = [1, 2, 2, 1]$$

✚ freqs 함수

```
freqs(b, a)
```

- ✓ 크기 응답(dB)과 위상 응답을 같이 그린다.

```
H = freqs(b, a, w)
```

- ✓ 주어진 각 주파수 w 에 대한 주파수 특성 값을 돌려준다.
- ✓ 각 주파수 w 는 로그 눈금 분포이다.

```
Hm = abs(H)
```

```
Hp = angle(H)
```

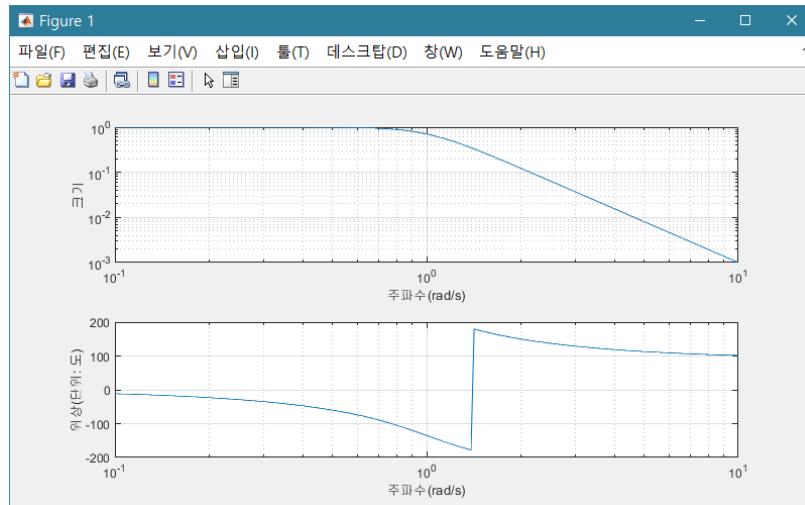
- ✓ 크기 응답과 위상 응답

✚ 아날로그 버터워스 저역 통과 필터

$$H(s) = \frac{1}{s^3 + 2s^2 + 2s + 1}$$

✓ 프로그램 1

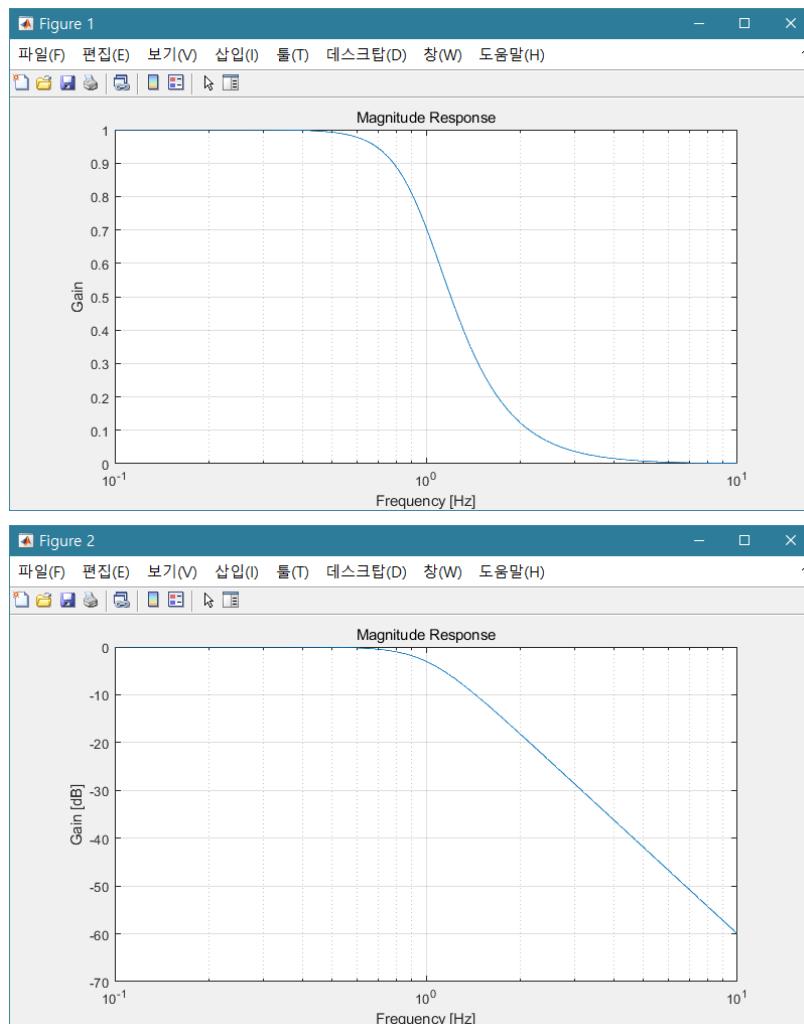
```
clc; close all;
figure
b = 1;
a = [1 2 2 1];
freqs(b,a)
grid on
```

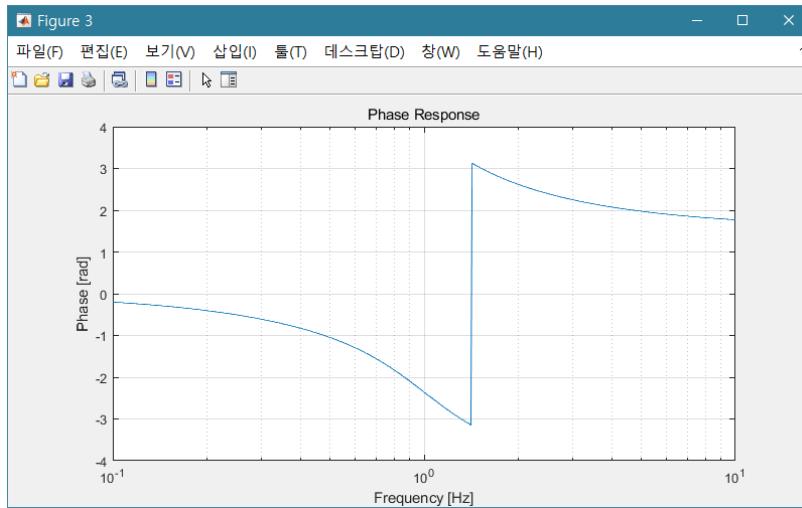


✓ 프로그램 2

```
clc; close all;
b = 1;
a = [1 2 2 1];
w = logspace(-1,1,500);
H = freqs(b,a,w);
Hm = abs(H);
Hm_dB = 20*log10(Hm);
Hp = angle(H);
figure(1)
semilogx(w,Hm) % 크기 응답
grid on
title('Magnitude Response')
```

```
xlabel('Frequency [Hz]')
ylabel('Gain')
figure(2)
semilogx(w,Hm_dB) % 크기 응답 [dB]
grid on
title('Magnitude Response')
xlabel('Frequency [Hz]')
ylabel('Gain [dB]')
figure(3)
semilogx(w,Hp) % 위상 응답
grid on
title('Phase Response')
xlabel('Frequency [Hz]')
ylabel('Phase [rad]')
```





4.6.2 디지털 시스템의 주파수 특성

➊ 디지털 시스템의 전달 함수

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_{m-1} z^{-(m-1)} + b_m z^{-m}}{a_0 + a_1 z^{-1} + \cdots + a_{n-1} z^{-(n-1)} + a_n z^{-n}}$$

- ✓ z^{-1} 에 대해 오름 차순으로 정렬한다.

➋ 디지털 시스템의 주파수 특성

$$H(e^{j\omega}) = \frac{b_0 + b_1 e^{-j\omega} + \cdots + b_{m-1} e^{-j\omega(m-1)} + b_m e^{-j\omega m}}{a_0 + a_1 e^{-j\omega} + \cdots + a_{n-1} e^{-j\omega(n-1)} + a_n e^{-j\omega n}}$$

- ✓ 디지털 시스템의 주파수는 선형 눈금이다.
- ✓ 디지털 시스템의 주파수 특성은 ω 에 대해 주기가 2π 인 주기 함수이다.

➌ 디지털 시스템의 주파수 특성 함수

- ✓ freqz

$$\mathbf{b} = [b_0, b_1, \dots, b_{m-1}, b_m]$$

$$\mathbf{a} = [a_0, a_1, \dots, a_{n-1}, a_n]$$

$$H(z) = \frac{0.3 + 0.3z^{-1} + 0.1z^{-2}}{1 - 0.5z^{-1} + 0.2z^{-2}} \Rightarrow \mathbf{b} = [0.3, 0.3, 0.1], \mathbf{a} = [1, -0.5, 0.2]$$

➍ freqz 함수

```
freqz(b, a)
freqz(b, a, 'whole')
```

- ✓ 크기 응답(dB)과 위상 응답을 반 주기($0 \leq \omega \leq \pi$) 동안 같이 그린다.

- ✓ 크기 응답(dB)과 위상 응답을 한 주기($0 \leq \omega \leq 2\pi$) 동안 같이 그린다.

```
H = freqz(b,a,w)
```

- ✓ 주어진 각 주파수 w 에 대한 주파수 특성 값을 돌려준다.

```
Hm = abs(H)
```

```
Hp = angle(H)
```

- ✓ 크기 응답과 위상 응답

```
[H,w] = freqz(b,a,n)
```

```
[H,w] = freqz(b,a,n,'whole')
```

- ✓ 반 주기($0 \leq \omega \leq \pi$) 동안 n 점에서의 주파수와 특성 값을 돌려준다.
- ✓ 한 주기($0 \leq \omega \leq 2\pi$) 동안 n 점에서의 주파수와 특성 값을 돌려준다.
- ✓ n 은 생략할 수 있다. 생략하면 $n=512$ 가 된다.

```
H = freqz(b,a,f,fs)
```

- ✓ 샘플링 주파수 fs 로 샘플링된 신호를 필터링하도록 설계된 디지털 시스템에 대해 주어진 주파수 f 에 대한 주파수 특성 값을 돌려준다.
- ✓ 샘플링 주파수 fs 는 각 주파수 $\omega=2\pi$ 에 대응한다.

```
[H,f] = freqz(b,a,n,fs)
```

```
[H,f] = freqz(b,a,n,'whole',fs)
```

- ✓ 샘플링 주파수 fs 로 샘플링된 신호를 필터링하도록 설계된 디지털 시스템에 대해 $0 \sim fs/2$ 사이에 있는 n 점에서의 주파수와 특성 값을 돌려준다. (반 주기)
- ✓ 샘플링 주파수 fs 로 샘플링된 신호를 필터링하도록 설계된 디지털 시스템에 대해 $0 \sim fs$ 사이에 있는 n 점에서의 주파수와 특성 값을 돌려준다. (한 주기)

▶ 디지털 저역 통과 필터

$$H(z) = \frac{0.3 + 0.3z^{-1} + 0.1z^{-2}}{1 - 0.5z^{-1} + 0.2z^{-2}}$$

- ✓ 프로그램 1

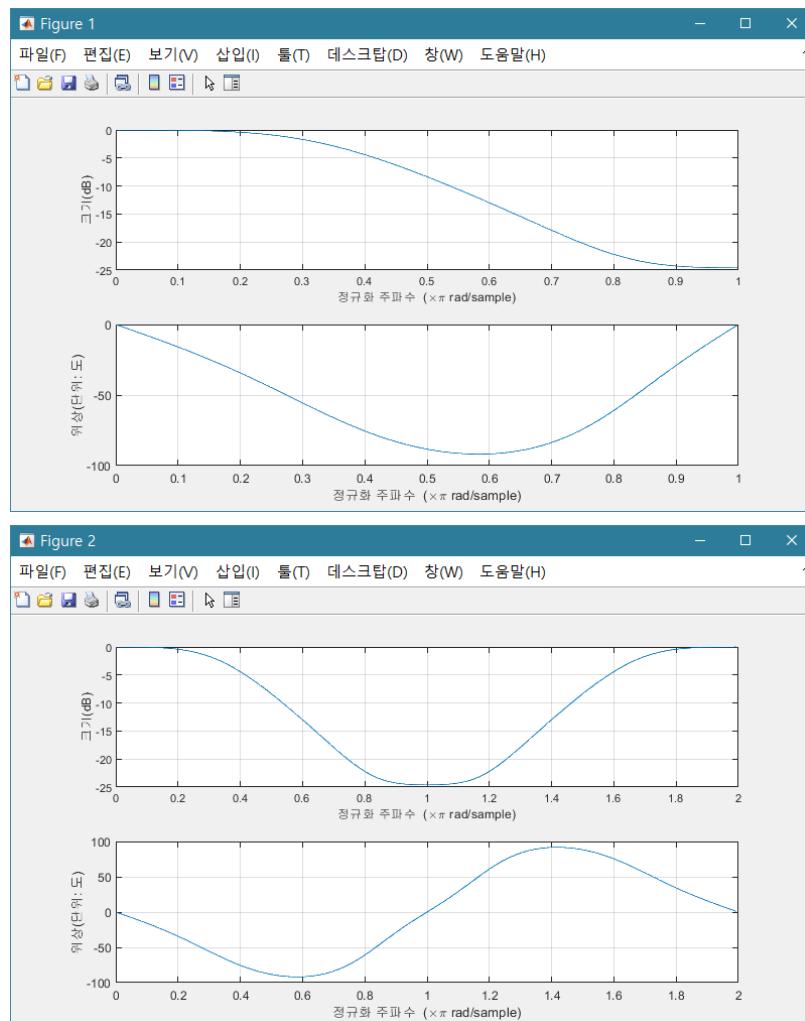
```

clc; close all;

b = [0.3 0.3 0.1];
a = [1 -0.5 0.2];
figure(1)
freqz(b,a) % 반 주기 표시
grid on

figure(2)
freqz(b,a,'whole') % 한 주기 표시
grid on

```



✖ 각 주파수 ω 가 ω/π 로 정규화되어 표시된다.

✓ 프로그램 2

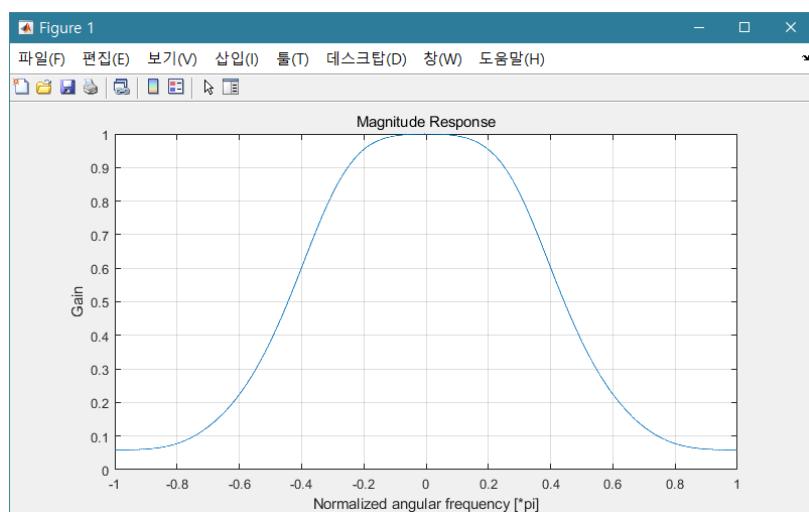
```

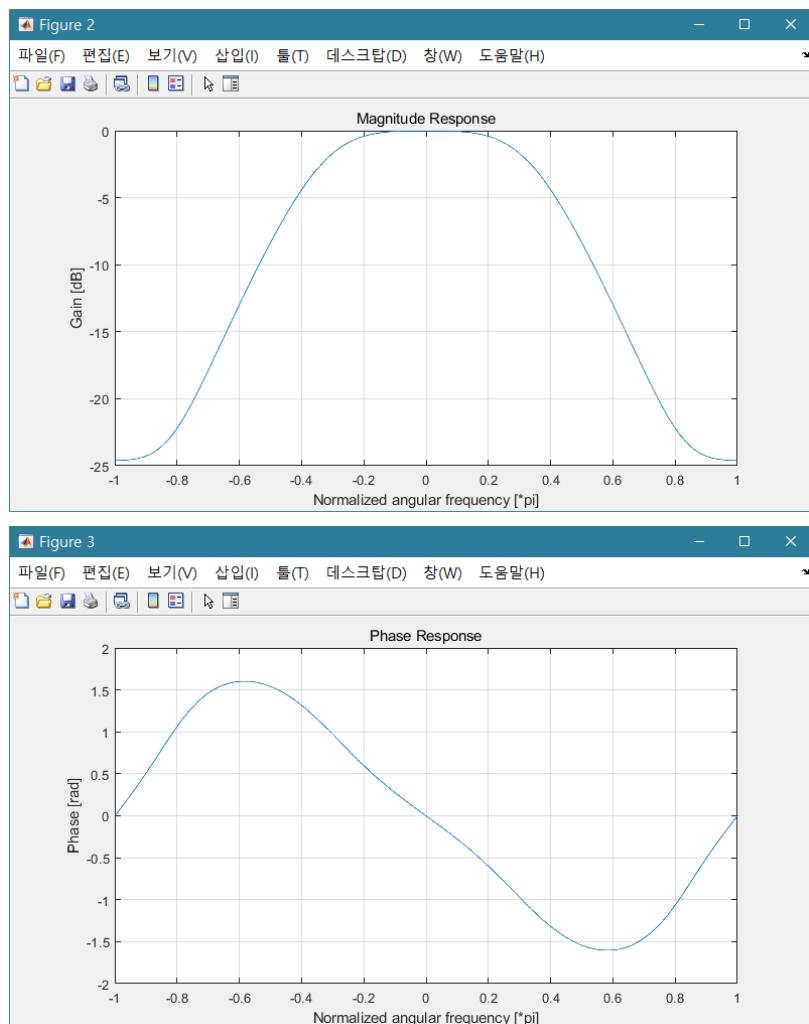
clc; close all;

b = [0.3 0.3 0.1];
a = [1 -0.5 0.2];

```

```
w = linspace(-pi,pi,1001); % 한 주기 주파수
H = freqz(b,a,w);
Hm = abs(H);
Hm_dB = 20*log10(Hm);
Hp = angle(H);
figure(1)
plot(w/pi,Hm) % 크기 특성
grid on
title('Magnitude Response')
xlabel('Normalized angular frequency [*pi]')
ylabel('Gain')
figure(2)
plot(w/pi,Hm_dB) % 크기(dB) 특성
grid on
title('Magnitude Response')
xlabel('Normalized angular frequency [*pi]')
ylabel('Gain [dB]')
figure(3)
plot(w/pi,Hp) % 위상 특성
grid on
title('Phase Response')
xlabel('Normalized angular frequency [*pi]')
ylabel('Phase [rad]')
```





- ✖ 각 주파수 ω 가 ω/π 로 정규화 되어 표시한다.
- ✖ 한 주기($-\pi \leq \omega \leq \pi$) 동안 표시한다.
- ✓ 프로그램 3
 - ✖ 주어진 디지털 필터가 샘플링 주파수 $f_s = 1\text{ kHz}$ 로 샘플링된 신호를 필터링하도록 설계된 저역 통과 필터일 때의 주파수 특성

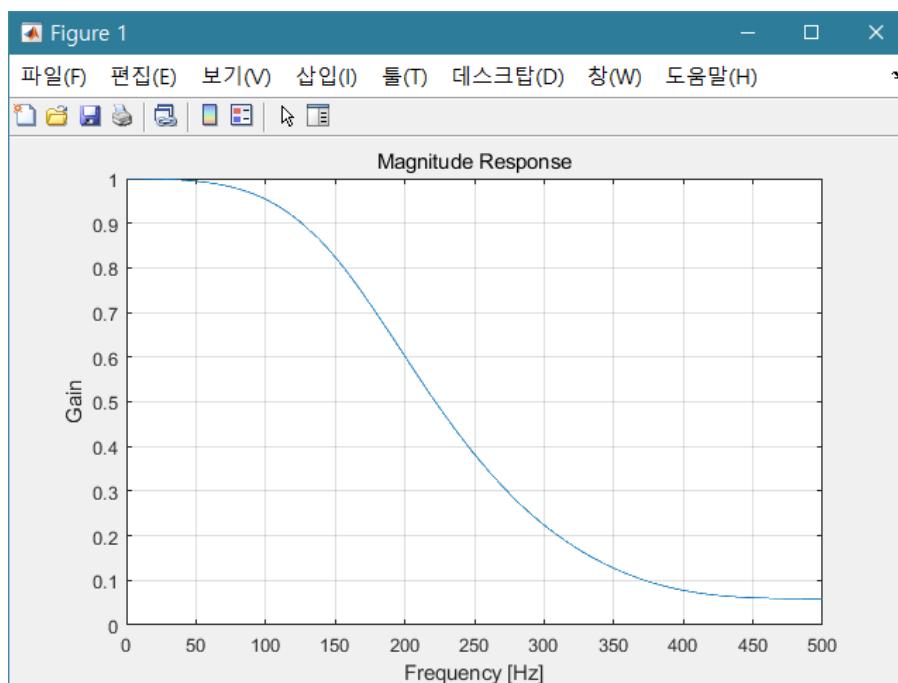
```

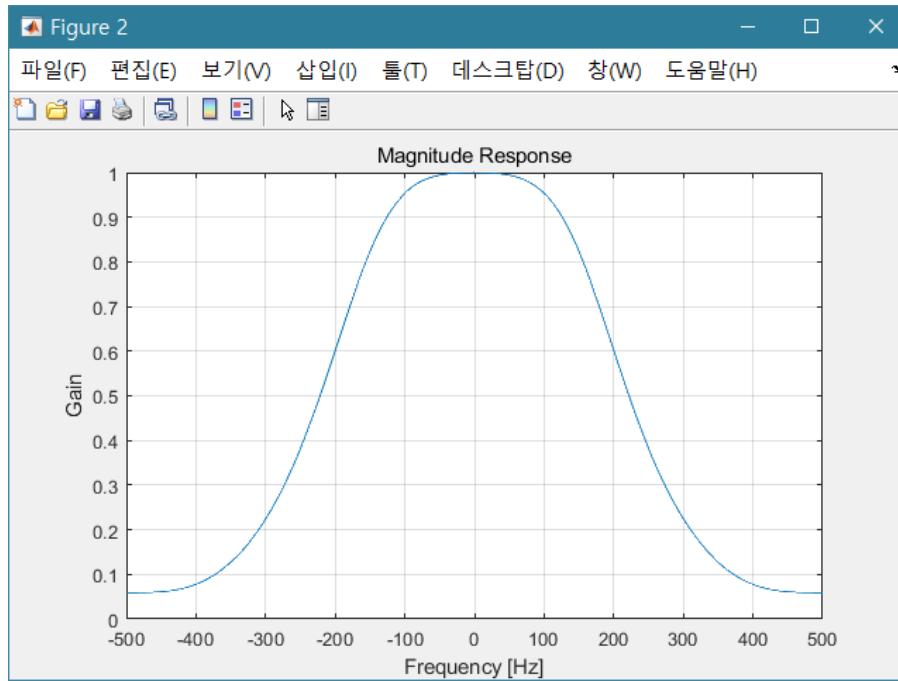
clc; clear; close all;

% Digital system
b = [0.3 0.3 0.1];
a = [1 -0.5 0.2];
fs = 1000;
% Calculate and plot frequency response during 0 ~ fs/2
f = linspace(0,fs/2,1001);
H = freqz(b,a,f,fs);
Hm = abs(H);

```

```
figure(1)
plot(f,Hm)
grid on
title('Magnitude Response')
xlabel('Frequency [Hz]')
ylabel('Gain')
% Calculate and plot frequency response during -fs/2 ~ fs/2
f = linspace(-fs/2,fs/2,1001);
H = freqz(b,a,f,fs);
Hm = abs(H);
figure(2)
plot(f,Hm)
grid on
title('Magnitude Response')
xlabel('Frequency [Hz]')
ylabel('Gain')
```





✓ 프로그램 4

- ✗ 주어진 디지털 필터가 샘플링 주파수 $f_s = 1\text{ kHz}$ 로 샘플링된 신호를 필터링하도록 설계된 저역 통과 필터일 때의 주파수 특성

```

clc; clear; close all;

% Digital system

b = [0.3 0.3 0.1];
a = [1 -0.5 0.2];
fs = 1000;

% calculate and plot frequency response during 0 ~ fs/2
[H,f] = freqz(b,a,1001,fs);
Hm = abs(H);

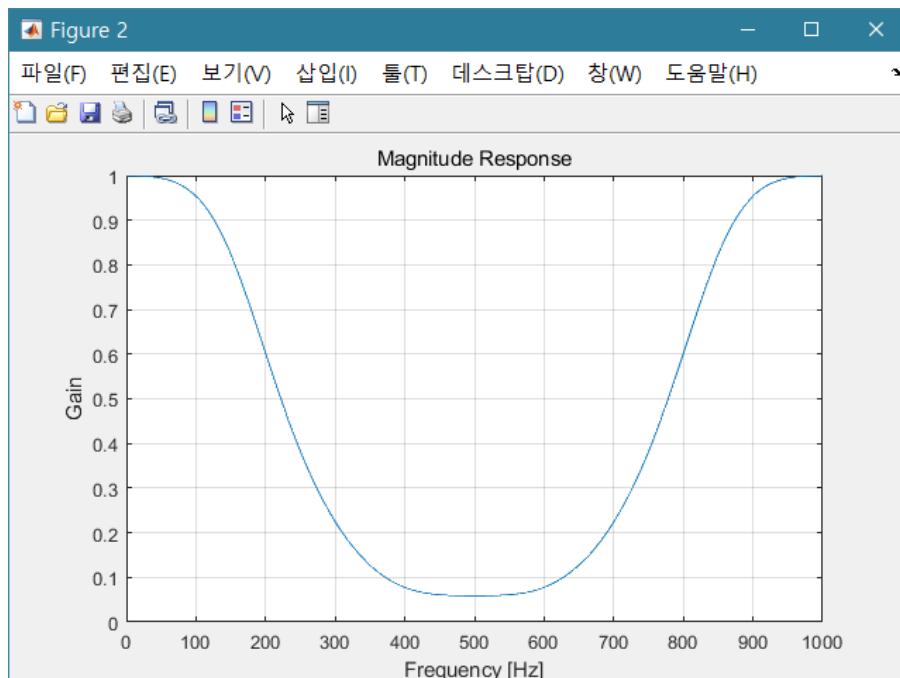
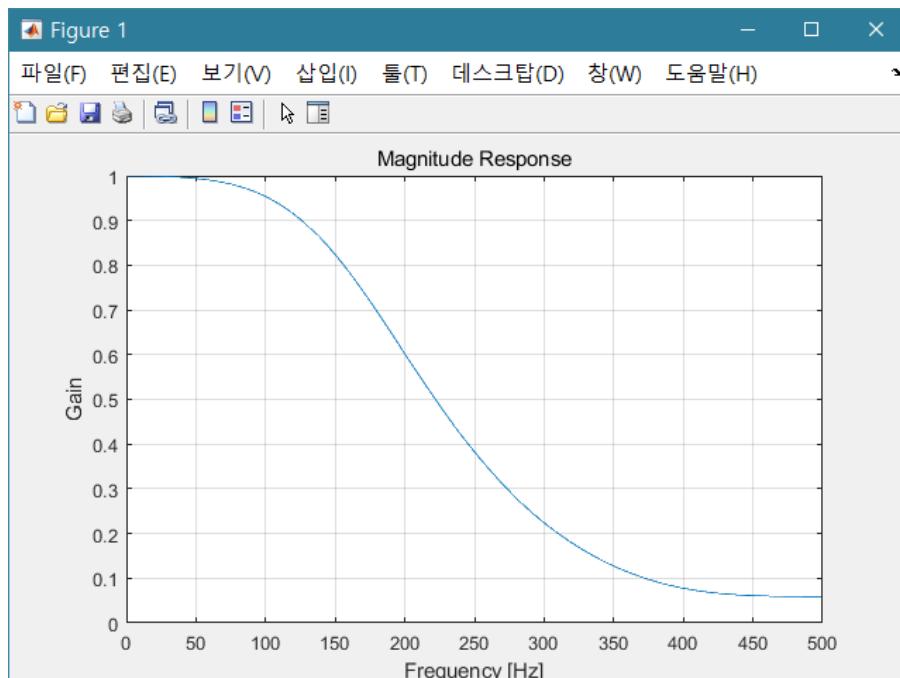
figure(1)
plot(f,Hm)
grid on
title('Magnitude Response')
xlabel('Frequency [Hz]')
ylabel('Gain')

% Calculate and plot frequency response during 0 ~ fs
[H,f] = freqz(b,a,1001,'whole',fs);
Hm = abs(H);

figure(2)

```

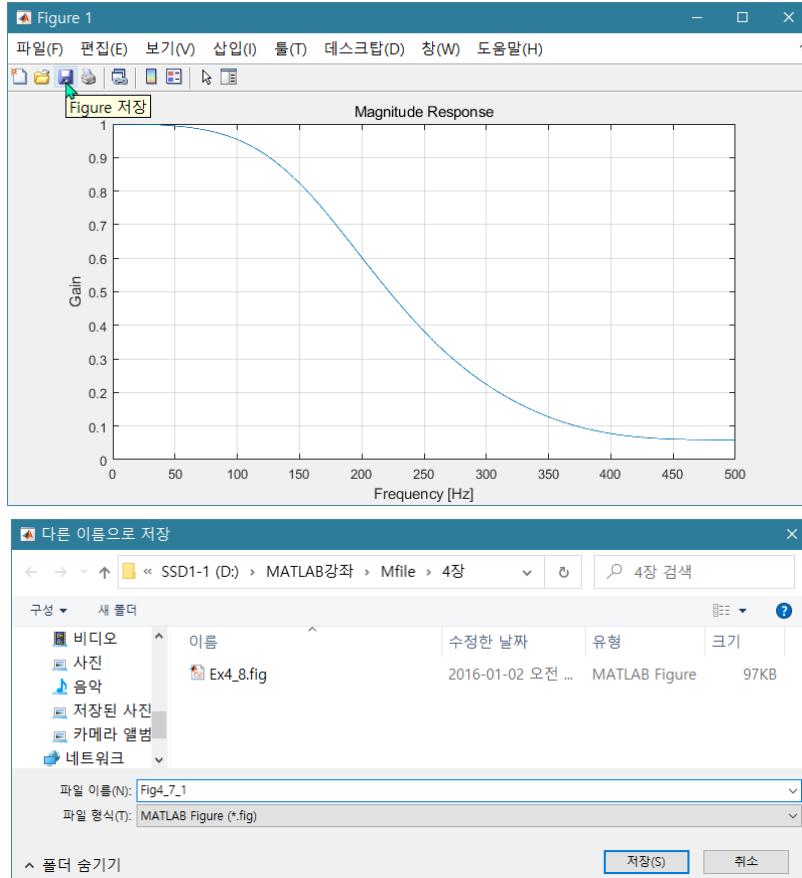
```
plot(f,Hm)
grid on
title('Magnitude Response')
xlabel('Frequency [Hz]')
ylabel('Gain')
```



4.7 그림의 저장 및 읽기

4.7.1 그림 저장

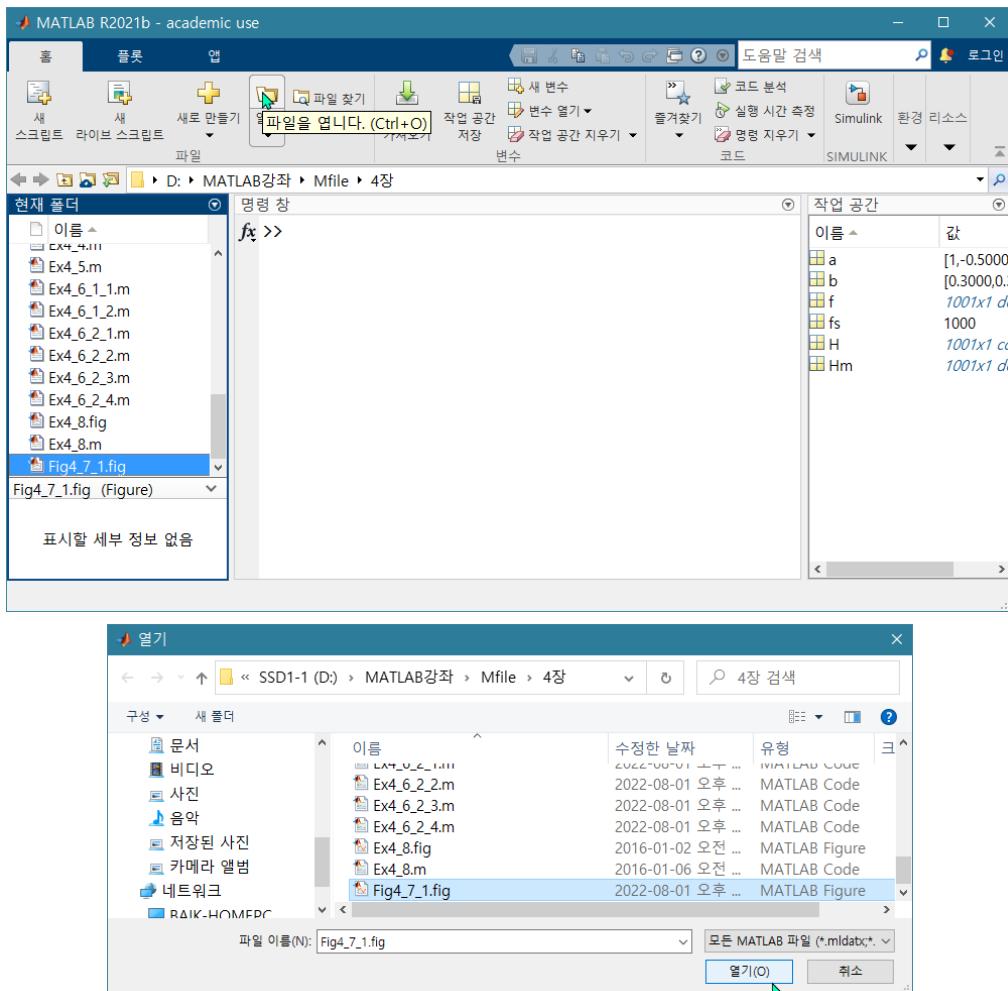
저장 아이콘을 클릭한다.



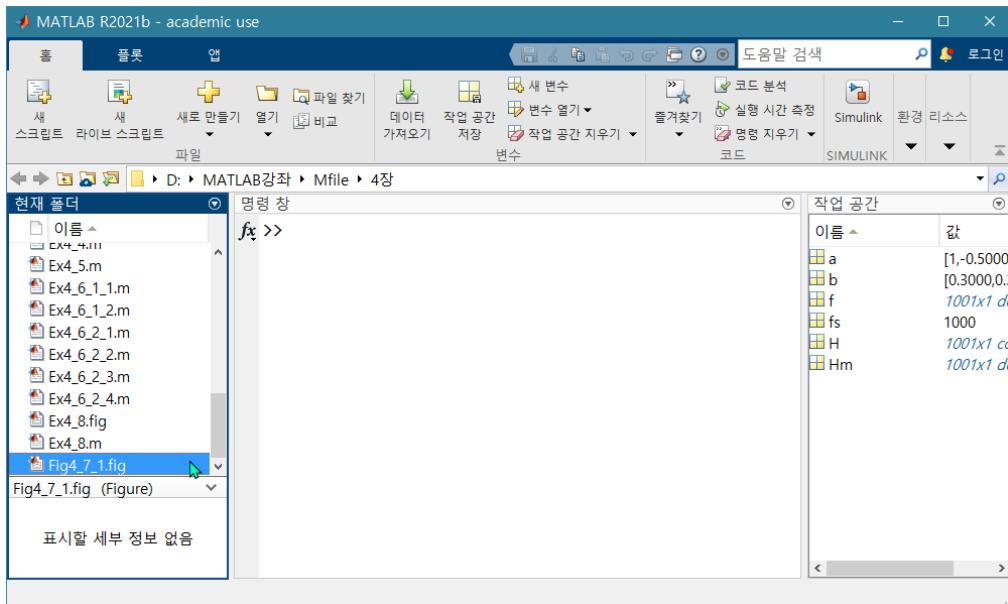
- ✓ 파일 형식을 선택하면 여러 종류의 그림으로 저장할 수 있다.
- ✗ MATLAB 자체 형식 : fig
- ✗ 기타 형식 : bmp, emf, jpg, png, ...

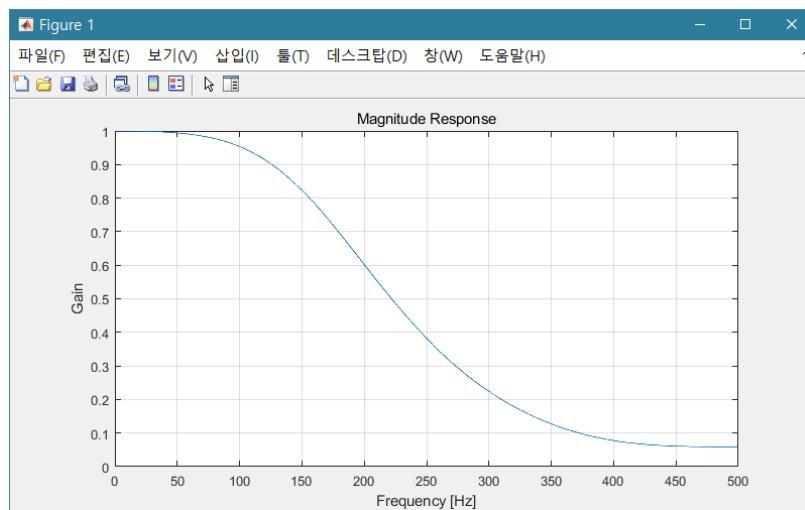
4.7.2 그림 읽기

파일 열기 아이콘을 클릭하여 그림 파일을 불러온다.



현재 폴더에서 해당 파일을 더블 클릭하여 그림 파일을 연다.

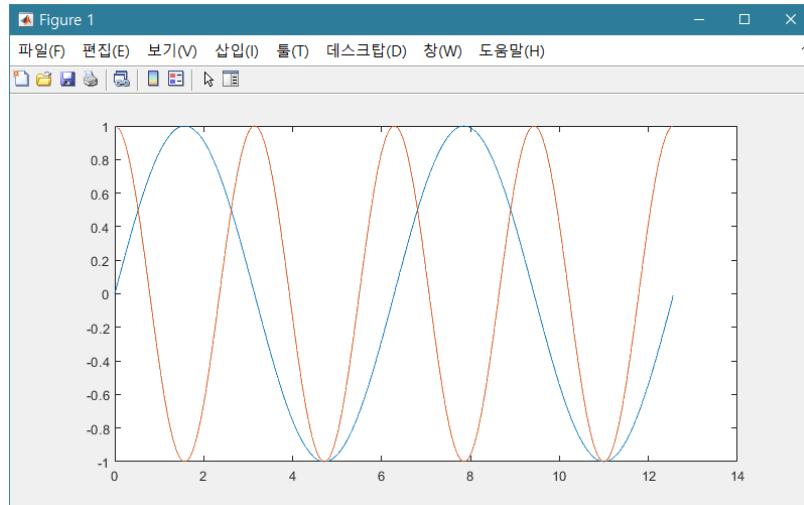




4.8 그림 편집

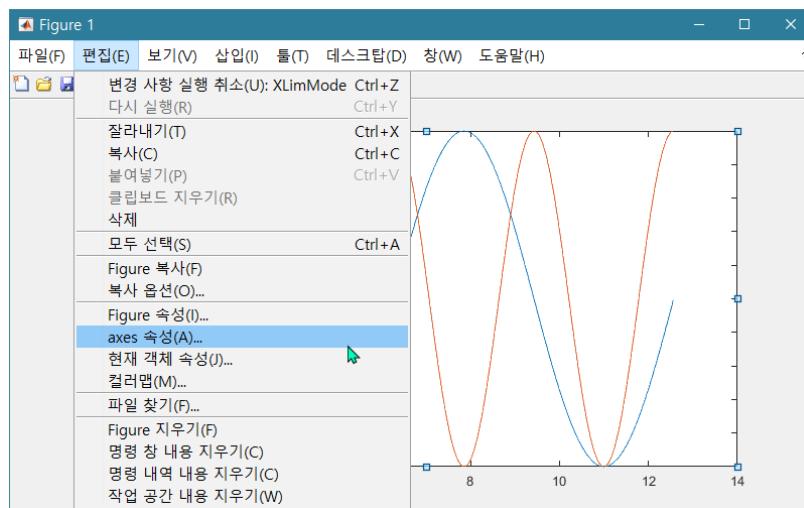
그림 작성

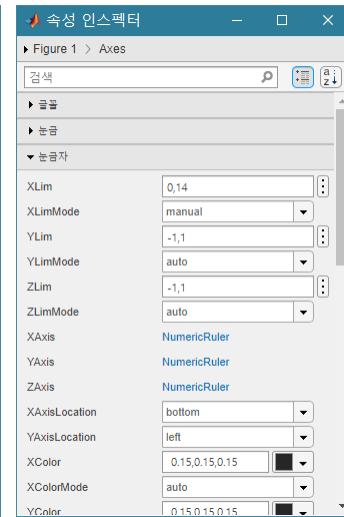
```
clc; close all;
figure
x = 0:0.01:4*pi;
y1 = sin(x);
y2 = cos(2*x);
plot(x,y1,x,y2)
```



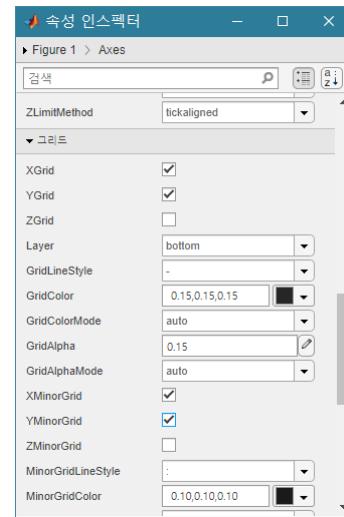
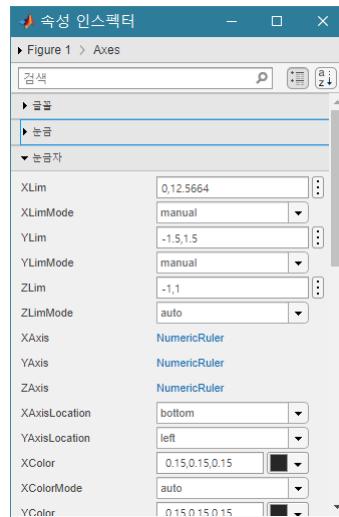
4.8.1 좌표 축 속성 편집

▶ 편집 – axes 속성 메뉴를 선택하여 속성 인스펙터 창을 연다.

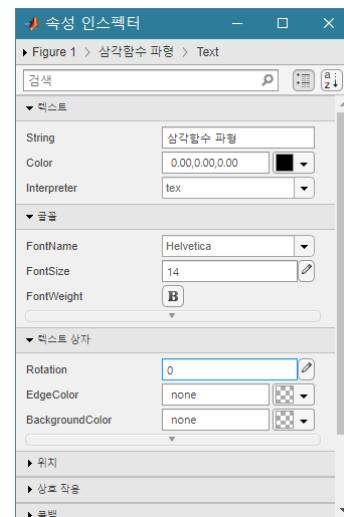
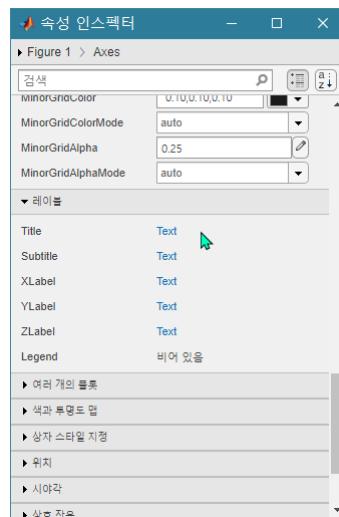


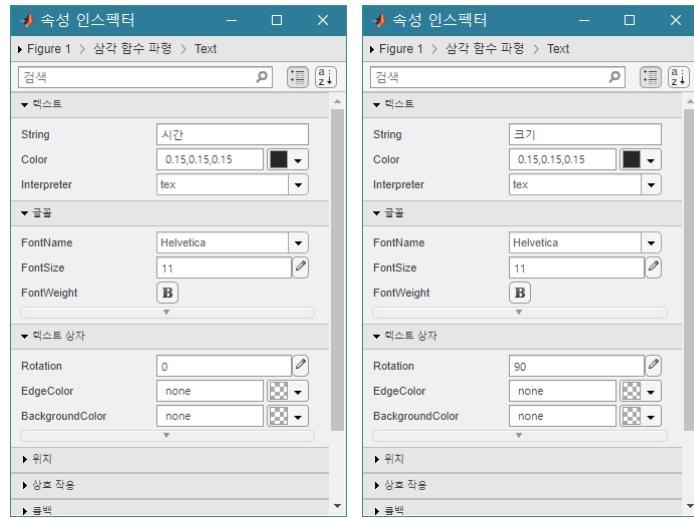


✓ x 축, y 축 범위 변경, 눈금 설정

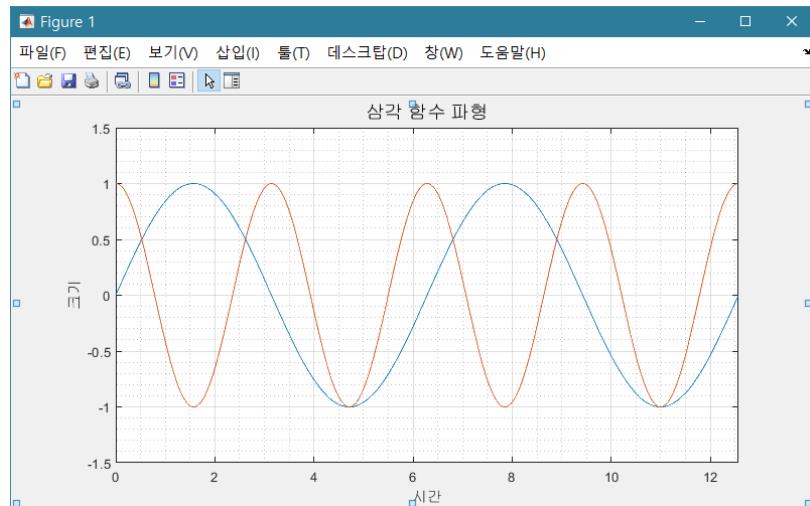


✓ Title, xlabel, ylabel 설정

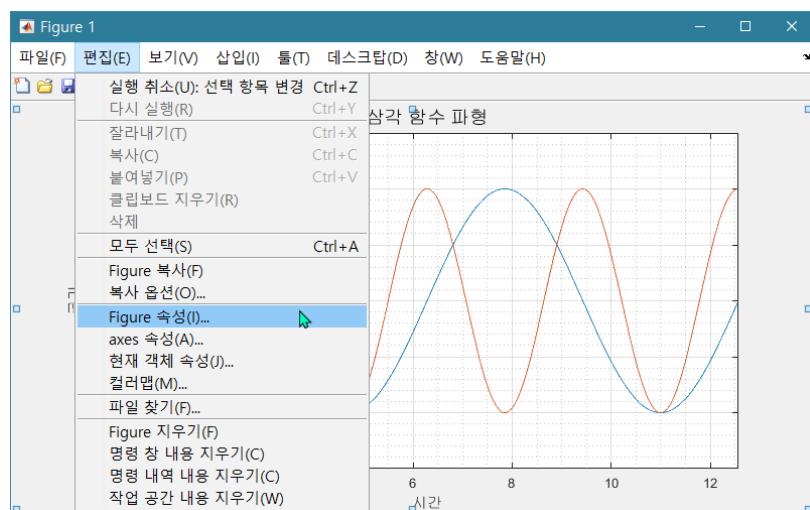


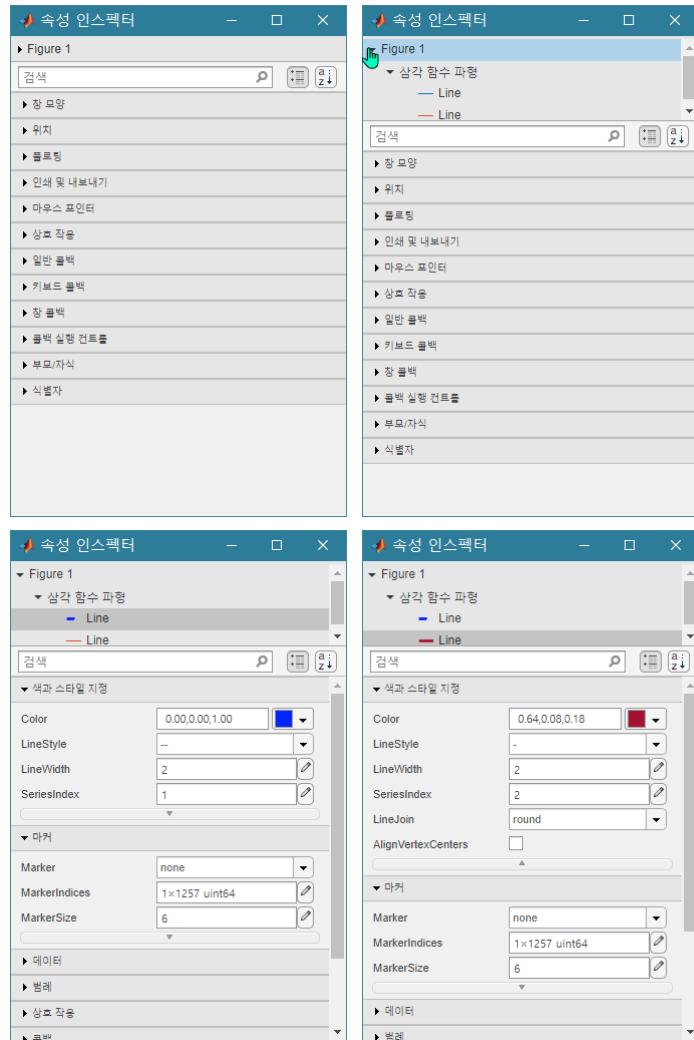


✓ 편집 결과 그림

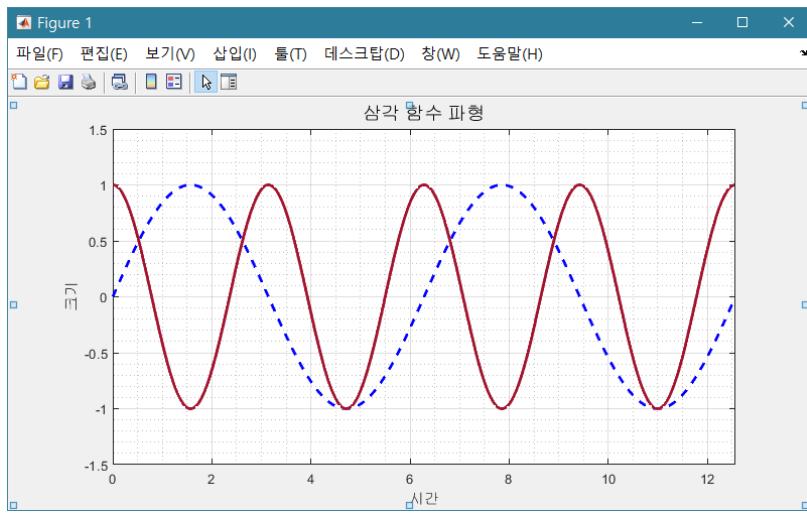


➊ 편집 – Figure 속성 메뉴를 선택하여 속성 인스펙터 창을 연다.



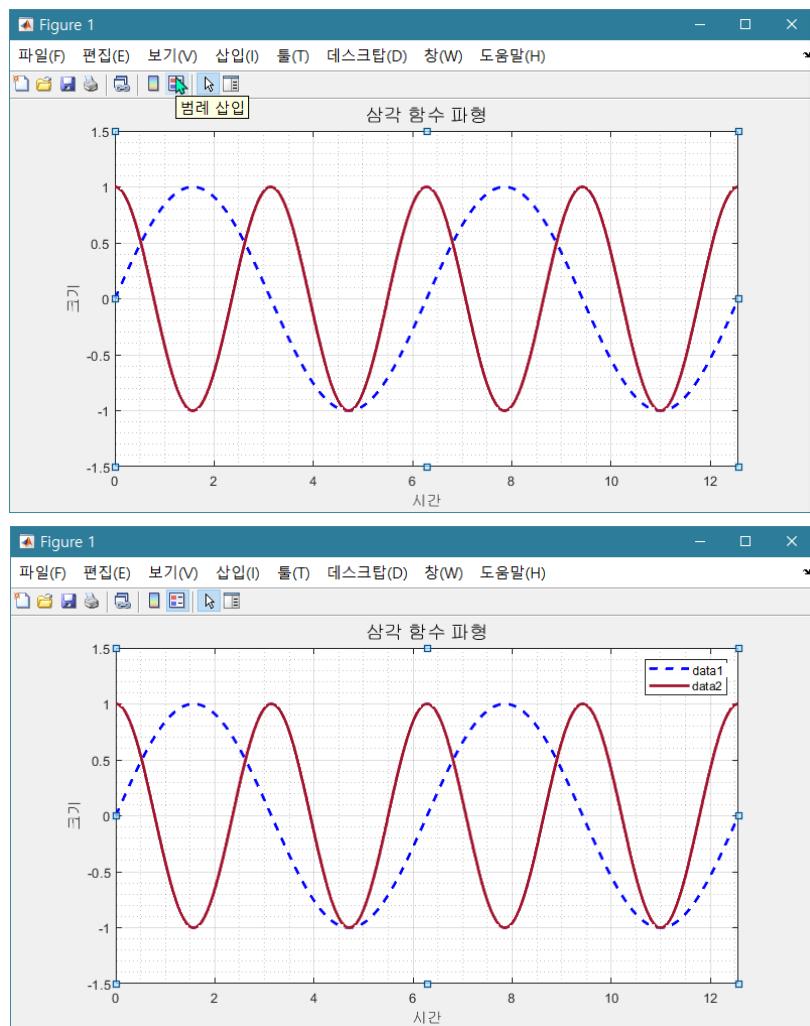


✓ 편집 결과 그림

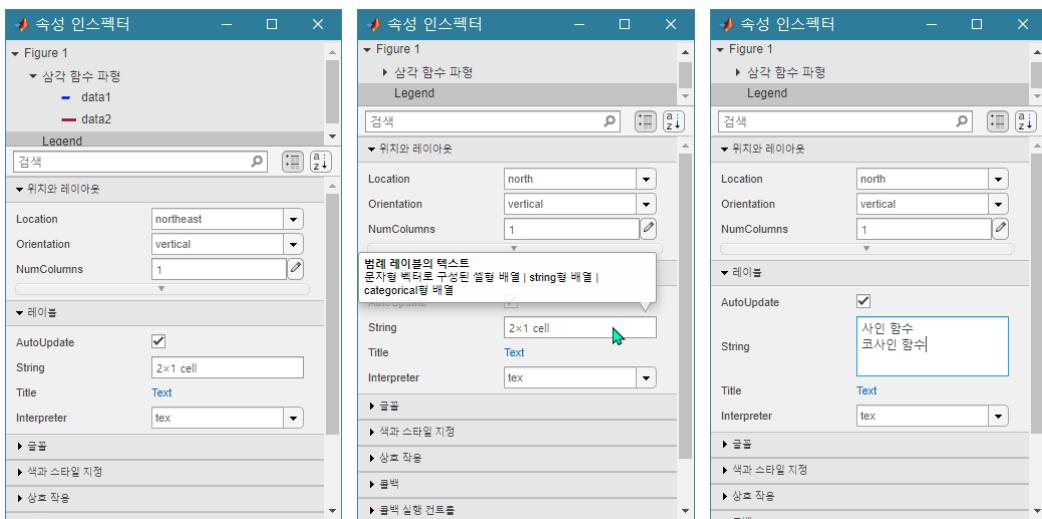


❖ 범례(legend) 삽입 및 편집

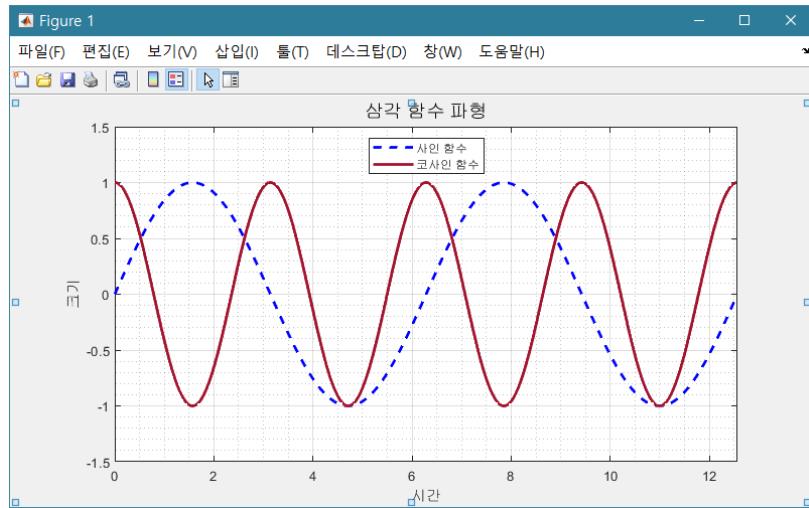
- ✓ 범례 삽입 아이콘을 클릭하여 범례를 삽입한다.



- ✓ 범례 상자를 더블 클릭하여 속성 인스펙터 창을 연다.



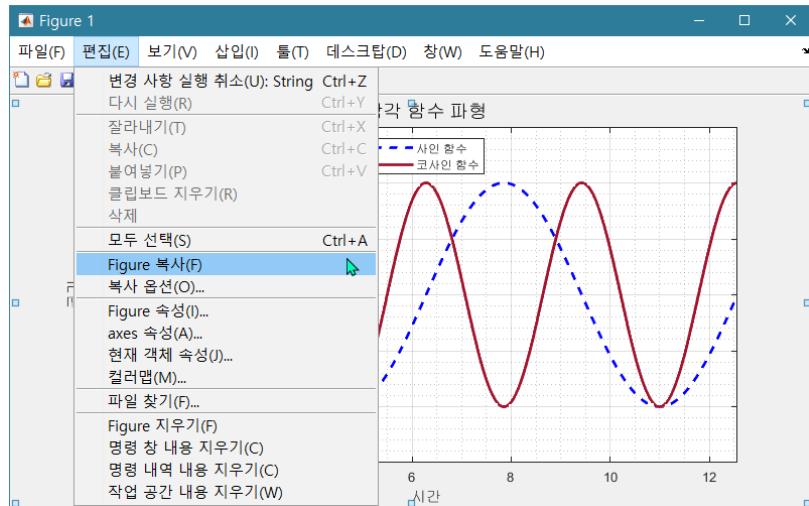
- ✓ 편집 결과 그림



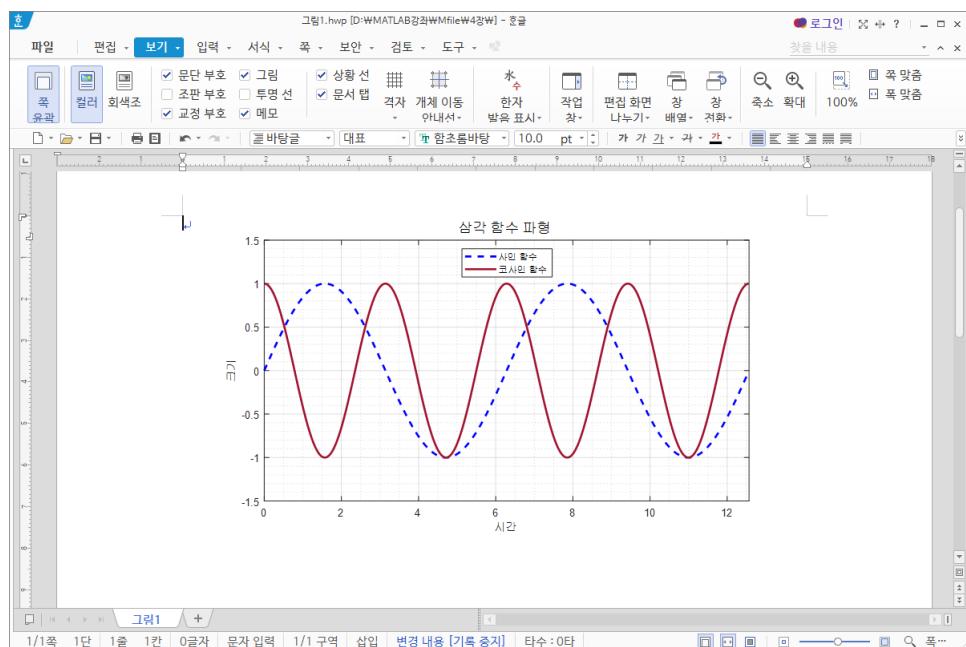
- ▶ 프로그램이 아닌 수동으로 그림을 편집할 경우 그림을 fig 확장자로 별도로 저장해야 나중에 다시 사용할 수 있다.

4.9 그림 붙여 넣기

- 편집 – Figure 복사 메뉴를 선택한다.



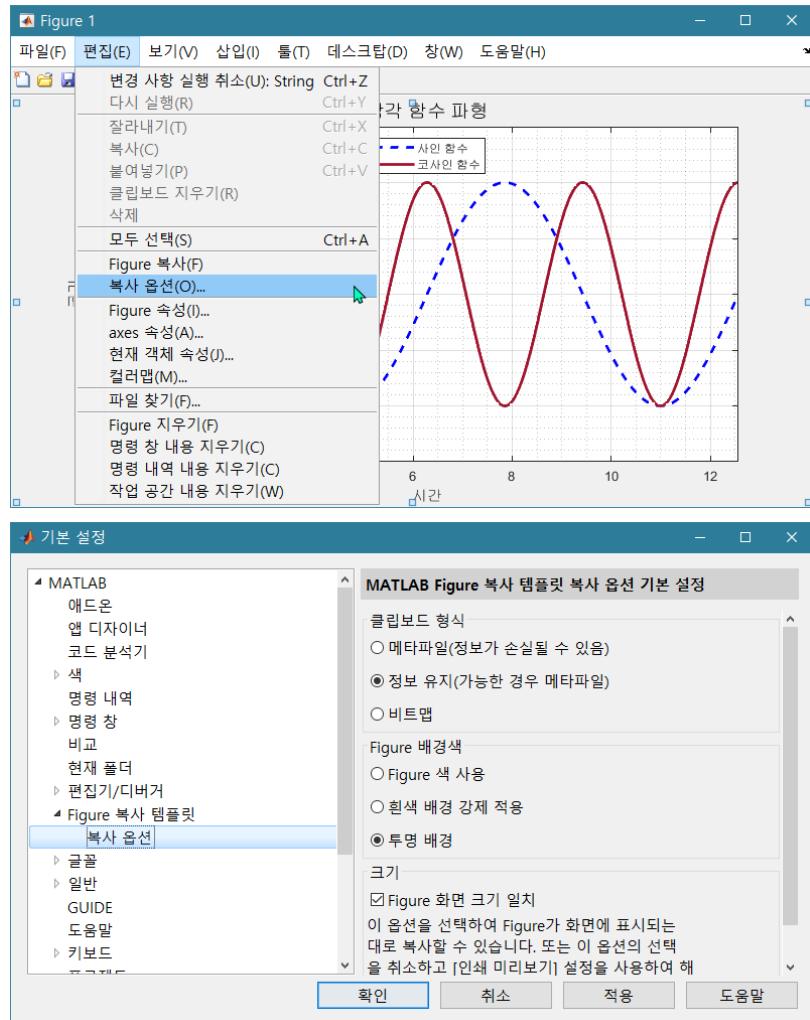
- 워드나 한글 파일에 붙여 넣기(Ctrl-V)한 후 크기를 조절한다.



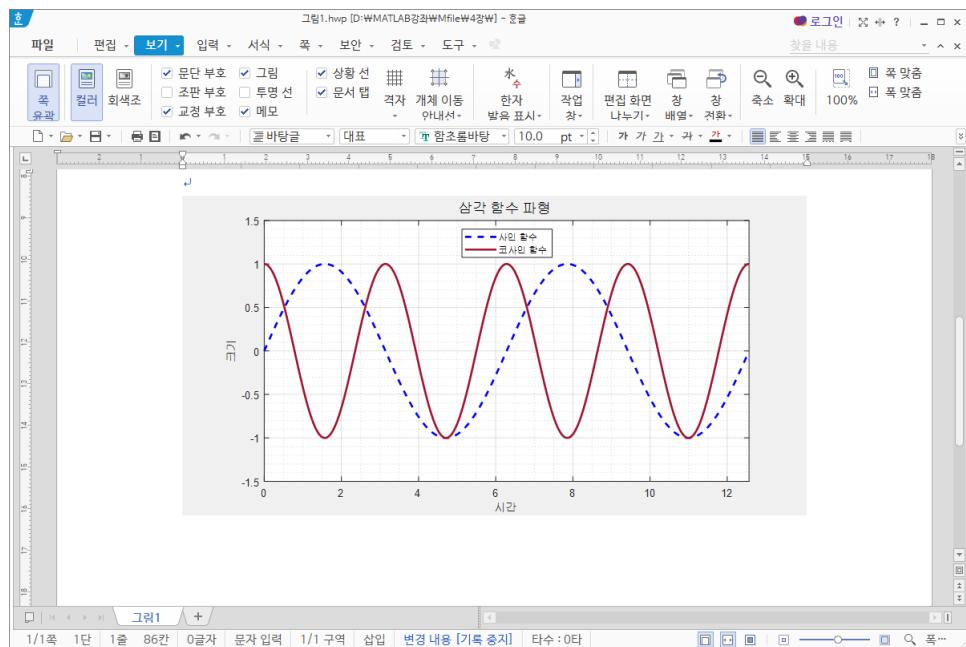
- 한글에서는 그림의 개체 속성에서 위치를 글자처럼 취급하는 게 편하다.
(스타일 적용하기에 좋다.)

- 복사 옵션

- 편집 – 복사 옵션 메뉴를 선택하여 기본 설정(복사 옵션) 창을 연다.



- ✓ Figure 배경색 – Figure 색 사용을 선택했을 때의 그림



4.10 연습 문제

- ▶ 문제 1. 실험에서 증폭기에 $V_i = 10 \text{ mV}$ 의 입력을 가했을 때 다음과 같이 출력 전압을 얻었다. 이를 이용하여 증폭기의 전압 이득 $A_v = V_o / V_i$ 의 주파수 특성을 그려라. 단 주파수 축은 로그 눈금을 사용하고 이득은 데시벨(dB)로 변환하라.

$f [\text{Hz}]$	10	20	40	70	100	200	400	700	1000	2000	4000	7000	10000
V_o	100	100	100	99	99	98	93	82	71	45	24	14	10

- ▶ 문제 2. 다음과 같은 타원과 타원체를 그려라.

$$x^2 + \frac{y^2}{4} = 1, \quad x^2 + \frac{y^2}{4} + \frac{z^2}{9} = 1$$

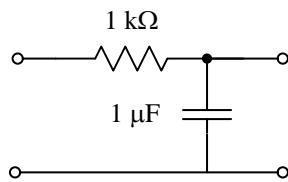
- ▶ 문제 3. 다음과 같은 전달 함수의 크기와 위상을 구간 $-\pi \leq \omega \leq \pi$ 동안 그려라.

$$H(e^{j\omega}) = \frac{1}{1 - 0.8e^{-j\omega}}$$

- ▶ 문제 4. 다음 식을 구간 $-20 \leq x \leq 20$ 동안 그래프로 그려라.

$$y = \frac{\sin x}{x}$$

- ▶ 문제 5. 다음과 같은 회로망의 전달 함수의 차단 주파수를 구하고 주파수 응답 (크기와 위상)을 그려라.



- ▶ 문제 6. 문제 5에 주어진 회로망의 주파수 응답을 freqs 함수를 이용하여 그려라.

- ▶ 문제 7. 다음과 같이 전달 함수가 주어지는 디지털 시스템의 주파수 응답 (크기와 위상)과 극-영점도를 그려라.

$$H(z) = \frac{0.05634(1+z^{-1})(1-1.0166z^{-1}+z^{-2})}{(1-0.683z^{-1})(1-1.4461z^{-1}+0.7957z^{-2})}$$

5. 파일 입출력(File I/O)

5.1 High-level 파일 입출력

5.1.1 데이터 저장하기

✚ save 함수

save filename

- ✓ Workspace 내의 모든 변수를 filename에 저장한다.

save filename variables

- ✓ Workspace 내의 변수 variable을 filename에 저장한다.
- ✓ 변수 명에 wildcard(*)도 가능하다.

save filename variables option

- ✓ Workspace 내의 변수 variable을 option format으로 filename에 저장한다.

✚ Option

Options	Result : How Data is Stored
-ascii	8-digit ASCII format
-ascii -double	16-digit ASCII format
-ascii -tabs	delimits with tabs
-ascii -double -tabs	16-digit ASCII format, tab delimited
-mat	Binary MAT-file form (default)
-v6 (4)	A format that MATLAB version 6(4) can open
-append	The specified existed MAT-file, appended to the end

5.1.2 데이터 불러 오기

load 함수

```
load filename
```

- ✓ filename에 저장된 모든 변수를 불러들인다.

```
load filename variables
```

- ✓ filename에 저장된 변수 variable을 불러들인다.

모든 데이터 저장

```
>> clc; clear
>> x1 = 5; x2 = 10; y = 15;
>> save data % data.mat에 모든 데이터 저장
>> save data.txt -ascii % data.txt에 ASCII로 저장
>> clear all
>> load data
>> whos
```

Name	Size	Bytes	Class	Attribute
x1	1x1	8	double	
x2	1x1	8	double	
y	1x1	8	double	

지정한 데이터만 저장

```
>> clc; clear
>> x1 = 5; x2 = 10; y = 15;
>> save datax x* % x로 시작하는 변수 저장
>> save datax.txt x* -ascii % x로 시작하는 변수를 ASCII로 저장
>> clear all
>> load datax
>> whos
```

Name	Size	Bytes	Class	Attribute
x1	1x1	8	double	
x2	1x1	8	double	

5.2 Low-level 파일 입출력

5.2.1 Low-level 파일 입출력 함수

▣ Open/close file

- ✓ fopen()
- ✓ fclose()

▣ Read/write ASCII data

- ✓ fscanf()
- ✓ fprintf()

▣ Read/write binary data

- ✓ fread()
- ✓ fwrite()

▣ Read/write text data

- ✓ fgets()
- ✓ fgetl()
- ✓ fileread()
- ✓ fprintf()

▣ Position

- ✓ ftell()
- ✓ fseek()
- ✓rewind()
- ✓ feof()

▣ fopen 함수

```
fid = fopen(filename)
```

- ✓ filename 의 파일을 연다.
- ✓ fid 는 file identifier 로 불리는 정수(integer)이다.
- ✓ 파일을 열 수 없으면 fid=-1 이다.

```
fid = fopen(filename,permission)
```

- ✓ permission 으로 주어지는 모드로 filename 의 파일을 연다.

```
[fid,message] = fopen(filename,permission)
```

- ✓ 파일을 열 수 없을 때 fid 는 -1 이 되고, message 에 에러 메시지를 담는다.

➊ Permission

'r'	Read (default).
'w'	Write (create if necessary)
'a'	Append (create if necessary)
'r+'	Read and write (do not create)
'w+'	Truncate or create for read and write
'a+'	Read and append (create if necessary)
'A'	Append without automatic flushing
'W'	Write without automatic flushing

➋ fclose 함수

```
status = fclose(fid)
```

- ✓ fid 에 해당하는 파일을 닫는다.
- ✓ 성공하면 status=0, 실패하면 status=-1 이다.

```
status = fclose('all')
```

- ✓ 모든 파일을 닫는다.
- ✓ 성공하면 status=0, 실패하면 status=-1 이다.

➌ fprintf 함수

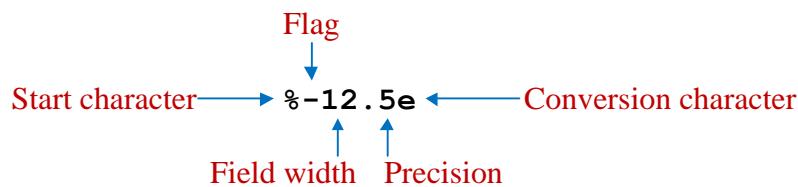
```
count = fprintf(fid,format,A,...)
```

- ✓ 행렬 A 의 실수 부분 데이터를 format 문자열에 맞는 format 으로 fid 에 해당하는 파일에 쓴다.
- ✓ 쓰여진 바이트의 개수를 count 로 돌려 준다.

➍ Format 문자열

- ✓ C 언어 format 문자열과 유사하다.

- ✓ 새로운 라인, 탭과 같은 인쇄가 되지 않는 escape 문자를 포함할 수 있다.
- ✓ format 문자열은 %로 시작하며 다음과 같은 것을 포함할 수 있다.
 - ✗ Flags (optional)
 - ✗ Width and precision fields (optional)
 - ✗ A subtype specifier (optional)
 - ✗ Conversion character (required)
- ✓ format 문자열은 다음과 같은 순서로 써야 한다.



Flag

-	Left-justifies the converted argument in its field.
+	Always prints a sign character (+ or -).
0	Pad with zeros rather than spaces.

변환 문자(conversion character)

%c	Single character
%s	String of characters
%d, %i	Decimal notation (signed)
%f	Fixed-point notation
%e, %E	Exponential notation
%u	Decimal notation (unsigned)
%o	Octal notation (unsigned)
%x, %X	Hexadecimal notation

Escape 문자

\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\\\	Backslash
\' , ''	Single quotation mark
%%	Percent character

fscanf 함수

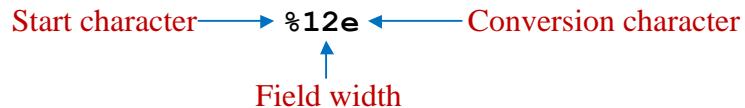
```
A = fscanf(fid,format)
```

- ✓ fid 에 해당하는 파일에서 모든 데이터를 읽고 format 문자열에 해당하는 format 으로 변환한 후 행렬 A 에 할당한다.
- ✓ format 은 읽을 데이터의 format 을 규정하는 문자열이다.

```
[A,count] = fscanf(fid,format,size)
```

- ✓ fid 에 해당하는 파일에서 size 만큼 데이터를 읽고 format 문자열에 해당하는 format 으로 변환한 후 행렬 A 에 할당하고 데이터 수를 count 에 할당한다.
- ✓ size 는 읽을 데이터의 개수를 나타내는 변수이다.

Format 문자열



▶ 변환 문자(conversion character)

%c	Sequence of characters; number specified by field width
%s	A series of non-white-space characters
%d	Decimal numbers
%e, %f, %g	Floating-point numbers
%i	Signed integer
%u	Signed decimal integer
%o	Signed octal integer
%x	Signed hexadecimal integer

▶ fwrite 함수

```
count = fwrite(fid,A,precision)
```

- ✓ fid에 해당하는 파일에 precision으로 규정된 정밀도로 행렬 A의 데이터를 쓴다.
- ✓ 데이터는 열 우선으로 쓰여진다.
- ✓ 성공적으로 쓰여진 데이터의 개수를 count에 돌려 준다.

▶ Precision

MATLAB	C, FORTRAN	Interpretation	MATLAB	C, FORTRAN	Interpretation
'schar'	'signed char'	Signed character, 8 bits	'int8'	'integer*1'	Integer, 8 bits
'uchar'	'unsigned char'	Unsigned character, 8 bits	'int16'	'integer*2'	Integer, 16 bits
'int8'	'integer*1'	Integer, 8 bits	'int32'	'integer*4'	Integer, 32 bits
'int16'	'integer*2'	Integer, 16 bits	'int64'	'integer*8'	Integer, 64 bits
'int32'	'integer*4'	Integer, 32 bits	'float32'	'real*4'	Floating-point, 32 bits
'int64'	'integer*8'	Integer, 64 bits	'float64'	'real*8'	Floating-point, 32 bits
			'double'	'real*8'	Floating-point, 64 bits

✚ fread 함수

```
A = fread(fid)
A = fread(fid,size)
A = fread(fid,size,precision)
[A,count] = fread(fid,size,precision)
```

- ✓ fid에 해당하는 파일에서 이진 데이터를 읽고 행렬 A에 할당한다.
- ✓ 성공적으로 읽을 경우 count에 데이터 개수를 할당한다.
- ✓ size가 주어지면 size만큼 데이터를 읽고, 주어지지 않으면 파일 끝까지 데이터를 읽는다.
- ✓ 유효한 size 형태

n	Reads n elements into a column vector
inf	Reads to the end of the file
[m,n]	Reads elements to fill an m -by- n matrix, in column order

✚ fgets 함수

```
tline = fgets(fid)
```

- ✓ fid에 해당하는 파일로부터 파일 위치 지시기(file position indicator)가 가리키는 다음 라인을 읽고 tline에 할당한다.
- ✓ fgets이 파일 끝 지시기(end-of-file indicator)을 만나면 -1을 돌려 준다.
- ✓ fgets는 텍스트 파일에만 사용할 수 있다.
- ✓ tline에는 새 라인 문자(new line character)가 포함된다.

```
tline = fgets(fid,nchar)
```

- ✓ fid에 해당하는 파일로부터 파일 위치 지시기가 가리키는 다음 라인의 nchar개수 만큼의 문자를 읽고 tline에 할당한다.

✚ fgetl 함수

```
tline = fgetl(fid)
```

- ✓ fid에 해당하는 파일로부터 파일 위치 지시기가 가리키는 다음 라인을 읽고 tline에 할당한다.

- ✓ fgets 이 파일 끝 지시기를 만나면 -1 을 돌려 준다.
- ✓ fgets 는 텍스트 파일에만 사용할 수 있다.
- ✓ tline 에는 새 라인 문자(new line character)가 포함되지 않는다.

✚ Position

```
position = ftell(fid)
```

- ✓ fid 에 해당하는 파일의 파일 위치 지시기의 위치(정수)를 알려 준다.
- ✓ 만일 실패하면 -1 을 돌려 준다.

```
status = fseek(fid,offset,origin)
```

- ✓ fid 에 해당하는 파일의 파일 위치 지시기의 위치를 origin 에 대해 offset 바이트만큼 이동시킨다.

```
frewind(fid)
```

- ✓ fid 에 해당하는 파일의 파일 위치 지시기의 위치를 파일의 처음으로 되돌린다.

5.2.2 ASCII 데이터 읽고 쓰기

✚ ASCII 데이터 쓰기

- ✓ 정수 데이터 쓰기

```

clc; clear;
x = 1:10;

fid = fopen('data-ai1.txt','wt');
fprintf(fid,'%i',x);
fclose(fid);

fid = fopen('data-ai2.txt','wt');
fprintf(fid,'%i\t',x);
fclose(fid);

fid = fopen('data-ai3.txt','wt');

```

```
fprintf(fid, '%10i', x);  
fclose(fid);
```

- ✓ 실수 데이터 쓰기

```
clc; clear;  
x = randn(1,10);  
  
fid = fopen('data-ar1.txt','wt');  
fprintf(fid, '%15.5f', x);  
fclose(fid);  
  
fid = fopen('data-ar2.txt','wt');  
fprintf(fid, '%15.5e', x);  
fclose(fid);  
  
fid = fopen('data-ar3.txt','wt');  
fprintf(fid, '%15.5e\n', x);  
fclose(fid);  
  
fid = fopen('data-ar4.txt','wt');  
fprintf(fid, '%15.5e%15.5e\n', x);  
fclose(fid);
```

✚ ASCII 데이터 읽기

- ✓ 정수 데이터 읽기

```
clc; clear;  
fid = fopen('data-ai1.txt','rt');  
x1 = fscanf(fid, '%i') % check x1  
fclose(fid);  
  
fid = fopen('data-ai2.txt','rt');  
x2 = fscanf(fid, '%i')  
fclose(fid);
```

```
fid = fopen('data-ai3.txt','rt');
x3 = fscanf(fid,'%10i')
fclose(fid);
```

✓ 실수 데이터 읽기.

```
clc; clear;
fid = fopen('data-ar1.txt','rt');
y1 = fscanf(fid,'%15f')
fclose(fid);
```

```
fid = fopen('data-ar2.txt','rt');
y2 = fscanf(fid,'%15e')
fclose(fid);
```

```
fid = fopen('data-ar3.txt','rt');
y3 = fscanf(fid,'%15e')
fclose(fid);
```

```
fid = fopen('data-ar4.txt','rt');
y4 = fscanf(fid,'%15e')
fclose(fid);
```

✚ ASCII 행렬 데이터 쓰기

```
clc; clear;
x = [1 2 3 4; 5 6 7 8; 9 10 11 12];

fid = fopen('data-am1.txt','wt');
fprintf(fid,'%5i%5i%5i\n',x);
fclose(fid);
```

✚ ASCII 행렬 데이터 읽기

```
clc; clear;
```

```

fid = fopen('data-am1.txt','rt');
x1 = fscanf(fid,'%i',[3,inf])
fclose(fid);

fid = fopen('data-am1.txt','rt');
x2 = fscanf(fid,'%i',[3,4])
fclose(fid);

```

5.2.3 이진(binary) 데이터의 읽고 쓰기

- ✚ 이진(binary) 데이터 쓰기
 - ✓ 정수 데이터 쓰기
-

```

clc; clear;
x = 1:10;

fid = fopen('data-bi1.bin','wb');
fwrite(fid,x,'int16');
fclose(fid);

fid = fopen('data-bi2.bin','wb');
fwrite(fid,x,'int32');
fclose(fid);

```

- ✓ 실수 데이터 쓰기
-

```

clc; clear;
x = randn(1,10);

fid = fopen('data-br1.bin','wb');
fwrite(fid,x,'float32');
fclose(fid);

fid = fopen('bdata-br2.bin','wb');
fwrite(fid,x,'float64');

```

```
fclose(fid);
```

- ✚ 이진(binary) 데이터 읽기
 - ✓ 정수 데이터 읽기

```
clc; clear;  
fid = fopen('data-bi1.bin','rb');  
x1 = fread(fid,'int16')  
fclose(fid);  
  
fid = fopen('data-bi2.bin','rb');  
x2 = fread(fid,'int32')  
fclose(fid);  
  
fid = fopen('data-bi1.bin','rb');  
x3 = fread(fid,[2,5],'int16')  
fclose(fid);
```

- ✓ 실수 데이터 읽기

```
clc; clear;  
fid = fopen('data-br1.bin','rb');  
x1 = fread(fid,'float32')  
fclose(fid);  
  
fid = fopen('data-br2.bin','rb');  
x2 = fread(fid,'float64')  
fclose(fid);  
  
fid = fopen('data-br1.bin','rb');  
x3 = fread(fid,[5,2],'float32')  
fclose(fid);
```

5.2.4 텍스트(text) 데이터의 읽고 쓰기

 텍스트(text) 데이터 쓰기

```
clc; clear;

s1 = 'This is MATLAB education course.'
s2 = 'My name is Baik Heung-Ki.'

fid = fopen('data-t1.txt','wt');
fprintf(fid,'%s\n',s1);
fprintf(fid,'%s\n',s2);
fclose(fid);

fid = fopen('data-t2.txt','wt');
fprintf(fid,'%s\n%s',s1,s2);
fclose(fid);
```

 텍스트(text) 데이터 읽기

```
clc; clear;

fid = fopen('data-t1.txt','rt');
s1 = fgets(fid)
s2 = fgets(fid)
fclose(fid);

fid = fopen('data-t2.txt','rt');
tline = fgetl(fid);
while ischar(tline)
    disp(tline)
    tline = fgetl(fid);
end
fclose(fid);

text = fileread('data-t2.txt')
```

5.3 표준 파일 포맷(standard file format)

5.3.1 텍스트 파일 읽고 쓰기

- 콤마로 분리된 데이터(comma separated value: CSV) 읽고 쓰기

```
writematrix(A,filename)
```

- 변수 A의 데이터를 filename에 저장한다.
- filename의 확장자는 txt, dat, csv가 될 수 있다.
- Excel 스프레드 시트 파일로 저장하려면 확장자는 xls, xlsx가 되어야 한다.

```
A = readmatrix(filename)
```

- filename의 파일에서 데이터를 읽어 변수 A에 저장한다.

```
clc; clear;
x = [1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15];
writematrix(X,'data-mat1.txt');
writematrix(X,'data-mat1.csv');
Y = readmatrix('data-mat1.txt')
Z = readmatrix('data-mat1.csv')
```

- 불규칙적인 자릿수를 차지하고 단순히 콤마로 분리되어 있는 데이터를 읽는데 사용된다.

5.3.2 스프레드 시트 파일 읽고 쓰기

- Spreadsheet 데이터 읽고 쓰기

```
writematrix(A,filename)
```

- Excel 스프레드 시트 파일로 저장하려면 확장자는 xls, xlsx가 되어야 한다.

```
writematrix(A, fname, 'WriteMode', 'overwritesheet')
writematrix(A, fname, 'WriteMode', 'append')
```

- ✓ 확실하게 덮어쓰기 하려면 WritrMode에 overwritesheet를 추가한다.
- ✓ 기존의 파일에 데이터를 추가하려면 WriteMode에 append를 추가한다.

```
writematrix(A, fname, 'Sheet', n, 'Range', 'B2:F4')
```

- ✓ 시트 번호와 범위를 지정할 수 있다.

```
A = readmatrix(fname, 'Sheet', n, 'Range', 'B2:F4')
```

- ✓ 주어진 시트와 범위의 데이터를 읽는다.

```
clc; clear;
A = [1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15]
B = A+15
fname1 = 'data-mat1.xlsx';
fname2 = 'data-mat2.xlsx';
% 엑셀 스프레드 시트에 데이터 쓰기
writematrix(A, fname1, 'WriteMode', 'overwritesheet');
writematrix(B, fname1, 'WriteMode', 'append');
writematrix(A, fname2, 'Sheet', 2, 'Range', 'B2:F4');
% 엑셀 스프레드 시트에서 데이터 읽기
C = readmatrix(fname1)
D = readmatrix(fname2, 'Sheet', 2, 'Range', 'B2:F4')
```

5.3.3 이미지 파일 읽고 쓰기

```
info = imfinfo(filename)
```

- ✓ filename의 파일의 이미지 정보를 알려 준다.

```
A = imread(filename)
```

- ✓ filename의 파일에서 데이터를 읽어 변수 A에 저장한다.

```
imwrite(A, filename)
```

- ✓ 변수 A의 데이터를 filename에 저장한다.

```
imshow(filename)
```

- ✓ filename 의 파일의 이미지를 보여 준다.

```
clc; clear all;
info = imfinfo('RosePic.jpg');
A = imread('RosePic.jpg');
imshow(A)
imwrite(A, 'Rose.jpg')
```

5.3.4 오디오 파일 읽고 쓰기

```
info = audioinfo(filename)
```

- ✓ filename 의 파일의 오디오 정보를 알려 준다.

```
[x, Fs] = audioread(filename)
```

- ✓ filename 의 파일에서 데이터를 읽어 변수 A 에, 샘플링 주파수는 변수 Fs 에 저장한다.

```
audiowrite(filename, x, Fs)
```

- ✓ 변수 x 의 데이터와 샘플링 주파수 Fs 를 filename 에 저장한다.

```
sound(x)
sound(x, Fs)
```

- ✓ 변수 x 의 데이터를 연주한다.

```
clc; clear all;
info = audioinfo('dtmf_sig.wav')
[x, Fs] = audioread('dtmf_sig.wav');
sound(x, Fs);
audiowrite('dtmf.wav', x, Fs);
```

5.4 연습 문제

- ▣ 문제 1. 앞에서 만든 함수를 이용하여 평균이 0이고 분산이 1인 가우시안 백색 잡음을 1000 개 생성하여 MATLAB 포맷으로 저장하고 이를 다시 읽는 프로그램을 작성하라.
- ▣ 문제 2. 문제 1에서 작성한 가우시안 백색 잡음을 ASCII 코드 (고정 소수점과 부동 소수점)로 저장하고 이를 다시 읽는 프로그램을 작성하라.
- ▣ 문제 3. 문제 1에서 작성한 가우시안 백색 잡음을 이진수 (단일 정밀도와 배정밀도)로 저장하고 이를 다시 읽는 프로그램을 작성하라.
- ▣ 문제 4. $1+j$, $1-j$, $-1+j$, $-1-j$ 를 랜덤하게 갖는 신호 x 를 1000 개 생성하고, 이를 이진수로 저장하고 다시 읽는 프로그램을 작성하라.
- ▣ 문제 5. 주어진 텍스트 문장을 저장하고 이를 다시 읽는 프로그램을 작성하라.
 - ✓ This is the first line.
 - ✓ This is the second line.
 - ✓ This is the third line.
- ▣ 문제 6. 인터넷에서 장미꽃 이미지 파일(RosePic.jpg)을 다운로드한 다음 MATLAB에서 읽고 흑백 이미지로 바꾸어 저장하라. RGB 파일을 gray 파일로 바꾸는 함수는 `rgb2gray` 이다.

6. 신호와 스펙트럼

6.1 신호

- ✚ 아날로그 신호와 디지털 신호의 주파수를 구분할 필요가 있다.

- ✓ 아날로그 신호의 주파수와 각 주파수

$$F, \Omega = 2\pi F$$

- ✓ 디지털 신호의 주파수와 각 주파수

$$f, \omega = 2\pi f$$

6.1.1 아날로그 신호

- ✚ 아날로그 (연속 시간) 신호

- ✓ 주파수 신호

$$x(t) = A \sin(\Omega t) = A \sin(2\pi F t)$$

- ✓ 백색 잡음 신호

✗ 연속 시간 백색 잡음은 하드웨어에 의해 생성해야 한다.

✗ MATLAB에서 얻은 백색잡음은 의사 랜덤(pseudo random) 잡음이다.

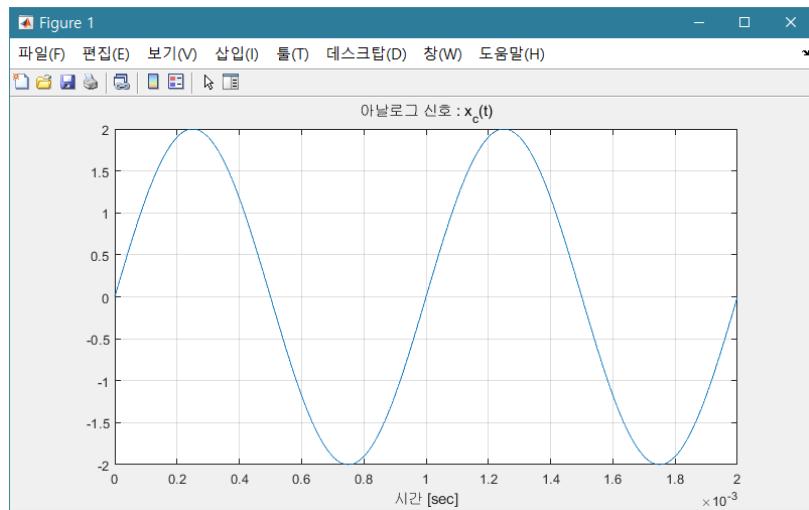
- ✚ 주파수 신호의 생성

$$x(t) = A \sin(2\pi F t), \quad A = 2, F = 1 \text{ kHz}$$

- ✓ 2 주기 신호 생성 및 그래프

```
% Plot analog signal
clc; clear;
% 연속 시간 신호의 생성
A = 2; f = 1000;
T = 1/f;
t = linspace(0, 2*T, 1001);
x = A*sin(2*pi*f*t);
% 연속 시간 신호의 그래프
figure(1)
plot(t, x)
grid on
```

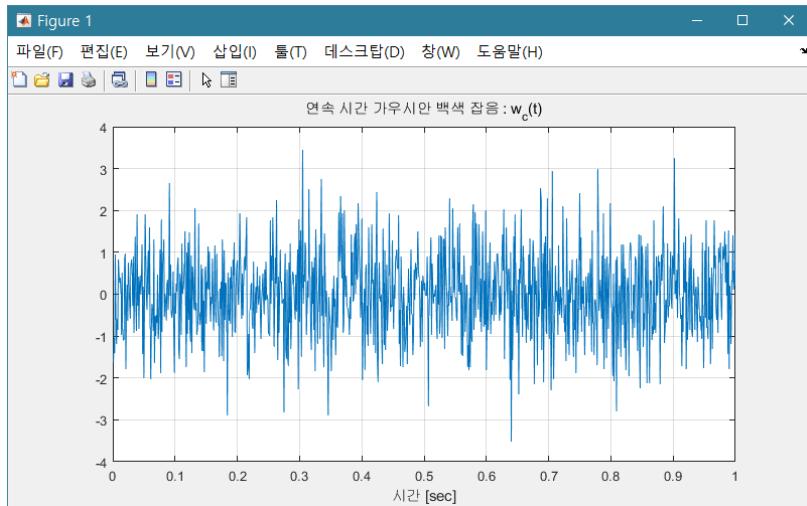
```
title('아날로그 신호 : x_c(t)')
xlabel('시간 [sec]')
```



▶ 백색 잡음 신호의 생성

- ✓ 평균이 0이고 분산이 1인 가우시안 백색 잡음
- ✓ MATLAB에서 얻은 백색 잡음이 1ms마다 얻은 신호라고 가정

```
% Plot analog white Gaussian noise
clc; clear;
% 아날로그 가우시안 백색 잡음의 생성
x = wgauss(0,1,1000);
dt = 1/1000;
t = dt*(1:length(x));
% 아날로그 가우시안 백색 잡음의 그래프
figure(1)
plot(t,x)
grid on
title('연속 시간 가우시안 백색 잡음 : w_c(t)')
xlabel('시간 [sec]')
```



✚ 잡음이 섞인 주파수 신호 생성

- ✓ 크기가 2, 주파수가 1 kHz 인 사인파에 평균이 0, 분산이 0.5 인 가우시안
백색 잡음이 섞인 신호

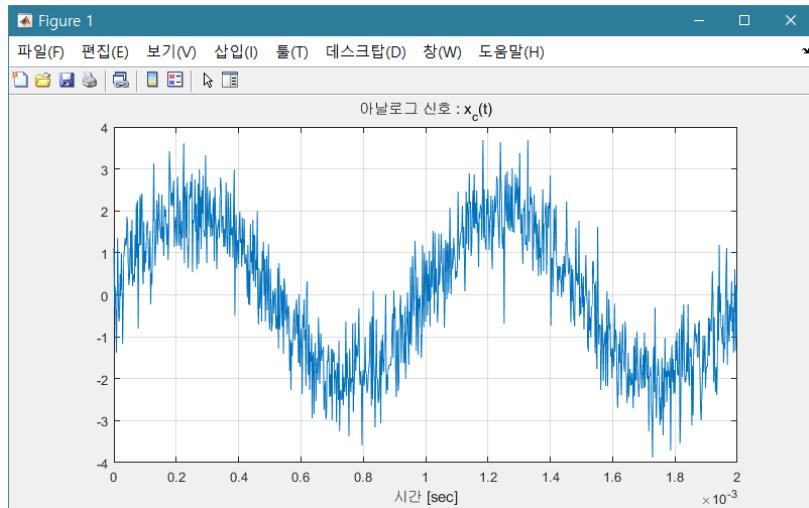
```
% Plot analog noise corrupted signal
clc; clear;

% 아날로그 신호의 생성
A = 2; F = 1000;
T = 1/F;
t = linspace(0, 2*T, 1001);
xs = A*sin(2*pi*F*t);

% 백색 잡음 신호의 생성
mx = 0; vx = 0.5; N = length(xs);
xn = wgauss(mx,vx,N);

% 잡음이 섞인 신호의 생성
x = xs+xn;

% 잡음이 섞인 아날로그 신호의 그래프
figure(1)
plot(t,x)
grid on
title('아날로그 신호 : x_c(t)')
xlabel('시간 [sec]')
```



6.1.2 디지털 신호

✚ 디지털 (이산 시간) 신호

- ✓ 주파수 신호

$$x[n] = A \sin(\omega n) = A \sin(2\pi f n)$$

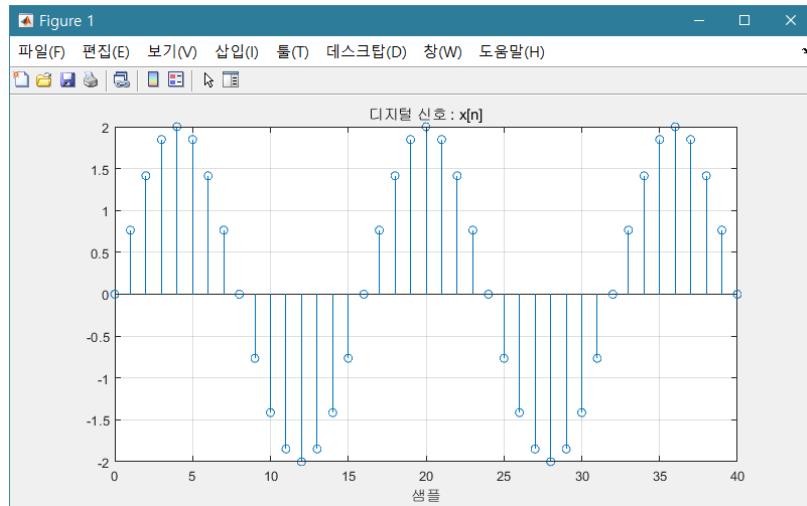
- ✓ 백색 잡음 신호

✗ MATLAB에서 얻은 백색잡음은 의사 랜덤(pseudo random) 잡음이다.

✚ 주파수 신호의 생성

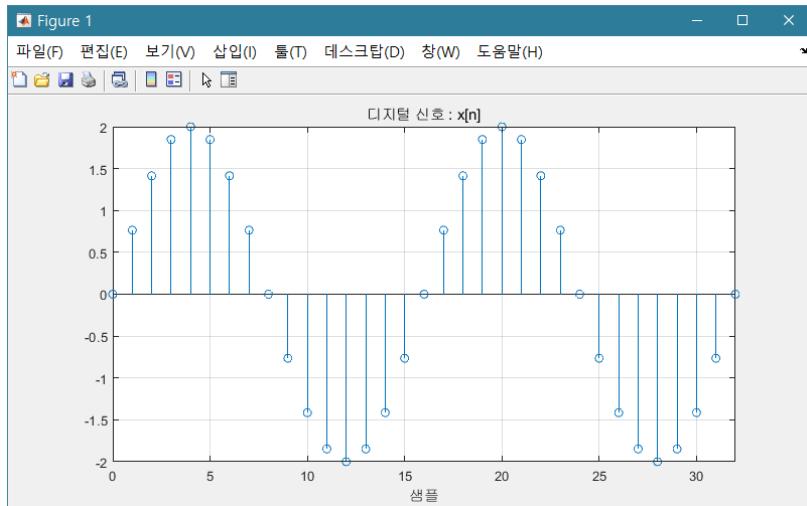
$$x[n] = 2 \sin(\omega n), \quad \omega = \pi / 8$$

```
% Plot digital signal
clc; clear;
% 디지털 신호의 생성
A = 2; w0 = pi/8;
n = 0:40;
x = A*sin(w0*n);
% 디지털 신호의 그래프
figure(1)
stem(n,x)
grid on
title('디지털 신호 : x[n]')
xlabel('샘플')
```



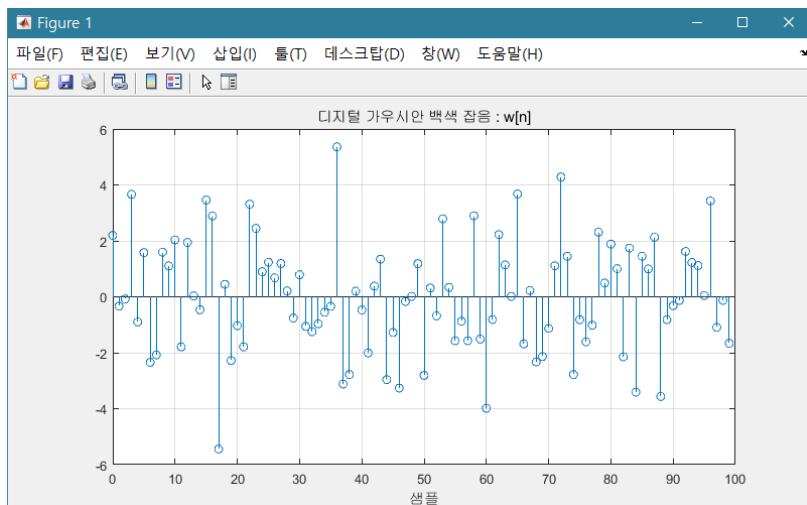
- ✓ 원하는 주기 만큼 그리는 방법은?
- ✗ 힌트 : 디지털 주파수 신호의 각 주파수 ω 는 주기가 2π 이다.

```
% Plot digital signal
clc; clear;
% 디지털 신호의 생성
A = 2; w = pi/8;
N = 2*(2*pi/w);
n = 0:N;
x = A*sin(w*n);
% 디지털 신호의 그래프
figure(1)
stem(n,x)
grid on
title('디지털 신호 : x[n]')
xlabel('샘플')
xlim([0,N])
```



▶ 백색 잡음 신호의 생성

```
% Plot digital white Gaussian noise
clc; clear;
% 디지털 가우시안 백색 잡음의 생성
mx = 0; vx = 4; Nx = 100;
x = wgauss(mx,vx,Nx);
n = 0:Nx-1;
% 이산 시간 가우시안 백색 잡음의 그래프
figure(1)
stem(n,x)
grid on
title('디지털 가우시안 백색 잡음 : w[n]')
xlabel('샘플')
```



▶ 잡음이 섞인 주파수 신호 생성

- ✓ 크기가 2, 각 주파수가 $\pi/8$ 인 사인파에 평균이 0, 분산이 0.5 인 가우시안 백색 잡음이 섞인 신호

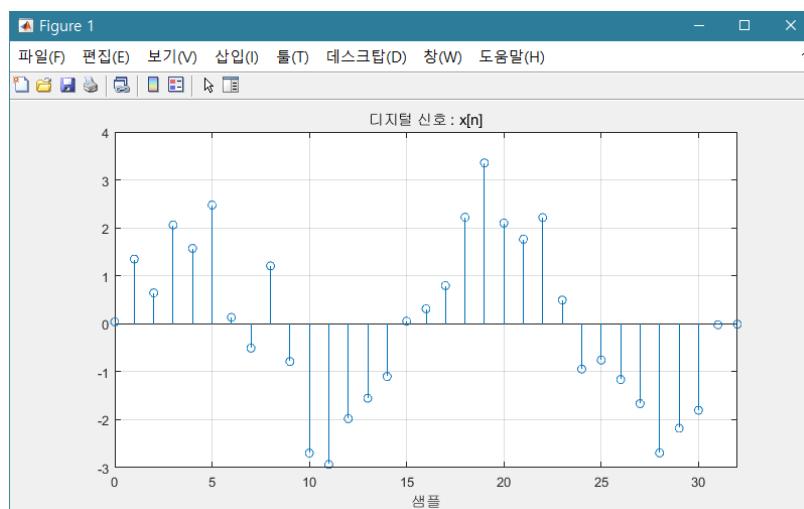
```
% Plot digital signal
clc; clear;

% 디지털 신호의 생성
A = 2; w0 = pi/8;
N = 2*(2*pi/w0);
n = 0:N;
xs = A*sin(w0*n);

% 가우시안 백색 잡음의 생성
mx = 0; vx = 0.5; Nx = length(xs);
xn = wgauss(mx,vx,Nx);

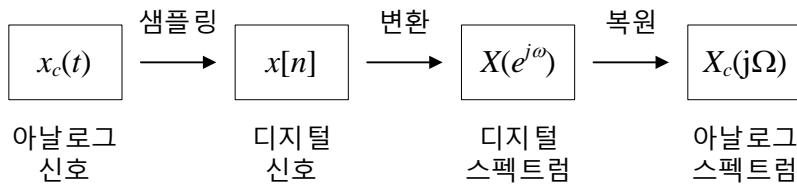
% 잡음이 섞인 신호의 생성
x = xs+xn;

% 디지털 신호의 그래프
figure(1)
stem(n,x)
grid on
title('디지털 신호 : x[n]')
xlabel('샘플')
xlim([0,N])
```



6.2 신호의 스펙트럼

✚ 아날로그/디지털 신호와 스펙트럼



- ✓ MATLAB에서 변환은 기본적으로 디지털 신호에서 디지털 스펙트럼으로의 변환이다.

6.2.1 디지털 신호의 스펙트럼

✚ 디지털 신호의 스펙트럼

- ✓ 길이가 유한한 신호에 대해 구한다.

$$x[n]: x[0], x[1], \dots, x[N-1]$$

$$X(e^{j\omega}) = \sum_{n=0}^{N-1} x[n] e^{-j\omega n}$$

- ✓ 디지털 신호의 스펙트럼은 각 주파수 ω 에 대해 주기가 2π 인 주기 함수이다.
- ✓ 디지털 신호의 스펙트럼은 고속 푸리에 변환 (FFT)을 이용하여 이산 푸리에 변환 (DFT)를 구하면 얻을 수 있다.

$$x[0], x[1], \dots, x[N-1] \Rightarrow X[0], X[1], \dots, X[N-1]$$

$$X[k] = X\left(e^{-j\frac{2\pi k}{N}}\right) = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi kn}{N}}$$

- ✓ $X[k]$ 는 $\omega = (2\pi k)/N$ 에서의 스펙트럼의 크기이다.
- ✓ N 점 FFT $X[k]$ 의 인수 k 와 디지털 각 주파수 ω 의 관계

FFT	$X[0]$	$X[1]$...	$X[k]$...	$X[N-1]$	$X[N]$
\downarrow	\downarrow	\downarrow		\downarrow		\downarrow	\downarrow
k	0	1	...	k	...	$N-1$	N
\downarrow	\downarrow	\downarrow		\downarrow		\downarrow	\downarrow
ω	0	$\frac{2\pi}{N}$...	$k \frac{2\pi}{N}$...	$(N-1) \frac{2\pi}{N}$	2π

✚ fft 함수

```
X = fft(x)
```

- ✓ 배열 x 의 DFT를 계산한다.
- ✓ 배열 X 는 각 주파수 $0 \leq \omega < 2\pi$ 의 구간을 나타낸다.

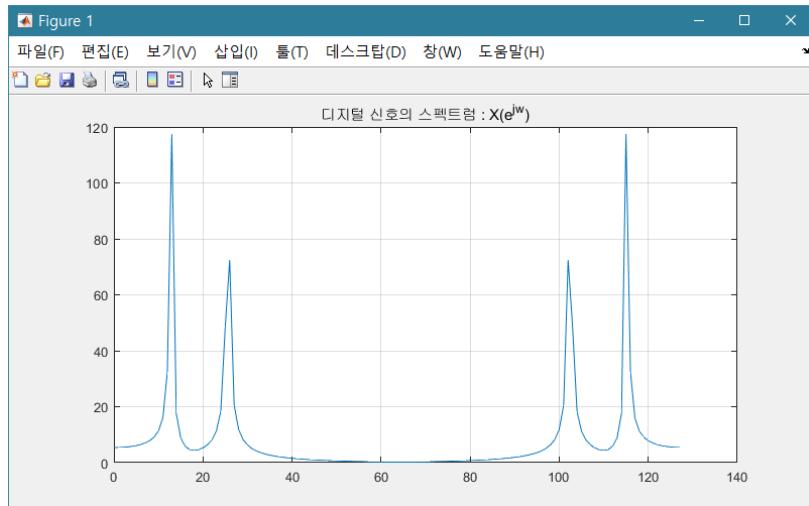
```
X = fft(x,n)
```

- ✓ 배열 x 의 길이가 n 보다 작은 경우 0을 채워 길이가 n 인 DFT를 계산한다.
(n 점 DFT)
- ✓ 배열 x 의 길이가 n 보다 큰 경우 x 를 n 만큼 잘라 DFT를 계산한다. (n 점 DFT)

```
Y = fftshift(X)
```

- ✓ 배열 X 의 왼쪽 절반과 오른쪽 절반을 바꾼다.
 - ✓ 배열 Y 는 각 주파수 $-\pi \leq \omega < \pi$ 의 구간을 나타낸다.
- ▶ 크기와 각 주파수가 각각 $A_1 = 2$, $\omega_1 = 0.2\pi$, $A_2 = 1.5$, $\omega_2 = 0.4\pi$ 인 사인파의 합인 신호의 스펙트럼

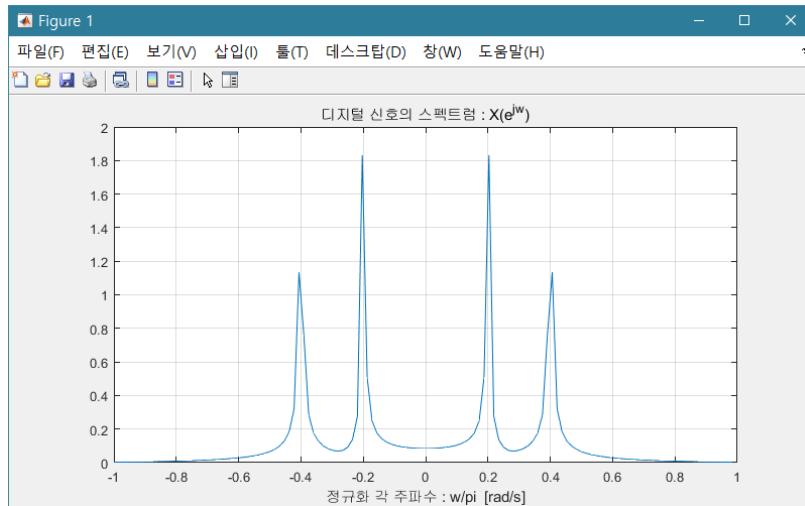
```
% Spectrum of digital signal
clc; clear;
% 디지털 사인파 신호
A1 = 2; w1 = 0.2*pi; A2 = 1.5; w2 = 0.4*pi;
N = 128;
n = 0:N-1;
x = A1*sin(w1*n) + A2*sin(w2*n);
% 신호의 스펙트럼
X = fft(x);
Xm = abs(X);
k = 0:N-1;
% 신호의 스펙트럼 그래프
figure(1)
plot(k,Xm)
grid on
title('디지털 신호의 스펙트럼 : X(e^{jw})')
```



- ✓ 문제점 : 가로 축, 각 주파수 범위, 스펙트럼 크기 등
- ✓ 수정한 프로그램

```
% Spectrum of digital signal
clc; clear;

% 디지털 사인파 신호
A1 = 2; w1 = 0.2*pi; A2 = 1.5; w2 = 0.4*pi;
N = 128;
n = 0:N-1;
x = A1*sin(w1*n) + A2*sin(w2*n);
% 신호의 스펙트럼
X = fft(x);
X = fftshift(X); % FFT 좌우 교체
Xm = abs(X) / (N/2); % 크기 조정
k = -N/2:N/2-1;
w = k*(2*pi)/N; % 구간을 -π ~ π로 변환
% 신호의 스펙트럼 그래프
figure(1)
plot(w/pi,Xm)
grid on
title('디지털 신호의 스펙트럼 :  $X(e^{jw})$ ')
xlabel('정규화 각 주파수 :  $w/\pi$  [rad/s]')
```



➊ 데이터 길이에 따른 스펙트럼 변화

- ✓ 데이터 길이가 32 개, 256 개일 때의 변화

```
% Spectrum of digital signal
clc; clear;

% 디지털 사인파 신호
A1 = 2; w1 = 7/32*pi; A2 = 1.5; w2 = 13/32*pi;
N = 256; N1 = 32; N2 = 256;
n = 0:N-1;
x = A1*sin(w1*n) + A2*sin(w2*n);

% 신호의 스펙트럼 (데이터 32 개)
X1 = fft(x, N1);
X1 = fftshift(X1);
X1m = abs(X1) / (N1/2);
k1 = -N1/2:N1/2-1;
w1 = k1*(2*pi)/N1;
figure(1)
plot(w1/pi,X1m)
grid on
ylim([0,2.2])
title('디지털 신호의 스펙트럼 : X(e^{jw})')
xlabel('정규화 각 주파수 : w/pi [rad/s]')

% 신호의 스펙트럼 (데이터 256 개)
x2 = fft(x,N2);
```

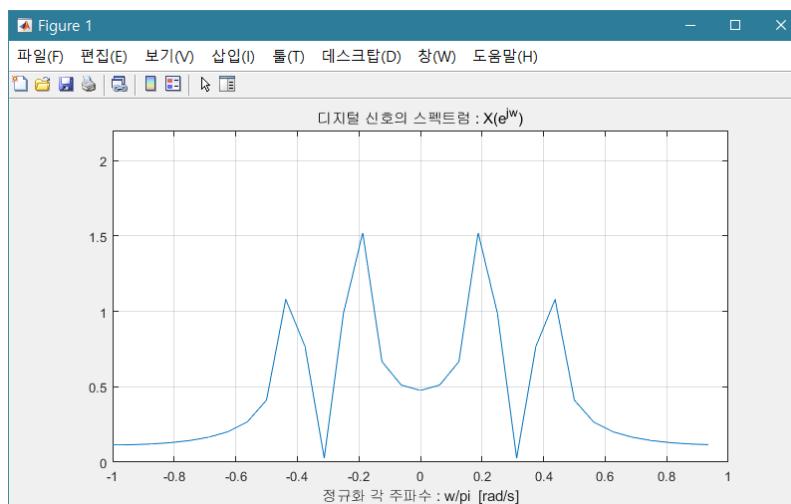
```

x2 = fftshift(X2);
X2m = abs(X2) / (N2/2);
k2 = -N2/2:N2/2-1;
w2 = k2*(2*pi)/N2;
figure(2)
plot(w2/pi,X2m)
grid on
ylim([0,2.2])
title('디지털 신호의 스펙트럼 : X(e^{jw})')
xlabel('정규화 각 주파수 : w/pi [rad/s]')

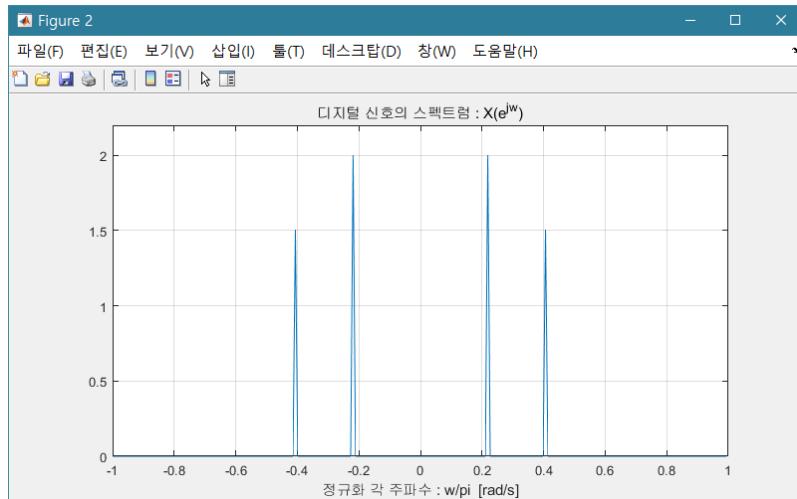
% 2개의 스펙트럼 비교
figure(3)
plot(w1/pi,X1m, w2/pi,X2m)
grid on
ylim([0,2.2])
title('디지털 신호의 스펙트럼 : X(e^{jw})')
xlabel('정규화 각 주파수 : w/pi [rad/s]')
legend('데이터 길이 32', '데이터 길이 512')

```

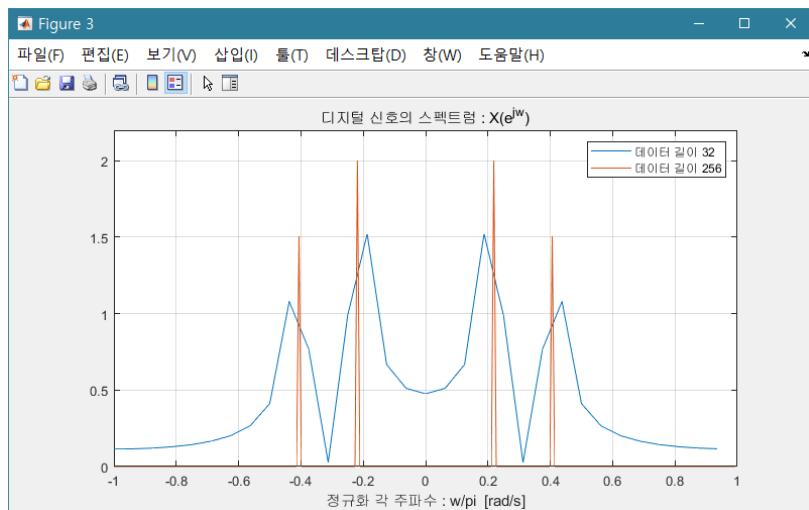
✓ 데이터 길이가 32 일 때의 스펙트럼



✓ 데이터 길이가 256 일 때의 스펙트럼



✓ 두 스펙트럼 비교



- ✓ 데이터의 길이가 길면 분해도가 좋아진다.
- ✚ 데이터의 길이가 짧을 때의 스펙트럼 처리
 - ✓ 데이터의 길이가 64 일 때 256 개의 스펙트럼
 - ✓ 데이터의 뒤에 192 개의 0 을 추가하여 256 개로 만든 후의 스펙트럼

```
% Spectrum of digital signal
clc; clear;

% 디지털 사인파 신호
A1 = 2; w1 = 7/32*pi; A2 = 1.5; w2 = 13/32*pi;
N = 32; N1 = 32; N2 = 256;
n = 0:N-1;
x = A1*sin(w1*n) + A2*sin(w2*n);
```

```
% 신호의 스펙트럼 (데이터 32 개)
```

```

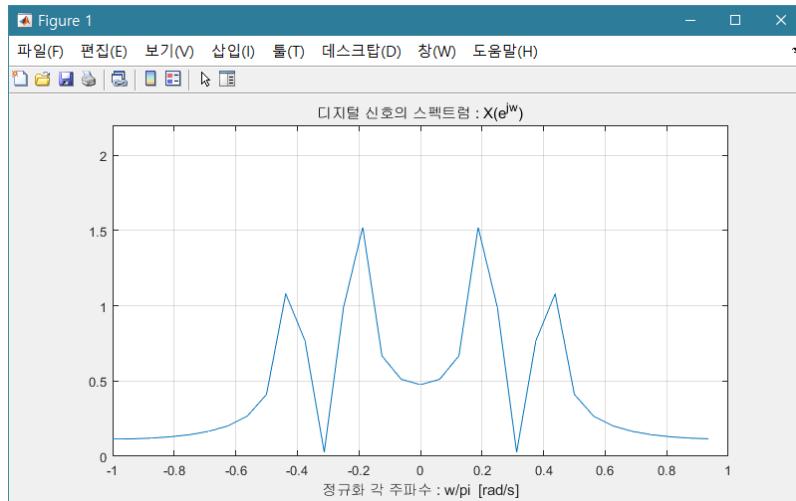
x1 = fft(x, N1);
x1 = fftshift(X1);
X1m = abs(X1) / (N1/2);
k1 = -N1/2:N1/2-1;
w1 = k1*(2*pi)/N1;
figure(1)
plot(w1/pi,X1m)
grid on
ylim([0,2.2])
title('디지털 신호의 스펙트럼 : X(e^{jw})')
xlabel('정규화 각 주파수 : w/pi [rad/s]')

% 신호의 스펙트럼 (데이터 32 개 + 0 224 개, 총 256 개)
x2 = fft(x,N2);
x2 = fftshift(X2);
X2m = abs(X2) / (N2/2);
k2 = -N2/2:N2/2-1;
w2 = k2*(2*pi)/N2;
figure(2)
plot(w2/pi,X2m)
grid on
ylim([0,2.2])
title('디지털 신호의 스펙트럼 : X(e^{jw})')
xlabel('정규화 각 주파수 : w/pi [rad/s]')

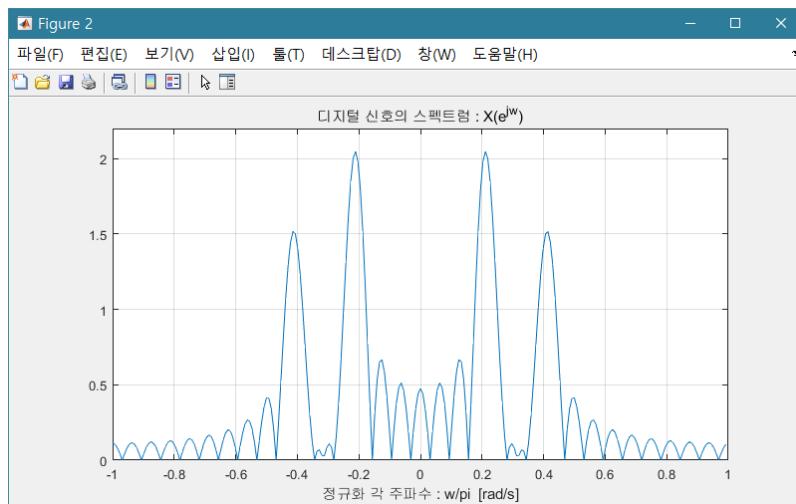
% 2 개의 스펙트럼 비교
figure(3)
plot(w1/pi,X1m, w2/pi,X2m)
grid on
ylim([0,2.2])
title('디지털 신호의 스펙트럼 : X(e^{jw})')
xlabel('정규화 각 주파수 : w/pi [rad/s]')
legend('데이터 길이 32', '데이터 길이 32 (224 개 0 삽입)')

```

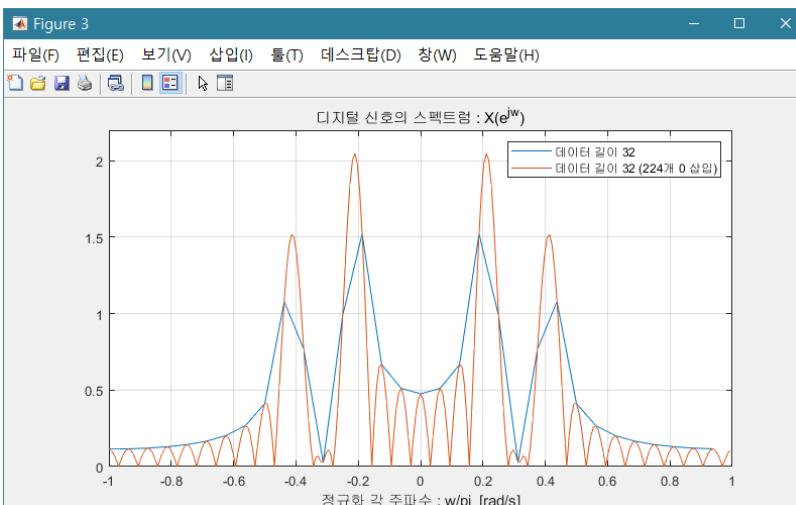
-
- ✓ 데이터의 길이가 32 일 때 32 점 스펙트럼



- ✓ 데이터의 뒤에 224 개의 0 을 추가하여 256 개로 만든 후의 스펙트럼



- ✓ 두 스펙트럼 비교



- ✓ 스펙트럼의 분해도는 변동이 없지만 많은 주파수의 값을 알 수 있다.
(촘촘한 그래프를 그릴 수 있다.)
- ✓ 촘촘하게 그리면 정확한 주파수를 알 수 있다.

▶ 결론

- ✓ 원본 데이터의 길이가 커야 스펙트럼의 분해도가 증가한다.
- ✓ 데이터의 길이가 작을 때는 0 을 추가하여 데이터의 길이를 증가시킨다.
(스펙트럼의 분해도는 증가하지 않는다.)

▶ 잡음의 크기가 신호 대 잡음 비 (signal to noise ratio: SNR)로 주어질 때

- ✓ 크기가 A이고 SNR 이 주어질 때 잡음 분산 구하는 방법

$$SNR = 10 \log_{10} \left(\frac{\text{var}_S}{\text{var}_N} \right)$$

$$\text{var}_S = \frac{A^2}{2} \Rightarrow \text{var}_N = \text{var}_S 10^{-\frac{SNR}{10}} = \frac{A^2}{2} 10^{-\frac{SNR}{10}}$$

▶ 크기가 $A=2$, 각 주파수가 $\omega_0 = 0.4\pi$ 인 사인파에 SNR 이 10 dB 인 가우시안 백색 잡음이 섞인 신호의 스펙트럼

```

clc; clear;

% 디지털 사인파 신호
A = 2; w0 = 0.4*pi;
N = 256; SNR = 10;
n = 0:N-1;
xs = A*sin(w0*n);

% 가우시안 백색 잡음
mn = 0;
vn = A^2/2*10^(-SNR/10);
xn = wgauss(mn,vn,N);

% 잡음이 섞인 신호
x = xs+xn;

% 신호의 스펙트럼
X = fft(x);
X = fftshift(X);
Xm = abs(X) / (N/2);
XmdB = 20*log10(Xm);
k = -N/2:N/2-1;
w = k*(2*pi)/N;

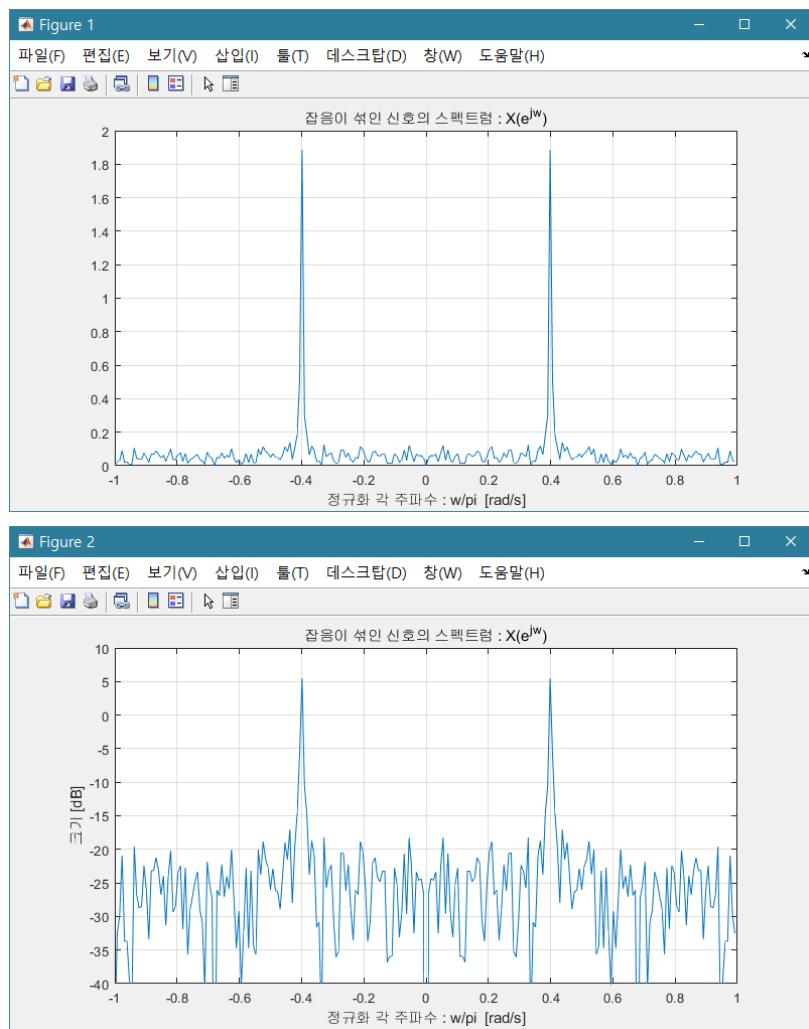
figure(1)
plot(w/pi,Xm)

```

```

grid on
title('잡음이 섞인 신호의 스펙트럼 : X(e^{jw})')
xlabel('정규화 각 주파수 : w/pi [rad/s]')
figure(2)
plot(w/pi,XmdB)
grid on
ylim([-40,10])
title('잡음이 섞인 신호의 스펙트럼 : X(e^{jw})')
xlabel('정규화 각 주파수 : w/pi [rad/s]')
ylabel('크기 [dB]')

```



6.2.2 디지털 신호의 전력 스펙트럼 밀도(power spectrum density: PSD)

- ➊ 디지털 신호의 전력 스펙트럼 밀도
 - ✓ Periodogram 방법 이용

$$P_{xx}(\omega) = \lim_{M \rightarrow \infty} E \left[\frac{1}{2M+1} \left| \sum_{n=-M}^M x[n] e^{-j\omega n} \right|^2 \right]$$

$$\hat{P}_{PER}(\omega) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] e^{-j\omega n} \right|^2$$

▶ periodogram 함수

```
periodogram(x)
```

- ✓ 배열 x의 PSD를 계산하고 그려준다.

```
Pxx = periodogram(x)
```

- ✓ 배열 x의 PSD를 계산한다.

```
Pxx = periodogram(x,window)
```

- ✓ 배열 x에 window에 해당하는 창을 써운 후 PSD를 계산한다.
- ✓ 창의 길이는 배열 x의 길이와 같아야 한다.

```
Pxx = periodogram(x,window,nfft)
```

- ✓ 배열 x의 길이가 nfft보다 작으면 0을 추가한다.

```
[Pxx,w] = periodogram(x,window,w)
```

- ✓ 주어진 w에 대해 PSD를 계산한 후 w와 함께 돌려 준다.

▶ 크기와 각 주파수가 각각 $A_1 = 2$, $\omega_1 = \pi/4$, $A_2 = 1.5$, $\omega_2 = \pi/2$ 인 사인파의 합인 신호의 PSD

```
clc; clear;
% 디지털 사인파 신호
A1 = 2; w1 = pi/4; A2 = 1.5; w2 = pi/2;
N = 256; N1 = 64; N2 = 256;
n = 0:N-1;
x = A1*sin(w1*n) + A2*sin(w2*n);
```

% 64개 신호의 PSD

```
figure(1)
periodogram(x(1:N1))
```

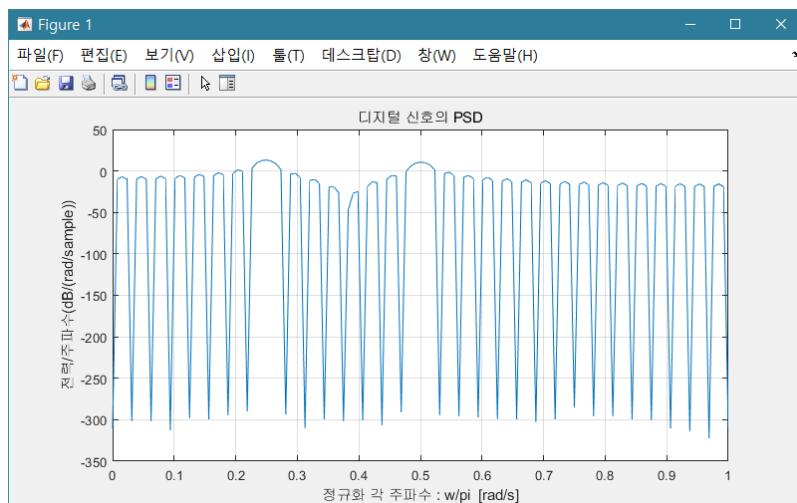
```

grid on
% ylim([-20,20])
title('디지털 신호의 PSD')
xlabel('정규화 각 주파수 : w/pi [rad/s]')

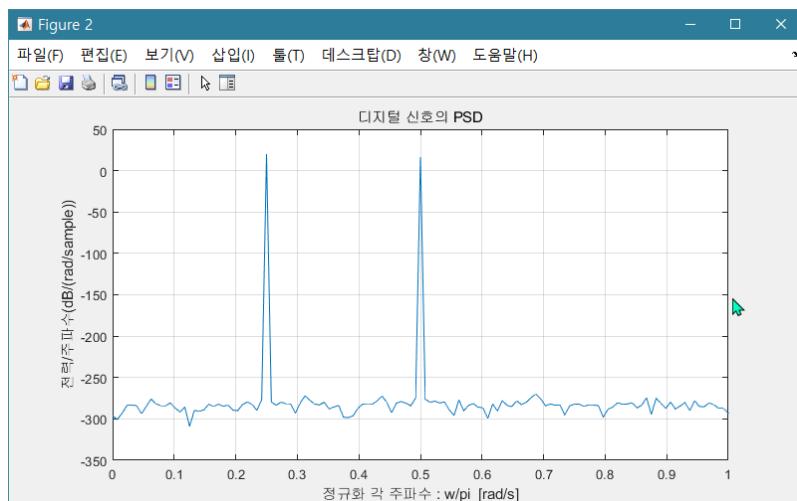
% 256 개 신호의 PSD
figure(2)
periodogram(x,hamming(N))
grid on
% ylim([-20,20])
title('디지털 신호의 PSD')
xlabel('정규화 각 주파수 : w/pi [rad/s]')

```

- ✓ 데이터 길이가 64 일 때의 PSD



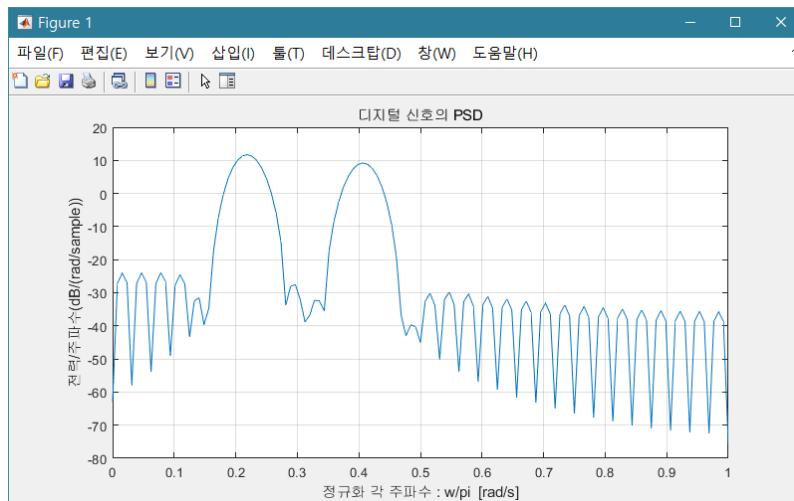
- ✓ 데이터 길이가 256 일 때의 PSD



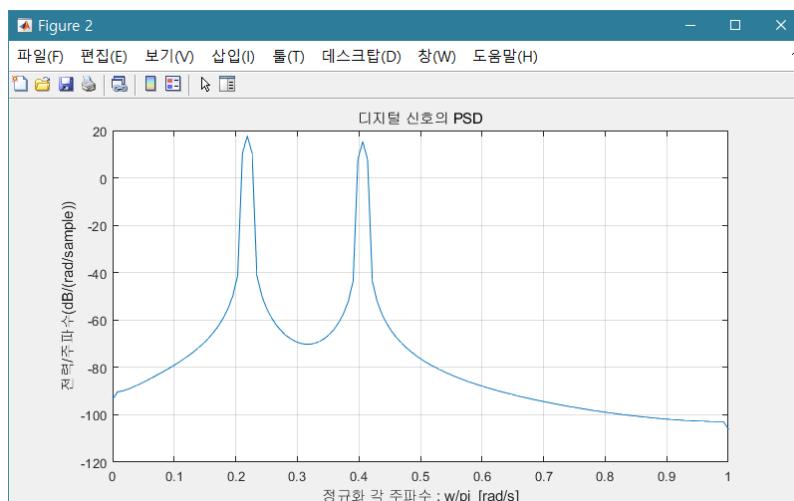
- ✓ periodogram 에 window 를 적용한 그림

```
periodogram(x(1:N1)) -> periodogram(x(1:N1),hamming(N1))
periodogram(x) -> periodogram(x,hamming(N))
```

- ✖ 데이터 길이가 64일 때의 PSD



- ✖ 데이터 길이가 256일 때의 PSD



- ✚ w 를 주고 해당하는 PSD 를 구하고 그리는 프로그램

```
clc; clear;
% 디지털 사인파 신호
A1 = 2; w1 = pi/4; A2 = 1.5; w2 = pi/2;
N = 256;
n = 0:N-1;
x = A1*sin(w1*n) + A2*sin(w2*n);

% 신호의 PSD 1
w = linspace(0,pi,1025);
[Pxx,w] = periodogram(x,[],w);
```

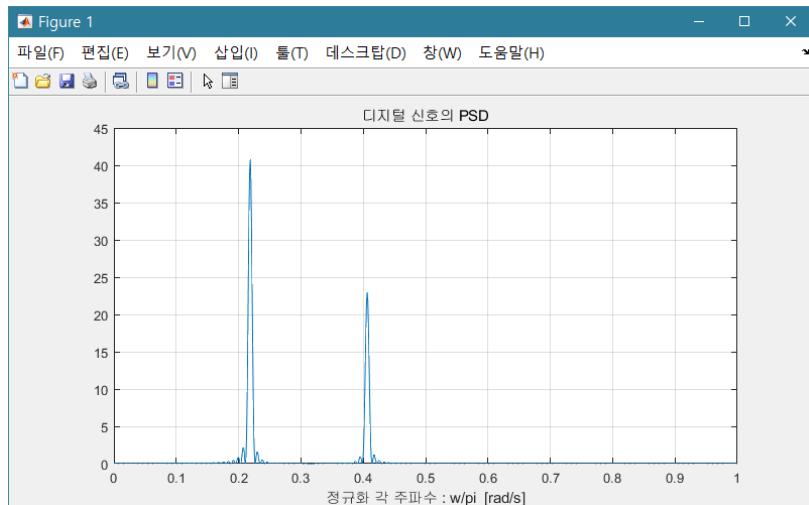
```

figure(1)
plot(w/pi,Pxx)
grid on
title('디지털 신호의 PSD')
xlabel('정규화 각 주파수 : w/pi [rad/s]')

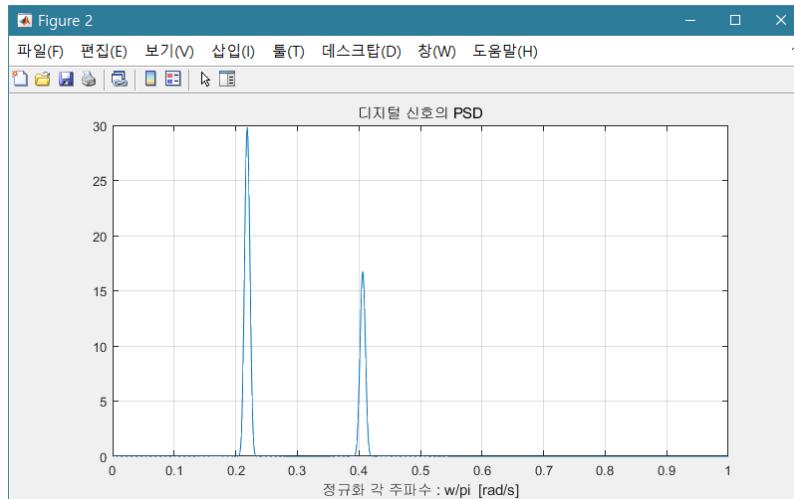
% 신호의 PSD 2 (해밍 창 사용)
w = linspace(0,pi,1025);
[Pxx,w] = periodogram(x,hamming(N),w);
figure(2)
plot(w/pi,Pxx)
grid on
title('디지털 신호의 PSD')
xlabel('정규화 각 주파수 : w/pi [rad/s]')

```

✓ 창을 사용하지 않았을 경우



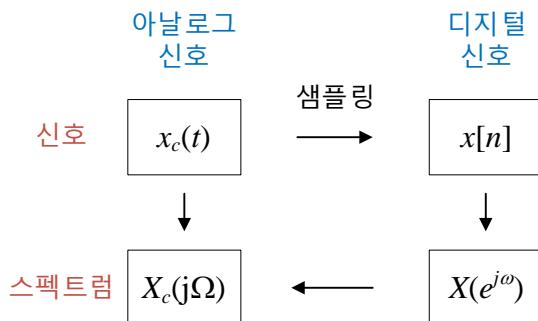
✓ 해밍 창을 사용했을 경우



- ✓ 해밍 창을 사용하면 주엽(main lobe)의 폭은 커지나 부엽(side lobe)의 크기가 감소한다.

6.2.3 아날로그 신호의 스펙트럼

✚ 아날로그/디지털 신호와 스펙트럼의 관계



- ✓ 아날로그 신호를 샘플링하여 디지털 신호를 만들고 스펙트럼을 구한 후 디지털 각 주파수를 아날로그 주파수로 변환한다. 이 때 샘플링 주파수는 나이키스트 샘플링 정리를 만족해야 한다.
- ✓ 아날로그 주파수와 디지털 주파수의 관계

$$x[n] = x_c(nT_s), \quad T_s = \frac{1}{F_s}$$

$$\Omega = \frac{\omega}{T_s} = F_s \omega, \quad F = \frac{f}{T_s} = F_s f$$

✗ 아날로그 주파수 F 는 $F_s / 2$ 까지 가능하다.

✚ FFT를 이용한 아날로그 신호의 스펙트럼

- ✓ 크기와 주파수가 각각 $A_1 = 2$, $F_1 = 1 \text{ kHz}$, $A_2 = 1.5$, $F_2 = 2 \text{ kHz}$ 인 사인파의 합인 아날로그 신호의 스펙트럼

- ✓ 샘플링 주파수를 $F_s = 10 \text{ kHz}$ 로 하면 $0 \sim 5 \text{ kHz}$ 동안 (또는 $-5 \sim 5 \text{ kHz}$ 동안) 스펙트럼을 관찰할 수 있다.
- ✓ N 점 FFT $X[k]$ 의 인수 k 와 디지털 각 주파수 ω 의 관계

FFT	$X[0]$	$X[1]$	\cdots	$X[k]$	\cdots	$X[N-1]$	$X[N]$
\Downarrow	\downarrow	\downarrow		\downarrow		\downarrow	\downarrow
k	0	1	\cdots	k	\cdots	$N-1$	N
\Downarrow	\downarrow	\downarrow		\downarrow		\downarrow	\downarrow
ω	0	$\frac{2\pi}{N}$	\cdots	$k \frac{2\pi}{N}$	\cdots	$(N-1) \frac{2\pi}{N}$	2π
\Downarrow	\downarrow	\downarrow		\downarrow		\downarrow	\downarrow
F	0	$\frac{F_s}{N}$		$k \frac{F_s}{N}$		$(N-1) \frac{F_s}{N}$	F_s

```

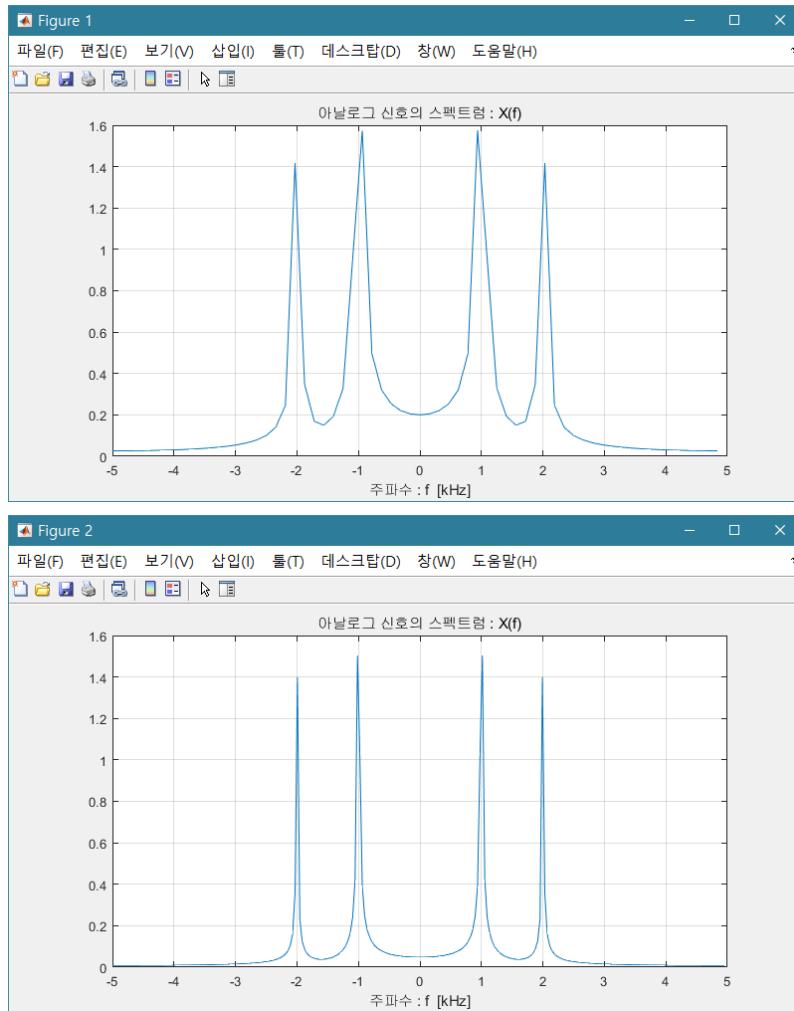
clc; clear;

% 아날로그 사인파 신호
A1 = 2; F1 = 1000; A2 = 1.5; F2 = 2000;
Fs = 10000; Ts = 1/Fs;
N = 256; N1 = 64; N2 = 256;
n = 0:N-1;
x = A1*sin(2*pi*F1*Ts*n)+A2*sin(2*pi*F2*Ts*n); %256 점 데이터
% 신호의 스펙트럼 (데이터 길이 64)
X = fft(x, N1); % 64 점 FFT
X = fftshift(X);
Xm = abs(X) / (N1/2);
k = -N1/2:N1/2-1;
F = k*Fs/N1; % 아날로그 주파수로 변환
figure(1)
plot(F/1000, Xm)
grid on
title('아날로그 신호의 스펙트럼 : X(f)')
xlabel('주파수 : f [kHz]')
% 신호의 스펙트럼 (데이터 길이 256)
X = fft(x, N2); % 256 점 FFT
X = fftshift(X);
Xm = abs(X) / (N2/2);
k = -N2/2:N2/2-1;

```

```
F = k*Fs/N2; % 아날로그 주파수로 변환

figure(2)
plot(F/1000,Xm)
grid on
title('아날로그 신호의 스펙트럼 : X(f)')
xlabel('주파수 : f [kHz]')
```



6.2.4 아날로그 신호의 전력 스펙트럼 밀도(PSD)

- ✚ 스펙트럼과 마찬가지로 아날로그 신호를 샘플링하여 디지털 신호를 만들고 PSD 를 구한 후 디지털 각 주파수를 아날로그 주파수로 변환한다. 이 때 샘플링 주파수는 나이키스트 샘플링 정리를 만족해야 한다.
- ✚ periodogram 함수

```
[Pxx,f] = periodogram(x,window,f,fs)
```

- ✓ 배열 x에 window에 해당하는 창을 써운 후 PSD를 계산한다.

- ✓ 창의 길이는 배열 x 의 길이와 같아야 한다.
- ✓ 주어진 f 에 대해 PSD를 계산한 후 f 와 함께 돌려준다.
- ✚ 크기와 주파수가 각각 $A_1 = 2$, $F_1 = 1\text{ kHz}$, $A_2 = 1.5$, $F_2 = 2\text{ kHz}$ 인 사인파의 합인 아날로그 신호의 PSD

```

clc; clear;

% 아날로그 사인파 신호

A1 = 2; F1 = 1000; A2 = 1.5; F2 = 2000;

Fs = 10000; Ts = 1/Fs;

N = 128;

n = 0:N-1;

x = A1*sin(2*pi*F1*Ts*n) + A2*sin(2*pi*F2*Ts*n);

% 신호의 PSD

F = -Fs/2:Fs/512:Fs/2;

[Pxx,F] = periodogram(x,[],F,Fs);

% [Pxx,F] = periodogram(x,hamming(N),F,Fs);

figure(1)

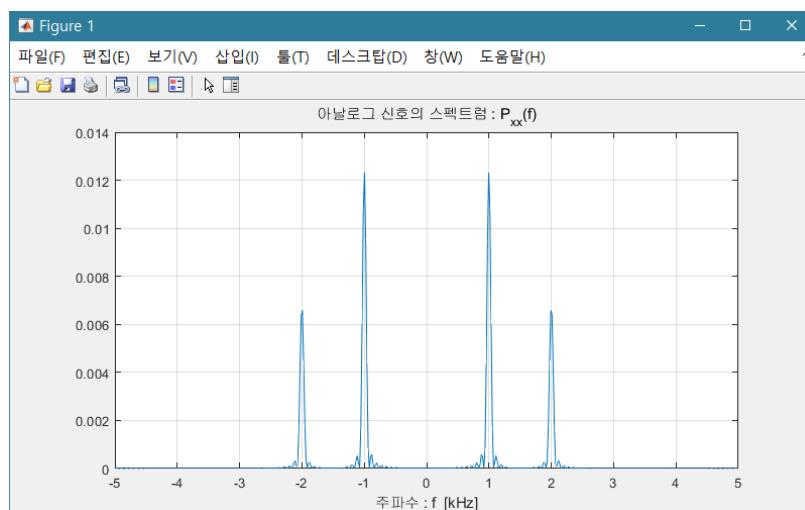
plot(F/1000,Pxx)

grid on

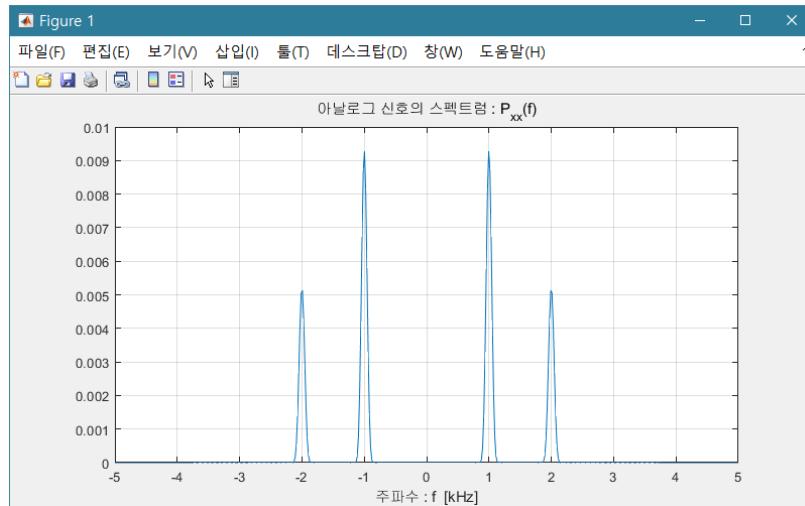
title('아날로그 신호의 스펙트럼 : P_{xx}(f)')

xlabel('주파수 : f [kHz]')

```



- ✓ hamming 창을 사용했을 때 그림



6.3 연습 문제

- ▶ 문제 1. 크기와 주파수가 각각 $A_1 = 2$, $F_1 = 1.8 \text{ kHz}$, $A_2 = 1.5$, $F_2 = 2 \text{ kHz}$ 인 사인파의 합인 아날로그 신호를 10 kHz 로 샘플링하여 디지털 신호를 얻었다. 이 신호로부터 64 개의 데이터 $x1(n)$, 64 개의 데이터에 나머지를 0 으로 채워 만든 512 개의 데이터 $x2(n)$, 512 개의 데이터 $x3(n)$ 의 스펙트럼을 FFT 를 사용하여 구하고 스펙트럼을 그려라. 스펙트럼을 구하고 그리는 프로그램은 함수를 사용하고 스펙트럼은 한 개의 창에 위 아래로 크기와 dB 로 나타내라. 이로부터 세 스펙트럼의 차이에 대해 설명하라.
- ▶ 문제 2. 크기와 주파수가 각각 $A_1 = 2$, $F_1 = 1.8 \text{ kHz}$, $A_2 = 1.5$, $F_2 = 2 \text{ kHz}$ 인 사인파의 합인 아날로그 신호를 10 kHz 로 샘플링하여 디지털 신호를 얻었다. 이 신호로부터 64 개의 데이터 $x1(n)$, 64 개의 데이터에 나머지를 0 으로 채워 만든 512 개의 데이터 $x2(n)$, 512 개의 데이터 $x3(n)$ 의 전력 스펙트럼 밀도(PSD)을 periodogram 을 사용하여 구하고 그려라. PSD 를 구하고 그리는 프로그램은 함수를 사용하라. 이로부터 세 PSD 의 차이에 대해 설명하라.

7. 필터 설계와 신호의 필터링

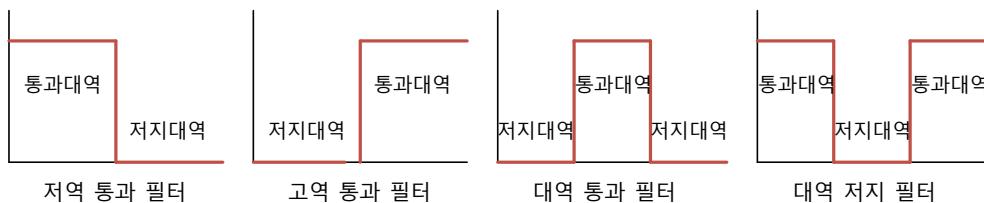
7.1 필터의 종류

필터 종류 1

- ✓ 아날로그 필터
- ✓ 디지털 필터

필터 종류 2 (주파수 특성)

- ✓ 저역 통과 필터 (lowpass filter: LPF)
- ✓ 고역 통과 필터 (highpass filter: HPF)
- ✓ 대역 통과 필터 (bandpass filter: BPF)
- ✓ 대역 저지 필터 (bandstop filter: BSF)



필터 종류 3 (전달 함수)

- ✓ FIR(finite impulse response) 필터
- ✓ IIR(infinite impulse response) 필터

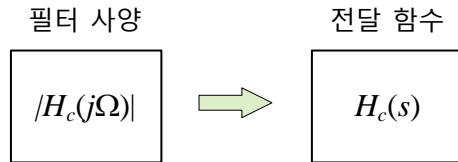
필터 종류 4 (설계 방법)

- ✓ IIR 필터
 - ✗ 버터워스 필터
 - ✗ 체비세프 필터
 - ✗ 타원 필터
- ✓ FIR 필터
 - ✗ 카이저 창을 이용한 필터
 - ✗ parks-McClellan 최적 필터
 - ✗ 기타

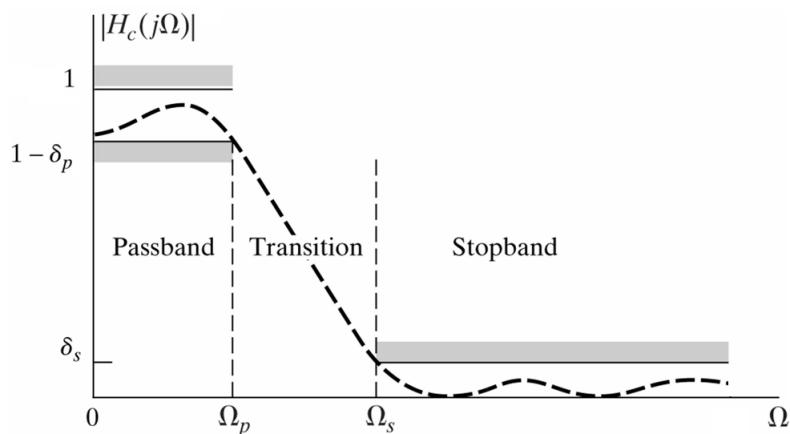
7.2 필터 설계

7.2.1 아날로그 필터 설계

✚ 아날로그 필터 설계



✓ 아날로그 필터 사양 (LPF)



$$\delta_p, \delta_s, \Omega_p, \Omega_s \Rightarrow H_c(s) = \frac{B(s)}{A(s)}$$

- ✓ 아날로그 필터는 기본적으로 IIR 필터이다.
- ✓ 아날로그 주파수는 기본적으로 로그 눈금이다.

✚ 아날로그 필터 설계 함수

- ✓ 입/출력 변수
 - ✗ $W_p = \Omega_p$, $W_s = \Omega_s$, $R_p = -20 \log_{10}(1 - \delta_p)$, $R_s = -20 \log_{10}(\delta_s)$
 - ✗ BPF와 BSF는 W_s 와 W_p 가 벡터이다.
 - ✗ N : 필터의 차수, W_n : 차단 주파수
 - ✗ $ftype$: 필터 종류, 문자열, 'low', 'high', 'bandpass', 'stop'
- ✓ 버터워스 LPF

```
[N,Wn] = buttord(Wp,Ws,Rp,Rs,'s')
[b,a] = butter(N,Wn,ftype,'s')
```

✓ 체비세프 1 LPF

```
[N,Wn] = cheb1ord(Wp,Ws,Rp,Rs,'s')
[b,a] = cheby1(N,Rp,Wn,ftype,'s')
```

✓ 체비세프 2 LPF

```
[N,Wn] = cheb2ord(Wp,Ws,Rp,Rs,'s')
[b,a] = cheby2(N,Rs,Wn,ftype,'s')
```

✓ 타원 LPF

```
[N,Wn] = ellipord(Wp,Ws,Rp,Rs,'s')
[b,a] = ellip(N,Rp,Rs,Wn,ftype,'s')
```

▶ 아래 예제에서 구한 아날로그 필터의 주파수 크기 응답을 그리는 함수

✓ 함수 이름 : plot_analog_freq_resp.m

```
function plot_analog_freq_resp(b,a,str1,str2)
% 아날로그 필터의 주파수 응답(크기)을 그리는 함수
F = logspace(2,4,500);
W = 2*pi*F;
H = freqs(b,a,W);
Hm = abs(H);
semilogx(F,Hm)
grid on
ylim([0,1.1])
str = ['아날로그 ',str1,' ',str2,' 주파수 크기 응답'];
title(str)
xlabel('주파수(F) : [Hz]')
```

▶ 저역 통과 필터 설계

$$F_p = 1 \text{ kHz}, \quad F_s = 2 \text{ kHz}, \quad \delta_p = 0.05, \quad \delta_s = 0.05$$

$$\Omega_p = 2\pi F_p, \quad \Omega_s = 2\pi F_s, \quad R_p = -20 \times \log_{10}(1 - \delta_p), \quad R_s = -20 \times \log_{10}(\delta_s)$$

```
%% Analog Lowpass Filter Design
```

```
clc; clear;
```

```
% 필터 사양
```

```

Fp = 1000; Fs = 2000;
dp = 0.05; ds = 0.05;
% 각 주파수와 감쇄율 (dB)
Wp = 2*pi*Fp; Ws = 2*pi*Fs;
Rp = -db(1-dp); Rs = -db(ds);

% 버터워스 필터
[N,Wn] = buttord(Wp,Ws,Rp,Rs,'s')
[b,a] = butter(N,Wn,'s');
figure(1)
plot_analog_freq_resp(b,a,'버터워스','LPF')

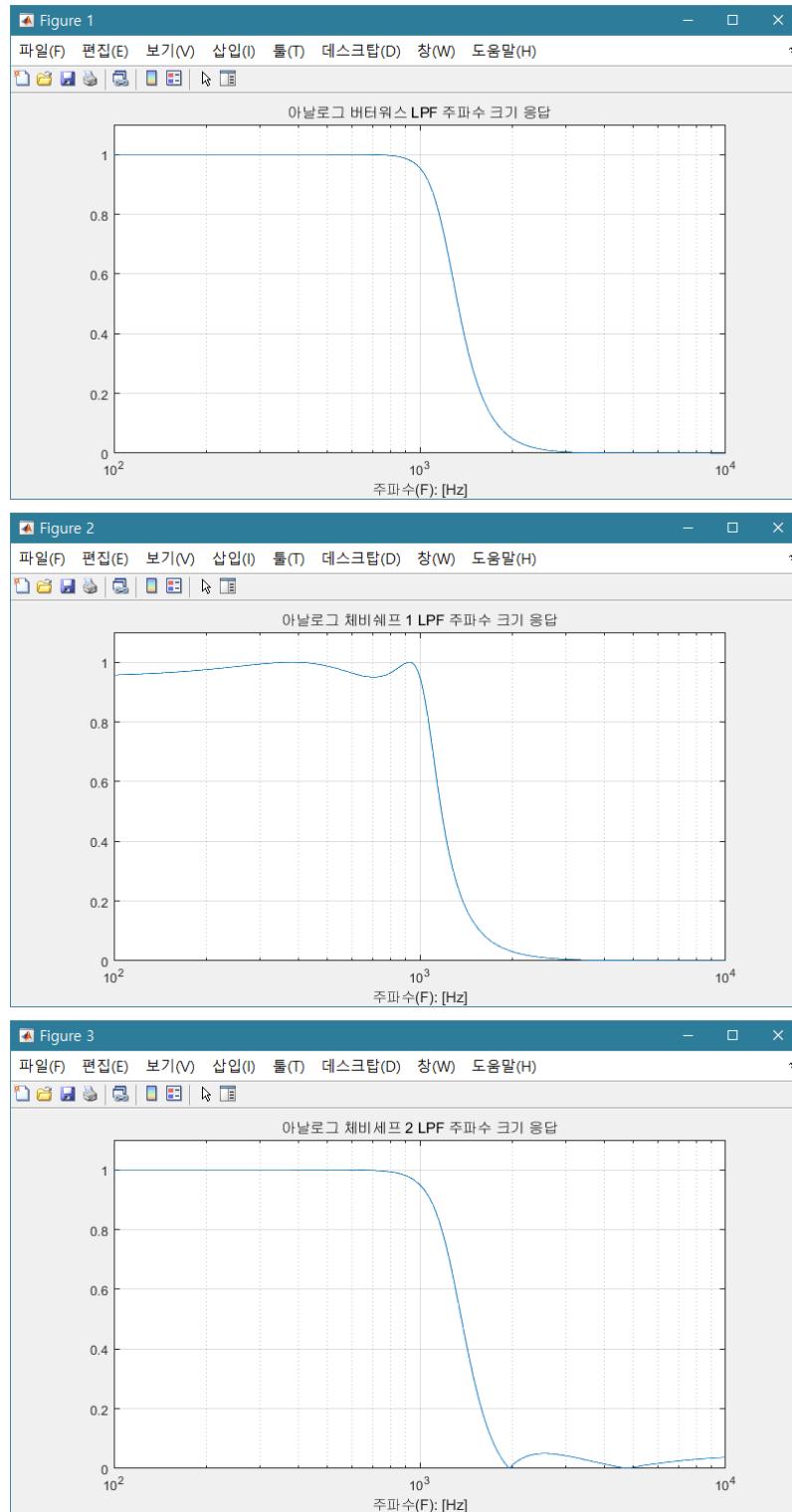
% 체비세프 1 필터
[N,Wn] = cheb1ord(Wp,Ws,Rp,Rs,'s')
[b,a] = cheby1(N,Rp,Wn,'s');
figure(2)
plot_analog_freq_resp(b,a,'체비쉐프 1','LPF')

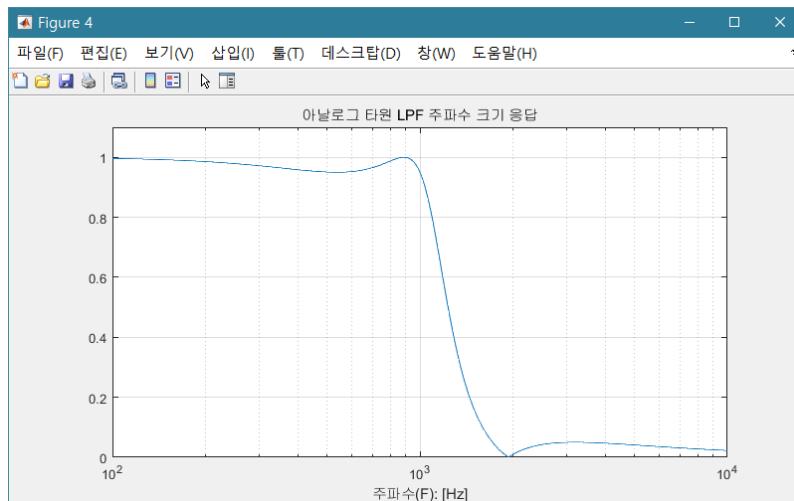
% 체비세프 2 필터
[N,Wn] = cheb2ord(Wp,Ws,Rp,Rs,'s')
[b,a] = cheby2(N,Rs,Wn,'s');
figure(3)
plot_analog_freq_resp(b,a,'체비세프 2','LPF')

% 타원 필터
[N,Wn] = ellipord(Wp,Ws,Rp,Rs,'s')
[b,a] = ellip(N,Rp,Rs,Wn,'s');
figure(4)
plot_analog_freq_resp(b,a,'타원','LPF')

```

- ✓ 주파수 응답 (크기)





✓ 필터 차수 비교

	Butterworth	Chebyshev 1	Chebyshev 2	Elliptic
Filter order	6	4	4	3

✚ 고역 통과 필터 설계

$$F_s = 1 \text{ kHz}, \quad F_p = 2 \text{ kHz}, \quad \delta_s = 0.05, \quad \delta_p = 0.05$$

$$\Omega_p = 2\pi F_p, \quad \Omega_s = 2\pi F_s, \quad R_p = -20 \times \log_{10}(1 - \delta_p), \quad R_s = -20 \times \log_{10}(\delta_s)$$

```
%% Analog Highpass Filter Design
clc; clear;

% 필터 사양
Fs = 1000; Fp = 2000;
ds = 0.05; dp = 0.05;
% 각 주파수와 감쇄율 (dB)
Wp = 2*pi*Fp; Ws = 2*pi*Fs;
Rp = -db(1-dp); Rs = -db(ds);

% 버터워스 필터
[N,Wn] = buttord(Wp,Ws,Rp,Rs,'s');
[b,a] = butter(N,Wn,'high','s');
figure(1)
plot_analog_freq_resp(b,a,'버터워스','HPF')

% 체비세프 1 필터
[N,Wn] = cheblord(Wp,Ws,Rp,Rs,'s');
[b,a] = cheby1(N,Rp,Wn,'high','s');
```

```
figure(2)
```

```
plot_analog_freq_resp(b,a,'체비세프 1','HPF')
```

% 체비세프 2 필터

```
[N,Wn] = cheb2ord(Wp,Ws,Rp,Rs,'s')
```

```
[b,a] = cheby2(N,Rs,Wn,'high','s');
```

```
figure(3)
```

```
plot_analog_freq_resp(b,a,'체비세프 2','HPF')
```

% 타원 필터

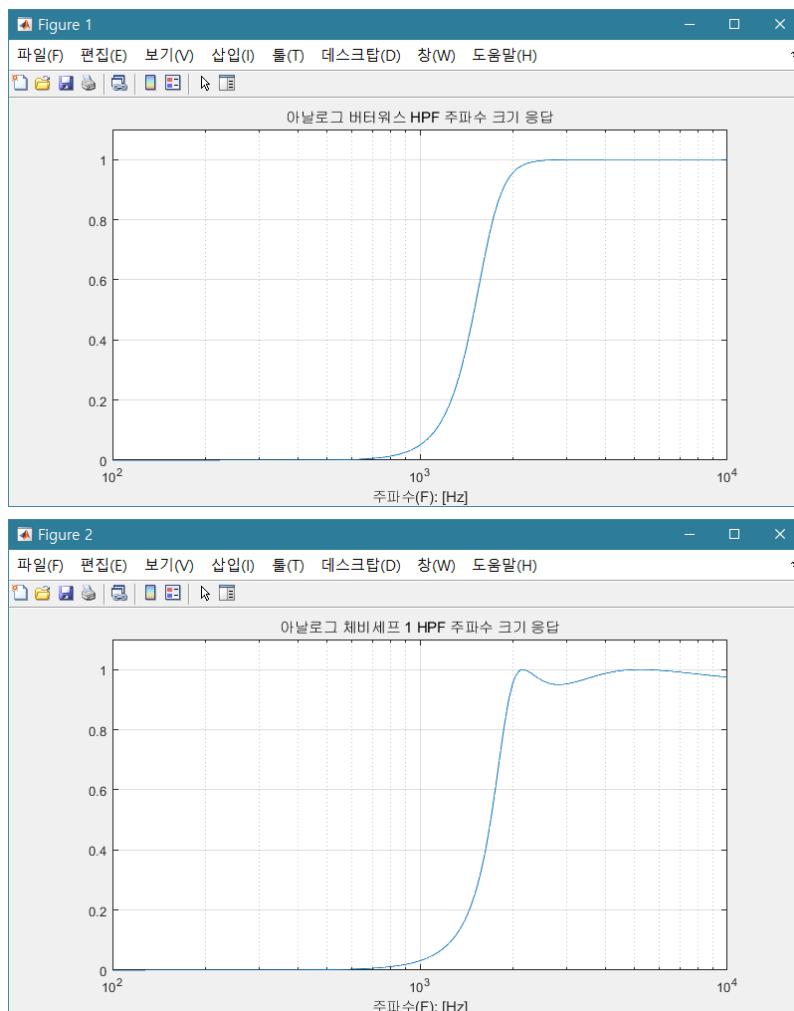
```
[N,Wn] = ellipord(Wp,Ws,Rp,Rs,'s')
```

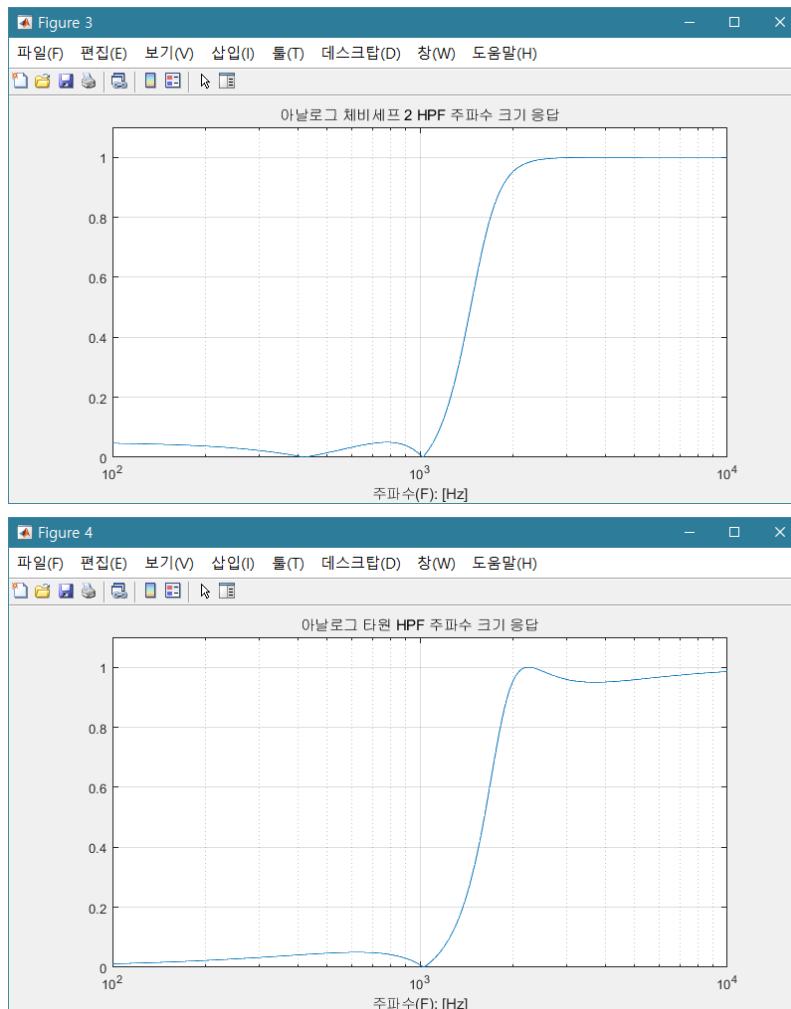
```
[b,a] = ellip(N,Rs,Wn,'high','s');
```

```
figure(4)
```

```
plot_analog_freq_resp(b,a,'타원','HPF')
```

✓ 주파수 응답 (크기)





✓ 필터 차수 비교

	Butterworth	Chebyshev 1	Chebyshev 2	Elliptic
Filter order	6	4	4	3

✚ 대역 통과 필터 설계

$$F_{s1} = 0.5 \text{ kHz}, \quad F_{p1} = 0.8 \text{ kHz}, \quad F_{p2} = 1.2 \text{ kHz}, \quad F_{s2} = 2 \text{ kHz}$$

$$\delta_p = 0.05, \quad \delta_s = 0.05$$

$$\Omega_p = 2\pi[F_{p1}, F_{p2}], \quad \Omega_s = 2\pi[F_{s1}, F_{s2}]$$

$$R_p = -20 \times \log_{10}(1 - \delta_p), \quad R_s = -20 \times \log_{10}(\delta_s)$$

```
%% Analog Bandpass Filter Design
```

```
clc; clear;
```

```
% 필터 사양
```

```
Fs1 = 500; Fp1 = 800; Fp2 = 1200; Fs2 = 2000;
```

```
dp = 0.05; ds = 0.05;
```

```
% 각 주파수와 감쇄율 (dB)
```

```
Wp = 2*pi*[Fp1,Fp2]; Ws = 2*pi*[Fs1,Fs2];
Rp = -db(1-dp); Rs = -db(ds)
```

% 버터워스 필터

```
[N,Wn] = buttord(Wp,Ws,Rp,Rs,'s')
[b,a] = butter(N,Wn,'bandpass','s');
figure(1)
plot_analog_freq_resp(b,a,'버터워스','BPF')
```

% 체비세프 1 필터

```
[N,Wn] = cheb1ord(Wp,Ws,Rp,Rs,'s')
[b,a] = cheby1(N,Rp,Wn,'bandpass','s');
figure(2)
plot_analog_freq_resp(b,a,'체비세프 1','BPF')
```

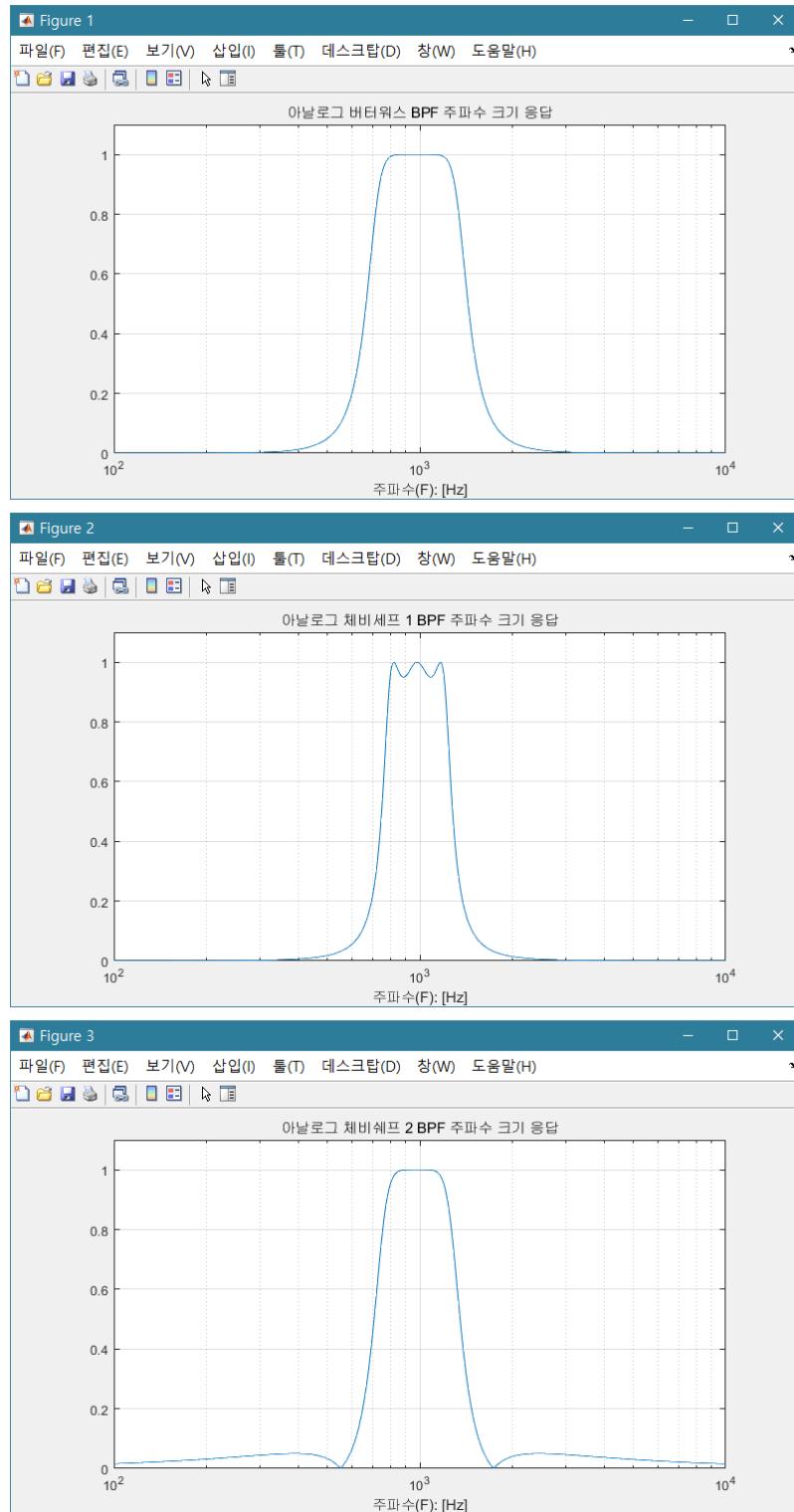
% 체비세프 2 필터

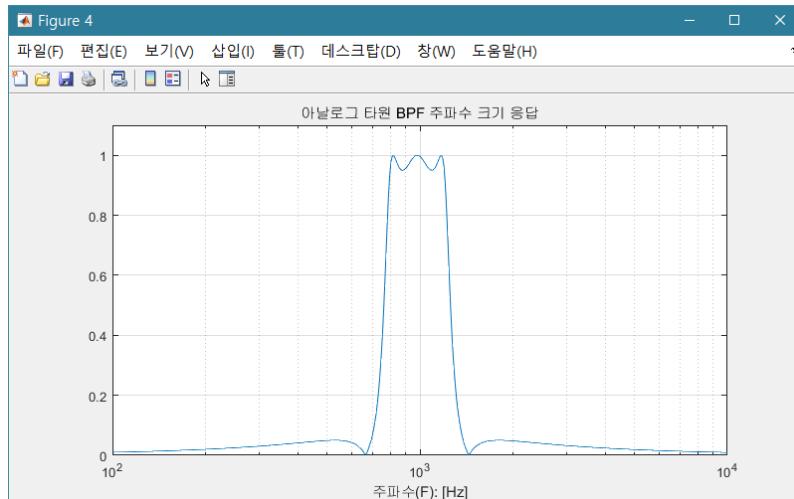
```
[N,Wn] = cheb2ord(Wp,Ws,Rp,Rs,'s')
[b,a] = cheby2(N,Rs,Wn,'bandpass','s');
figure(3)
plot_analog_freq_resp(b,a,'체비쉐프 2','BPF')
```

% 타원 필터

```
[N,Wn] = ellipord(Wp,Ws,Rp,Rs,'s')
[b,a] = ellip(N,Rp,Rs,Wn,'bandpass','s');
figure(4)
plot_analog_freq_resp(b,a,'타원','BPF')
```

- ✓ 주파수 응답 (크기)





✓ 필터 차수 비교

	Butterworth	Chebyshev 1	Chebyshev 2	Elliptic
Filter order	4	3	3	3

✚ 대역 저지 필터 설계

$$F_{p1} = 0.5 \text{ kHz}, \quad F_{p2} = 2 \text{ kHz}, \quad F_{s1} = 0.8 \text{ kHz}, \quad F_{s2} = 1.2 \text{ kHz}$$

$$\delta_p = 0.05, \quad \delta_s = 0.05$$

$$\Omega_p = 2\pi[F_{p1}, F_{p2}], \quad \Omega_s = 2\pi[F_{s1}, F_{s2}]$$

$$R_p = -20 \times \log_{10}(1 - \delta_p), \quad R_s = -20 \times \log_{10}(\delta_s)$$

```
%% Analog Bandstop Filter Design
```

```
clc; clear;
```

```
% 필터 사양
```

```
Fp1 = 500; Fs1 = 800; Fs2 = 1200; Fp2 = 2000;
```

```
dp = 0.05; ds = 0.05;
```

```
% 각 주파수와 감쇄율 (dB)
```

```
Wp = 2*pi*[Fp1,Fp2]; Ws = 2*pi*[Fs1,Fs2];
```

```
Rp = -db(1-dp); Rs = -db(ds)
```

```
% 버터워스 필터
```

```
[N,Wn] = buttord(Wp,Ws,Rp,Rs,'s')
```

```
[b,a] = butter(N,Wn,'stop','s');
```

```
figure(1)
```

```
plot_analog_freq_resp(b,a,'버터워스','BSF')
```

% 체비세프 1 필터

```
[N,Wn] = cheb1ord(Wp,Ws,Rp,Rs,'s')
[b,a] = cheby1(N,Rp,Wn,'stop','s');
figure(2)
plot_analog_freq_resp(b,a,'체비세프 1','BSF')
```

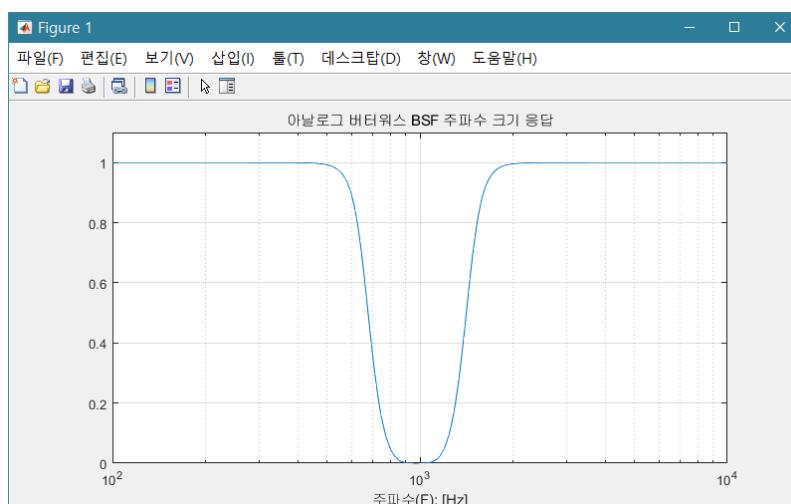
% 체비세프 2 필터

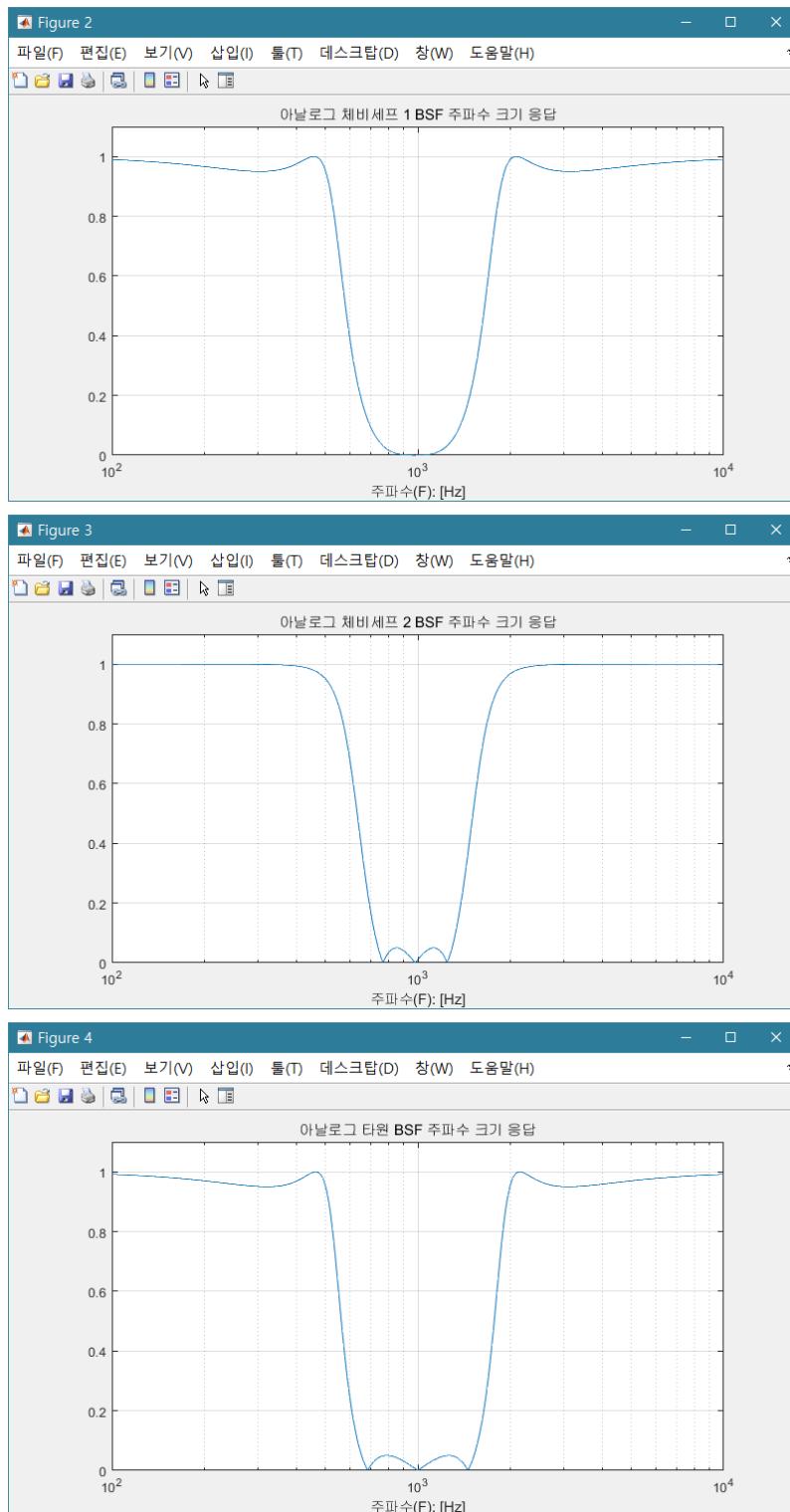
```
[N,Wn] = cheb2ord(Wp,Ws,Rp,Rs,'s')
[b,a] = cheby2(N,Rs,Wn,'stop','s');
figure(3)
plot_analog_freq_resp(b,a,'체비세프 2','BSF')
```

% 타원 필터

```
[N,Wn] = ellipord(Wp,Ws,Rp,Rs,'s')
[b,a] = ellip(N,Rp,Rs,Wn,'stop','s');
figure(4)
plot_analog_freq_resp(b,a,'타원','BSF')
```

✓ 주파수 응답 (크기)



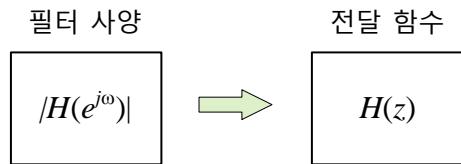


✓ 필터 차수 비교

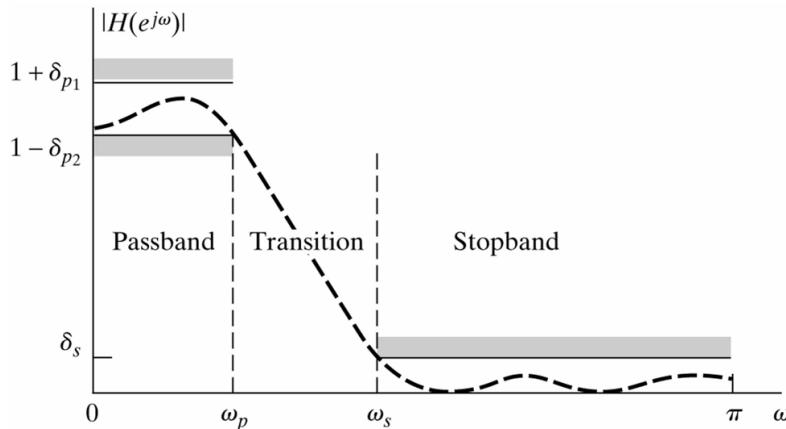
	Butterworth	Chebyshev 1	Chebyshev 2	Elliptic
Filter order	4	3	3	3

7.2.2 디지털 필터 설계

➊ 디지털 필터 설계



✓ 디지털 필터 사양 (LPF)



$$\delta_{p1}, \delta_{p2}, \delta_s, \omega_p, \omega_s \Rightarrow H(z) = \frac{B(z)}{A(z)}, H(z) = B(z)$$

- ✓ 일반적으로 $\delta_{p1} = \delta_{p2} = \delta_p$ 이다.
- ✓ 디지털 주파수는 기본적으로 선형 눈금이고, 범위는 $0 \sim 0.5$ (각 주파수는 $0 \sim \pi$)이다.
- ✓ 디지털 IIR 필터는 아날로그 IIR 필터 설계 방법을 이용한다.

▣ 디지털 IIR 필터 설계 함수

✓ 입/출력 변수

- ✗ $w_p = \omega_p / \pi, w_s = \omega_s / \pi, R_p = -20 \log_{10}(1 - \delta_p), R_s = -20 \log_{10}(\delta_s)$
- ✗ 함수에 사용된 wp, ws는 모두 정규화된 각 주파수이다.
- ✗ BPF와 BSF는 wp와 ws가 벡터이다.
- ✗ N : 필터의 차수, Wn : 차단 주파수
- ✗ ftype : 필터 종류, 문자열, 'low', 'high', 'bandpass', 'stop'

✓ 버터워스 LPF

```
[N, Wn] = buttord(wp, ws, Rp, Rs)
[b, a] = butter(N, Wn, ftype)
```

✓ 체비세프 1 LPF

```
[N,Wn] = cheb1ord(wp,ws,Rp,Rs)
[b,a] = cheby1(N,Rp,Wn,ftype)
```

- ✓ 체비세프 2 LPF

```
[N,Wn] = cheb2ord(wp,ws,Rp,Rs)
[b,a] = cheby2(N,Rs,Wn,ftype)
```

- ✓ 타원 LPF

```
[N,Wn] = ellipord(wp,ws,Rp,Rs)
[b,a] = ellip(N,Rp,Rs,Wn,ftype)
```

- ✚ 아래 예제에서 구한 디지털 필터의 주파수 크기 응답을 그리는 함수

- ✓ 함수 이름 : plot_digital_freq_resp.m

```
function plot_digital_freq_resp(b,a,str1,str2)
% 디지털 필터의 주파수 응답(크기)을 그리는 함수
w = linspace(0,pi,1025);
H = freqz(b,a,w);
Hm = abs(H);
plot(w/pi,Hm)
grid on
str = ['디지털 ',str1,' ',str2,' 주파수 크기 응답'];
title(str)
xlabel('정규화 주파수 (w/pi): [rad/s]')
ylim([0,1.1]);
```

- ✚ 저역 통과 필터 설계

$$\omega_p = 0.4\pi \text{ rad/s}, \quad \omega_s = 0.6\pi \text{ rad/s}, \quad \delta_p = 0.05, \quad \delta_s = 0.05$$

$$R_p = -20 \times \log_{10}(1 - \delta_p), \quad R_s = -20 \times \log_{10}(\delta_s)$$

```
%% Digital Lowpass Filter Design
clc; clear;
% 필터 사양
wp = 0.4*pi; ws = 0.6*pi;
dp = 0.05; ds = 0.05;
% 감쇄율 (dB)
```

```
Rp = -db(1-dp); Rs = -db(ds)
```

% 버터워스 필터

```
[N,Wn] = buttord(wp/pi,ws/pi,Rp,Rs)
[b,a] = butter(N,Wn);
figure(1)
plot_digital_freq_resp(b,a,'버터워스','LPF')
```

% 체비세프 1 필터

```
[N,Wn] = cheblord(wp/pi,ws/pi,Rp,Rs)
[b,a] = cheby1(N,Rp,Wn);
figure(2)
plot_digital_freq_resp(b,a,'체비세프 1','LPF')
```

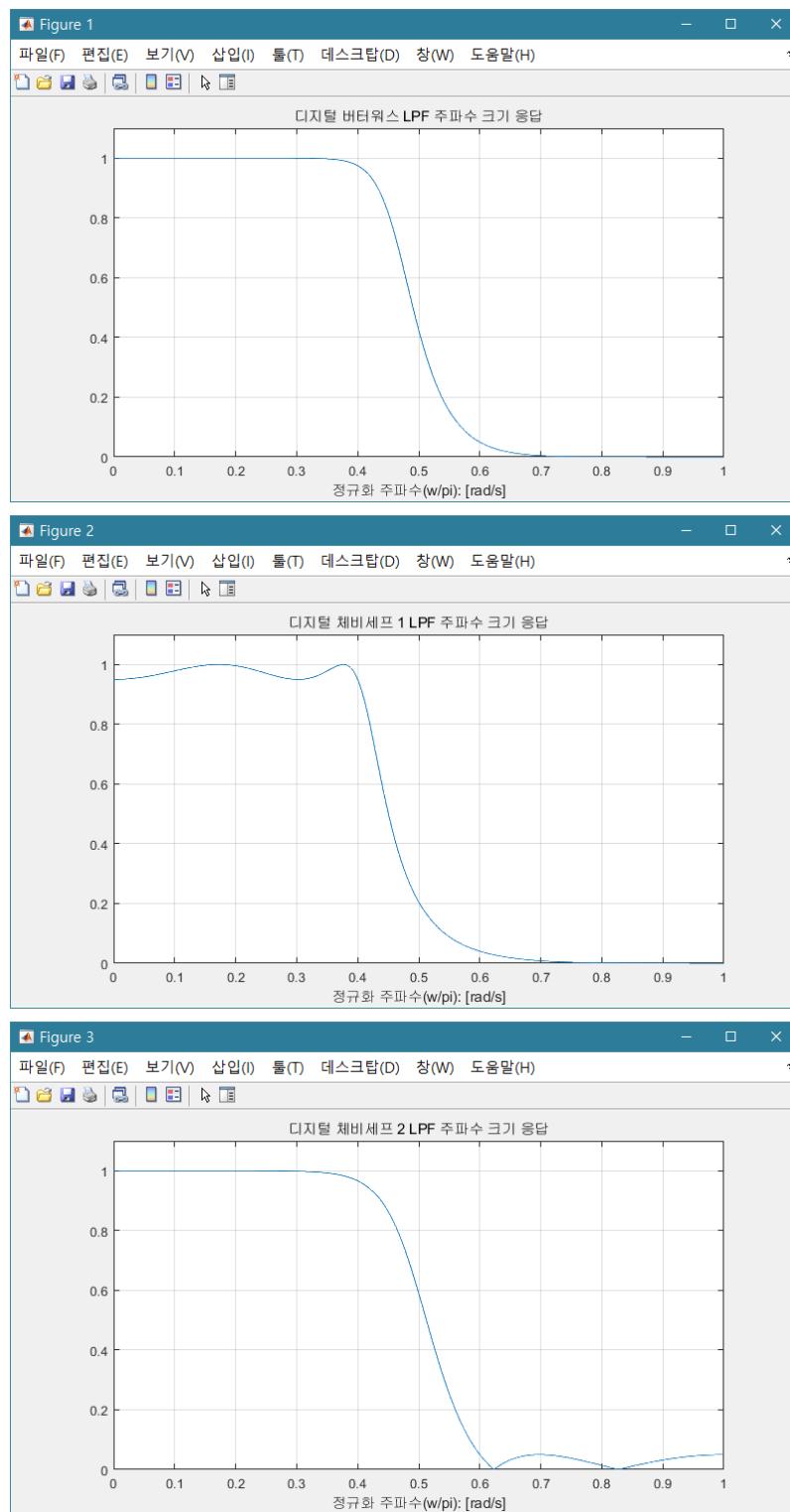
% 체비세프 2 필터

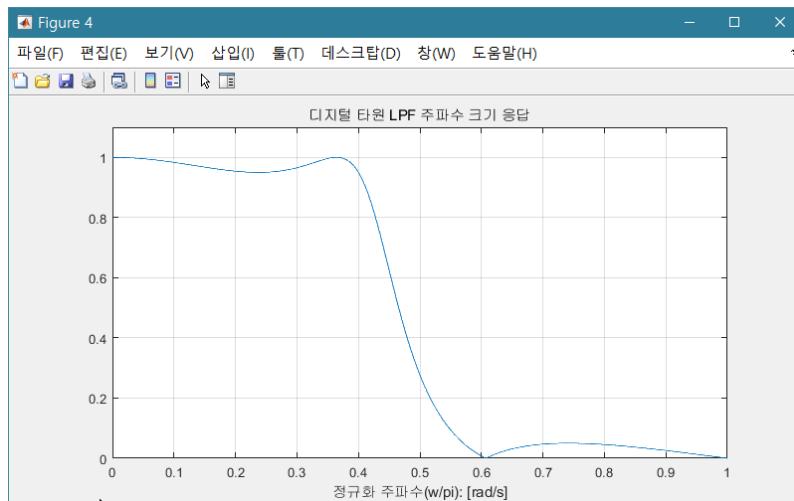
```
[N,Wn] = cheb2ord(wp/pi,ws/pi,Rp,Rs)
[b,a] = cheby2(N,Rs,Wn);
figure(3)
plot_digital_freq_resp(b,a,'체비세프 2','LPF')
```

% 타원 필터

```
[N,Wn] = ellipord(wp/pi,ws/pi,Rp,Rs)
[b,a] = ellip(N,Rp,Rs,Wn);
figure(4)
plot_digital_freq_resp(b,a,'타원','LPF')
```

✓ 주파수 응답 (크기)





✓ 필터 차수 비교

	Butterworth	Chebyshev 1	Chebyshev 2	Elliptic
Filter order	7	4	4	3

✚ 고역 통과 필터 설계

$$\omega_p = 0.6\pi \text{ rad/s}, \quad \omega_s = 0.4\pi \text{ rad/s}, \quad \delta_p = 0.05, \quad \delta_s = 0.05$$

$$R_p = -20 \times \log_{10}(1 - \delta_p), \quad R_s = -20 \times \log_{10}(\delta_s)$$

```
%% Digital Highpass Filter Design
clc; clear;

% 필터 사양
ws = 0.4*pi; wp = 0.6*pi;
ds = 0.05; dp = 0.05;
% 감쇄율 (dB)
Rp = -db(1-dp); Rs = -db(ds);

% 버터워스 필터
[N,Wn] = buttord(wp/pi, ws/pi, Rp, Rs)
[b,a] = butter(N,Wn, 'high');
figure(1)
plot_digital_freq_resp(b,a, '버터워스', 'HPF')

% 체비세프 1 필터
[N,Wn] = cheblord(wp/pi, ws/pi, Rp, Rs)
[b,a] = cheby1(N, Rp, Wn, 'high');
figure(2)
```

```
plot_digital_freq_resp(b,a,'체비세프 1','HPF')
```

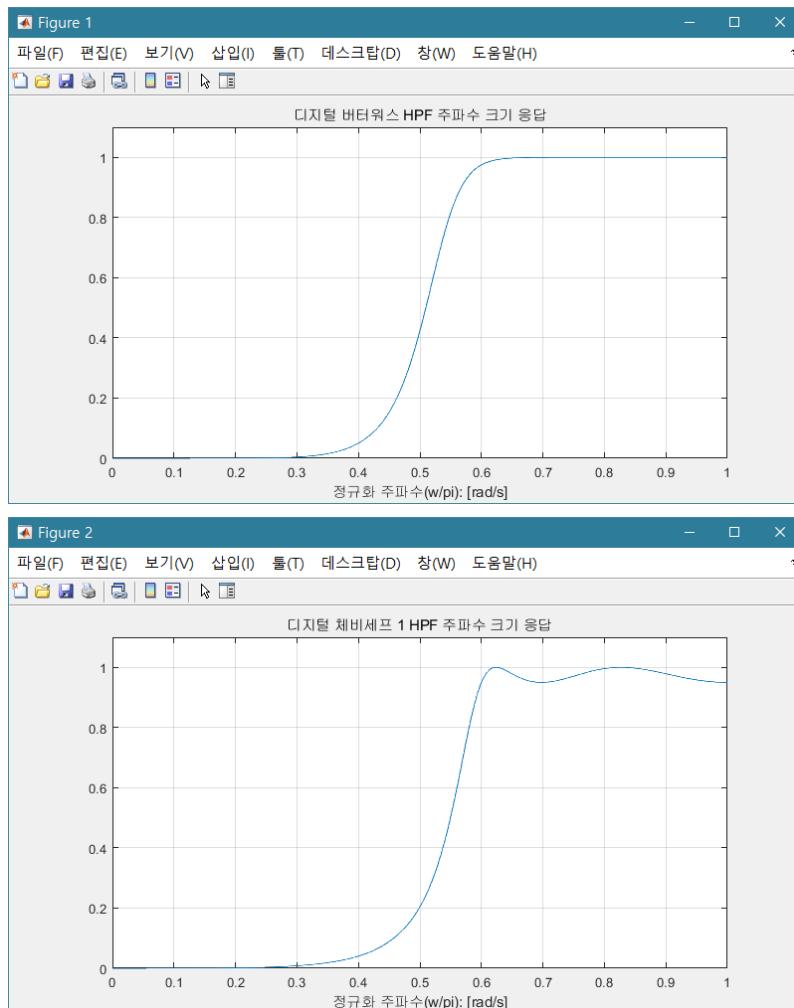
% 체비세프 2 필터

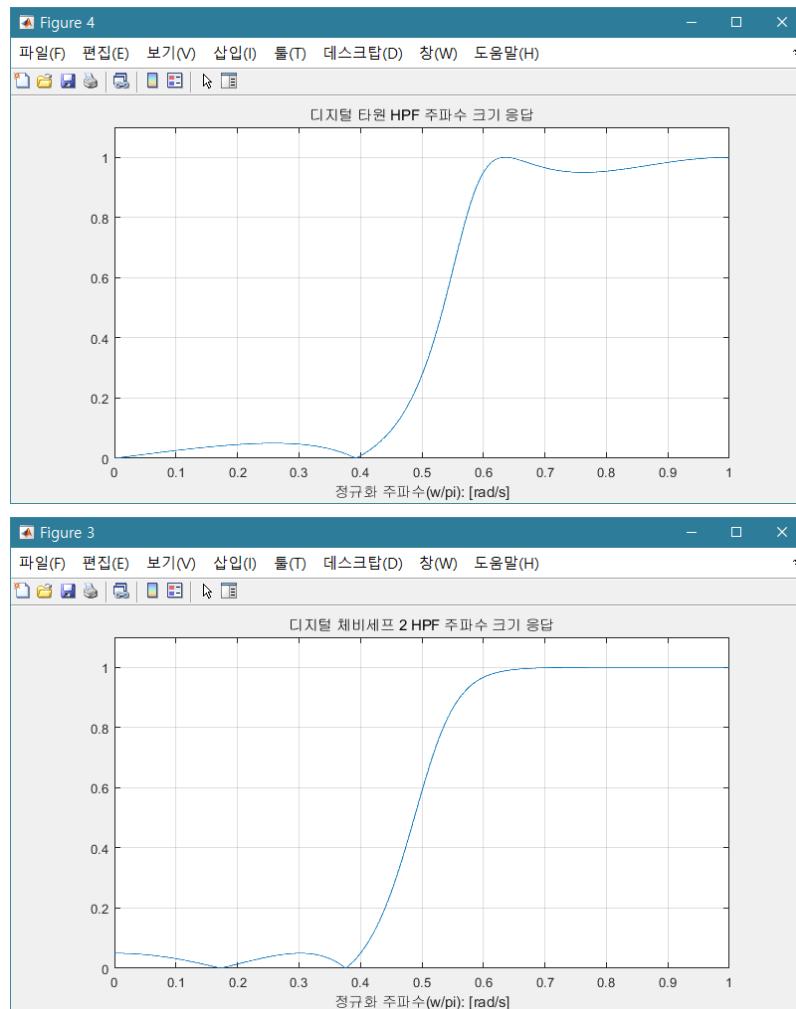
```
[N,Wn] = cheb2ord(wp/pi,ws/pi,Rp,Rs)
[b,a] = cheby2(N,Rs,Wn,'high');
figure(3)
plot_digital_freq_resp(b,a,'체비세프 2','HPF')
```

% 타원 필터

```
[N,Wn] = ellipord(wp/pi,ws/pi,Rp,Rs)
[b,a] = ellip(N,Rp,Rs,Wn,'high');
figure(4)
plot_digital_freq_resp(b,a,'타원','HPF')
```

✓ 주파수 응답 (크기)





✓ 필터 차수 비교

	Butterworth	Chebyshev 1	Chebyshev 2	Elliptic
Filter order	7	4	4	3

▶ 대역 통과 필터 설계

$$\omega_{p1} = 0.4\pi \text{ rad/s}, \quad \omega_{p2} = 0.6\pi \text{ rad/s}, \quad \omega_{s1} = 0.3\pi \text{ rad/s}, \quad \omega_{s2} = 0.7\pi \text{ rad/s}$$

$$\delta_p = 0.05, \quad \delta_s = 0.05$$

$$R_p = -20 \times \log_{10}(1 - \delta_p), \quad R_s = -20 \times \log_{10}(\delta_s)$$

```
%% Digital bandpass Filter Design
```

```
clc; clear;
```

% 필터 사양

```
ws1 = 0.3*pi; wp1 = 0.4*pi; wp2 = 0.6*pi; ws2 = 0.7*pi;
dp = 0.05; ds = 0.05;
```

% 각 주파수와 감쇄율 (dB)

```
wp = [wp1,wp2]; ws = [ws1,ws2];
Rp = -db(1-dp); Rs = -db(ds);
```

% 버터워스 필터

```
[N,Wn] = buttord(wp/pi,ws/pi,Rp,Rs)
[b,a] = butter(N,Wn,'bandpass');
figure(1)
plot_digital_freq_resp(b,a,'버터워스','BPF')
```

% 체비세프 1 필터

```
[N,Wn] = cheb1ord(wp/pi,ws/pi,Rp,Rs)
[b,a] = cheby1(N,Rp,Wn,'bandpass');
figure(2)
plot_digital_freq_resp(b,a,'체비세프 1','BPF')
```

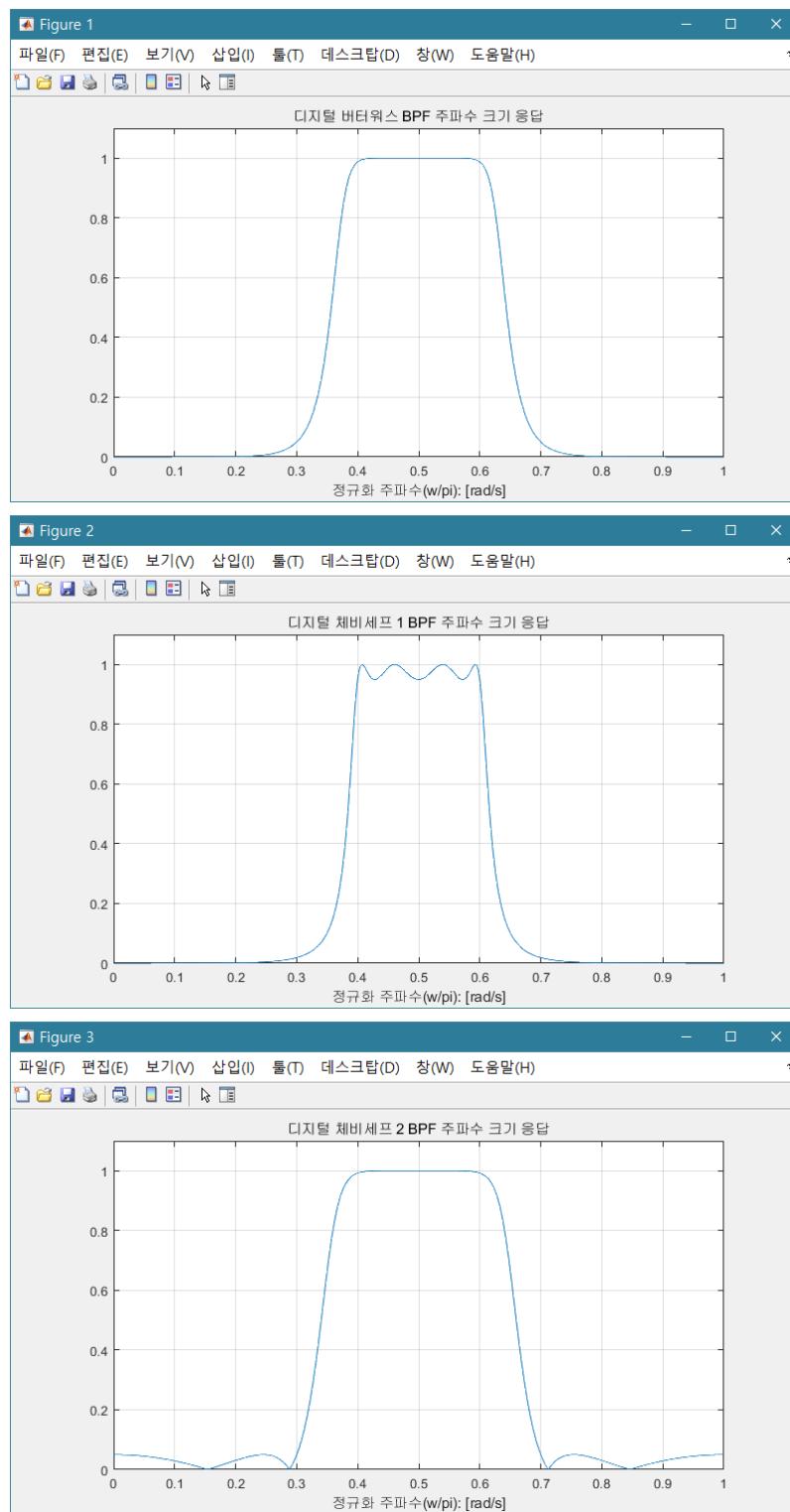
% 체비세프 2 필터

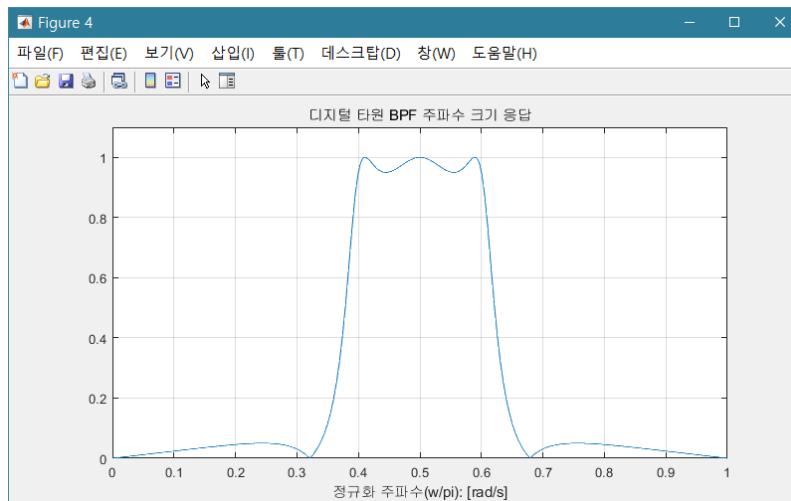
```
[N,Wn] = cheb2ord(wp/pi,ws/pi,Rp,Rs)
[b,a] = cheby2(N,Rs,Wn,'bandpass');
figure(3)
plot_digital_freq_resp(b,a,'체비세프 2','BPF')
```

% 타원 필터

```
[N,Wn] = ellipord(wp/pi,ws/pi,Rp,Rs)
[b,a] = ellip(N,Rp,Rs,Wn,'bandpass');
figure(4)
plot_digital_freq_resp(b,a,'타원','BPF')
```

✓ 주파수 응답 (크기)





✓ 필터 차수 비교

	Butterworth	Chebyshev 1	Chebyshev 2	Elliptic
Filter order	6	4	4	3

✚ 대역 저지 필터 설계

$$\omega_{p1} = 0.3\pi \text{ rad/s}, \quad \omega_{p2} = 0.7\pi \text{ rad/s}, \quad \omega_{s1} = 0.4\pi \text{ rad/s}, \quad \omega_{s2} = 0.6\pi \text{ rad/s}$$

$$\delta_p = 0.05, \quad \delta_s = 0.05$$

$$R_p = -20 \times \log_{10}(1 - \delta_p), \quad R_s = -20 \times \log_{10}(\delta_s)$$

```

%% Digital Bandstop Filter Design
clc; clear;

% 필터 사양
wp1 = 0.3*pi; ws1 = 0.4*pi; ws2 = 0.6*pi; wp2 = 0.7*pi;
dp = 0.05; ds = 0.05;
% 각 주파수와 감쇄율 (dB)
wp = [wp1,wp2]; ws = [ws1,ws2];
Rp = -db(1-dp); Rs = -db(ds);

% 버터워스 필터
[N,Wn] = buttord(wp/pi,ws/pi,Rp,Rs)
[b,a] = butter(N,Wn,'stop');
figure(1)
plot_digital_freq_resp(b,a,'버터워스','BSF')

% 체비세프 1 필터
[N,Wn] = cheblord(wp/pi,ws/pi,Rp,Rs)

```

```
[b,a] = cheby1(N,Rp,Wn,'stop');

figure(2)
plot_digital_freq_resp(b,a,'체비세프 1','BSF')
```

% 체비세프 2 필터

```
[N,Wn] = cheb2ord(wp/pi,ws/pi,Rp,Rs)

[b,a] = cheby2(N,Rs,Wn,'stop');

figure(3)
plot_digital_freq_resp(b,a,'체비세프 2','BSF')
```

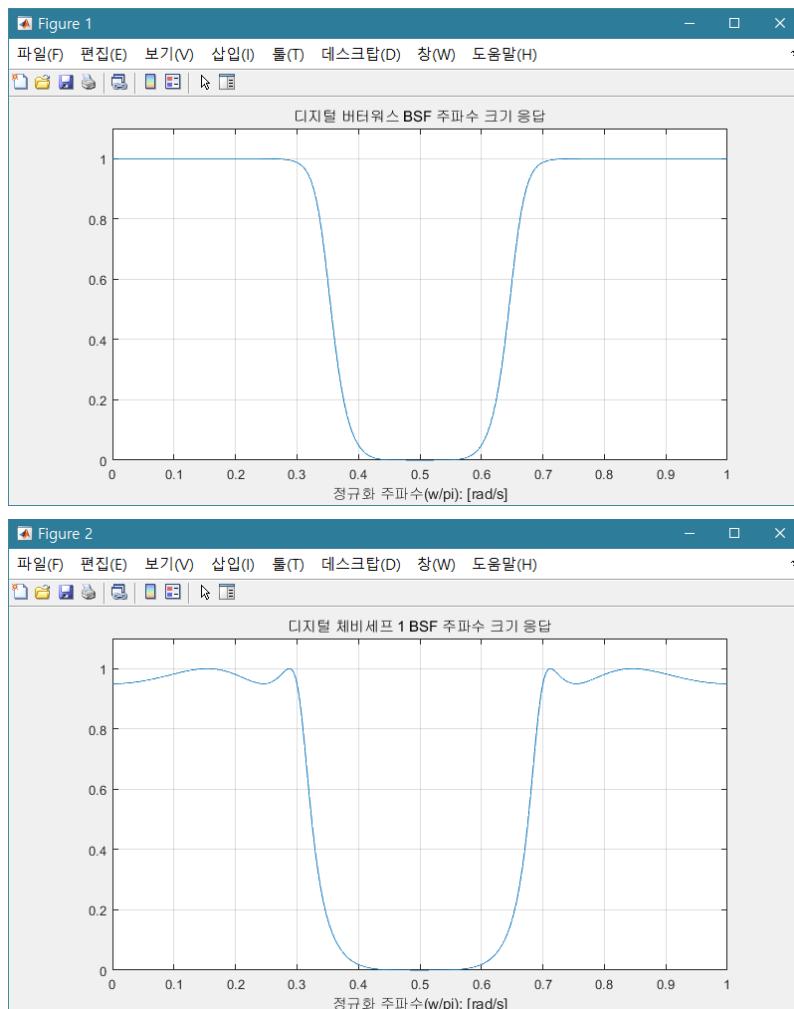
% 타원 필터

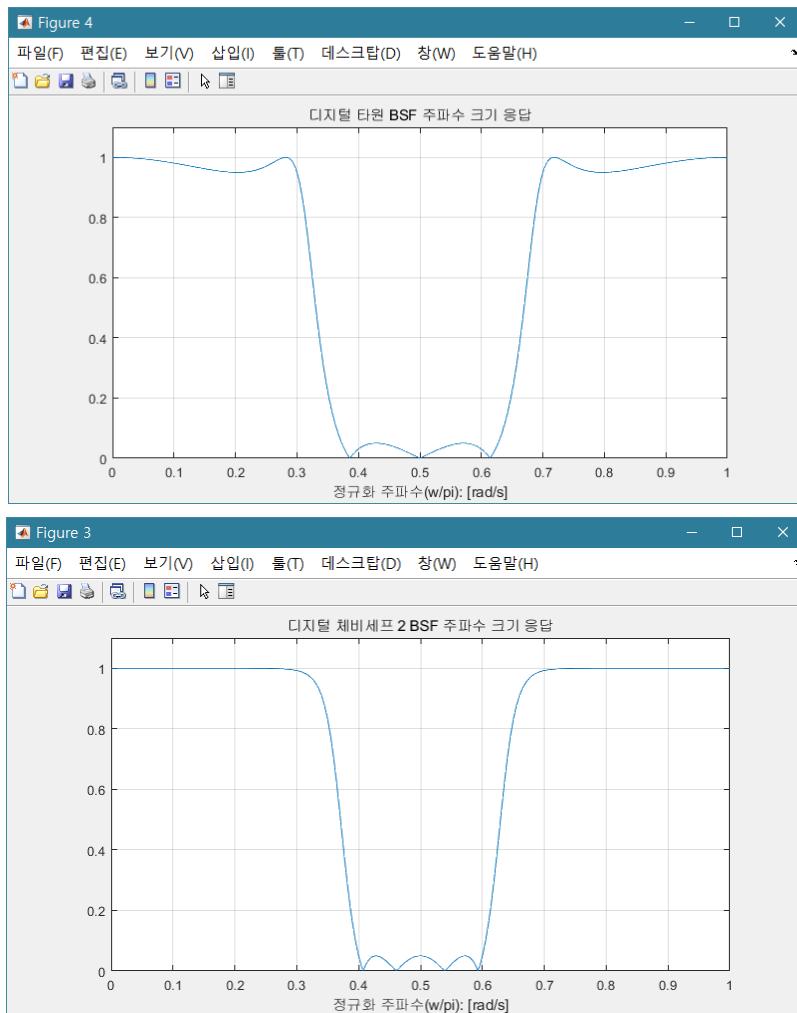
```
[N,Wn] = ellipord(wp/pi,ws/pi,Rp,Rs)

[b,a] = ellip(N,Rp,Rs,Wn,'stop');

figure(4)
plot_digital_freq_resp(b,a,'타원','BSF')
```

✓ 주파수 응답 (크기)





✓ 필터 차수 비교

	Butterworth	Chebyshev 1	Chebyshev 2	Elliptic
Filter order	6	4	4	3

▣ 디지털 FIR 필터 설계 함수

✓ 입/출력 변수

- ✗ f : 정규화된 각 주파수 벡터 (ω 를 π 로 나눈 값)
- ✗ a : 각 대역의 이상적인 크기 값 벡터
- ✗ dev : 각 대역의 오차 벡터

✓ 필터 종류에 따른 입력 변수

- ✗ LPF : $f = [\omega_p, \omega_s]/\pi$, $a = [1, 0]$, $dev = [\delta_p, \delta_s]$
- ✗ HPF : $f = [\omega_s, \omega_p]/\pi$, $a = [0, 1]$, $dev = [\delta_s, \delta_p]$
- ✗ BPF : $f = [\omega_{s1}, \omega_{p1}, \omega_{p2}, \omega_{s2}]/\pi$, $a = [0, 1, 0]$, $dev = [\delta_{s1}, \delta_p, \delta_{s2}]$
- ✗ BSF : $f = [\omega_{p1}, \omega_{s1}, \omega_{s2}, \omega_{p2}]/\pi$, $a = [1, 0, 1]$, $dev = [\delta_{p1}, \delta_s, \delta_{p2}]$

✓ 카이저 창 필터 설계 함수

```
[N,Wn,beta,ftype] = kaiserord(f,a,dev)
```

```
w = Kaiser(L,beta)
```

✖ L = N+1

```
b = fir1(N,Wn,ftype>window)
```

✖ window는 창 함수로 위에서 구한 카이저 창을 사용한다.

```
b = fir1(N,Wn,ftype>window,scaleopt)
```

✖ scaleopt : FIR 필터는 일반적으로 통과 대역의 범위가 $1+\delta_p \sim 1-\delta_p$ 이므로 가장 큰 값을 1로 맞추기 위해서 사용하는 변수, 'scale' 이면 1로 맞추고 'noscale'이면 그대로 둔다. 'scale'이 default이다.

✓ PM 최적 필터 설계 함수

```
[N,fo,ao,ftype] = firpmord(f,a,dev)
```

```
b = firpm(N,fo,ao)
```

▶ 아래 예제에서 구한 디지털 필터의 주파수 크기 응답을 그리는 함수

✓ 함수 이름 : plot_digital_freq_resp_dB.m

```
function plot_digital_freq_resp_dB(b,a,str1,str2)
% 디지털 필터의 주파수 응답(크기, dB)을 그리는 함수
w = linspace(0,pi,1025);
H = freqz(b,a,w);
Hm = abs(H);
HmdB = db(Hm);
plot(w/pi,HmdB)
grid on
str = ['디지털 ',str1,' ',str2,' 주파수 크기 응답'];
title(str)
xlabel('정규화 주파수(w/pi): [rad/s]')
ylim([-80,10]);
```

▶ 저역 통과 필터 설계

$$\omega_p = 0.4\pi \text{ rad/s}, \quad \omega_s = 0.5\pi \text{ rad/s}, \quad \delta_p = 0.01, \quad \delta_s = 0.01$$

```

%% Digital FIR Lowpass Filter Design

clc; clear;

% 필터 사양

wp = 0.4*pi; ws = 0.5*pi;
dp = 0.01; ds = 0.01;

% 필터 설계 파라미터

fc = [wp,ws]/pi;
a = [1,0];
dev = [dp,ds];

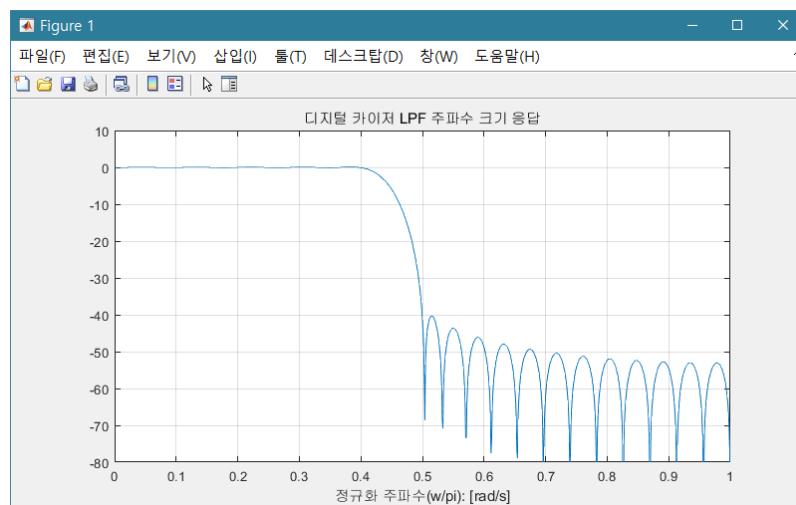
% 카이저 창 필터 설계

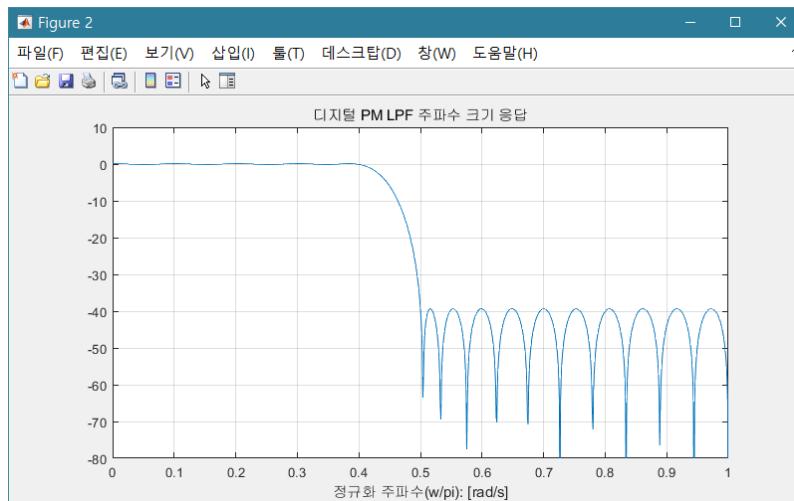
[N,Wn,beta,ftype] = kaiserord(fc,a,dev);
wk = kaiser(N+1,beta);
h = fir1(N,Wn,ftype,wk,'noscale');
figure(1)
plot_digital_freq_resp_dB(h,1,'카이저','LPF')

% Parks-McClellan 최적 필터 설계

[N,fo,ao,w] = firpmord(fc,a,dev);
h = firpm(N,fo,ao,w);
figure(2)
plot_digital_freq_resp_dB(h,1,'PM','LPF')

```





✓ 필터 차수 비교

	Kaiser window	PM optimal
Filter order	45	39

✚ 고역 통과 필터 설계

$$\omega_p = 0.6\pi \text{ rad/s}, \quad \omega_s = 0.5\pi \text{ rad/s}, \quad \delta_p = 0.01, \quad \delta_s = 0.01$$

```
% Digital FIR HighPass Filter Design
```

```
clc; clear;
% 필터 사양
ws = 0.5*pi; wp = 0.6*pi;
dp = 0.01; ds = 0.01;
% 필터 설계 파라미터
fc = [ws,wp]/pi;
a = [0,1];
dev = [ds,dp];
```

% 카이저 창 필터 설계

```
[N,Wn,beta,ftype] = kaiserord(fc,a,dev);
wk = kaiser(N+1,beta);
h = fir1(N,Wn,ftype,wk,'noscale');
figure(1)
plot_digital_freq_resp_dB(h,1,'카이저','HPF')
```

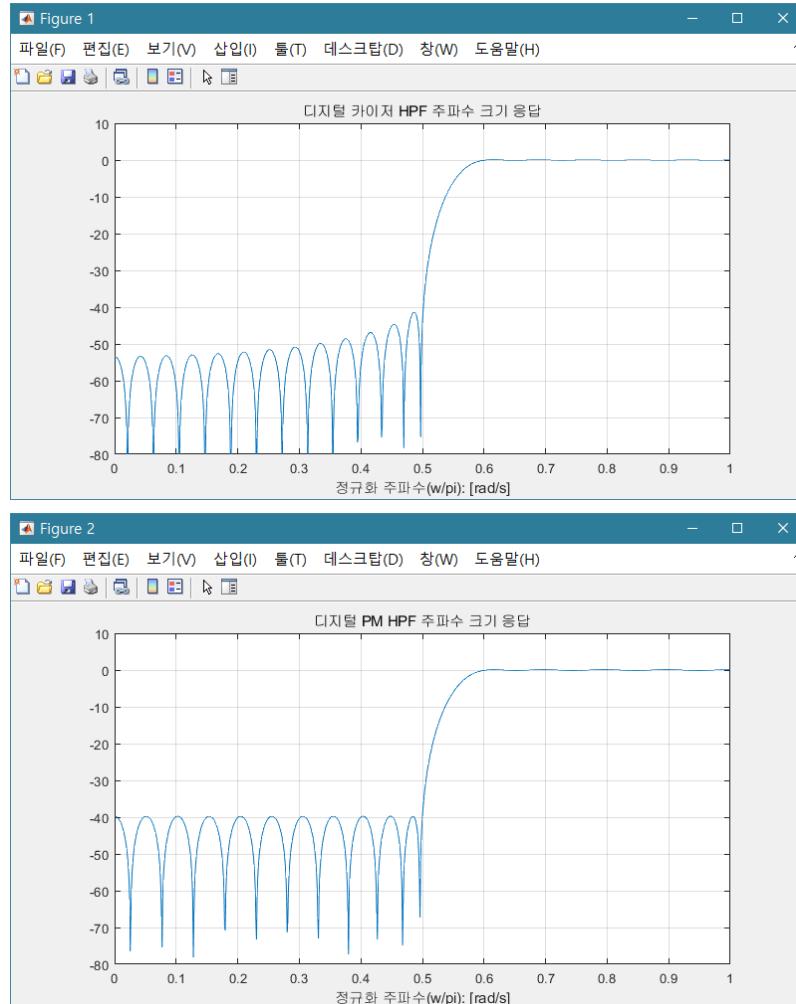
% Parks-McClellan 최적 필터 설계

```
[N,fo,ao,w] = firpmord(fc,a,dev);
```

```

h = firpm(N, fo, ao, w);
figure(2)
plot_digital_freq_resp_dB(h, 1, 'PM', 'HPF')

```



✓ 필터 차수 비교

	Kaiser window	PM optimal
Filter order	46	40

✚ 대역 통과 필터 설계

$$\omega_{p1} = 0.4\pi \text{ rad/s}, \quad \omega_{p2} = 0.6\pi \text{ rad/s}, \quad \omega_{s1} = 0.3\pi \text{ rad/s}, \quad \omega_{s2} = 0.7\pi \text{ rad/s}$$

$$\delta_p = 0.01, \quad \delta_{s1} = 0.01, \quad \delta_{s2} = 0.001$$

```

%% Digital FIR Bandpass Filter Design
clc; clear;
% 필터 사양
ws1 = 0.3*pi; wp1 = 0.4*pi; wp2 = 0.6*pi; ws2 = 0.7*pi;
ds1 = 0.01; dp = 0.01; ds2 = 0.001;

```

```
% 필터 설계 파라미터
```

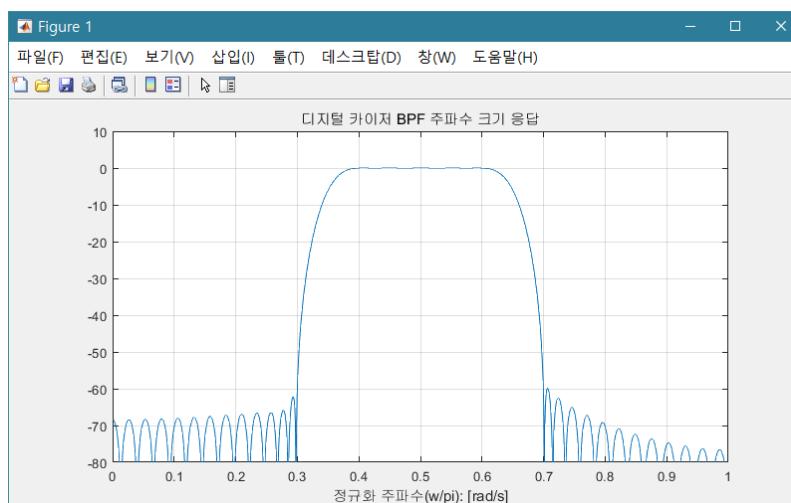
```
fc = [ws1,wp1,wp2,ws2]/pi;
a = [0,1,0];
dev = [ds1,dp,ds2];
```

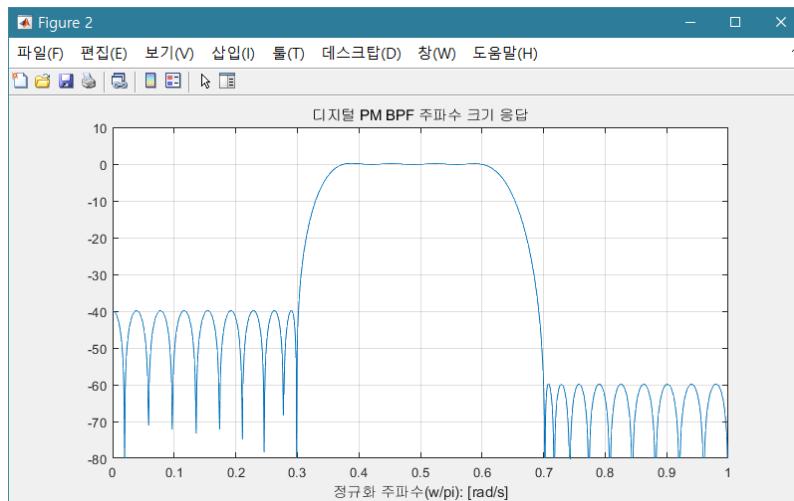
```
% 카이저 창 필터 설계
```

```
[N,Wn,beta,ftype] = kaiserord(fc,a,dev)
wk = kaiser(N+1,beta);
h = fir1(N,Wn,ftype,wk,'noscale');
figure(1)
plot_digital_freq_resp_dB(h,1,'카이저','BPF')
```

```
% Parks-McClellan 최적 필터 설계
```

```
[N,fo,ao,w] = firpmord(fc,a,dev)
h = firpm(N,fo,ao,w);
figure(2)
plot_digital_freq_resp_dB(h,1,'PM','BPF')
```





✓ 필터 차수 비교

	Kaiser window	PM optimal
Filter order	73	51

✚ 대역 저지 필터 설계

$$\omega_{p1} = 0.3\pi \text{ rad/s}, \quad \omega_{p2} = 0.7\pi \text{ rad/s}, \quad \omega_{s1} = 0.4\pi \text{ rad/s}, \quad \omega_{s2} = 0.6\pi \text{ rad/s}$$

$$\delta_{p1} = 0.01, \quad \delta_{p2} = 0.001, \quad \delta_s = 0.01$$

```

%% Digital FIR Bandstop Filter Design
clc; clear;

% 필터 사양
wp1 = 0.3*pi; wp2 = 0.7*pi; ws1 = 0.4*pi; ws2 = 0.6*pi;
dp1 = 0.01; ds = 0.01; dp2 = 0.001;

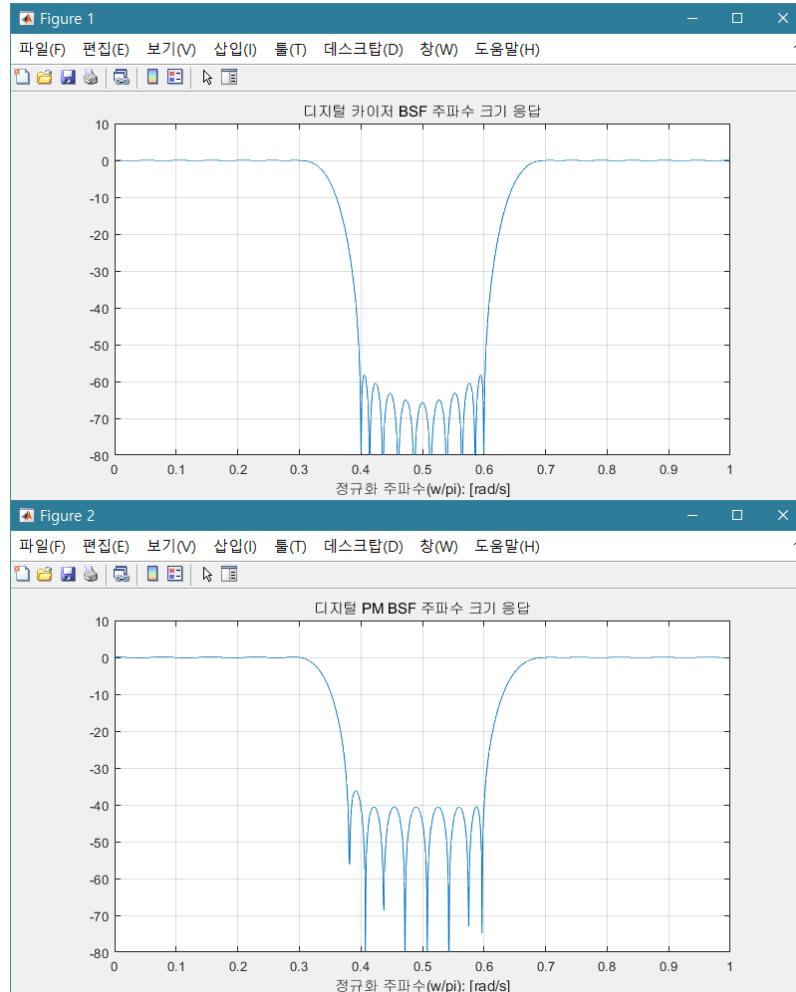
% 필터 설계 파라미터
fc = [wp1,ws1,ws2,wp2]/pi;
a = [1,0,1];
dev = [dp1,ds,dp2];

% 카이저 창 필터 설계
[N,Wn,beta,ftype] = kaiserord(fc,a,dev)
wk = kaiser(N+1,beta);
h = fir1(N,Wn,ftype,wk,'noscale');
figure(1)
plot_digital_freq_resp_dB(h,1,'카이저','BSF')

```

% Parks-McClellan 필터 설계

```
[N,fo,ao,w] = firpmord(fc,a,dev)
h = firpm(N,fo,ao,w);
figure(2)
plot_digital_freq_resp_dB(h,1,'PM','BSF')
```



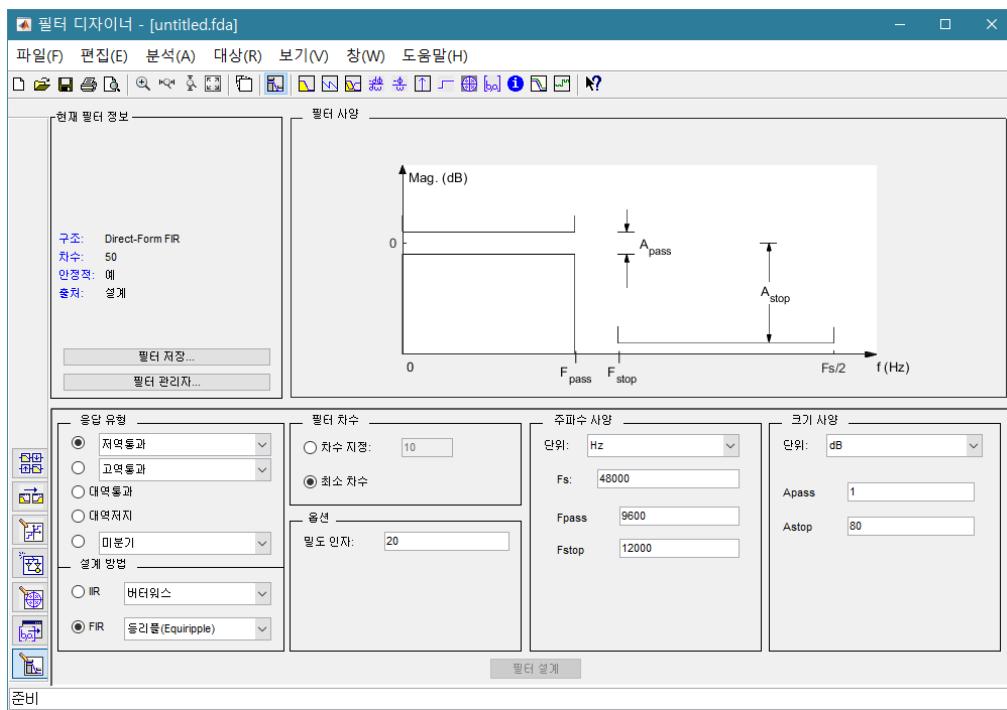
✓ 필터 차수 비교

	Kaiser window	PM optimal
Filter order	74	52

7.2.3 필터 설계 GUI

filterDesigner 함수

✓ 명령창에서 filterDesigner 실행



1) 아날로그 IIR 버터워스 LPF 설계

필터 사양

- ✓ 통과 대역 주파수 : $F_p = 1 \text{ kHz}$
- ✓ 저지 대역 주파수 : $F_s = 1.5 \text{ kHz}$
- ✓ 통과 대역 감쇄 : $A_p = 1 \text{ dB}$
- ✓ 저지 대역 감쇄 : $A_s = 80 \text{ dB}$
- ✓ 샘플링 주파수 : $F_{\text{samp}} = 8 \text{ kHz}$

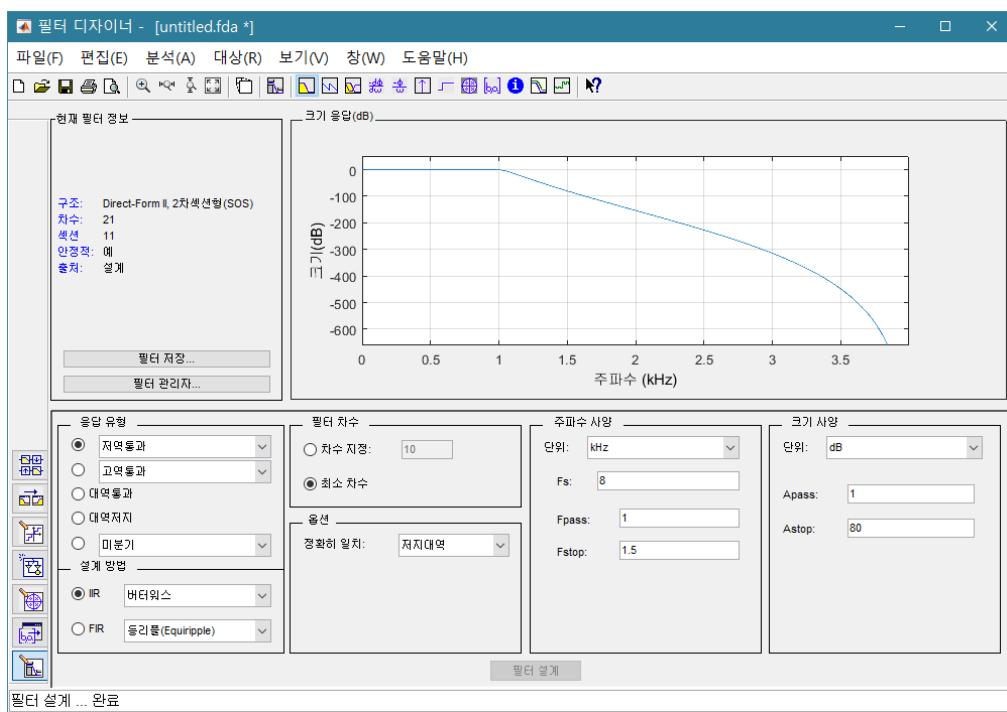
filterDesigner 파라미터

- ✓ 응답 유형 : 저역 통과
- ✓ 설계 방법 : IIR – 버터워스
- ✓ 필터 차수 : 최소 차수
- ✓ 옵션 : 정확히 일치 – 저지 대역
- ✓ 주파수 사양 : 단위 – kHz, $f_s = 8$, $F_{\text{pass}} = 1$, $F_{\text{stop}} = 1.5$
- ✓ 크기 사양 : 단위 – dB, $A_{\text{pass}} = 1$, $A_{\text{stop}} = 80$

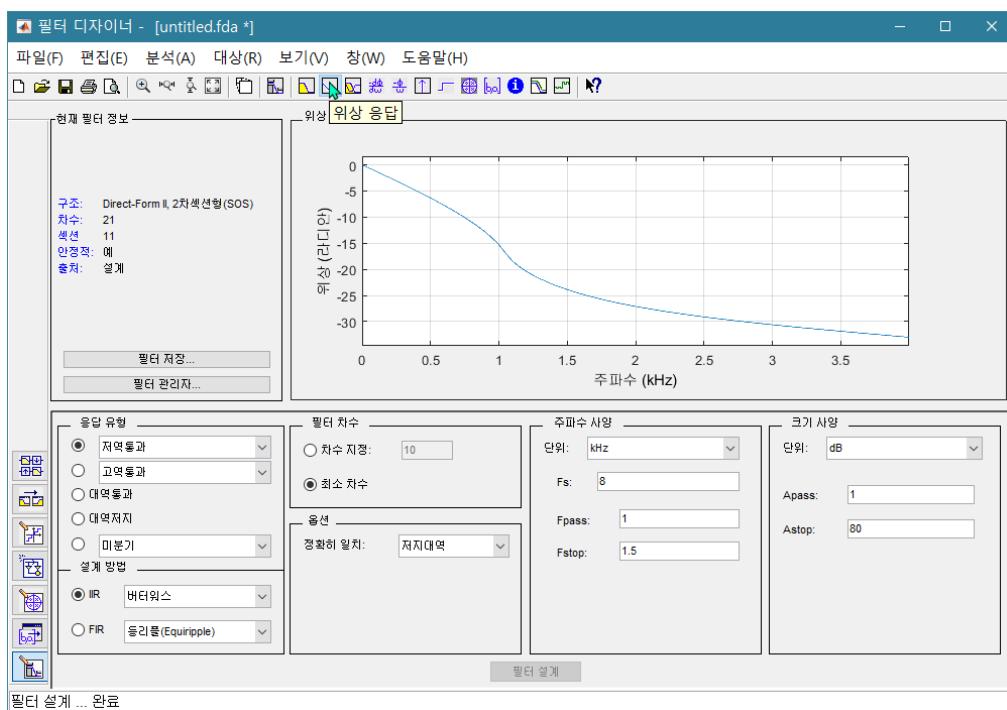
설계 실행

- ✓ 필터 설계 버튼 클릭

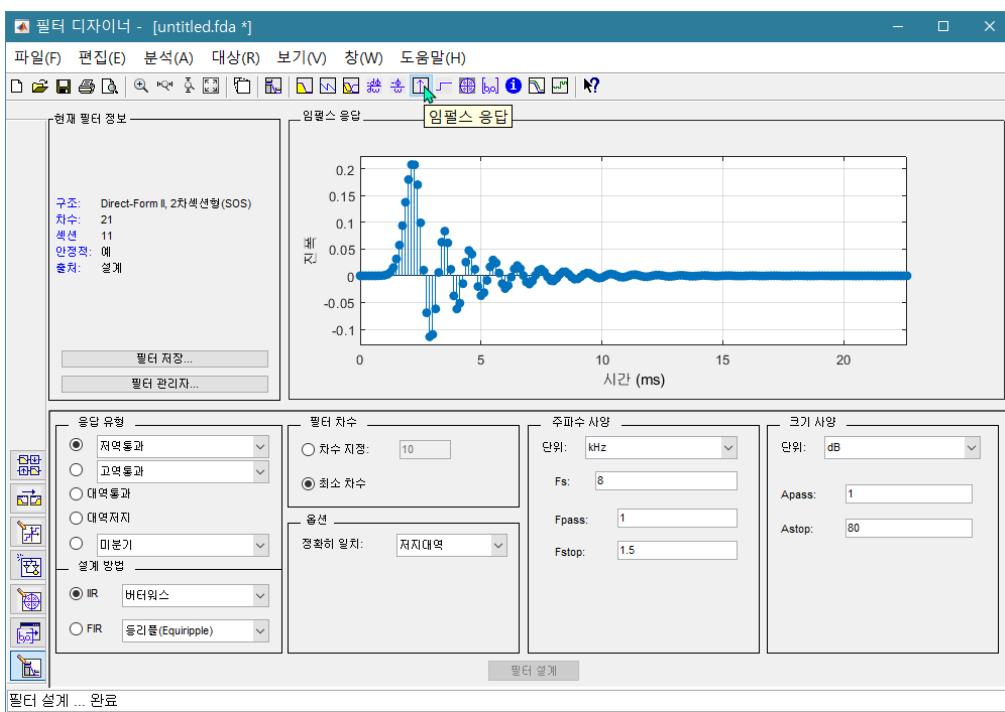
필터 크기 특성



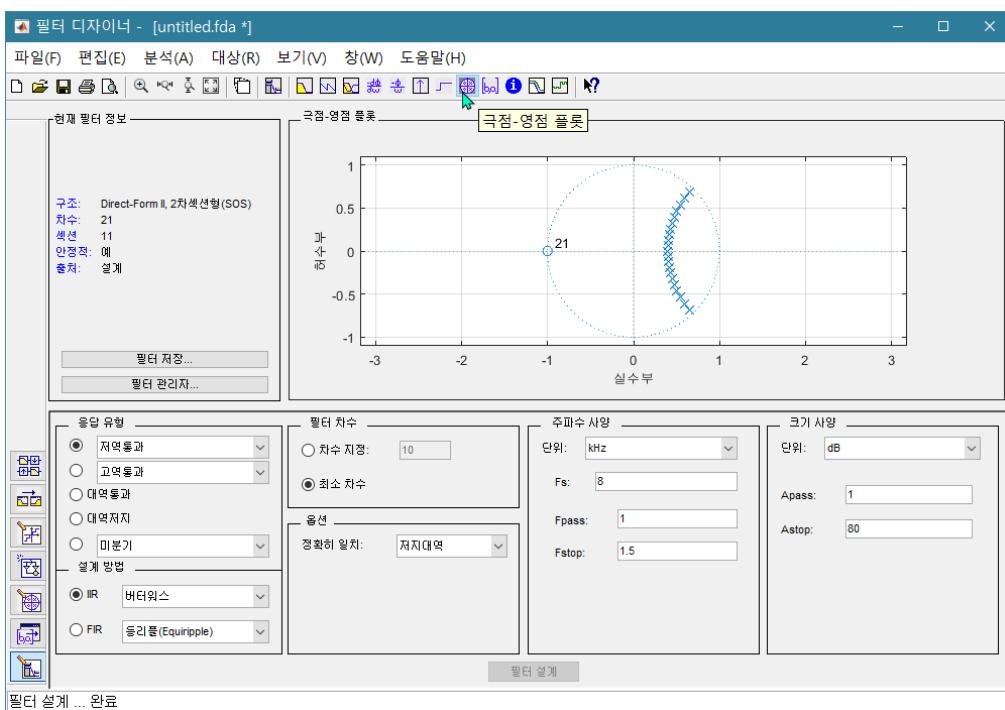
필터 위상 특성



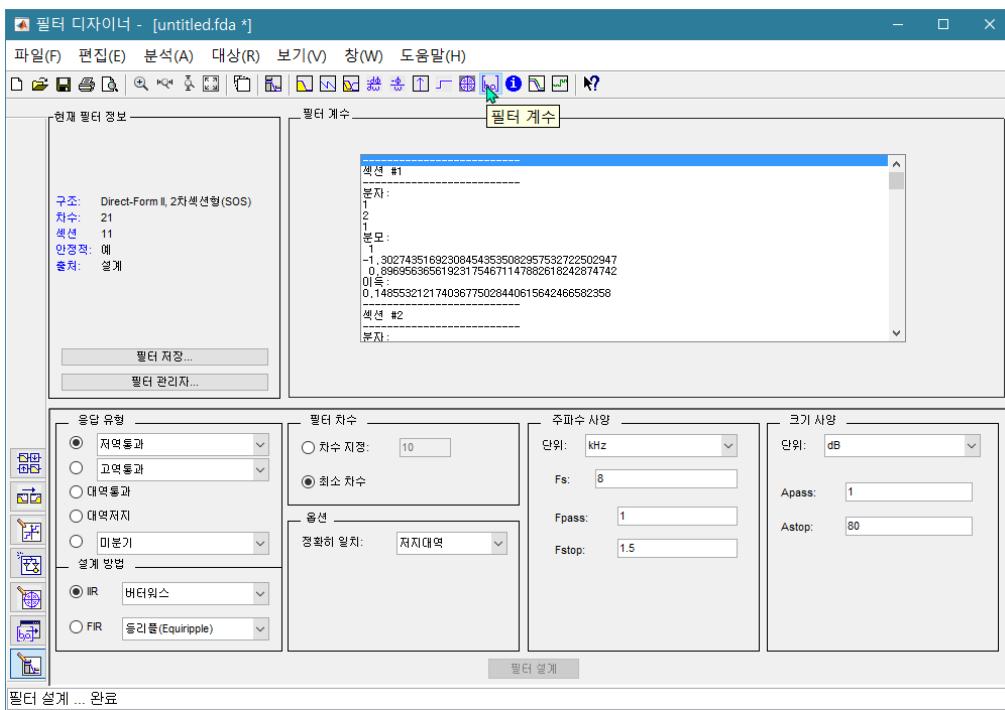
임펄스 응답



▣ 극-영점도

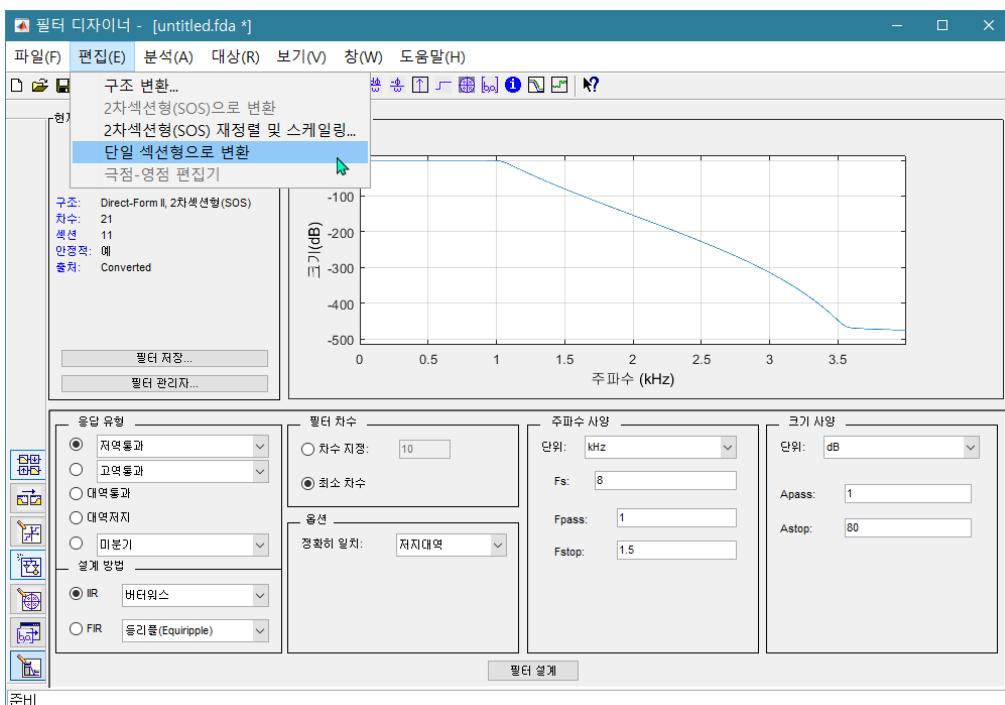


▣ 필터 계수

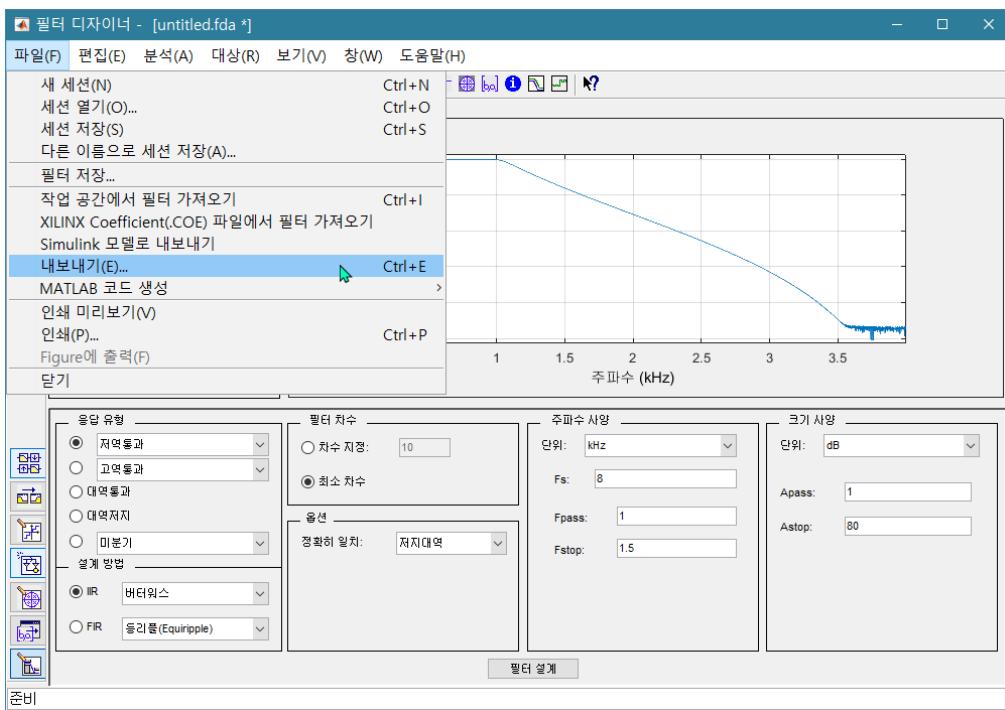


필터 계수 내보내기

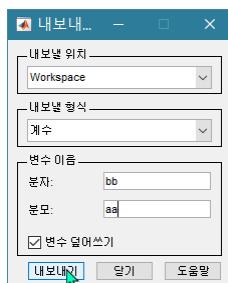
- ✓ 필터 계수는 필터 구조에 따라 달라진다.
- ✓ 필터 설계에서 익숙한 전달 함수의 분모, 분자 (b,a)의 계수를 얻으려면 필터 구조를 단일 섹션형으로 바꾸어야 한다.



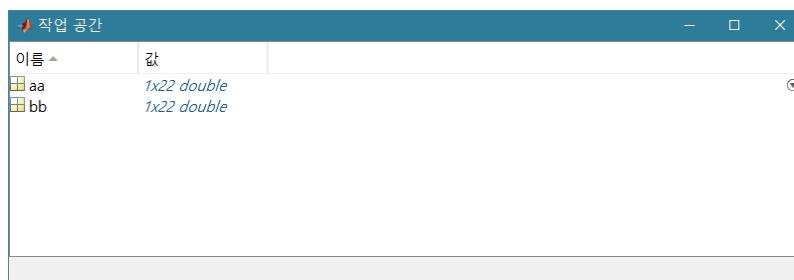
- ✓ 파일 – 내보내기 메뉴를 선택하여 내보내기 창을 연다.



- ✓ 문자와 분모에 원하는 변수 이름을 적고 내보내기를 클릭한다.



- ✓ 작업 공간에 필터 계수 변수가 보인다.



- ✓ 저장된 필터 계수로 그린 주파수 응답

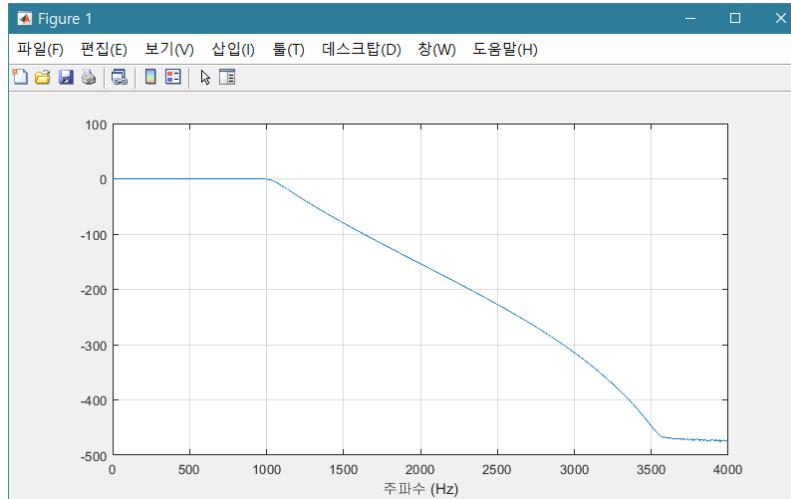
```

clc; % clear 사용 금지
Fs = 8000; % 샘플링 주파수
w = linspace(0,pi,1001);
H = freqz(bb,aa,w);
Hm = abs(H);
HmdB = db(Hm);
f = w*Fs/(2*pi); % 디지털 각 주파수를 아날로그 주파수로 변환

```

```
figure(1)
plot(f, HmdB)
grid on
xlabel('주파수 (Hz)')
```

✓ 주파수 크기 특성



2) 디지털 IIR 체비세프 2 HPF 설계

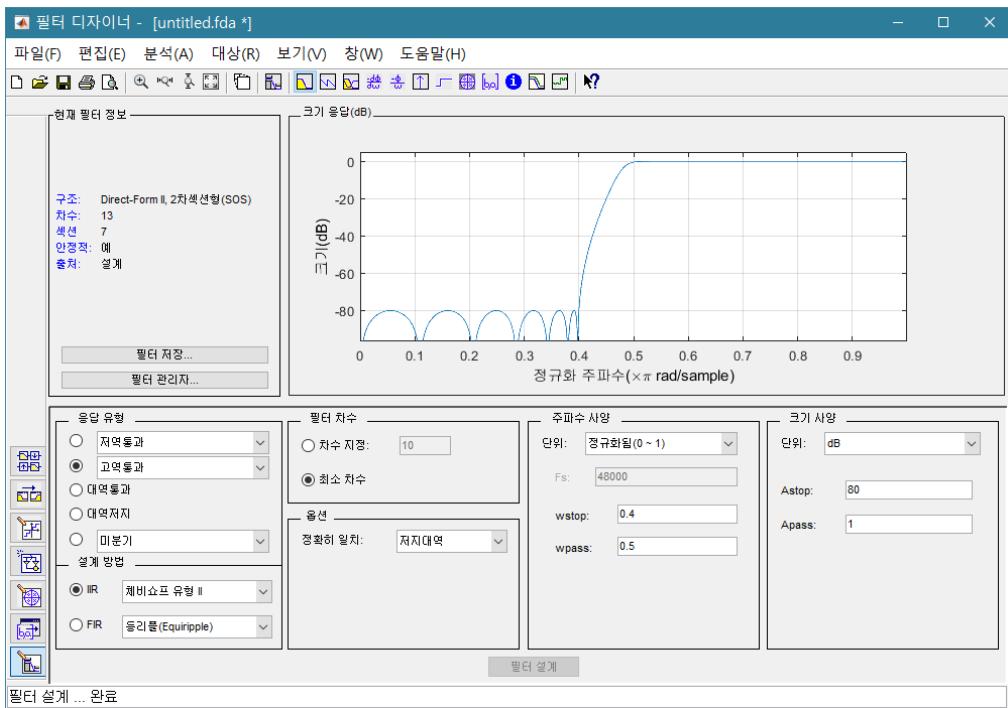
▣ 필터 사양

- ✓ 통과 대역 주파수 : $\omega_p = 0.5\pi$
- ✓ 저지 대역 주파수 : $\omega_s = 0.4\pi$
- ✓ 통과 대역 감쇄 : $A_p = 1 \text{ dB}$
- ✓ 저지 대역 감쇄 : $A_s = 80 \text{ dB}$

▣ filterDesigner 파라미터

- ✓ 응답 유형 : 고역 통과
- ✓ 설계 방법 : IIR – 타원(Elliptic)
- ✓ 필터 차수 : 최소 차수
- ✓ 옵션 : 정확히 일치 – 둘 다
- ✓ 주파수 사양 : 단위 – 정규화됨(0~1), wpass : 0.2, wstop : 0.3
- ✓ 크기 사양 : 단위 – dB, Apass : 1, Astop : 80

▣ 필터 크기 특성



3) 아날로그 IIR 타원 BPF 설계

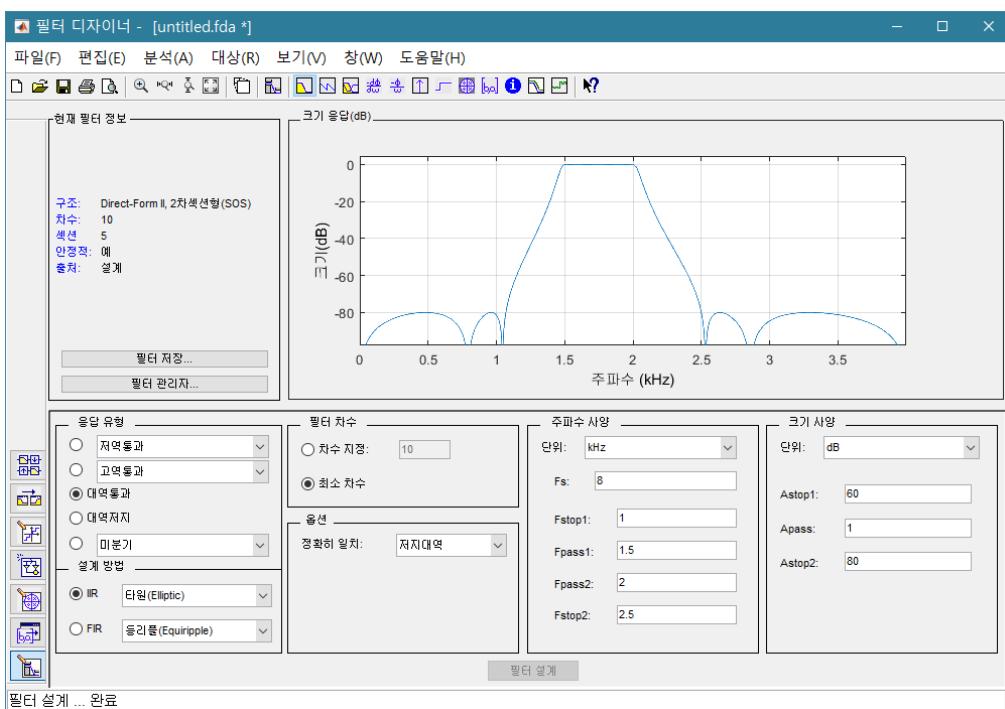
필터 사양

- ✓ 통과 대역 주파수 : $F_{p1} = 1.5 \text{ kHz}$, $F_{p2} = 2 \text{ kHz}$
- ✓ 저지 대역 주파수 : $F_{s1} = 1 \text{ kHz}$, $F_{s2} = 2.5 \text{ kHz}$
- ✓ 통과 대역 감쇄 : $A_p = 1 \text{ dB}$
- ✓ 저지 대역 감쇄 : $A_s = 80 \text{ dB}$
- ✓ 샘플링 주파수 : $F_{\text{samp}} = 8 \text{ kHz}$

filterDesigner 파라미터

- ✓ 응답 유형 : 저역 통과
- ✓ 설계 방법 : IIR – 버터워스
- ✓ 필터 차수 : 최소 차수
- ✓ 옵션 : 정확히 일치 – 저지 대역
- ✓ 주파수 사양 : 단위 – kHz, Fs : 8, Fstop1 : 1, Fpass1 : 1.5, Fpass2 : 2, Fstop2 : 2.5
- ✓ 크기 사양 : 단위 – dB, Astop1 : 60, Apass : 1, Astop2 : 80

필터 크기 특성



4) 카이저 창 함수를 이용한 아날로그 FIR LPF 설계

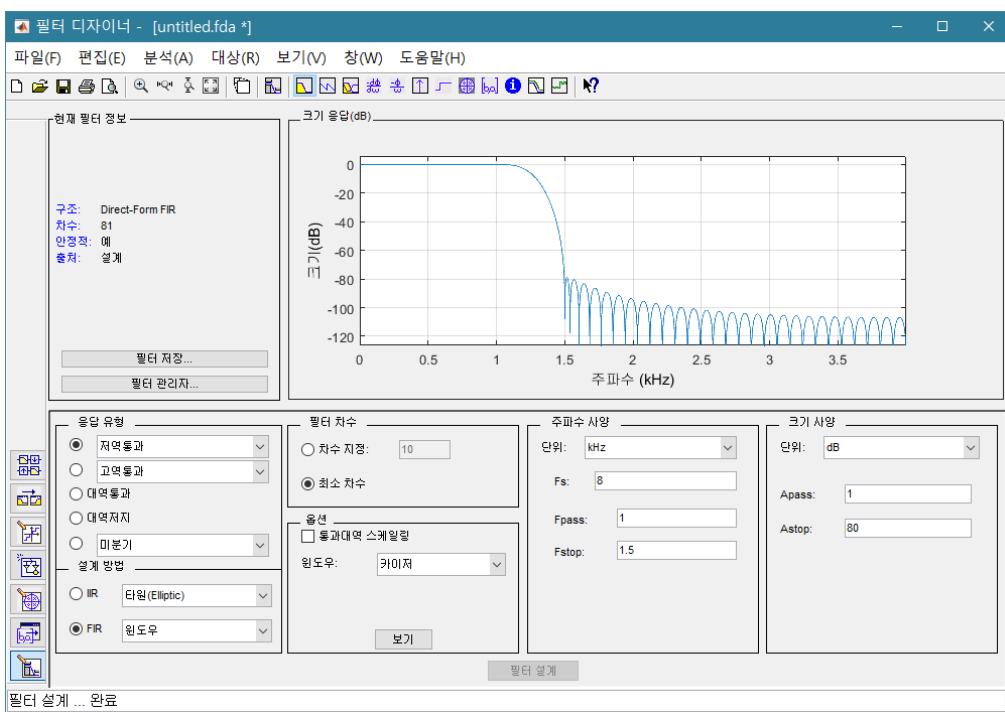
필터 사양

- ✓ 통과 대역 주파수 : $F_p = 1\text{ kHz}$
- ✓ 저지 대역 주파수 : $F_s = 1.5\text{ kHz}$
- ✓ 통과 대역 감쇄 : $A_p = 1\text{ dB}$
- ✓ 저지 대역 감쇄 : $A_s = 80\text{ dB}$
- ✓ 샘플링 주파수 : $F_{\text{samp}} = 8\text{ kHz}$

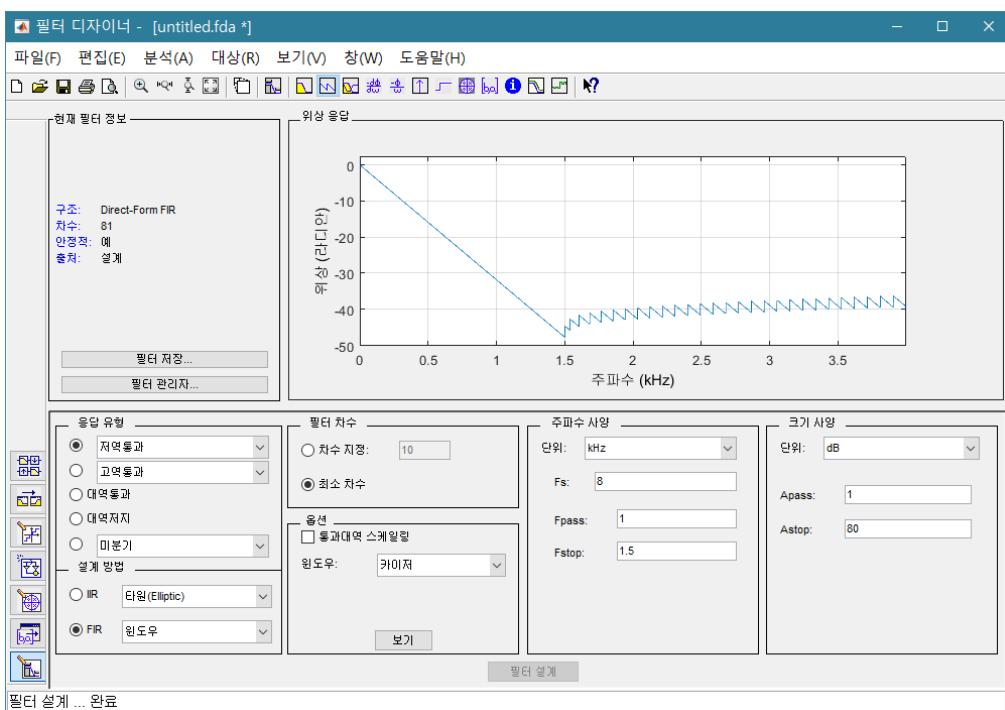
filterDesigner 파라미터

- ✓ 응답 유형 : 저역 통과
- ✓ 설계 방법 : FIR – 원도우
- ✓ 필터 차수 : 최소 차수
- ✓ 옵션 : 통과대역 스케일링 – 체크 해제, 원도우 – 카이저
- ✓ 주파수 사양 : 단위 – kHz, Fs : 8, Fpass : 1, Fstop : 1.5
- ✓ 크기 사양 : 단위 – dB, Apass : 1, Astop : 80

필터 크기 특성



필터의 위상 특성



5) PM 최적 디지털 FIR BPF 설계

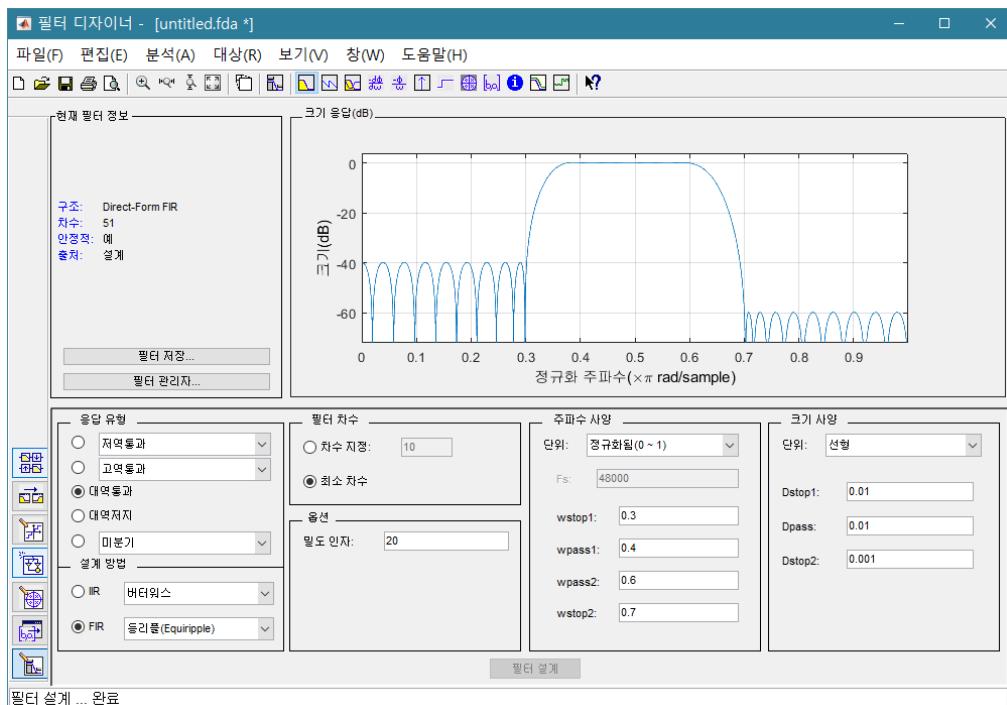
필터 사양

- ✓ 통과 대역 각 주파수 : $\omega_{p1} = 0.4\pi \text{ rad/s}$, $\omega_{p2} = 0.6\pi \text{ rad/s}$
- ✓ 저지 대역 주파수 : $\omega_{s1} = 0.3\pi \text{ rad/s}$, $\omega_{s2} = 0.7\pi \text{ rad/s}$
- ✓ 통과 대역 오차 : $\delta_p = 0.01$
- ✓ 저지 대역 오차 : $\delta_{s1} = 0.01$, $\delta_{s2} = 0.001$

filterDesigner 파라미터

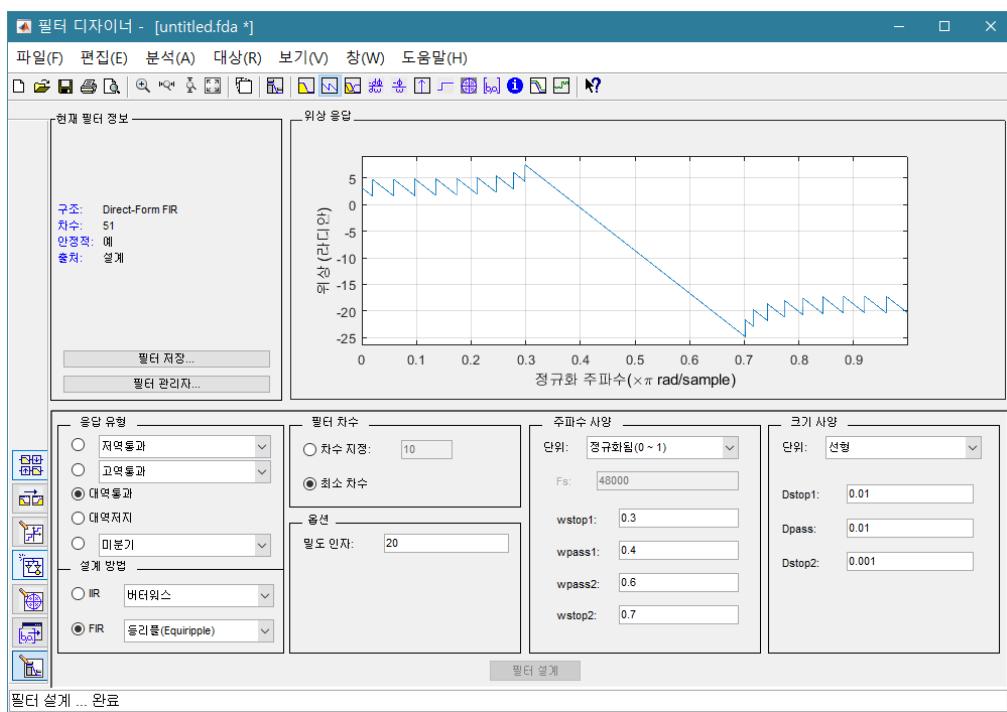
- ✓ 응답 유형 : 대역 통과
- ✓ 설계 방법 : FIR – 등리플(Equiripple)
- ✓ 필터 차수 : 최소 차수
- ✓ 주파수 사양 : 단위 – 정규화됨, wstop1 : 0.3, wpass1 : 0.4, wpass2 : 0.6, wstop2 : 0.7
- ✓ 크기 사양 : 단위 – 선형, Dstop1 : 0.01, Spass : 0.01, Dstop2 : 0.001

필터 크기 특성

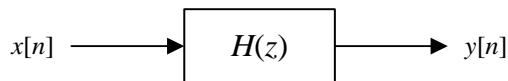


- ✓ 앞의 예제와 동일함.

필터 위상 특성



7.3 신호의 필터링



필터링 함수

```
y = filter(b,a,x)
```

- ✓ b, a : 필터의 분자, 분모 계수
- ✓ x, y : 입력 신호와 출력 신호

```
y = filter(b,a,x,zi)
```

- ✓ zi : 필터의 지연에 대한 초기 조건

필터 입/출력 스펙트럼 비교

- ✓ 디지털 필터

$$H(z) = \frac{0.3 + 0.3z^{-1} + 0.1z^{-2}}{1 - 0.5z^{-1} + 0.2z^{-2}}$$

- ✓ 입력 신호
- ✗ 평균이 0이고 분산이 1인 가우시안 백색 잡음

출력 신호 생성과 스펙트럼 비교

```
% Colored Noise Generation
clc; clear;

% 필터 계수
b = [0.3, 0.3, 0.1];
a = [1, -0.5, 0.2];

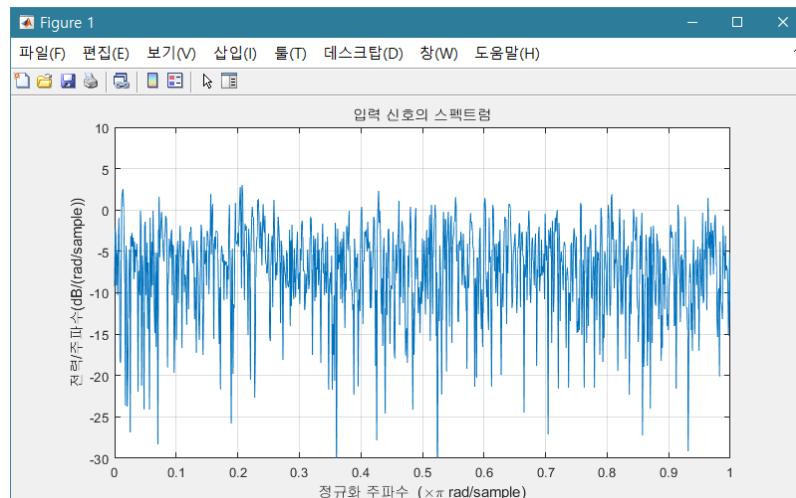
% 입력 신호와 스펙트럼
N = 2000; mx = 0; vx = 1;
x = wgauss(mx, vx, N);
figure(1)
periodogram(x, hamming(N))
grid on
ylim([-30, 10])
title('입력 신호의 스펙트럼')
```

```
% 출력 신호와 스펙트럼
y = filter(b,a,x);
figure(2)
periodogram(y,hamming(N))
grid on
ylim([-40,5])
title('출력 신호의 스펙트럼')
```

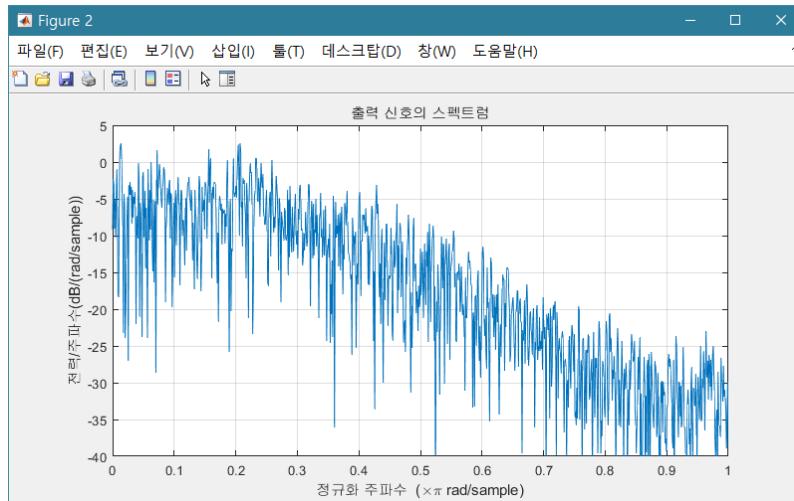
```
% 필터 주파수 응답
w = linspace(0,pi,1001);
H = freqz(b,a,w);
Hm = abs(H);
HmdB = db(Hm);
figure(3)
plot(w/pi,HmdB)
grid on
title('필터의 주파수 응답 (크기)');
xlabel('정규화된 각 주파수 (w/pi)');
ylabel('크기 [dB]');
```

✓ 스펙트럼 비교

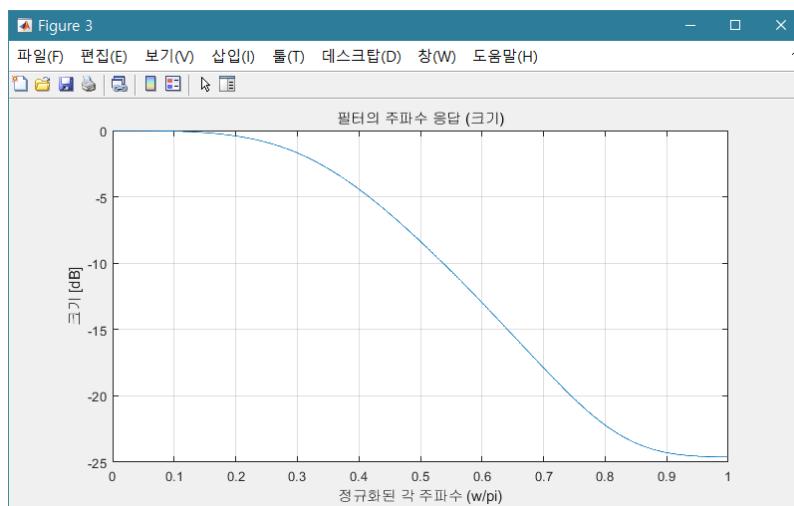
✗ 입력 신호의 스펙트럼



✗ 출력 신호의 스펙트럼



✖ 필터의 주파수 특성 (크기, dB)



7.4 연습 문제

- ▶ 문제 1. 다음과 같은 사양을 갖는 아날로그 타원 대역 통과 필터를 설계하고 주파수 크기(dB) 특성을 그려라. 주파수 범위는 100 ~ 10000 Hz 이고 로그 눈금이다.

$$F_{s1} = 0.5 \text{ kHz}, F_{p1} = 0.7 \text{ kHz}, F_{p2} = 1.3 \text{ kHz}, F_{s2} = 2 \text{ kHz}$$

$$\delta_{s1} = 0.01, \delta_p = 0.01, \delta_{s2} = 0.001$$

- ▶ 문제 2. filterDesigner 를 사용하여 다음과 같은 사양을 갖는 아날로그 타원 대역 통과 필터를 설계하고 주파수 크기(dB) 특성을 그려라

$$F_{s1} = 0.5 \text{ kHz}, F_{p1} = 0.7 \text{ kHz}, F_{p2} = 1.3 \text{ kHz}, F_{s2} = 2 \text{ kHz}$$

$$\delta_{s1} = 0.01, \delta_p = 0.01, \delta_{s2} = 0.001$$

- ▶ 문제 3. 다음과 같은 사양을 갖는 디지털 타원 대역 통과 필터를 설계하고 주파수 크기(dB) 특성을 그려라.

$$\omega_{s1} = 0.4\pi \text{ rad/s}, \omega_{p1} = 0.45\pi \text{ rad/s}, \omega_{p2} = 0.55\pi \text{ rad/s}, \omega_{s2} = 0.6\pi \text{ rad/s}$$

$$\delta_{s1} = 0.01, \delta_p = 0.01, \delta_{s2} = 0.001$$

- ▶ 문제 4. filterDesigner 를 사용하여 다음과 같은 사양을 갖는 디지털 타원 대역 통과 필터를 설계하고 주파수 크기(dB) 특성을 그려라.

$$\omega_{s1} = 0.4\pi \text{ rad/s}, \omega_{p1} = 0.45\pi \text{ rad/s}, \omega_{p2} = 0.55\pi \text{ rad/s}, \omega_{s2} = 0.6\pi \text{ rad/s}$$

$$\delta_{s1} = 0.01, \delta_p = 0.01, \delta_{s2} = 0.001$$

- ▶ 문제 5. Park-McClellan 방법을 사용하여 다음과 같은 사양을 갖는 디지털 FIR 대역 통과 필터를 설계하고 주파수 크기(dB) 특성을 그려라.

$$\omega_{s1} = 0.4\pi \text{ rad/s}, \omega_{p1} = 0.45\pi \text{ rad/s}, \omega_{p2} = 0.55\pi \text{ rad/s}, \omega_{s2} = 0.6\pi \text{ rad/s}$$

$$\delta_{s1} = 0.01, \delta_p = 0.01, \delta_{s2} = 0.001$$

- ▶ 문제 6. filterDesigner 를 사용하여 다음과 같은 사양을 갖는 등리플 디지털 FIR 대역 통과 필터를 설계하고 주파수 크기(dB) 특성을 그려라.

$$\omega_{s1} = 0.4\pi \text{ rad/s}, \omega_{p1} = 0.45\pi \text{ rad/s}, \omega_{p2} = 0.55\pi \text{ rad/s}, \omega_{s2} = 0.6\pi \text{ rad/s}$$

$$\delta_{s1} = 0.01, \delta_p = 0.01, \delta_{s2} = 0.001$$

8. Symbolic Math

8.1 Symbolic Math 란 무엇인가?

▣ 수치 데이터가 아닌 수식으로 된 연산 결과를 얻을 수 있다.

▣ Symbolic Math Tool box 필요

▣ 예제

✓ $x(t) = e^{-2t}$ 의 라플라스 변환을 구하라.

✓ $x[n] = 0.5^n$ 의 z 변환을 구하라.

✓ $x(t) = 3x^2 + 5x + 2$ 를 미분하라.

✓ $x(t) = 3x^2 + 5x + 2$ 를 적분하라.

✓ $x(t) = 3x^2 + 5x + 2$ 의 그래프를 그려라.

▣ 관련 사이트

✓ <https://kr.mathworks.com/products/symbolic.html>

8.2 Symbolic Math 관련 함수

✚ Symbolic Math 주요 관련 함수

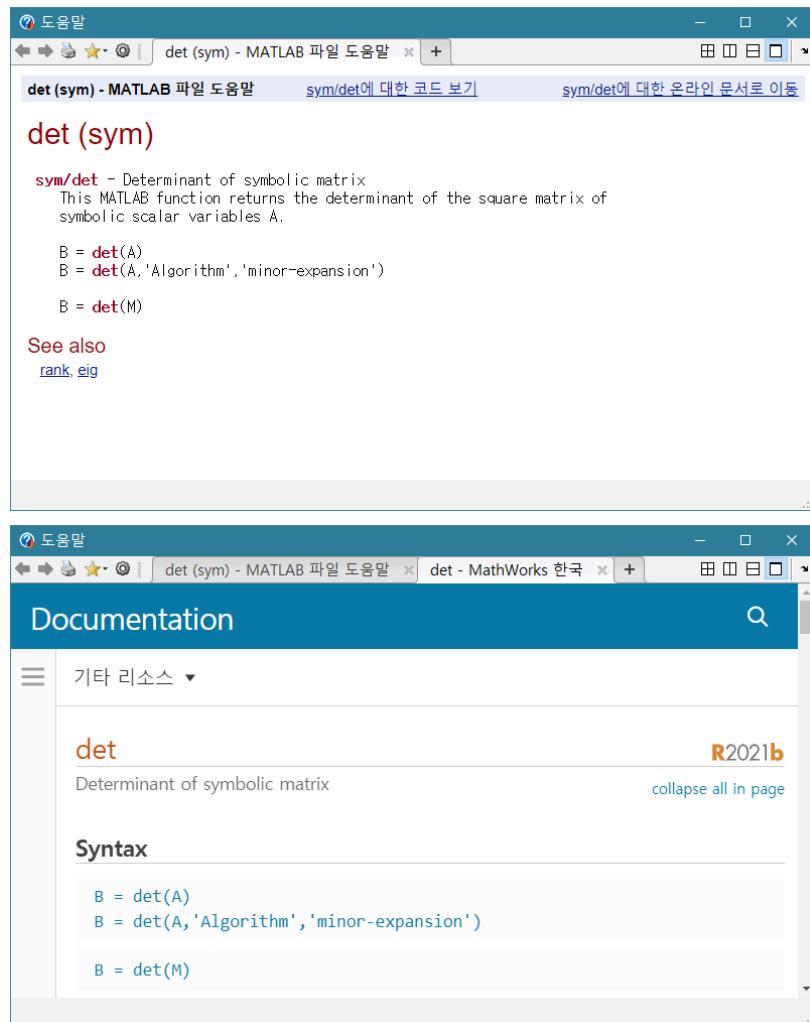
- ✓ Basic Operation : sym, syms, pretty, simplify
- ✓ Calculus : diff, int, limit, taylor
- ✓ Solution of Equations : solve, dsolve
- ✓ Linear Algebra : det, eig, inv, rank
- ✓ Transforms : fourier, ifourier, laplace, ilaplace, ztrans, iztrans
- ✓ Simplification : factor, simplify
- ✓ Graphical Applications : ezplot, ezplot3

✚ Symbolic Math 관련 함수 목록

- ✓ https://kr.mathworks.com/help/symbolic/referencelist.html?type=function&s_t_id=CRUX_topnav

8.3 Symbolic Math 의 도움말

- help sym/function_name
- helpwin sym/function_name
- doc sym/function_name



8.4 Symbolic 객체 생성

✚ Symbolic 숫자 생성

- ✓ Symbolic 숫자는 정확한 분수로 표시된다.

```
a = 1/3
b = sym(1/3) % Symbolic 숫자 생성
c = sin(pi)
d = sin(sym(pi))
```

```
a =
0.3333
b =
1/3
c =
1.2246e-16
d =
0
```

✚ Symbolic 변수 생성

```
syms x % Symbolic 변수 생성
y1 = 3*x^2 + 5*x + 2
y2 = str2sym('3*x^2 + 5*x + 2') % sym 함수 이용
```

```
y1 =
3*x^2 + 5*x + 2
y2 =
3*x^2 + 5*x + 2
```

✚ Symbolic 함수 생성

- ✓ symfun

```
syms x y
f1 = symfun(x^2+y^2-1,[x,y])
f2(x,y) = x^2 + y^2 - 1
f1(1,1)
```

```
f2(1,1)
```

```
f1(x, y) =
x^2 + y^2 - 1
f2(x, y) =
x^2 + y^2 - 1
ans =
1
ans =
1
```

▶ 벡터를 이용하여 symbolic 다항식 생성

```
syms t
c = [1 2 3 4];
p1 = poly2sym(c) % 변수가 x인 다항식 생성
p2 = poly2sym(c,t) % 변수가 t인 다항식 생성
```

```
p1 =
x^3 + 2*x^2 + 3*x + 4
p2 =
t^3 + 2*t^2 + 3*t + 4
```

8.5 Symbolic 과 수치 데이터의 변환

- ✚ Symbolic 을 숫자로 변환하는 함수

- ✓ single
- ✓ double
- ✓ sym2poly

```
x = sym(pi);
y1 = single(x) % 단일 정밀도 숫자로 변환
y2 = double(x) % 배 정밀도 숫자로 변환
e = y1 - y2
y = str2sym('x^3 + 2*x^2 + 3*x + 4');
p = sym2poly(y) % 계수 벡터로 변환
```

```
y1 =
3.1416
y2 =
3.1416
e =
8.7423e-08
p =
1    2    3    4
```

- ✚ Symbolic 함수에 값을 대입하는 함수

```
R = subs(S)
R = subs(S,new)
R = subs(S,old,new)
```

- ✓ Symbolic 함수 S 에 workspace 에 있는 변수 값 대입
- ✓ Symbolic 함수 S 의 기존 변수에 값 new 대입
- ✓ Symbolic 함수 S 의 변수 old 에 값 new 대입

- ✚ Symbolic 과 수치 데이터의 변환의 예

$$x(t) = A \cos(\omega_0 t), \quad A = 2, \omega_0 = \frac{\pi}{2} \Rightarrow x_1(t) = 2 \cos\left(\frac{\pi}{2}t\right)$$

$$x_1(t) = 2 \cos\left(\frac{\pi}{2}t\right) \Rightarrow x_1(2) = 2 \cos(\pi) = -2$$

```
syms t A w0
x = A*cos(w0*t);
A = 2; w0 = pi/2;
x1 = subs(x)
xv = subs(x1,2)          % 기존 변수 t에 2 대입
xv = subs(x1,t,2)        % 변수 t에 2 대입
xv1 = single(xv)         % 단일 정밀도 숫자로 변환
```

```
x1 =
2*cos((pi*t)/2)
xv =
-2
xv =
-2
xv1 =
-2
```

8.6 Symbolic 연산

8.6.1 행렬 연산

행렬의 사칙 연산

$$\mathbf{A} = \begin{bmatrix} \cos x & -\sin x \\ \sin x & \cos x \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} y & 0 \\ 0 & -y \end{bmatrix}$$

$$\mathbf{C} = \mathbf{A} + \mathbf{B} = \begin{bmatrix} \cos x + y & -\sin x \\ \sin x & \cos x - y \end{bmatrix}$$

$$\mathbf{D} = \mathbf{AB} = \begin{bmatrix} y \cos x & y \sin x \\ -y \sin x & -y \cos x \end{bmatrix}$$

```
syms x y
A = [cos(x) -sin(x); -sin(x) cos(x)];
B = [y 0; 0 -y];
C = A+B % 행렬의 덧셈
D = A*B % 행렬의 곱셈
```

```
C =
[y+cos(x), -sin(x)]
[ sin(x), cos(x)-y]
D =
[ y*cos(x), y*sin(x)]
[-y*sin(x), -y*cos(x)]
```

행렬의 고유값

$$\mathbf{A} = \begin{bmatrix} \cos x & -\sin x \\ \sin x & \cos x \end{bmatrix} \Rightarrow \mathbf{v} = \begin{bmatrix} \cos x - j \sin x \\ \cos x + j \sin x \end{bmatrix}$$

```
syms x y
A = [cos(x) -sin(x); sin(x) cos(x)];
v = eig(A) % 행렬의 고유 벡터
pretty(v)
```

```
v =
cos(x) - sin(x)*1i
```

```
cos(x) + sin(x)+1i
/ cos(x) - sin(x)*i \
|           |
\ cos(x) + sin(x)*i /
```

8.6.2 다항식의 연산

▣ 다항식의 미분, 적분

$$y = 3x^2 + 5x + 2$$

$$y' = 6x + 5, \quad y'' = 6$$

$$\int y dx = x^3 + \frac{5}{2}x^2 + 2x$$

```
syms x
y = 3*x^2 + 5*x + 2;
dy = diff(y)                                % 미분
d2y = diff(y,2)                             % 이중 미분
iy = int(y)                                 % 적분
```

```
dy =
6*x + 5
d2y =
6
iy =
(x*(2*x^2 + 5*x + 4))/2
```

▣ 다항식의 편미분

```
syms x y
z = x^2 + 2*y^2 + 3*x*y + 4*x + 5*y + 6;
dzx = diff(z,x)                            % x에 대해 편미분
dzy = diff(z,y)                            % y에 대해 편미분
d2zx = diff(z,x,2)                         % x에 대해 이중 편미분
```

```
dzx =
2*x + 3*y + 4
dzy =
```

3*x + 4*y+ +5

d2zx =

2

8.6.3 각종 변환(transform)

➊ 푸리에(Fourier) 변환

- ✓ 시간 영역 변수 : x , 주파수 영역 변수 : ω

fourier(F)

- ✓ 변수가 x 인 함수 F 를 푸리에 변환하고 변수 ω 로 출력

fourier(F, 'v')

- ✓ 변수가 x 인 함수 F 를 푸리에 변환하고 변수 ω 대신 v 로 출력

fourier(F, 't', 'v')

- ✓ 변수가 t 인 함수 F 를 푸리에 변환하고 변수 ω 대신 v 로 출력

➋ 푸리에(Fourier) 역 변환

ifourier(F)

- ✓ 변수가 ω 인 함수 F 를 푸리에 역 변환하고 변수 x 로 출력

ifourier(F, 't')

- ✓ 변수가 ω 인 함수 F 를 푸리에 역 변환하고 변수 x 대신 t 로 출력

ifourier(F, 'v', 't')

- ✓ 변수가 v 인 함수 F 를 푸리에 역 변환하고 변수 x 대신 t 로 출력

➌ 푸리에 변환 예제

$$f(x) = \exp[-x^2] \Leftrightarrow F(\omega) = \sqrt{\pi} \exp\left[-\frac{\omega^2}{4}\right]$$

```
syms x w v t
fx = exp(-x^2);
ft = exp(-t^2);
Fw = fourier(fx)
```

% 푸리에 변환

<code>Fv = fourier(fx, 'v')</code>	% 푸리에 변환 후 변수 v로 출력
<code>Fw = fourier(ft, 't', 'w')</code>	% 푸리에 변환
<code>Fv = fourier(ft, 't', 'v')</code>	% 푸리에 변환 후 변수 v로 출력
<code>fx = ifourier(Fw)</code>	% 푸리에 역 변환
<code>ft = ifourier(Fv, 'v', 't')</code>	% 푸리에 역 변환 후 변수 t로 출력

`Fw =`
 $\pi^{(1/2)} * \exp(-w^2/4)$
`Fv =`
 $\pi^{(1/2)} * \exp(-v^2/4)$
`Fw =`
 $\pi^{(1/2)} * \exp(-w^2/4)$
`Fv =`
 $\pi^{(1/2)} * \exp(-v^2/4)$
`fx =`
 $\exp(-x^2)$
`ft =`
 $\exp(-t^2)$

✚ 라플라스(Laplace) 변환

- ✓ 시간 영역 변수 : t , 주파수 영역 변수 : s

`laplace(F)`

- ✓ 변수가 t 인 함수 F 를 라플라스 변환하고 변수 s 로 출력

`laplace(F, 'w')`

- ✓ 변수가 t 인 함수 F 를 라플라스 변환하고 변수 s 대신 w 로 출력

`laplace(F, 'x', 'w')`

- ✓ 변수가 x 인 함수 F 를 라플라스 변환하고 변수 s 대신 w 로 출력

✚ 라플라스(Laplace) 역 변환

`ilaplace(F)`

- ✓ 변수가 s 인 함수 F 를 라플라스 역 변환하고 변수 t 로 출력

```
ilaplace(F, 'x')
```

- ✓ 변수가 s 인 함수 F 를 라플라스 역 변환하고 변수 t 대신 x 로 출력

```
ilaplace(F, 'w', 'x')
```

- ✓ 변수가 w 인 함수 F 를 라플라스 역 변환하고 변수 t 대신 x 로 출력

➊ 라플라스 변환 예제

$$f(t) = \exp[-2t] \Leftrightarrow F(s) = \frac{1}{s+2}$$

```
syms t s x w
ft = exp(-2*t);
fx = exp(-2*x);
Fs = laplace(ft)                                % 라플라스 변환
Fw = laplace(ft, 'w')                           % 라플라스 변환 후 변수 w로 출력
Fs = laplace(fx, 'x', 's')                      % 라플라스 변환
Fw = laplace(fx, 'x', 'w')                      % 라플라스 변환 후 변수 w로 출력
ft = ilaplace(Fs)                               % 라플라스 역 변환
fx = ilaplace(Fw, 'w', 'x')                     % 라플라스 역변환 후 변수 v로 출력
```

```
Fs =
1/(s+2)
Fw =
1/(w+2)
Fs =
1/(s+2)
Fw =
1/(w+2)
ft =
exp(-2*t)
fx =
exp(-2*x)
```

➋ z 변환

```
ztrans(F)
```

- ✓ 변수가 n 인 함수 F 를 z 변환하고 변수 z 로 출력

```
ztrans(F, 'w')
```

- ✓ 변수가 n 인 함수 F 를 z 변환하고 변수 z 대신 w 로 출력

```
ztrans(F, 'm', 'w')
```

- ✓ 변수가 m 인 함수 F 를 z 변환하고 변수 z 대신 w 로 출력

▶ z 역 변환

```
iztrans(F)
```

- ✓ 변수가 z 인 함수 F 를 z 역 변환하고 변수 n 으로 출력

```
iztrans(F, 'm')
```

- ✓ 변수가 z 인 함수 F 를 z 역 변환하고 변수 n 대신 m 으로 출력

```
iztrans(F, 'w', 'm')
```

- ✓ 변수가 w 인 함수 F 를 z 역 변환하고 변수 n 대신 m 으로 출력

▶ z 변환 예제

$$f[n] = 0.5^n \Leftrightarrow F(z) = \frac{1}{1 - 0.5z^{-1}} = \frac{z}{z - 1/2}$$

```
syms n z m w
fn = 0.5^n;
fm = 0.5^m;
Fz = ztrans(fn) % z 변환
Fw = ztrans(fn, 'w') % z 변환 후 변수 w로 출력
Fz = ztrans(fm, 'm', 'z') % z 변환
Fw = ztrans(fm, 'm', 'w') % z 변환 후 변수 w로 출력
fn = iztrans(Fz) % z 역 변환
fm = iztrans(Fw, 'w', 'm') % z 역 변환 후 변수 m으로 출력
```

```
Fz =
z/(z-1/2)
Fw =
w/(w-1/2)
```

```

Fz =
z / (z-1/2)

Fw =
w / (w-1/2)

fn =
(1/2)^n

fm =
(1/2)^m

```

8.6.4 방정식의 해

비선형 방정식 해 구하기

```
x1 = solve(f,x)
```

✓ $f(x)=0$ 의 해를 구한다.

```
[x1, x2, ..., xn] = solve(f1, f2, ..., fn, x1, x2, ..., xn)
```

✓ $\mathbf{f}(\mathbf{x})=\mathbf{0}$ 의 해를 구한다.

$$\begin{aligned} f_1(x_1, x_2, \dots, x_n) &= 0 \\ f_2(x_1, x_2, \dots, x_n) &= 0 \\ &\dots \\ f_n(x_1, x_2, \dots, x_n) &= 0 \end{aligned}$$

비선형 방정식 예제

$$f(x) = \frac{\sin^2 x + \cos x - 1}{(12 + 2 \sin x)^4} = 0 \Rightarrow x = 0, \pi/2$$

```

syms x
f = (sin(x)^2+cos(x)-1) / (12+2*sin(x))^4;
x1 = solve(f,x)
double(x1)                                % 배 정밀도 문자로 변환

```

```

x1 =
pi/2
0
ans =
1.5708

```

0

 비선형 연립 방정식 예제

$$\begin{cases} x_1 + 2x_2 - x_1x_2 = 3 \\ 2x_1 + x_2 - x_1x_2 = -2 \end{cases} \Rightarrow \begin{cases} x_1 = -2\sqrt{2} - 1 = -3.8284, & 2\sqrt{2} - 1 = 1.8284 \\ x_2 = 4 - 2\sqrt{2} = 1.1716, & 2\sqrt{2} + 4 = 6.8284 \end{cases}$$

```
syms x1 x2
f1 = x1+2*x2-x1*x2-3;
f2 = 2*x1+x2-x1*x2+2;
[x1,x2] = solve(f1,f2,x1,x2)
double([x1,x2])
```

```
x1 =
-2*2^(1/2) - 1
2*2^(1/2) - 1
xx2 =
4 - 2*2^(1/2)
2*2^(1/2) + 4
ans =
-3.8284    1.1716
1.8284    6.8284
```

8.6.5 미분 방정식의 해

 미분 방정식 해

f = dsolve(eqn)

f = dsolve(eqn,initial_values)

 미분 방정식 해의 예

$$y''(t) = ay \Rightarrow y(t) = C_1 e^{-\sqrt{a}t} + C_2 e^{\sqrt{a}t}$$

$$y''(t) = a^2 y(t), \quad y'(0) = 1, \quad y(0) = b \Rightarrow y(t) = \frac{e^{at}(ab+1)}{2a} + \frac{e^{-at}(ab-1)}{2a}$$

```
syms y(t) a b;
eqn = diff(y,t,2) == a*y;
```

```
y1 = dsolve(eqn)
Dy = diff(y,t); D2y = diff(y,t,2);
eqn = D2y == a^2*y;
cond = [y(0)==b, Dy(0)==1];
y2 = dsolve(eqn,cond)
```

```
y1 =
C1*exp(-a^(1/2)*t) + C2*exp(a^(1/2)*t)
y2 =
(exp(a*t)*(a*b+1))/(2*a) + (exp(-a*t)*(a*b-1))/(2*a)
```

연립 미분 방정식 해

```
[f1, ..., fn] = dsolve(eqn1, ..., eqnn, ivl, ..., ivn)
```

연립 미분 방정식 해의 예

$$\begin{cases} y_1'(t) = y_2(t) \\ y_2'(t) = -y_1(t) \end{cases}, \quad y_1(0) = 1, y_2(0) = 2 \Rightarrow \begin{cases} y_1(t) = \cos t + 2\sin t \\ y_2(t) = 2\cos t - \sin t \end{cases}$$

```
syms y1(t) y2(t);
Dy1 = diff(y1,t); Dy2 = diff(y2,t);
eq1 = Dy1==y2; eq2 = Dy2===-y1;
cond = [y1(0)==1,y2(0)==2];
[y1,y2] = dsolve(eq1,eq2,cond)
```

```
y1 =
cos(t) + 2*sin(t)
y2 =
2*cos(t) - sin(t)
```

8.6.6 기타 유용한 함수

유용한 함수

- pretty
- simplify

```
syms t s x
```

```
xt = exp(-2*t);  
xs = laplace(xt)  
pretty(xs)  
simplify(sin(x)^2+cos(x)^2)  
simplify((x^2-1)/(x-1))  
simplify(cos(x)^2-sin(x)^2)
```

```
1  
-----  
s + 2  
ans =  
1  
ans =  
x + 1  
ans =  
cos(2*x)
```

8.7 Symbolic Math 와 그래프

✚ $y = f(x)$ 의 그래프

ezplot(y)

✓ $y = f(x)$ 를 $-2\pi < x < 2\pi$ 구간에서 그린다.

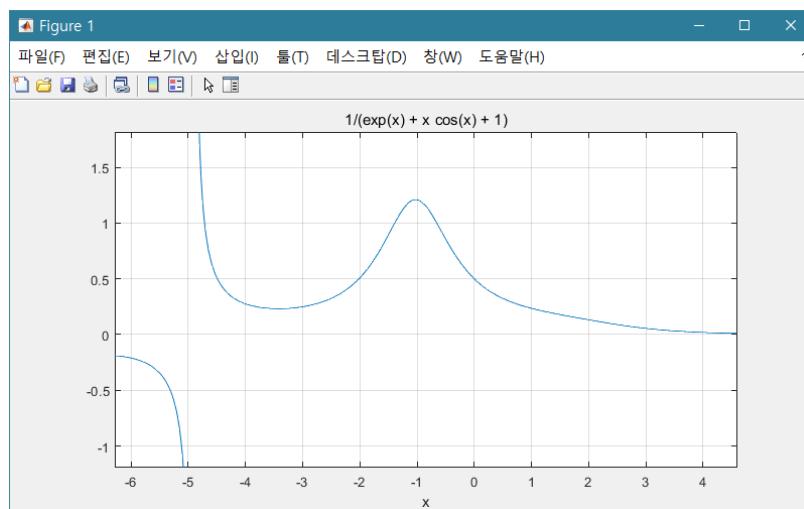
ezplot(y, [a,b])

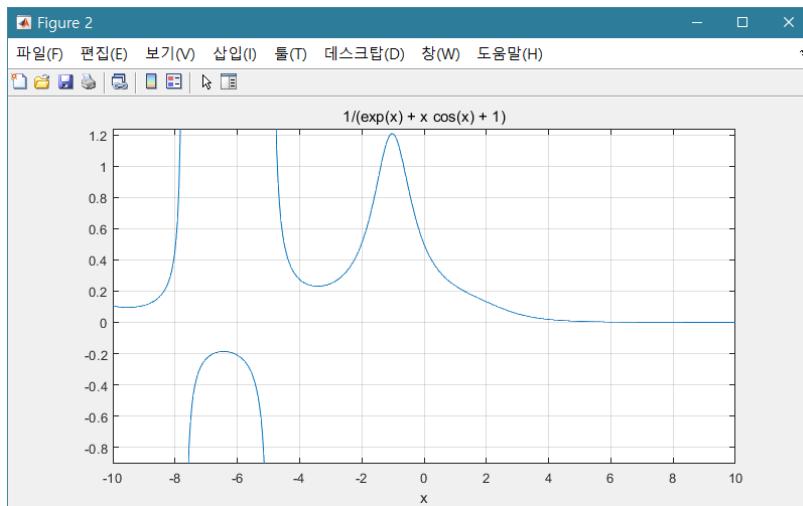
✓ $y = f(x)$ 를 $a < x < b$ 구간에서 그린다.

✚ $y = f(x)$ 의 그래프의 예

$$y = \frac{1}{1 + x \cos x + e^x}$$

```
clc; clear all; close all;
syms x
y = 1/(1+x*cos(x)+exp(x));
figure(1)
ezplot(y)
grid on
figure(2)
ezplot(y, [-10,10])
grid on
```





▶ $f(x, y)=0$ 의 그래프

ezplot(f)

- ✓ $f(x, y)=0$ 을 $-2\pi < x < 2\pi, -2\pi < y < 2\pi$ 구간에서 그린다.

ezplot(y, [a,b])

- ✓ $f(x, y)=0$ 을 $a < x < b$ 구간에서 그린다.

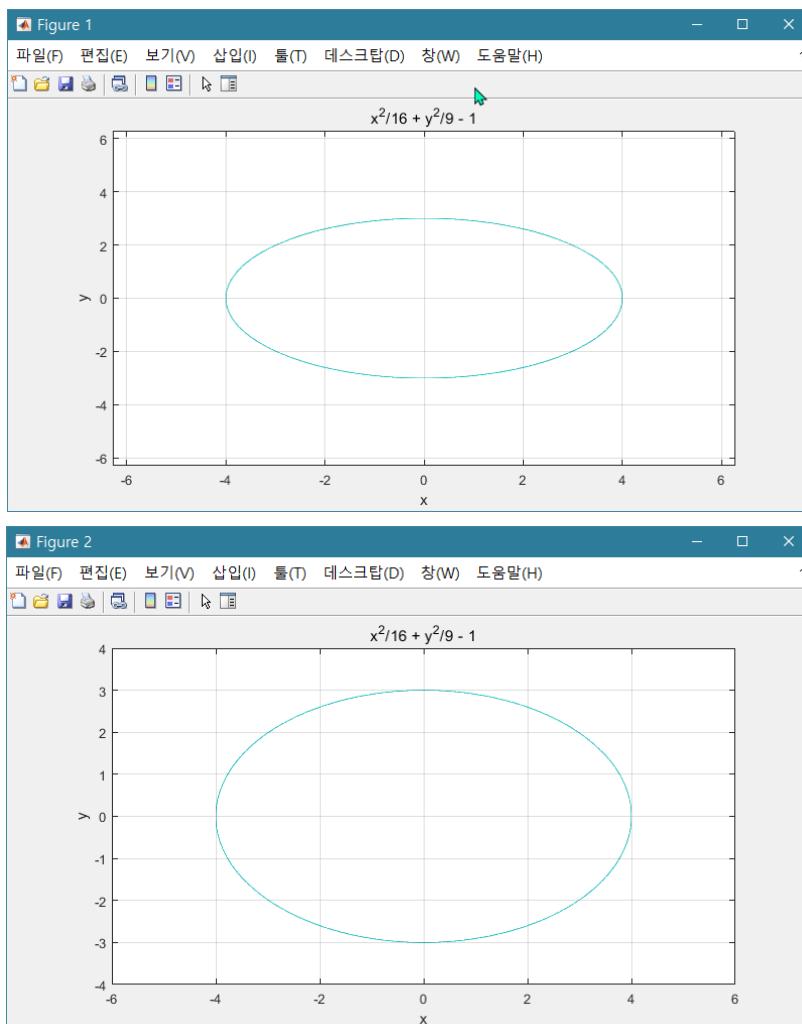
ezplot(y, [x1, x2, y1, y2])

- ✓ $f(x, y)=0$ 을 $x_{\min} < x < x_{\max}, y_{\min} < y < y_{\max}$ 구간에서 그린다.

▶ $f(x, y)=0$ 의 그래프의 예

$$\frac{x^2}{16} + \frac{y^2}{9} = 1 \Rightarrow f(x, y) = \frac{x^2}{16} + \frac{y^2}{9} - 1 = 0$$

```
clc; clear all; close all;
syms x y
f = x^2/16+y^2/9-1;
figure(1)
ezplot(f)
grid on
figure(2)
ezplot(f, [-6, 6, -4, 4])
grid on
```



$x = x(t)$, $y = y(t)$ 의 그래프

ezplot(x,y)

$y = f(x)$ 를 $0 < t < 2\pi$ 구간에서 그린다.

ezplot(x,y,[a,b])

$y = f(x)$ 를 $a < t < b$ 구간에서 그린다.

$x = x(t)$, $y = y(t)$ 의 그래프의 예

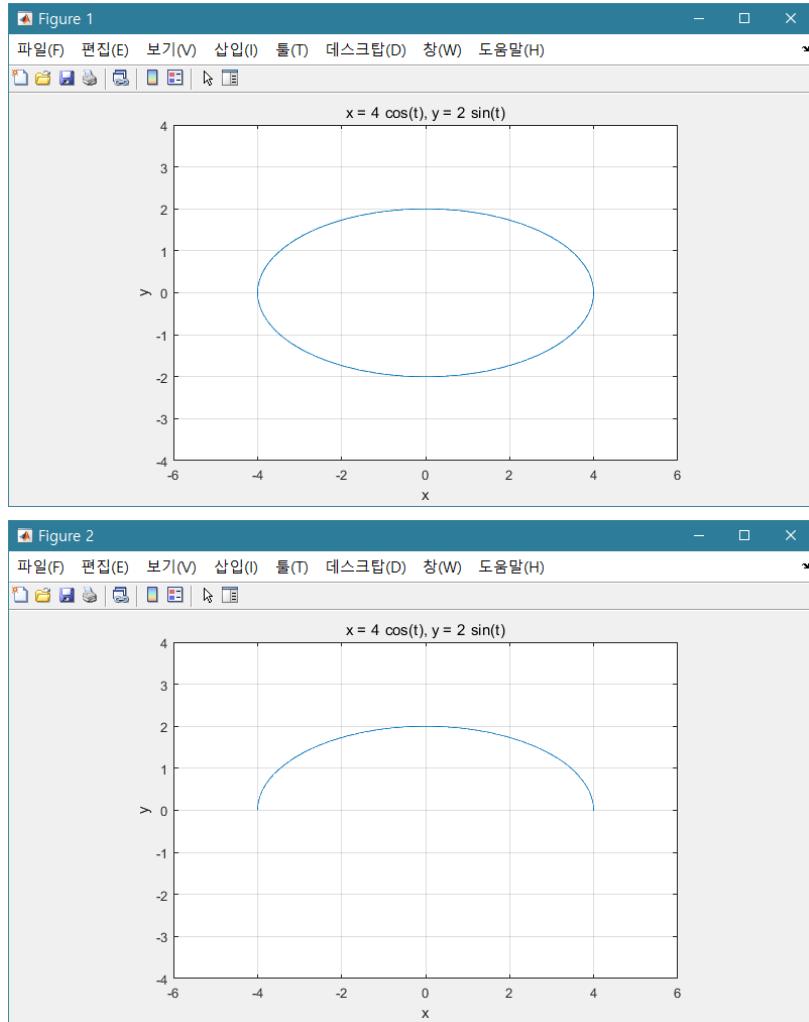
$$x(t) = 4 \cos t, \quad y(t) = 2 \sin t$$

```
clc; clear all; close all;
syms t
x = 4*cos(t);
y = 2*sin(t);
figure(1)
ezplot(x,y)
```

```

axis([-6,6,-4,4])
grid on
figure(2)
ezplot(x,y,[0,pi]) % 0<=t<=pi
axis([-6,6,-4,4])
grid on

```



▶ 주파수 특성 그리기

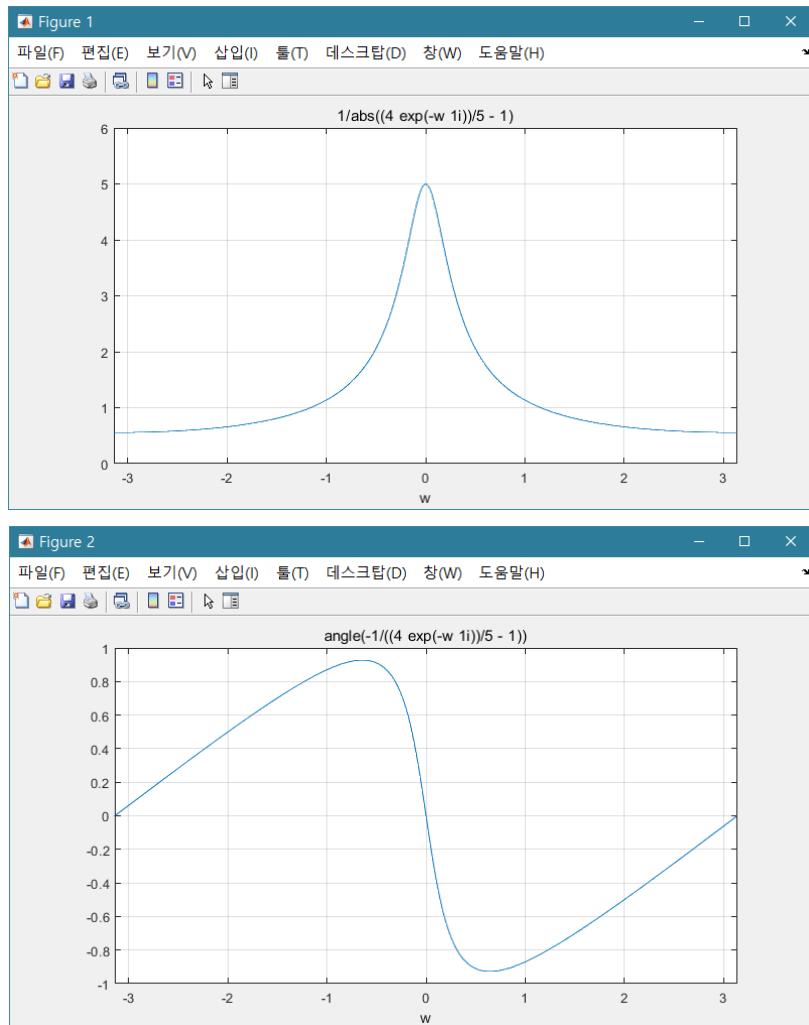
$$X(e^{j\omega}) = \frac{1}{1 - ae^{-j\omega}}, \quad a = 0.8$$

```

clc; clear all; close all;
syms w
a = 0.8;
Xw = 1/(1-a*exp(-j*w));
figure(1)
ezplot(abs(Xw))

```

```
grid on  
axis([-pi,pi,0,6])  
figure(2)  
ezplot(angle(Xw))  
grid on  
axis([-pi,pi,-1,1])
```



8.8 연습 문제

▶ 문제 1. 다음 식의 라플라스 변환과 역 변환을 구하라.

$$x(t) = 2\exp[-2t]u(t) + 3\exp[-3t]u(t)$$

$$X(s) = \frac{1}{s^2 + 3s + 2}$$

▶ 문제 2. 다음 식의 z 변환과 역 변환을 구하라

$$x[n] = 2\left(\frac{1}{2}\right)^n u[n] + 3\left(\frac{1}{3}\right)^n u[n]$$

$$X(z) = \frac{2+3z^{-1}}{1-\frac{5}{6}z^{-1}+\frac{1}{6}z^{-2}}$$

▶ 문제 3. 다음 미분 방정식을 풀어라.

$$y'' + 2y' + y = e^{-x}, \quad y(0) = -1, y'(0) = 1$$

▶ 문제 4. 다음 연립 미분 방정식을 풀어라.

$$\begin{cases} x_1'(t) = x_2(t) + \sin t \\ x_2'(t) = x_1(t) + \cos t \end{cases}, \quad x_1(0) = 0, x_2(0) = 0$$

▶ 문제 5. 다음 전달 함수의 주파수 응답을 구간 $-\pi < \omega < \pi$ 동안 그려라

$$H(e^{j\omega}) = \frac{1}{(1-0.9e^{-j\omega})(1+0.8e^{-j\omega})}$$

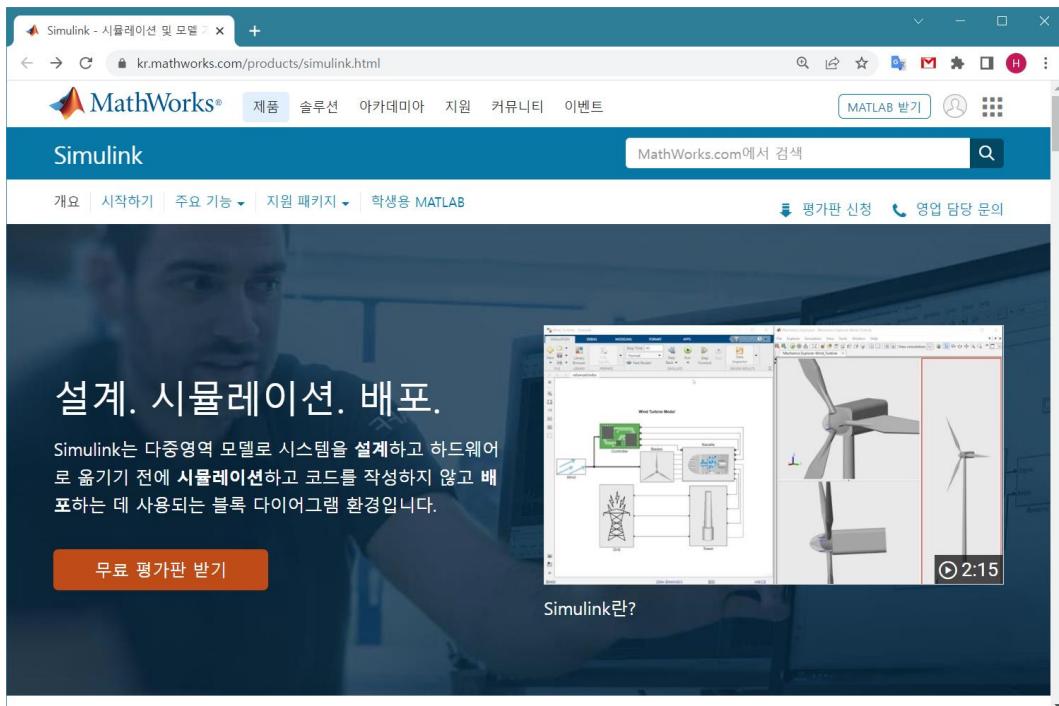
9. Simulink

9.1 Simulink 란 무엇인가?

- ✚ Simulink는 멀티 도메인 시뮬레이션과 모델 기반 설계를 위한 블록 다이어그램 환경입니다. 시스템 레벨 설계, 시뮬레이션, 자동 코드 생성과 임베디드 시스템의 지속적인 테스트 및 검증을 지원합니다.
- ✚ Simulink는 그래픽 편집기, 사용자 정의 가능한 블록 라이브러리, 동적 시스템의 모델링 및 시뮬레이션을 위한 solver를 제공합니다. MATLAB과 통합되므로 MATLAB 알고리즘을 모델로 통합하고 추후 분석을 위해 시뮬레이션 결과를 MATLAB으로 내보낼 수 있습니다.

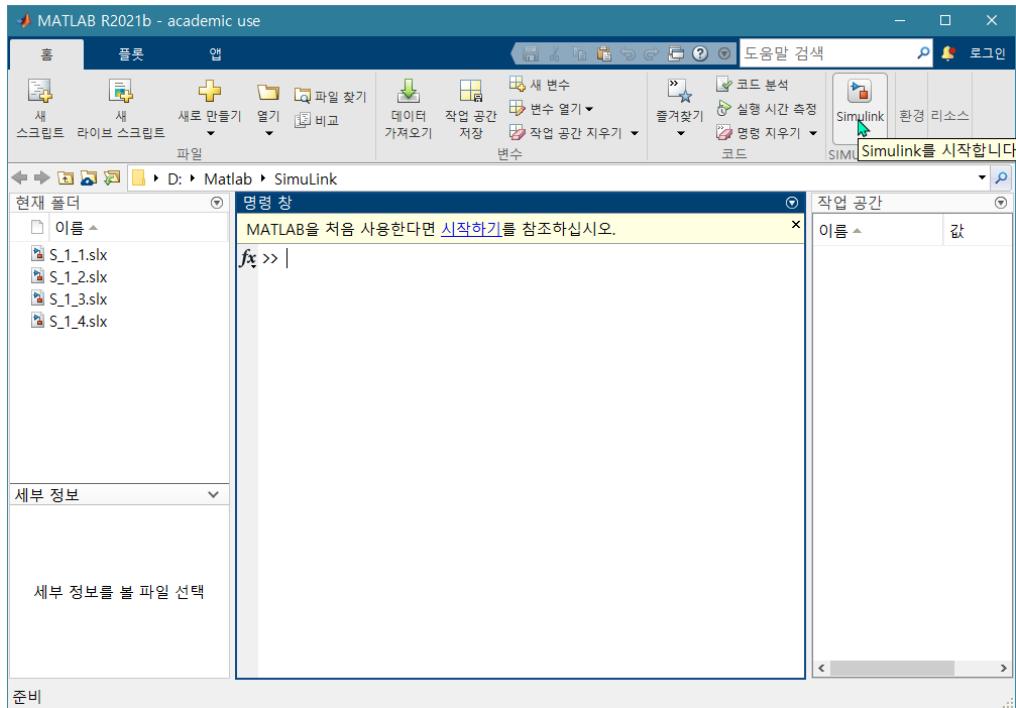
✚ Simulink 관련 사이트

- ✓ <http://kr.mathworks.com/products/simulink.html>

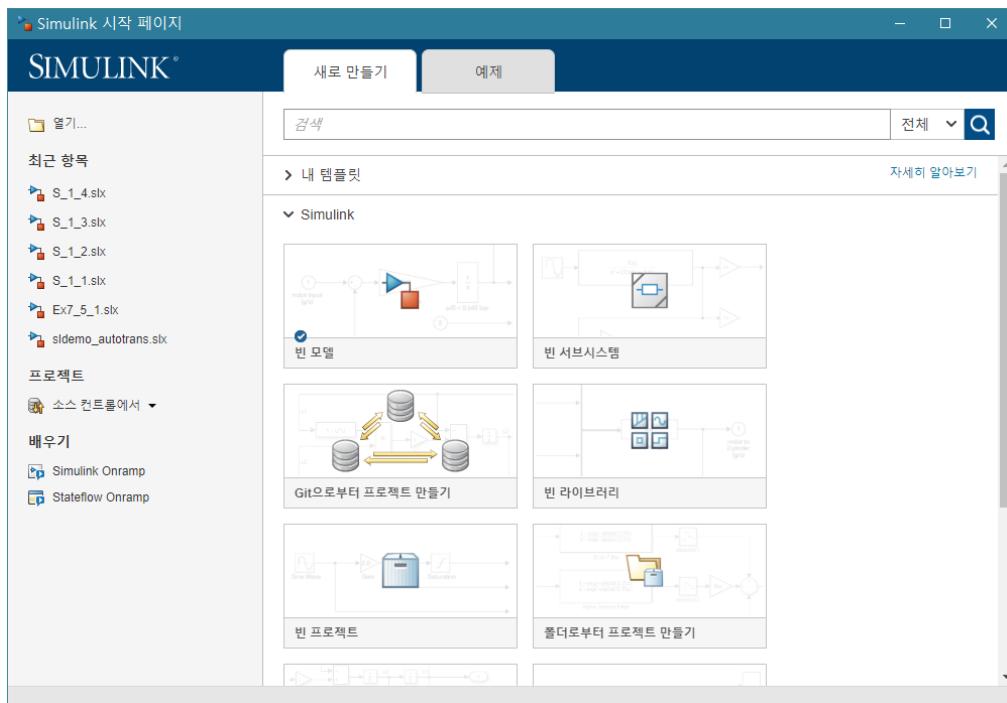


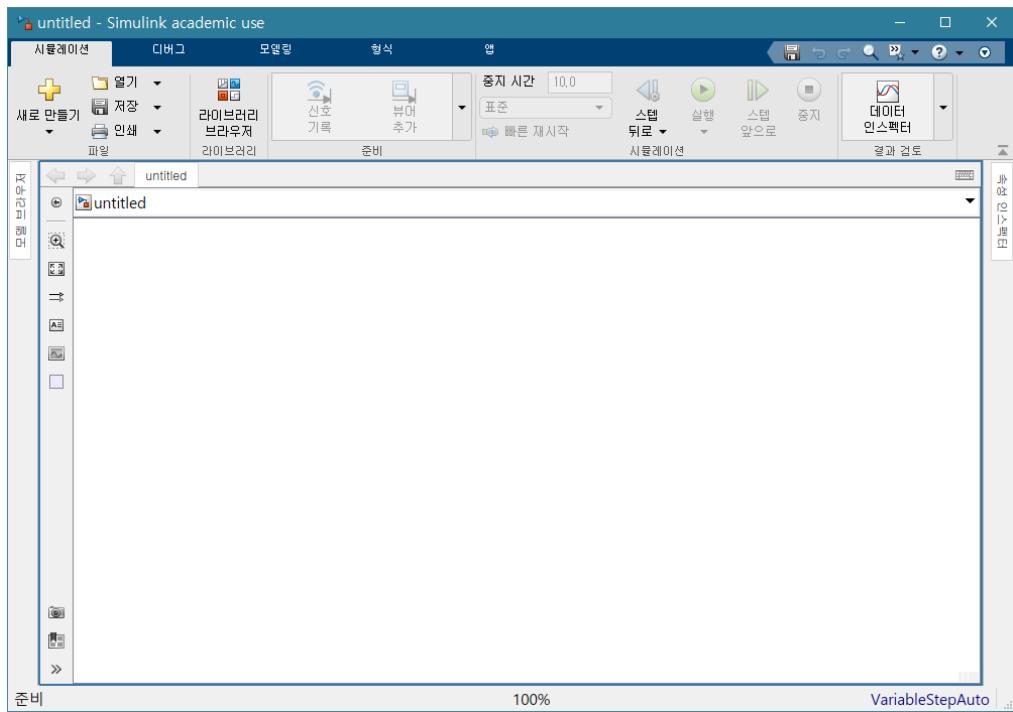
9.2 Simulink 의 시작

- Simulink 라이브러리를 클릭하여 Simulink 라이브러리 브라우저 창을 연다.



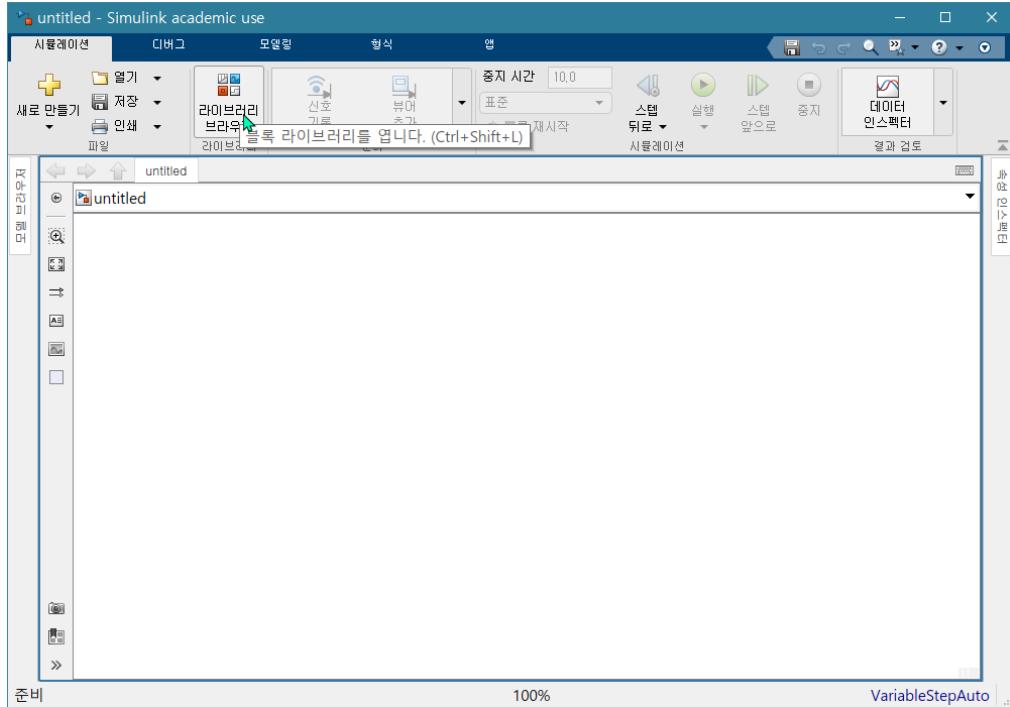
- 빈 모델을 선택한다.



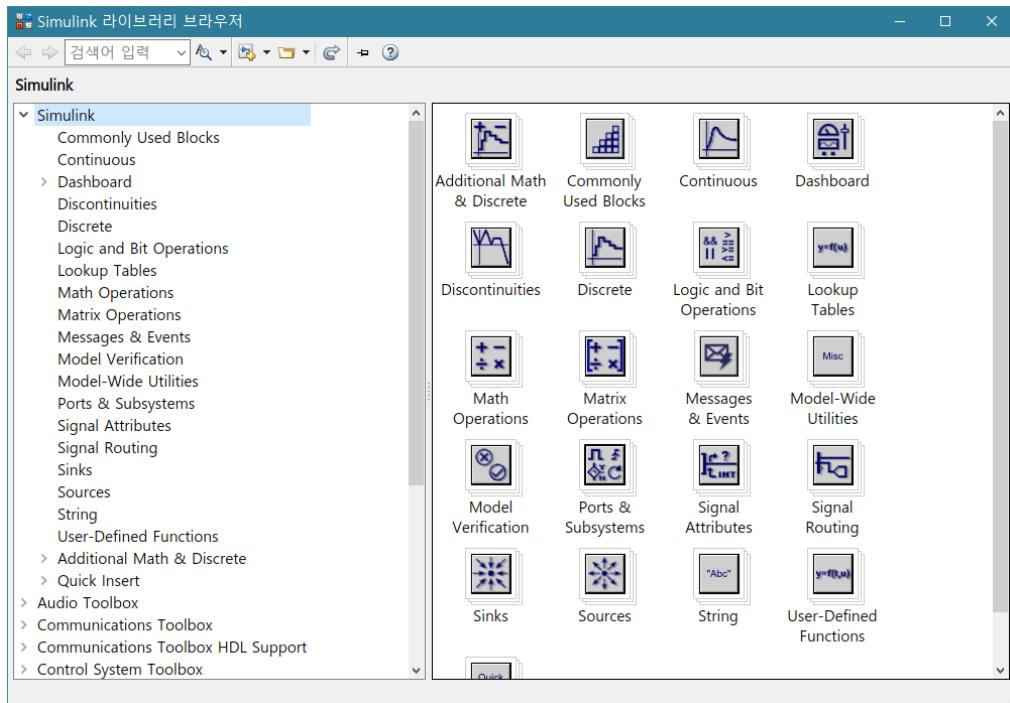


9.3 Simulink 의 구성

9.3.1 Simulink 라이브러리 브라우저

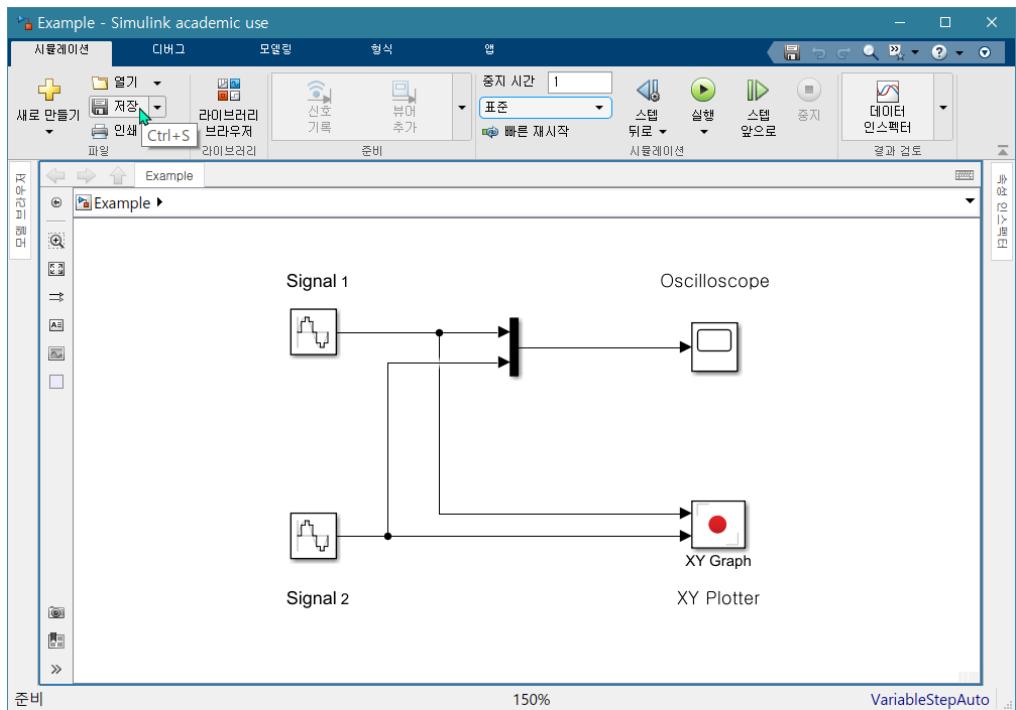


多样한 Simulink 라이브러리가 모여 있는 창이다.



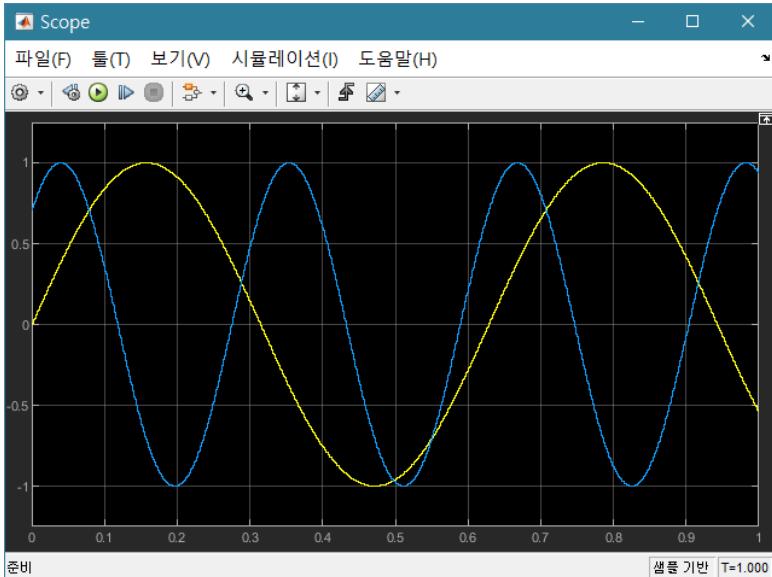
9.3.2 모델 창

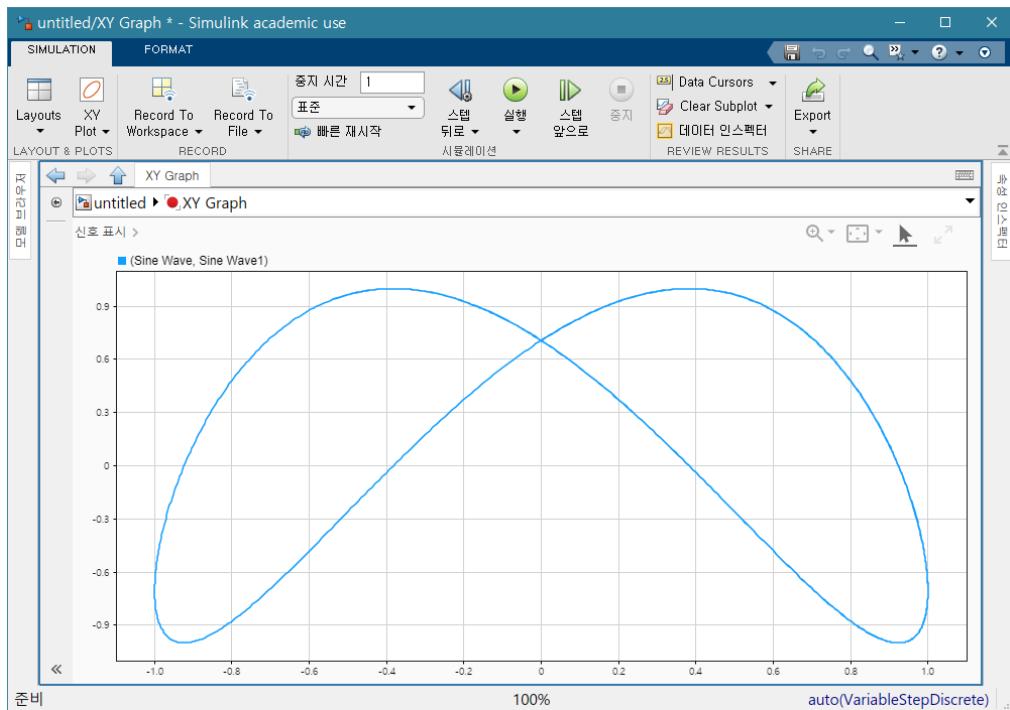
Simulink 시뮬레이션을 실행할 회로가 그려진 창이다.



9.3.3 그래프 창

▶ Simulink 시뮬레이션이 실행되었을 때 나타나는 그래프 창으로 자동적으로 나타나거나 블록을 클릭하면 나타나는 창이다.

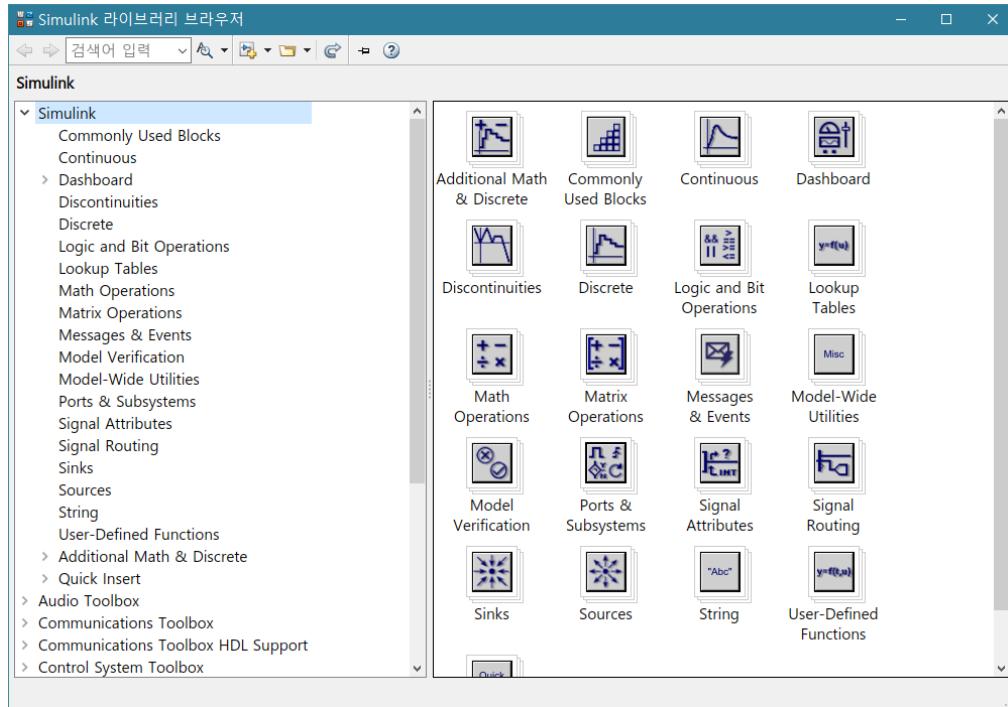




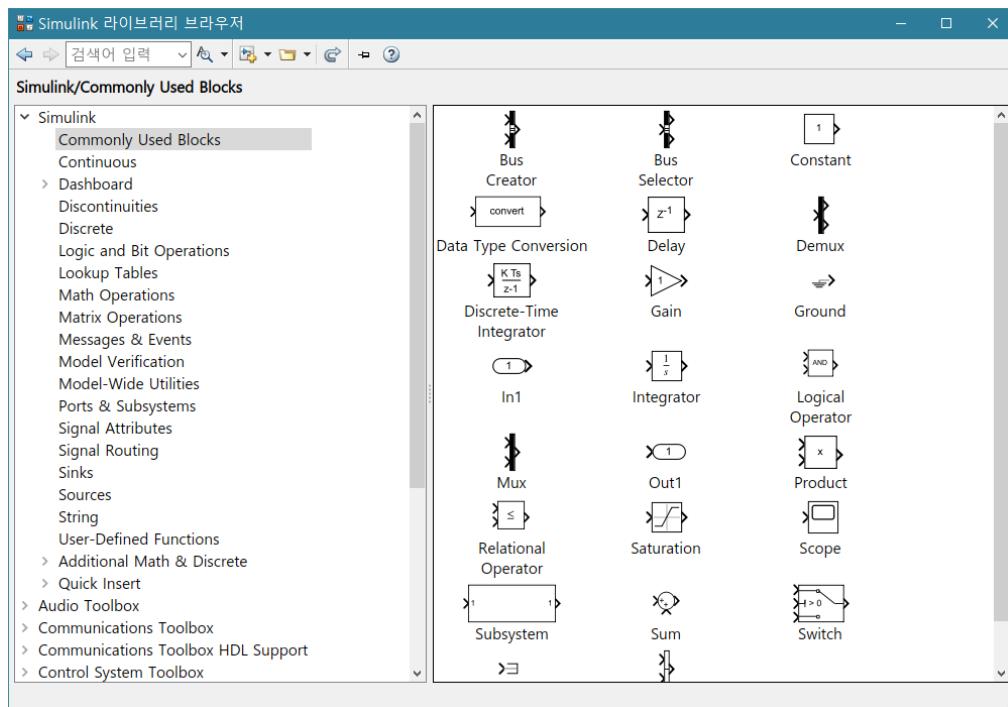
9.4 Simulink 블록 세트

9.4.1 Simulink 기본 블록 세트

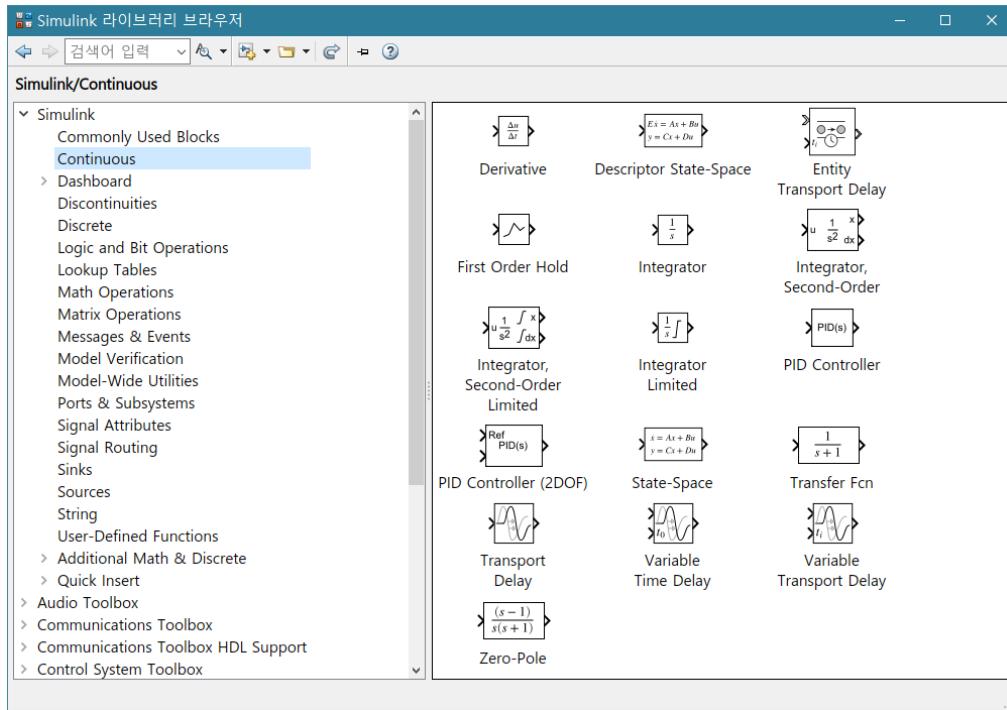
▶ 기본적으로 많이 사용되는 블록 세트로 분야별로 구분되어 있다.



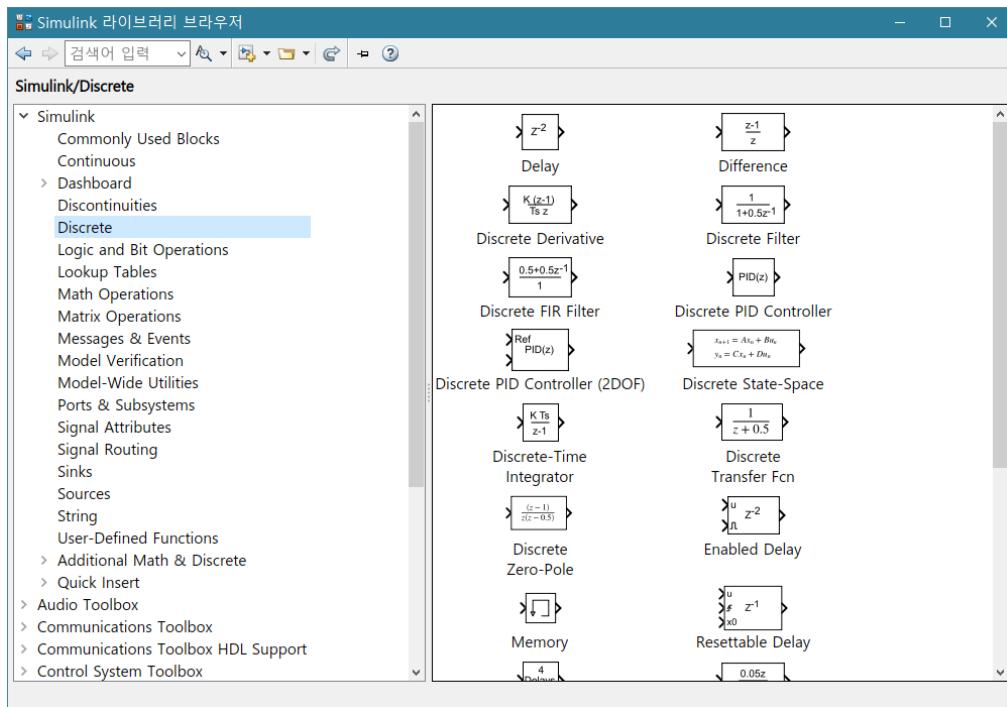
▶ Commonly Used Blocks



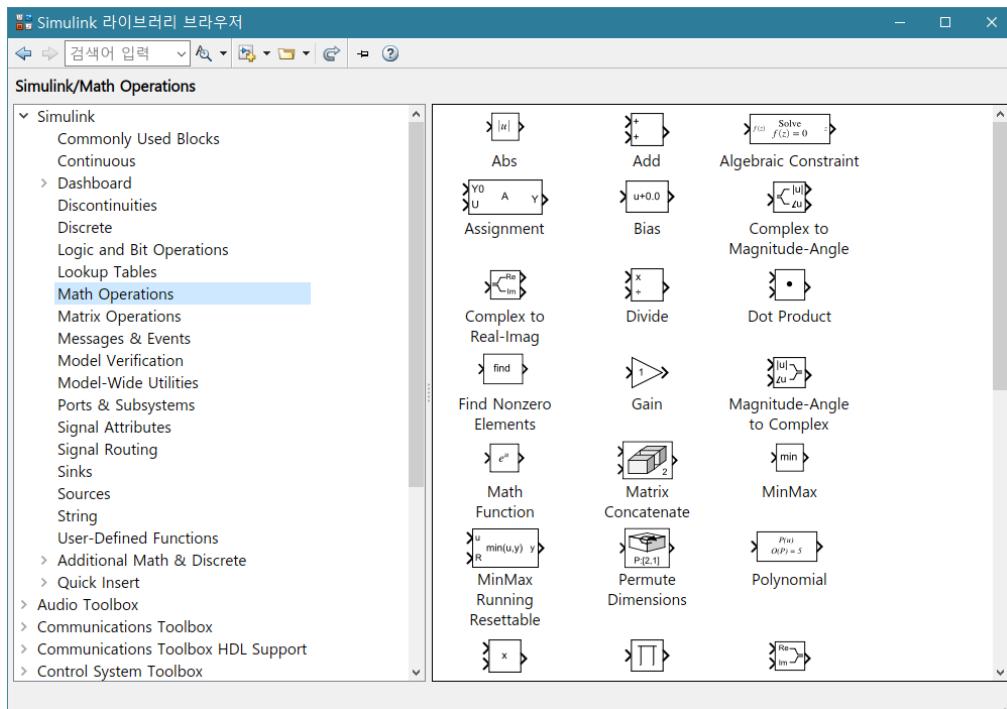
Continuous Blocks



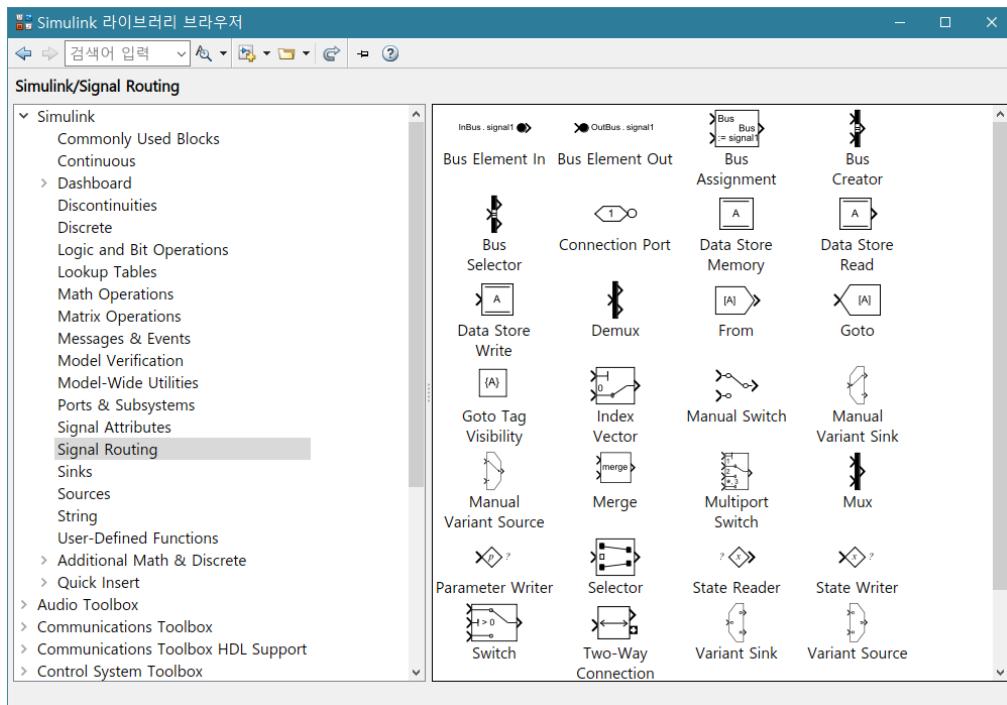
Discrete Blocks



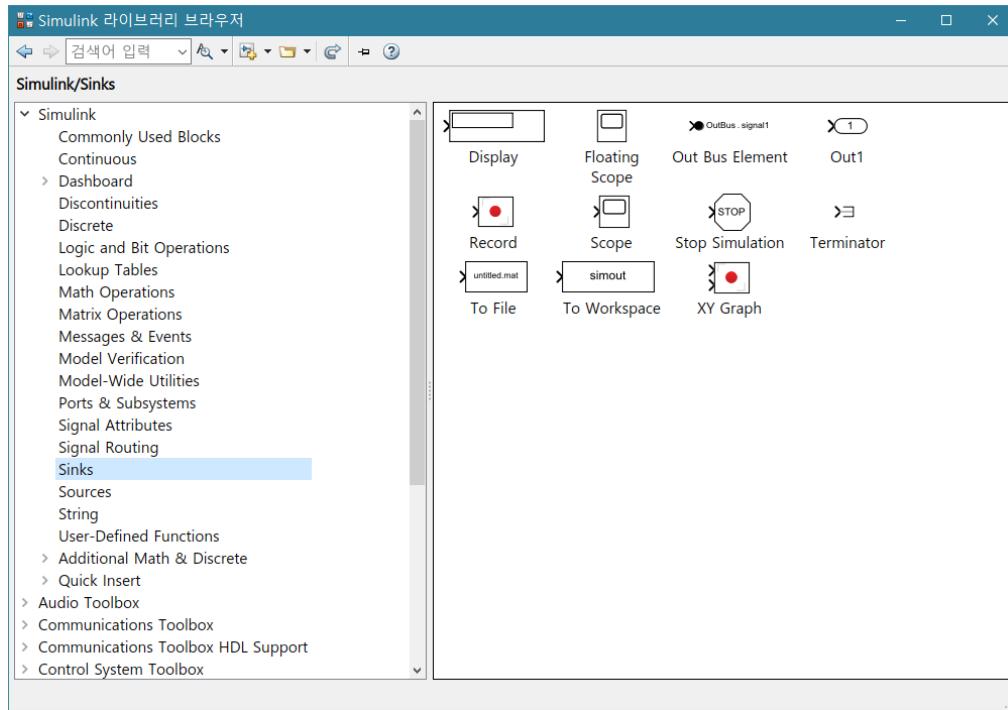
Math Operation Blocks



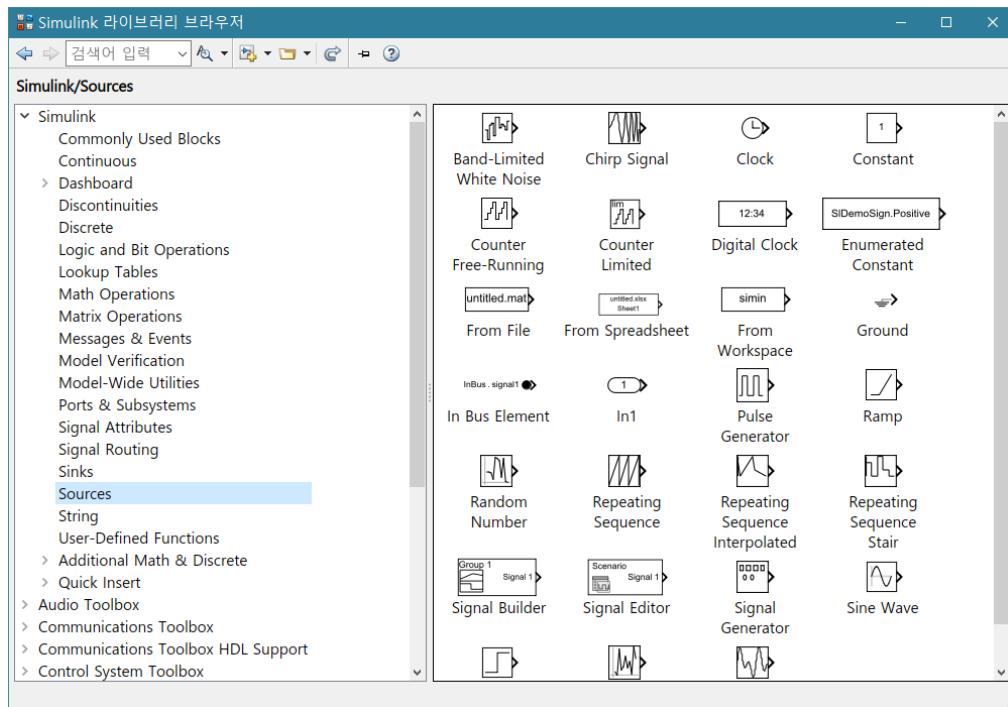
Signal Routing Blocks



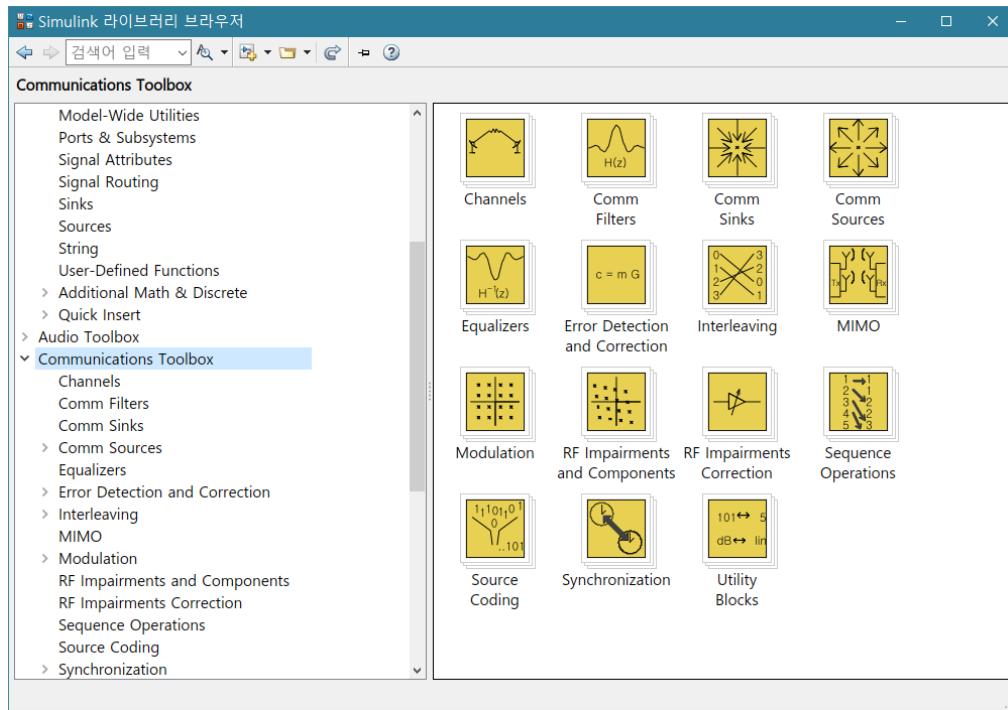
Sink Blocks



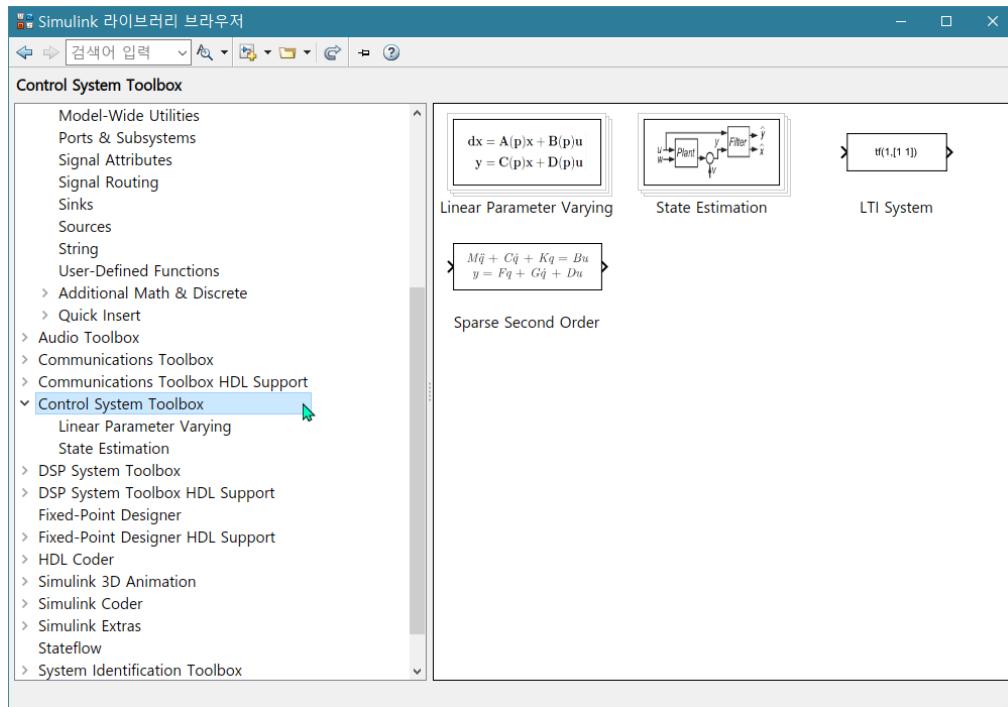
Source Blocks



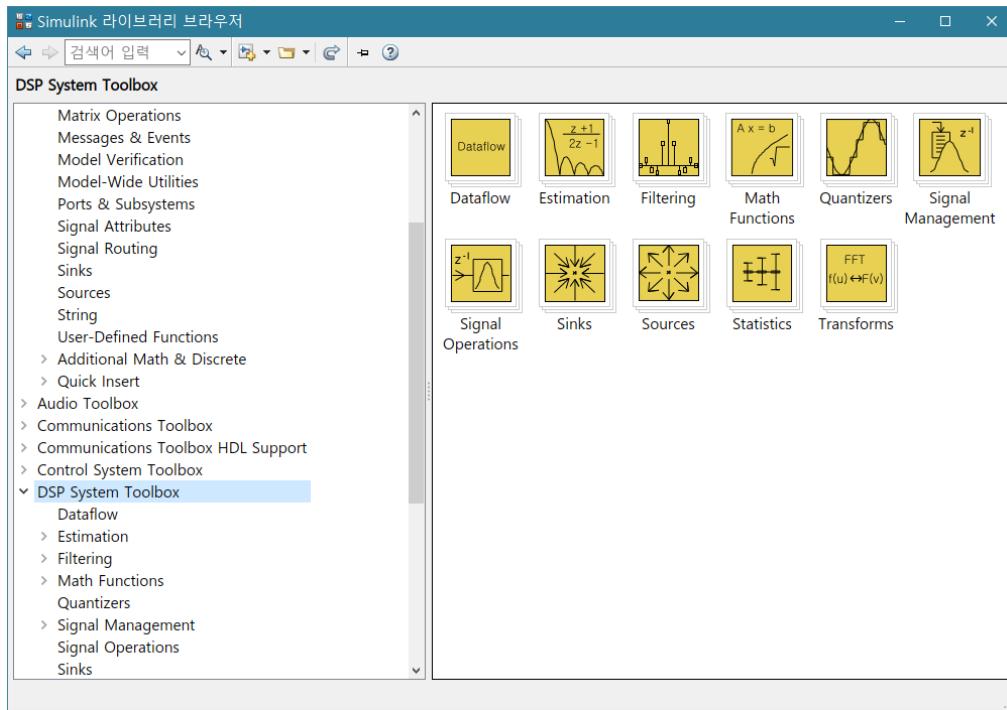
9.4.2 Communications Toolbox Blocks



9.4.3 Control System Toolbox Blocks



9.4.4 DSP System Toolbox Blocks



✚ Estimation Blocks

- ✓ Power Spectrum Estimation Blocks

✚ Filtering Blocks

- ✓ Filter Implementations Blocks

✚ Signal Operations Blocks

✚ Sinks Blocks

✚ Sources Blocks

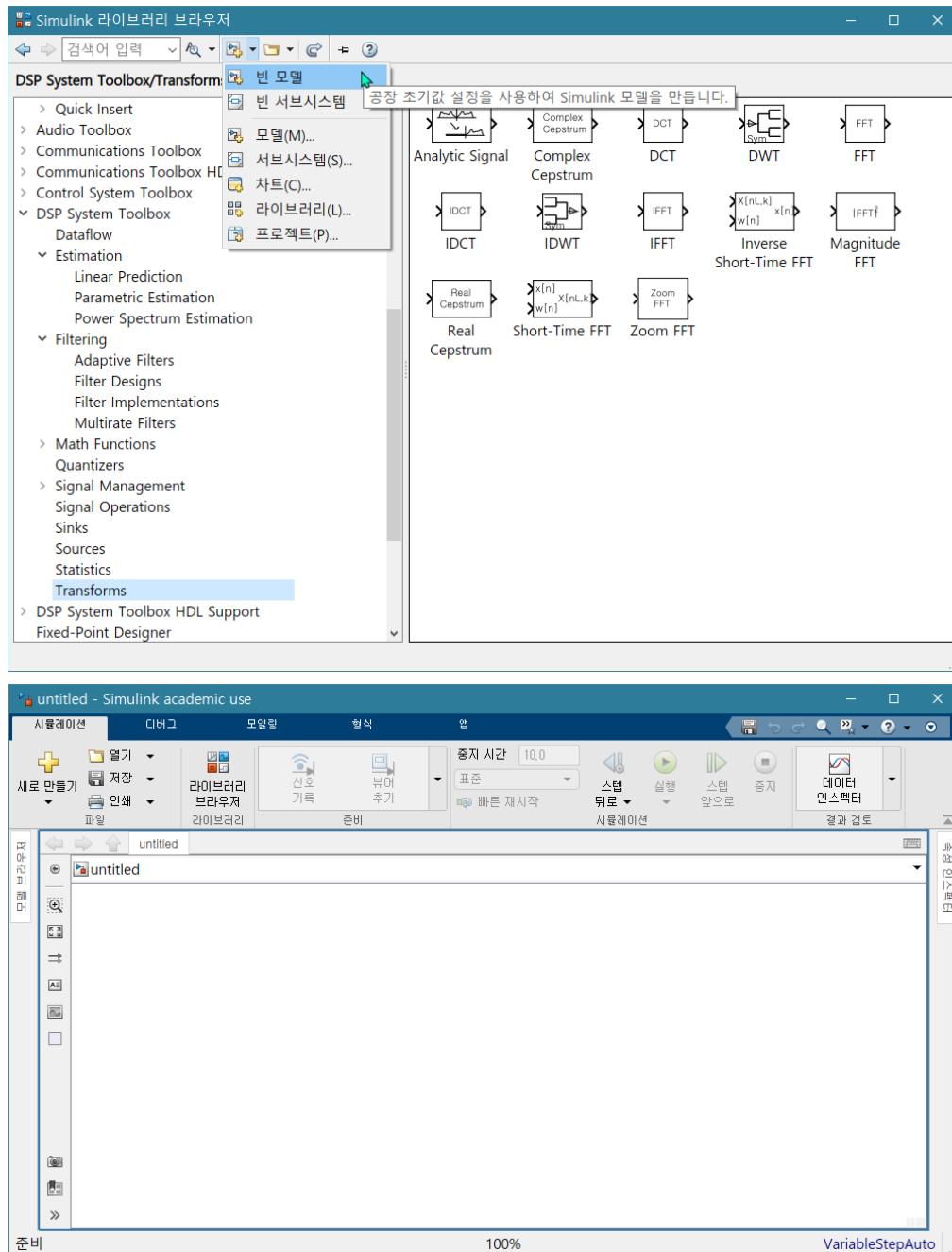
✚ Statistics Blocks

✚ Transforms Blocks

9.5 블록 모델

9.5.1 새 블록 모델 창 만들기

➊ 빈 모델 메뉴를 선택하면 빈 모델 창이 만들어진다.

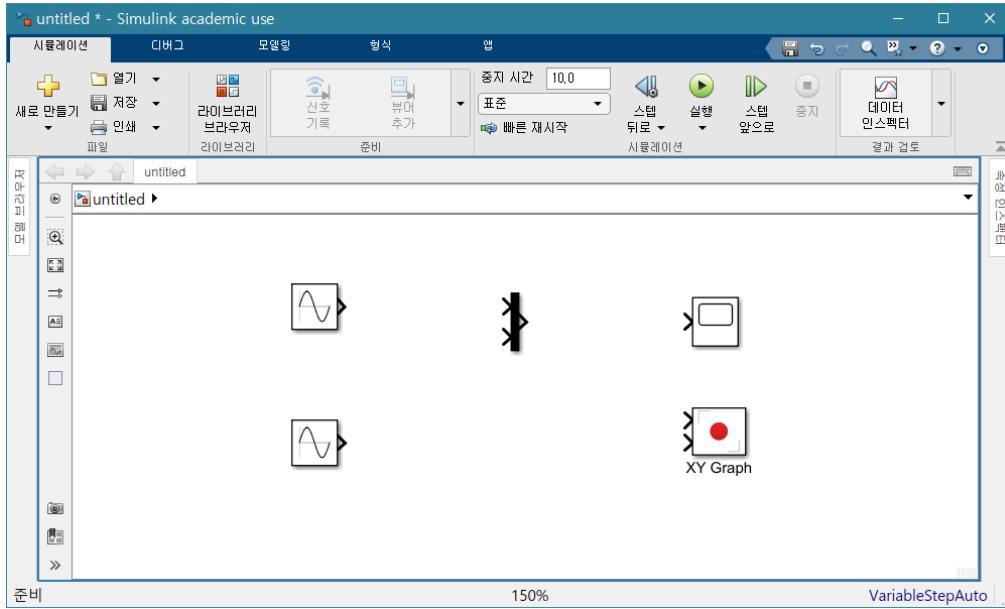


9.5.2 블록 불러오기

➋ Simulink Library Browser에서 아이콘을 드래그하여 블록을 불러온다.

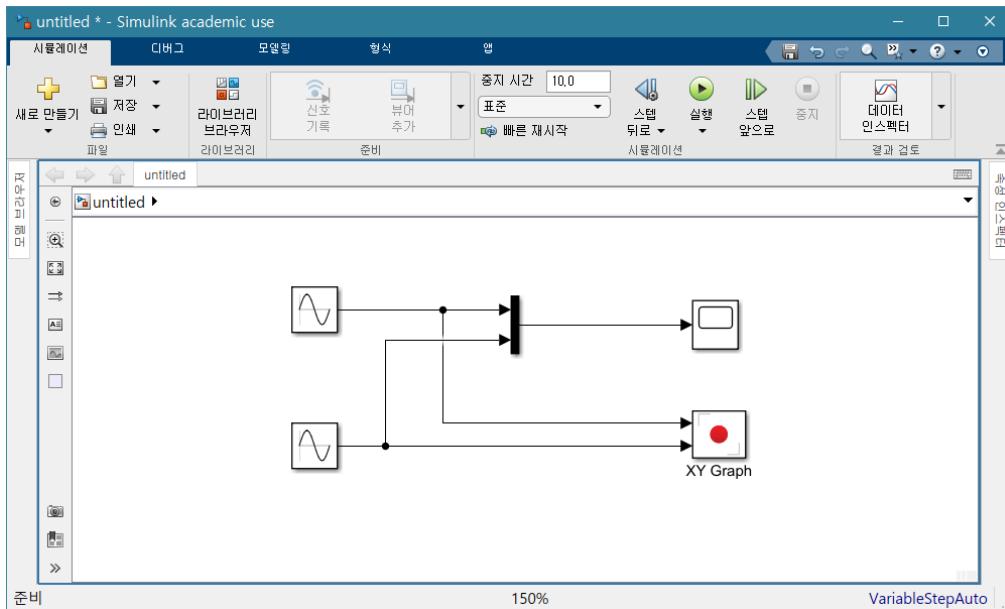
- ✓ Simulink – Sources – Sine Wave

- ✓ Simulink – Sinks – Scope
- ✓ Simulink – Sinks – XY Graph
- ✓ Simulink – Signal Routing – Mux



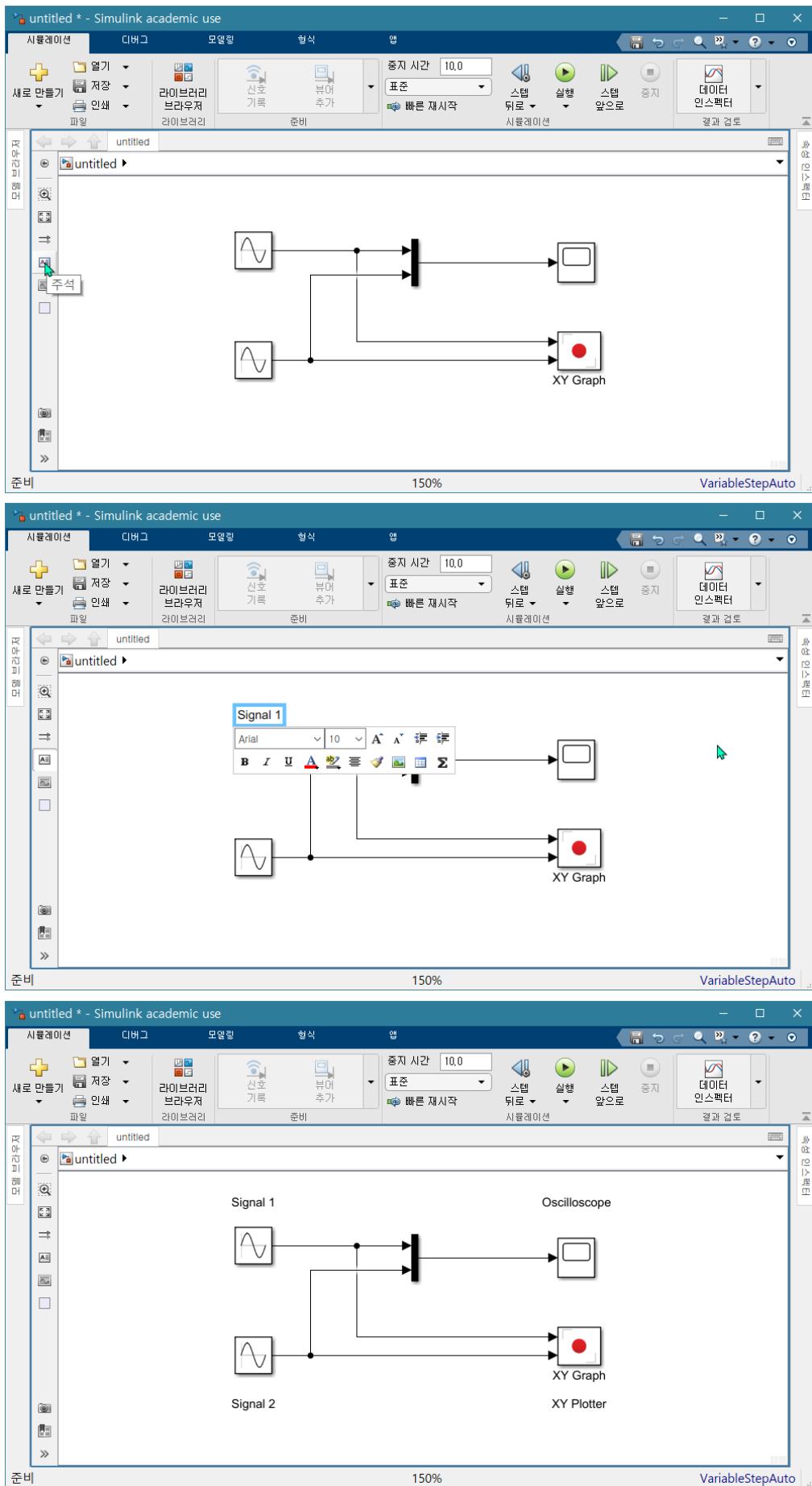
9.5.3 선 연결

- ✚ 마우스 왼쪽(또는 오른쪽) 버튼을 누른 상태로 블록의 단자를 연결한다.



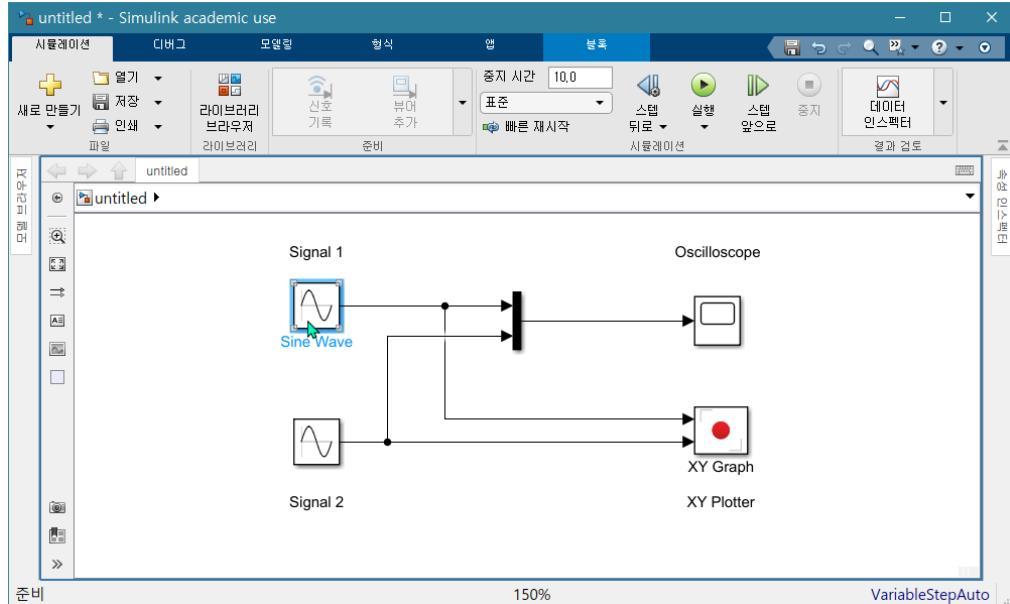
9.5.4 레이블 입력

- ✚ Annotation 아이콘을 누르고 레이블을 넣고자 하는 위치에 글씨를 입력한다.
- ✚ 완성된 글씨는 크기, 색 등을 수정하고 적당한 위치로 이동시킨다.

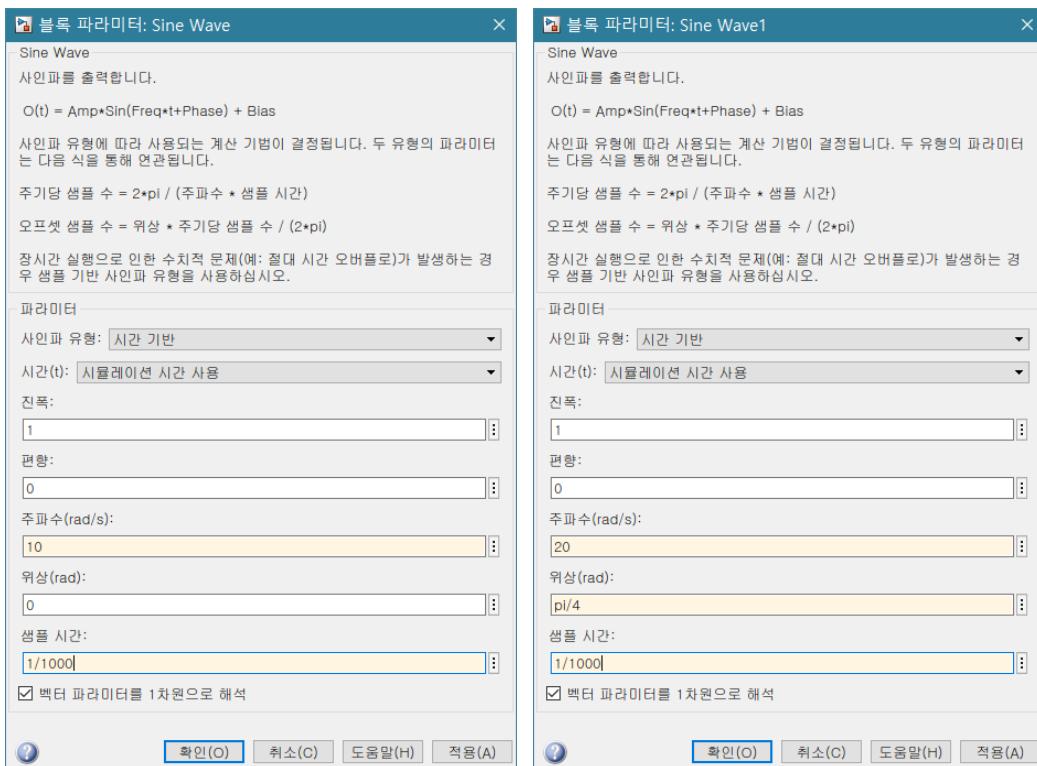


9.5.5 블록 파라미터 설정

- ▶ 블록을 더블 클릭하면 블록 파라미터 창을 열 수 있다.



- ✓ 블록 파라미터 창



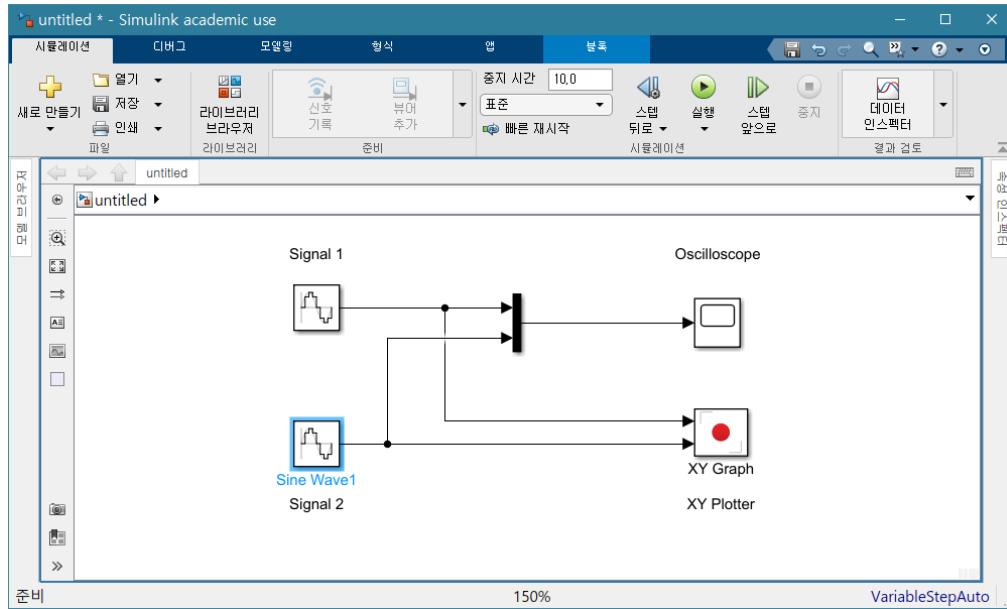
- ▶ Sine Wave 블록 파라미터 수정

- ✓ 진폭 : 1
- ✓ 주파수 : 10
- ✓ 위상 : 0

- ✓ 샘플 시간 : 1/1000

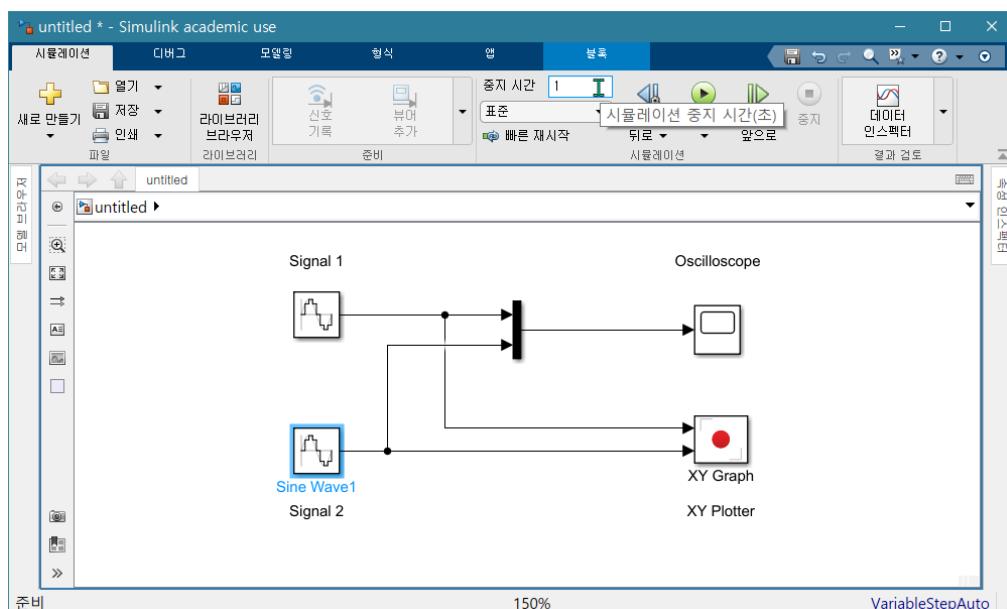
✚ Sine Wave 1 블록 파라미터 설정

- ✓ 진폭 : 1
- ✓ 주파수 : 20
- ✓ 위상 : $\pi/4$
- ✓ 샘플 시간 : 1/1000

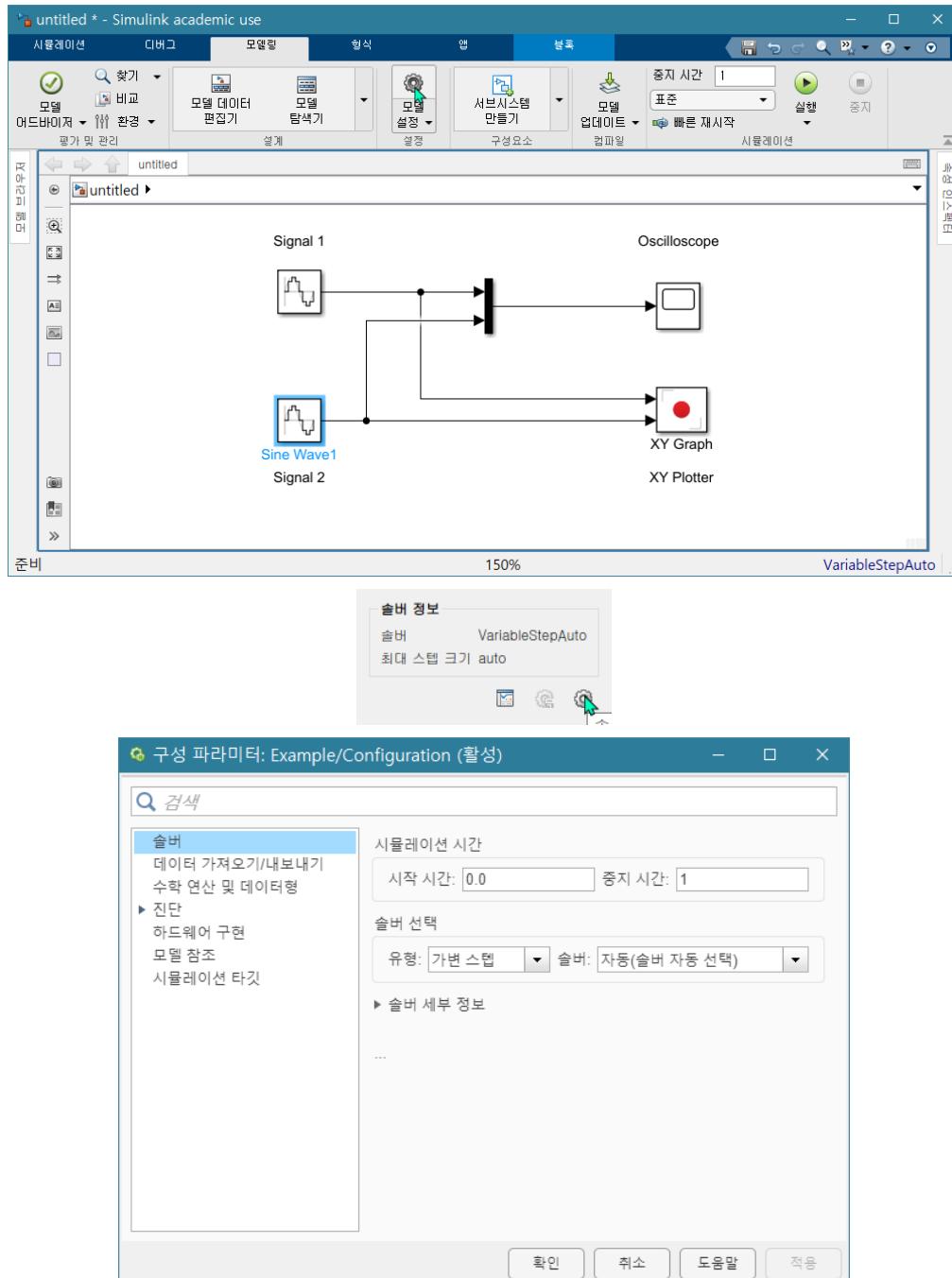


9.5.6 모델 시뮬레이션 구성 파라미터 설정

✚ 중지 시간을 1로 변경하고 시뮬레이션 모드를 표준을 선택한다.

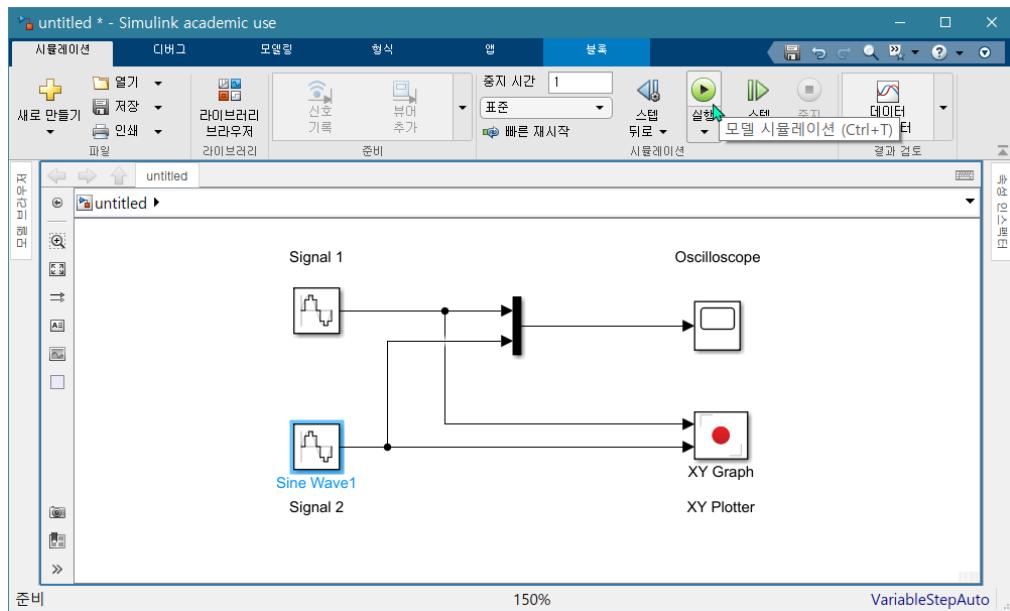


- 자세한 구성 파라미터를 변경하려면 메뉴 모델링 – 모델 설정을 클릭하거나 오른쪽 아래의 VariableStepAuto 를 클릭하여 설정 아이콘을 클릭하면 구성 파라미터 창을 열 수 있다..

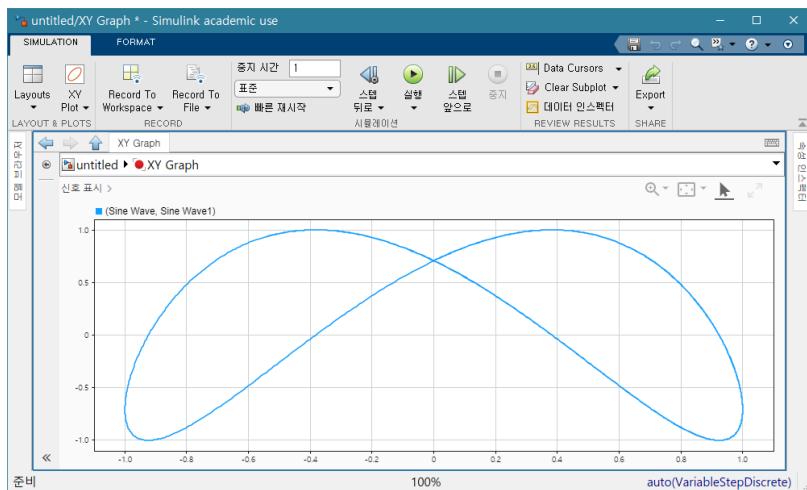


9.5.7 블록 모델 실행

- Run 아이콘 클릭한다.

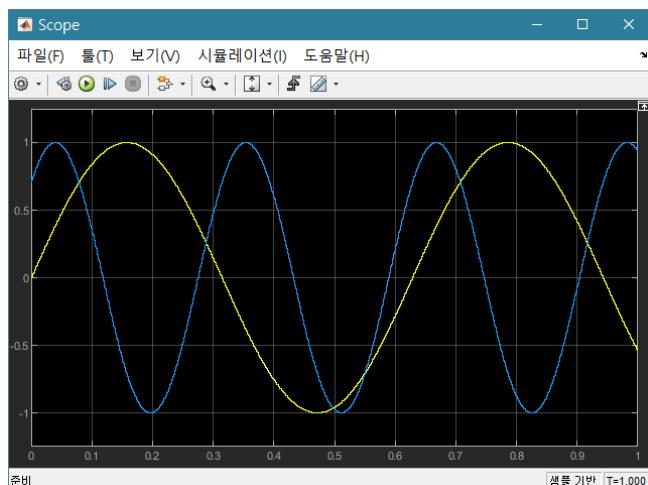


XY Graph 아이콘을 더블 클릭하면 파형 창을 볼 수 있다.



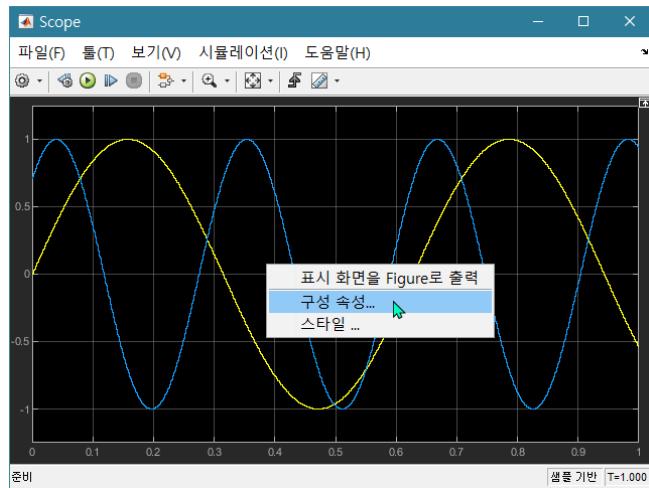
Oscilloscope 아이콘 더블 클릭하면 오실로스코프 파형 창을 볼 수 있다.

- ✓ 오실로스코프 창이 미리 나오도록 설정할 수 있다. 그러나 파형이 나타나려면 시뮬레이션을 실행해야 한다.

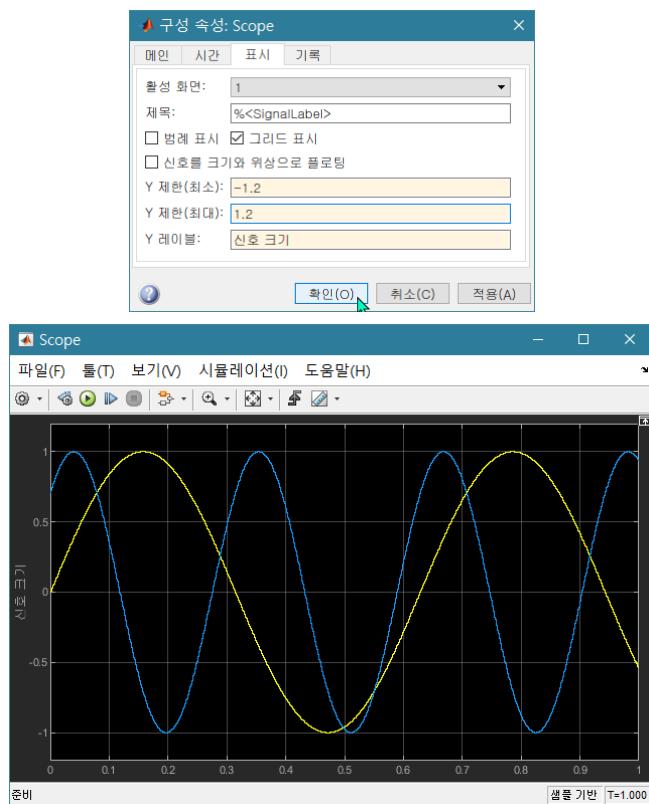


9.5.8 파형 그림 수정

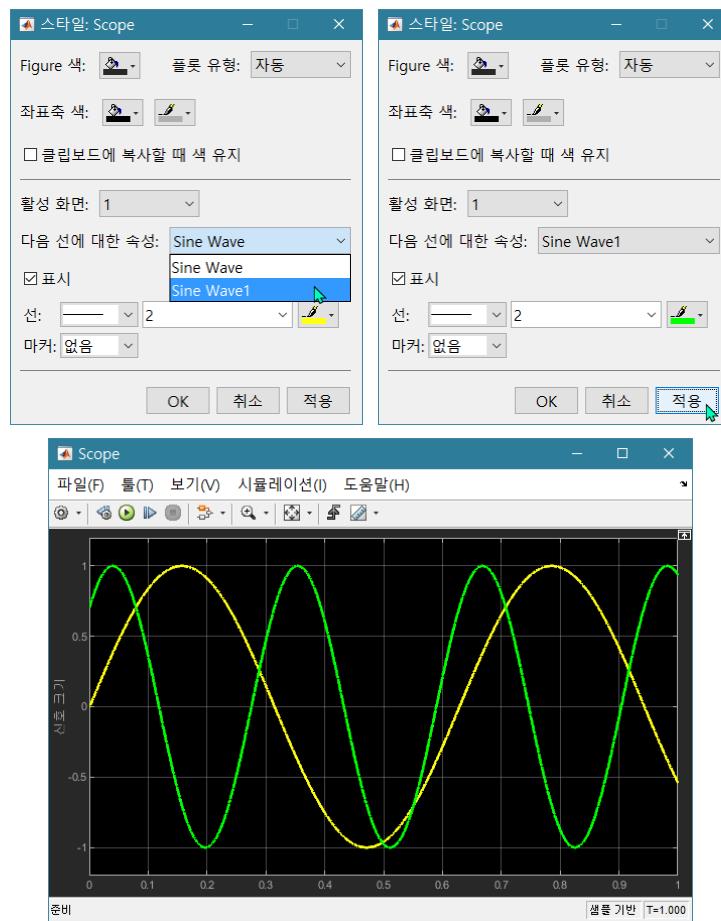
- 파형 창에서 마우스 오른쪽 버튼을 누르면 구성 속성과 그래프 스타일을 설정할 수 있다.



- 좌표축 범위 설정, 그리드 on/off, y 축 레이블 작성

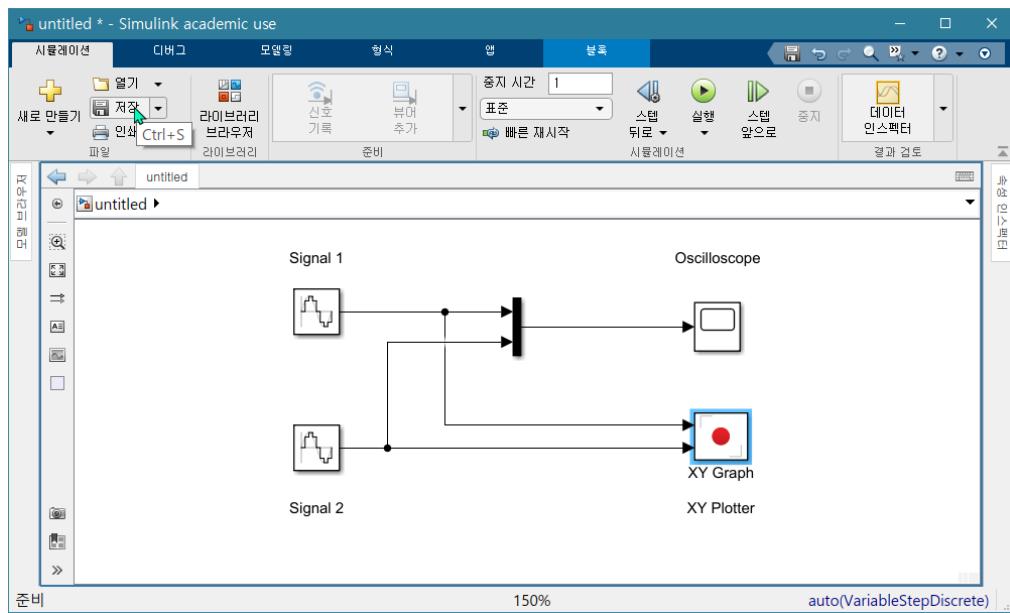


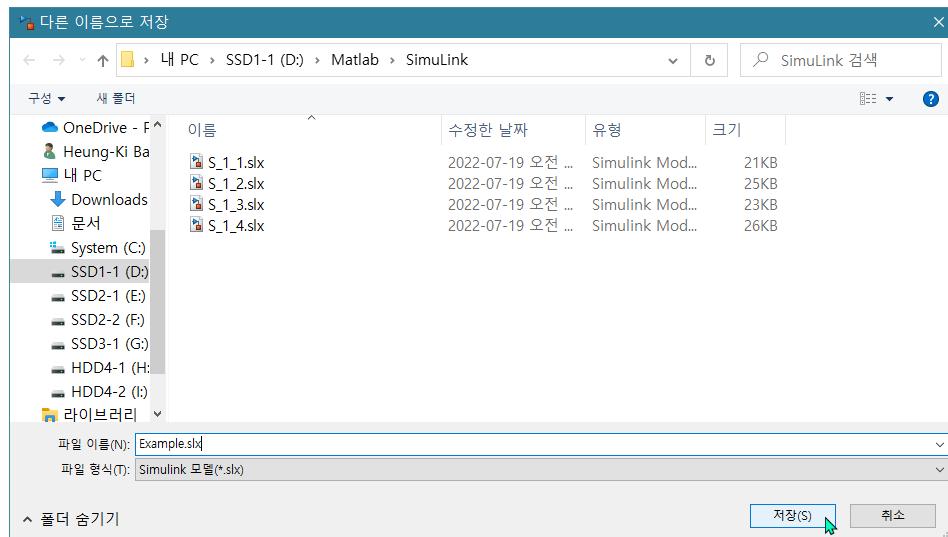
- 그래프 선 굵기, 색상 변경



9.5.9 블록 모델 저장

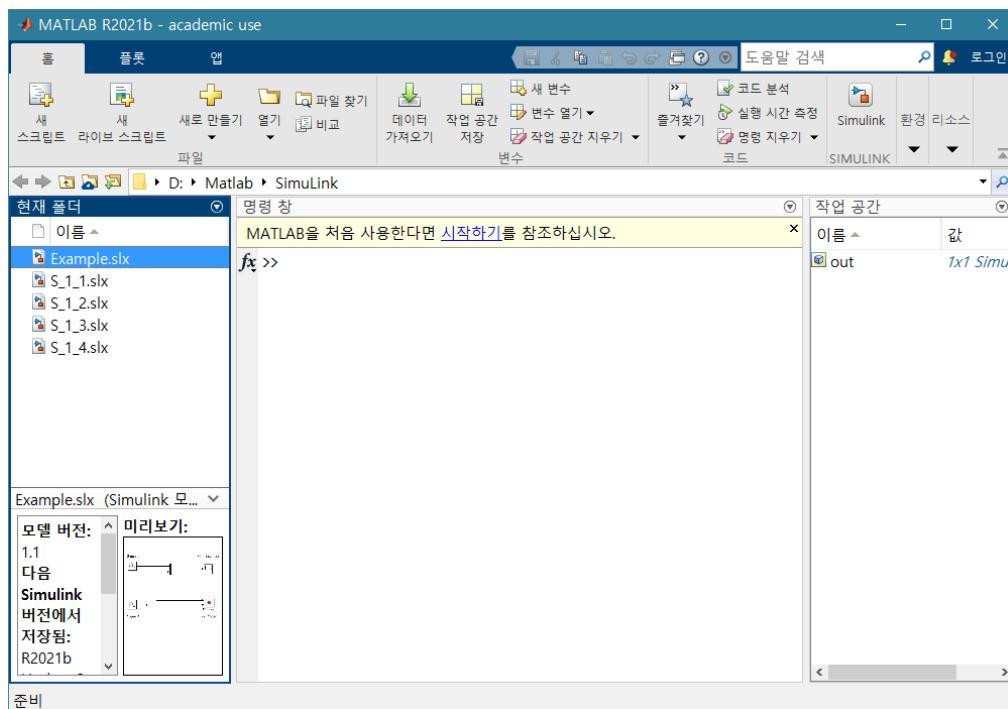
저장 아이콘을 클릭한다.

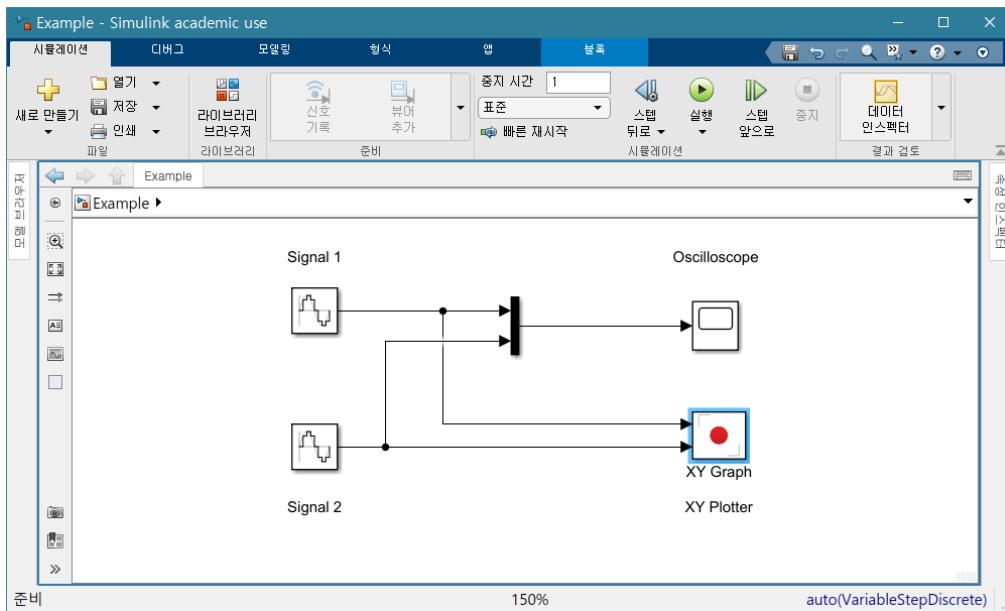




9.5.10 블록 모델 불러 오기

MATLAB 메인 창에서 Simulink 파일을 더블 클릭하거나 Simulink 창의 열기 아이콘을 선택하여 Simulink 모델을 불러 올 수 있다.





9.5.11 블록 편집

▶ 블록을 선택하고 마우스 우측 버튼을 클릭하면 블록을 편집할 수 있다.

- ✓ 블록 이름 on/off
- ✓ 블록 이름과 블록 반전
- ✓ 블록 회전
- ✓ 블록 전경색/배경색 변경
- ✓ 캔버스 색 변경



9.6 신호의 파형

- Simulink에서 모든 신호는 샘플링하여 이산 시간 신호로 변환하여 처리한다.
크기가 A , 주파수가 F_0 Hz인 정현파 신호 $x_c(t)$ 를 샘플링 주파수 F_s 로 샘플링하여 얻은 이산 시간 신호 $x[n]$ 은 다음과 같다.

$$x_c(t) = \sin(\Omega_0 t) = \sin(2\pi F_0 t)$$

$$x[n] = x_c(nT) = \sin(\Omega_0 nT) = \sin(\omega_0 n) = \sin(2\pi f_0 n), \quad \omega_0 = \Omega_0 T$$

신호의 파형을 보는 블록

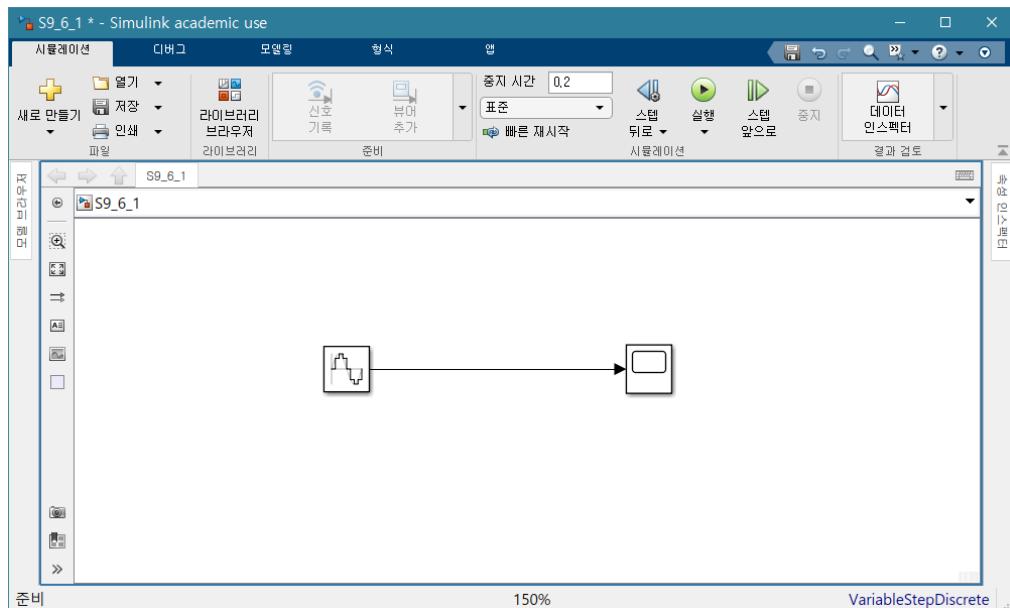
- ✓ Simulink – Sinks – Scope
- ✓ DSP System Toolbox – Sinks – Time Scope
- ✓ DSP System Toolbox – Sinks – Array Plot

9.6.1 아날로그 정현파 신호

- 크기가 1, 주파수 100Hz인 정현파를 1kHz로 샘플링하여 파형 관찰.

$$x(t) = A_1 \sin(2\pi F_1 t), \quad A_1 = 1, F_1 = 100 \text{ Hz}$$

블록 모델



- ✓ Simulink – Sources – Sine Wave
- ✓ Simulink – Sinks – Scope

블록 파라미터

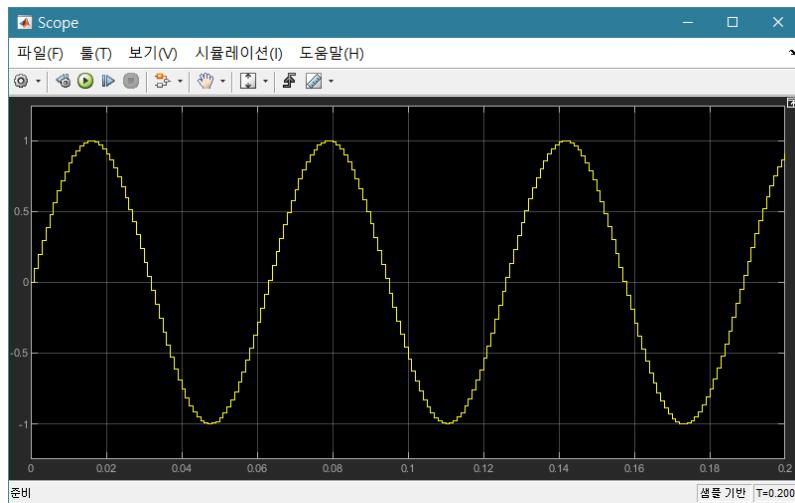
✓ Sine Wave

- ✗ 진폭 : 1
- ✗ 주파수 : 100
- ✗ 샘플 시간 : 1/1000

✚ 시뮬레이션 구성 파라미터

- ✓ 증지 시간 : 0.2

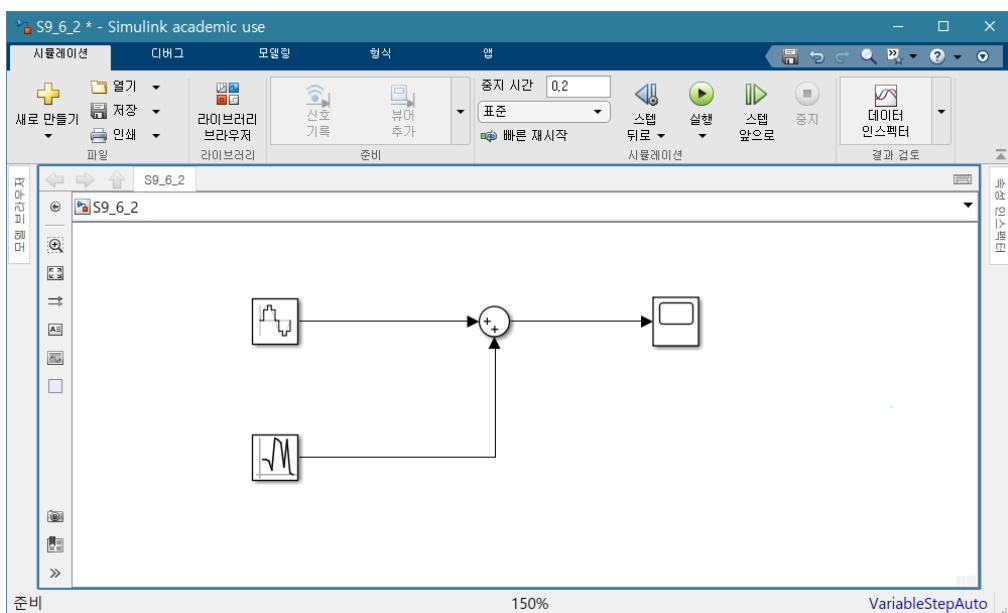
✚ 출력 파형



9.6.2 잡음이 포함된 아날로그 정현파 신호

$$x(t) = A_1 \sin(2\pi F_1 t) + w(t), \quad A_1 = 1, F_1 = 100 \text{ Hz}, \sigma_w^2 = 0.1$$

✚ 블록 모델



- ✓ Simulink – Sources – Sine Wave
- ✓ Simulink – Sources – Random Number
- ✓ Simulink – Math Operation – Sum
- ✓ Simulink – Sinks – Scope

▶ 블록 파라미터

- ✓ Sine Wave
 - ✗ 진폭 : 1
 - ✗ 주파수 : 100
 - ✗ 샘플 시간 : 1/1000
- ✓ Random Source
 - ✗ 평균 : 0
 - ✗ 분산 : 0.1
 - ✗ 샘플 시간 : 1/1000

▶ 시뮬레이션 구성 파라미터

- ✓ 중지 시간 : 0.2

▶ 출력 파형

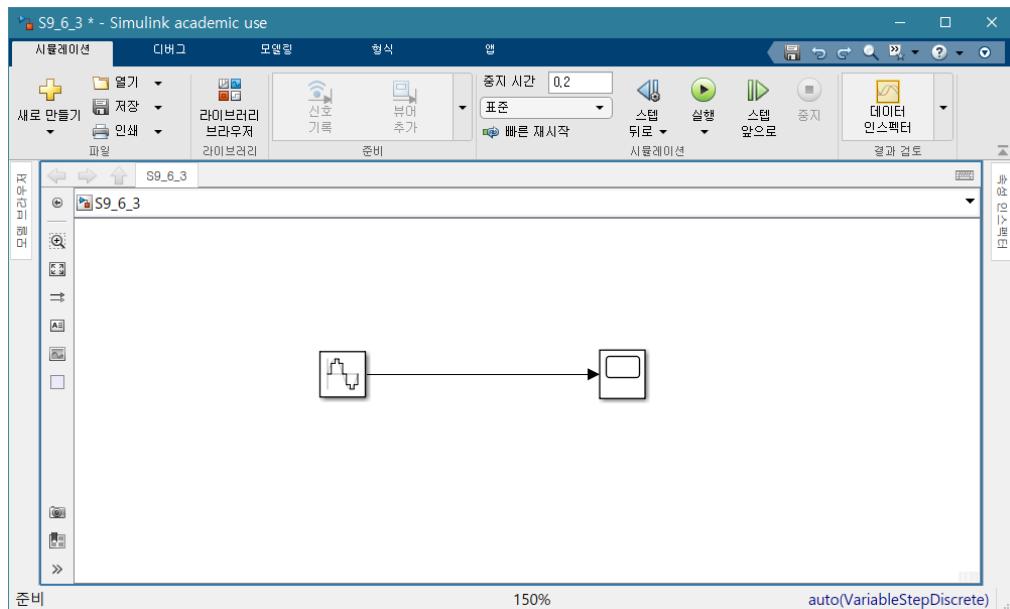


9.6.3 아날로그 정현파 신호 (2 채널)

$$x_1(t) = A_1 \sin(2\pi F_1 t + \phi_1), \quad A_1 = 1, F_1 = 50 \text{ Hz}, \phi_1 = 0$$

$$x_2(t) = A_2 \sin(2\pi F_2 t + \phi_2), \quad A_2 = 1, F_2 = 100 \text{ Hz}, \phi_2 = \pi / 4$$

▶ 블록 모델



✓ Simulink – Sources – Sine Wave

✓ Simulink – Sinks – Scope

▶ 블록 파라미터

✓ Sine Wave

✗ 진폭 : [1 1]

✗ 주파수 : [50 100]

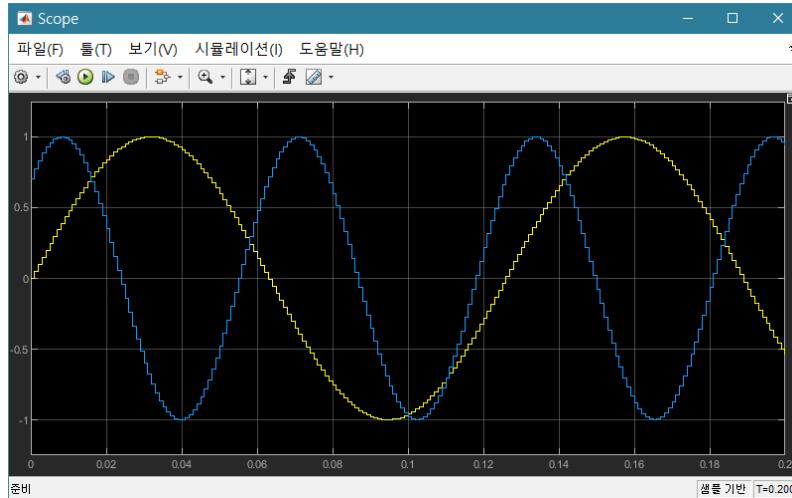
✗ 위상(rad) : [0 pi/4]

✗ 샘플 시간 : 1/1000

▶ 시뮬레이션 구성 파라미터

✓ 중지 시간 : 0.2

▶ 출력 파형

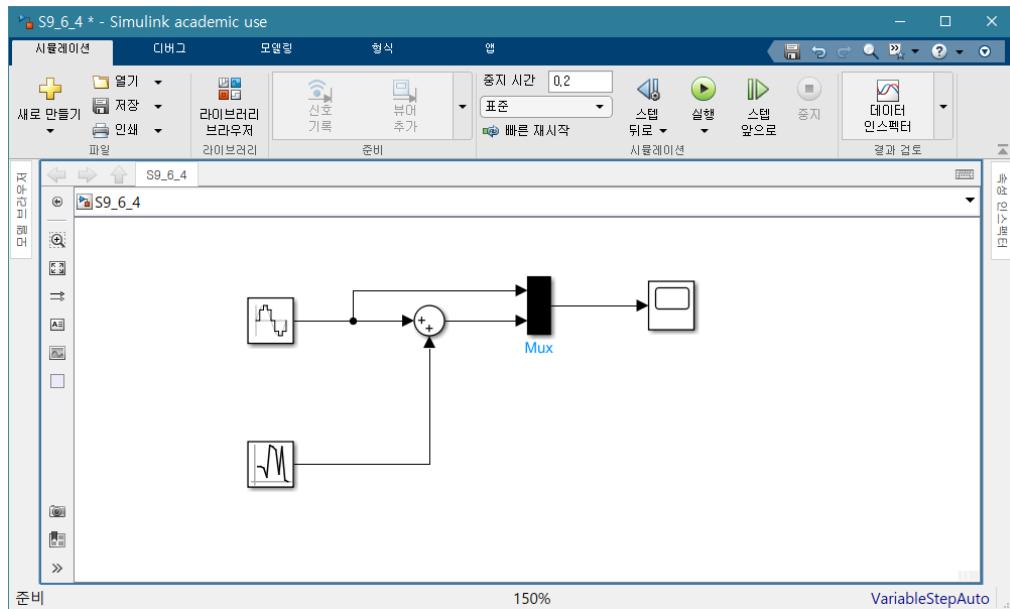


9.6.4 잡음이 포함된 아날로그 신호 (2 채널)

$$x_1(t) = A_1 \sin(2\pi F_1 t), \quad A_1 = 1, F_1 = 100 \text{ Hz}$$

$$x_2(t) = x_1(t) + w(t), \quad \sigma_w^2 = 0.1$$

블록 모델



- ✓ Simulink – Sources – Sine Wave
- ✓ Simulink – Sources – Random Number
- ✓ Simulink – Math Operation – Sum
- ✓ Simulink – Signal Routing – Mux
- ✓ Simulink – Sinks – Scope

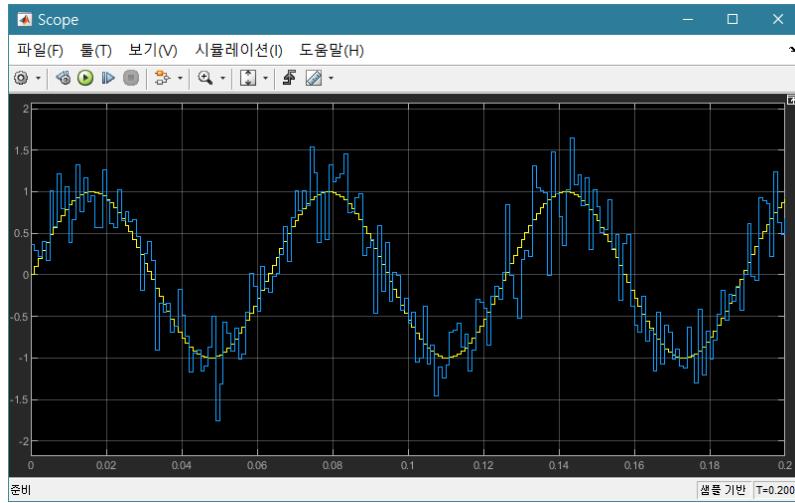
블록 파라미터

- ✓ Sine Wave
 - ✗ 진폭 : 1
 - ✗ 주파수 : 100
 - ✗ 샘플 시간 : 1/1000
- ✓ Random Source
 - ✗ 평균 : 0
 - ✗ 분산 : 0.1
 - ✗ 샘플 시간 : 1/1000

시뮬레이션 구성 파라미터

- ✓ 중지 시간 : 0.2

▶ 출력 파형

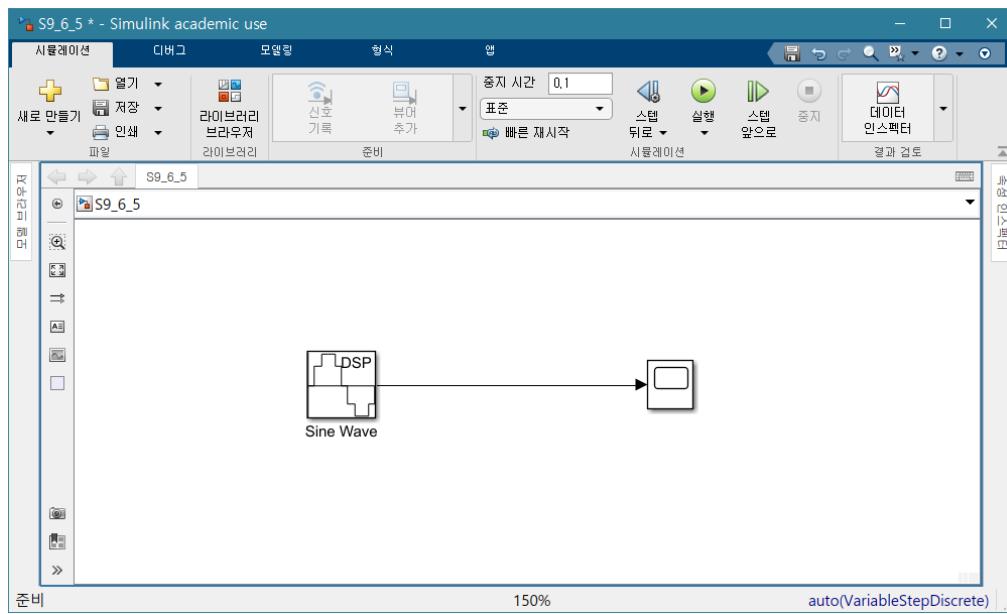


9.6.5 디지털 정현파 신호

$$x(t) = A_i \sin(2\pi F_i t), \quad A_i = 1, F_i = 100 \text{ Hz}$$

$$F_s = 1 \text{ kHz} \Rightarrow x[n] = x_c(nT_s) = A_i \sin\left(\frac{2\pi F_i}{F_s} n\right) = A_i \sin(\omega_i n)$$

▶ 블록 모델



- ✓ DSP System Toolbox – Sources – Sine Wave
- ✓ DSP System Toolbox – Sinks – Time Scope

▶ 블록 파라미터

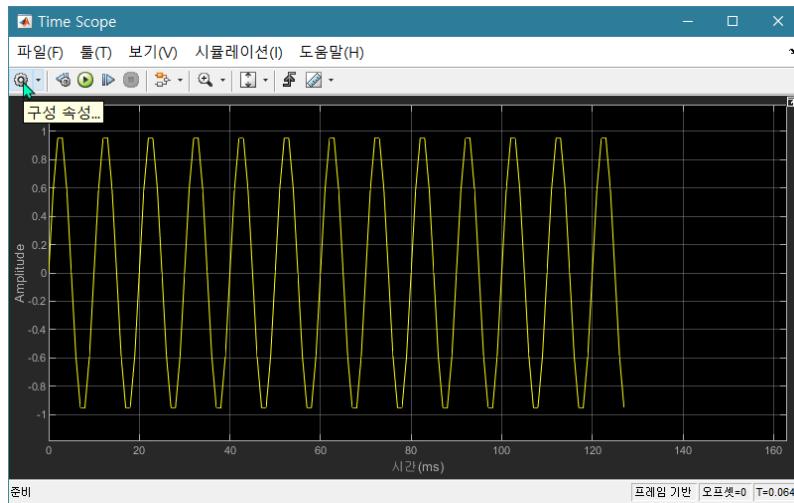
✓ Sine Wave

- ✗ Main – Amplitude : 1
- ✗ Main – Frequency : 100
- ✗ Main – Sample time : 1/1000
- ✗ Main – Samples per frame : 64

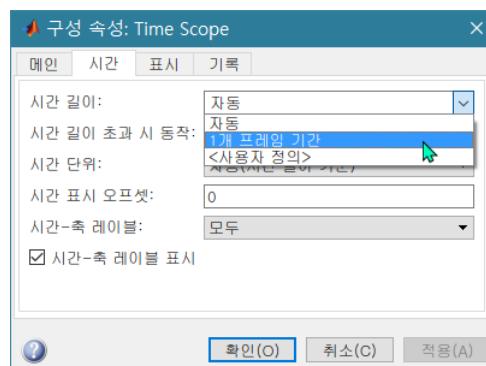
➊ 시뮬레이션 구성 파라미터

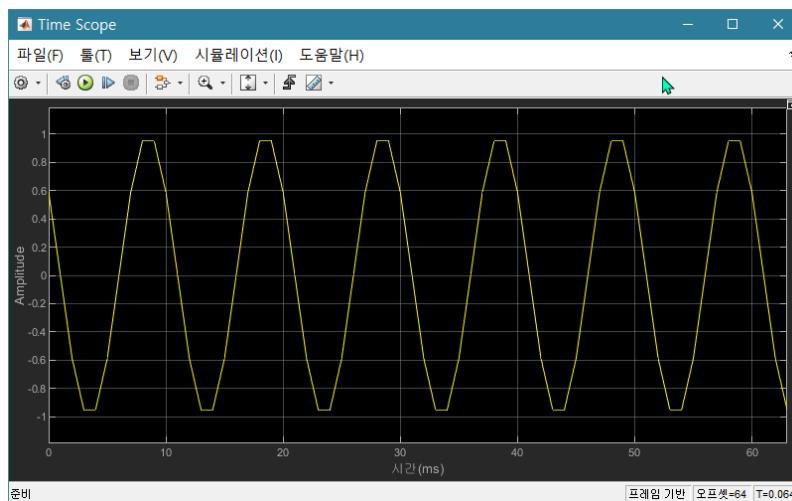
- ✓ 중지 시간 : 0.1

➋ 출력 파형



- ✓ 시간 축 설정 변경 : 파형 창에서 마우스 오른쪽 버튼을 누르거나 위 그림의 구성 속성 아이콘을 클릭하여 구성 속성 창을 연 후 시간 탭을 선택하고 시간 길이를 1개 프레임 기간으로 선택한다.
- ✓ 중지 시간을 크게 주어도 구성 속성 창에서 시간 축의 범위를 조절하면 원하는 시간만큼의 파형을 볼 수 있다.



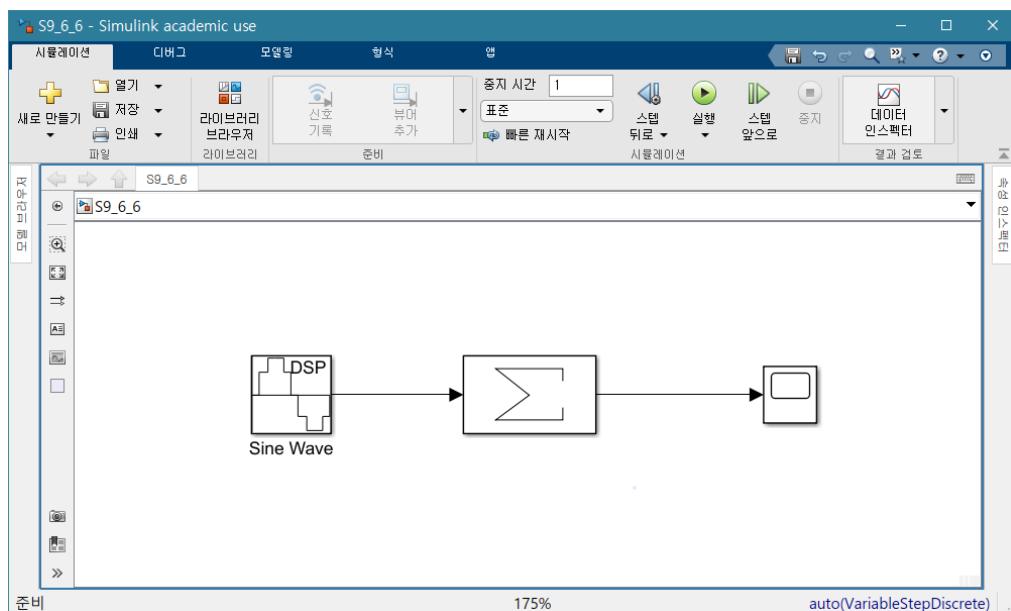


9.6.6 디지털 정현파 신호의 합

$$x(t) = A_1 \sin(2\pi F_1 t) + A_2 \sin(2\pi F_2 t), \quad A_1 = A_2 = 1, F_1 = 100 \text{ Hz}, \quad F_2 = 300 \text{ Hz}, \quad F_s = 5 \text{ kHz}$$

$$x[n] = x_c(nT_s) = A_1 \sin\left(\frac{2\pi F_1}{F_s} n\right) + A_2 \sin\left(\frac{2\pi F_2}{F_s} n\right) = A_1 \sin(\omega_1 n) + A_2 \sin(\omega_2 n)$$

▶ 블록 모델

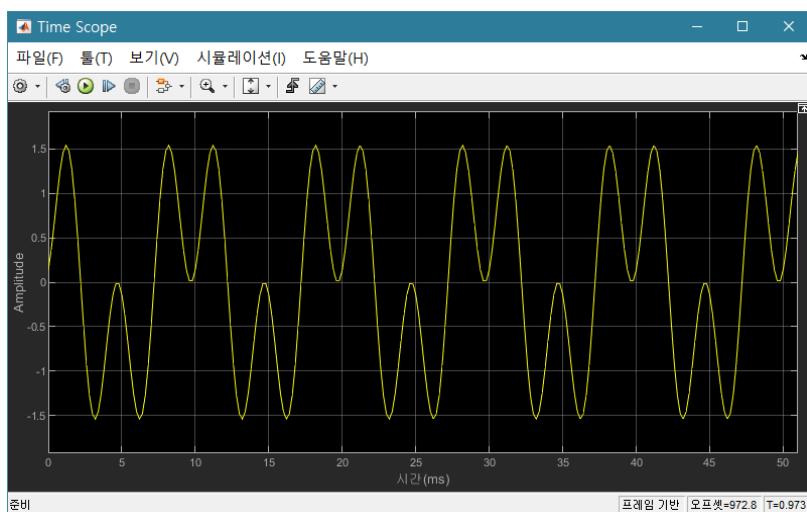


- ✓ DSP System Toolbox – Sources – Sine Wave
- ✓ DSP System Toolbox – Math Functions – Matrices and Linear Algebra – Matrix Operations – Matrix Sum
- ✓ DSP System Toolbox – Sinks – Time Scope

▶ 블록 파라미터

- ✓ Sine Wave

- ✗ Main – Amplitude : 1
- ✗ Main – Frequency : [100 300]
- ✗ Main – sample time : 1/5000
- ✗ Main – Samples per frame : 256
- ✓ Matrix Sum
 - ✗ 합 대상 : 지정된 차원
 - ✗ 차원 : 2
- ✓ Time Scope
 - ✗ 구성 속성 창 – 시간 탭 – 시간 길이 : 1개 프레임 기간
- ➊ 시뮬레이션 구성 파라미터
 - ✓ 중지 시간 : 1
- ➋ 출력 파형



- ➌ Sine Wave 2 개와 Sum 블록을 이용하여 만들어도 같은 결과가 나온다.
- ✓ Matrix Sum 블록을 사용하면 회로가 간단해진다.

9.7 신호의 스펙트럼

➊ 신호의 스펙트럼 보는 블록

✓ DSP System Toolbox – Sinks – Spectrum Analyzer

- ✗ 신호를 직접 연결할 경우 Input domain에 Time을 선택하면 블록 내부에서 스펙트럼을 구하여 보여준다.
- ✗ 신호를 주파수 변환하여 연결할 경우 Input domain에 frequency를 선택한다.

✓ DSP System Toolbox – Sinks – Array Plot

- ✗ 신호를 주파수 변환하여 연결한다.

➋ 주의 사항

- ✓ 모든 블록의 Sample time은 같아야 한다.
- ✓ 모든 블록의 Sample per frame도 같아야 한다.

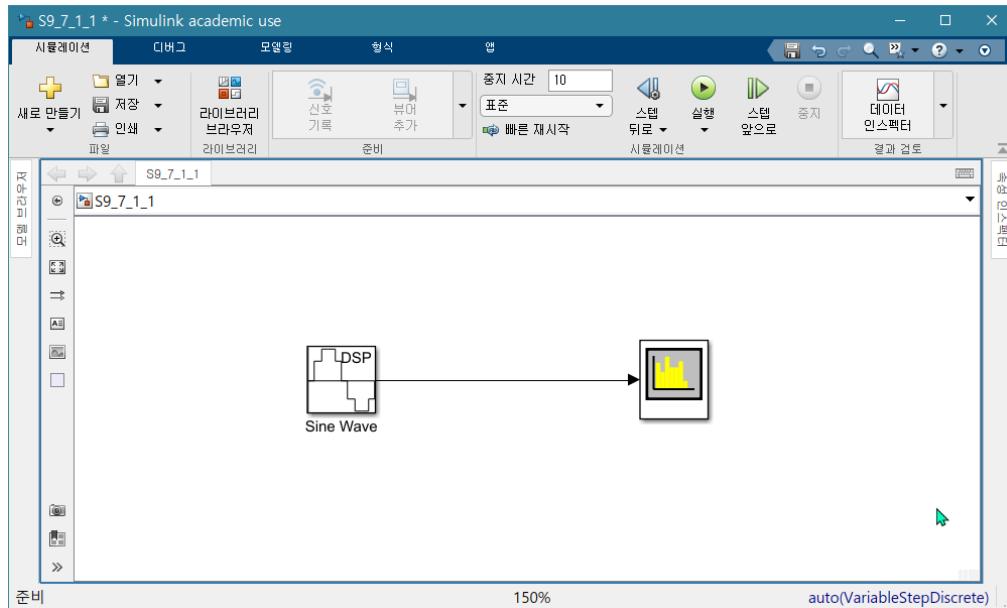
9.7.1 단일 정현파 신호의 스펙트럼

$$x(t) = A_1 \sin(2\pi F_1 t), \quad A_1 = 1, F_1 = 100 \text{ Hz}, F_s = 1 \text{ kHz}$$

$$x[n] = x_c(nT_s) = A_1 \sin\left(\frac{2\pi F_1}{F_s} n\right) = A_1 \sin(\omega_1 n)$$

1) 스펙트럼 분석기를 이용한 스펙트럼 관찰

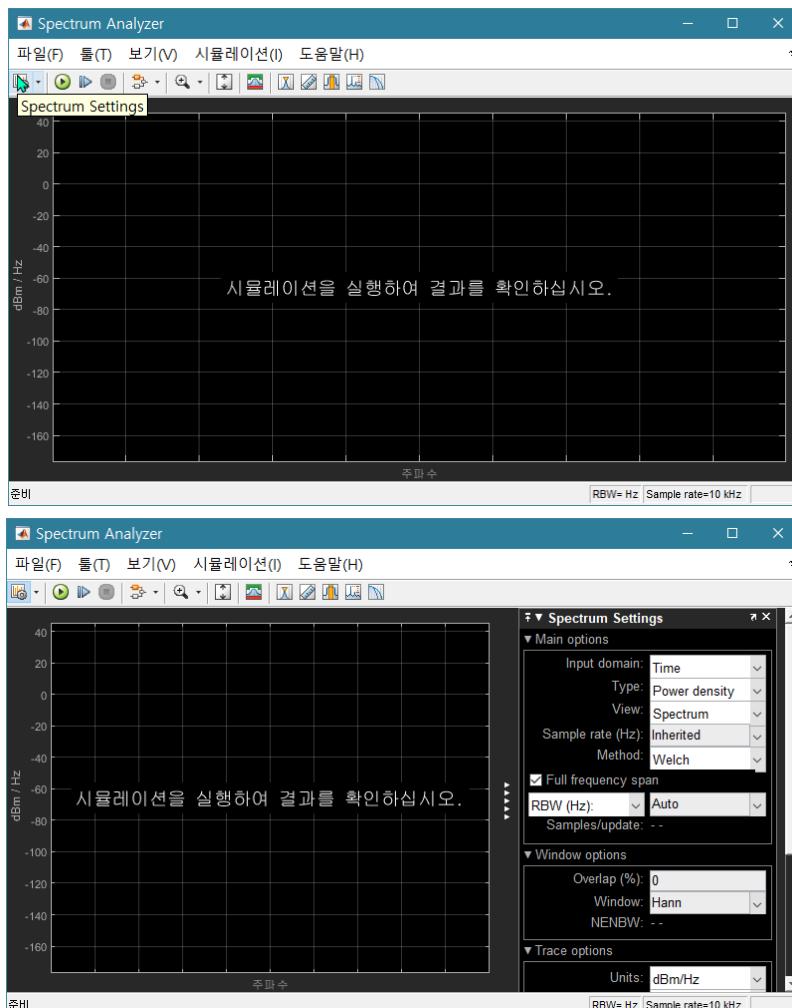
➌ 블록 모델



- ✓ DSP System Toolbox – Sources – Sine Wave
- ✓ DSP System Toolbox – Sinks – Spectrum Analyzer

▶ 블록 파라미터

- ✓ Sine Wave
 - ✗ Main – Amplitude : 1
 - ✗ Main – Frequency : 100
 - ✗ Main – Sample time : 1/1000
 - ✗ Main – Samples per frame : 256
- ✓ Spectrum Analyzer
 - ✗ Spectrum Settings 아이콘을 클릭하여 설정 창을 열고 수정

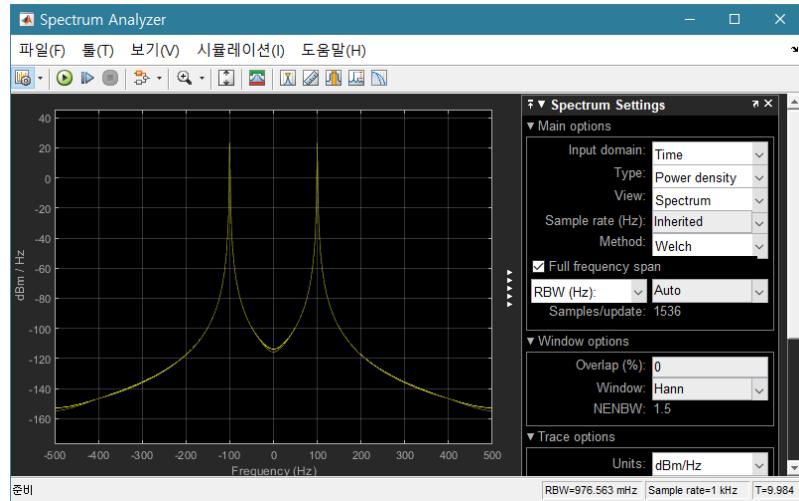


- ✗ Input domain : Time
- ✗ Type : Power density

▶ 시뮬레이션 구성 파라미터

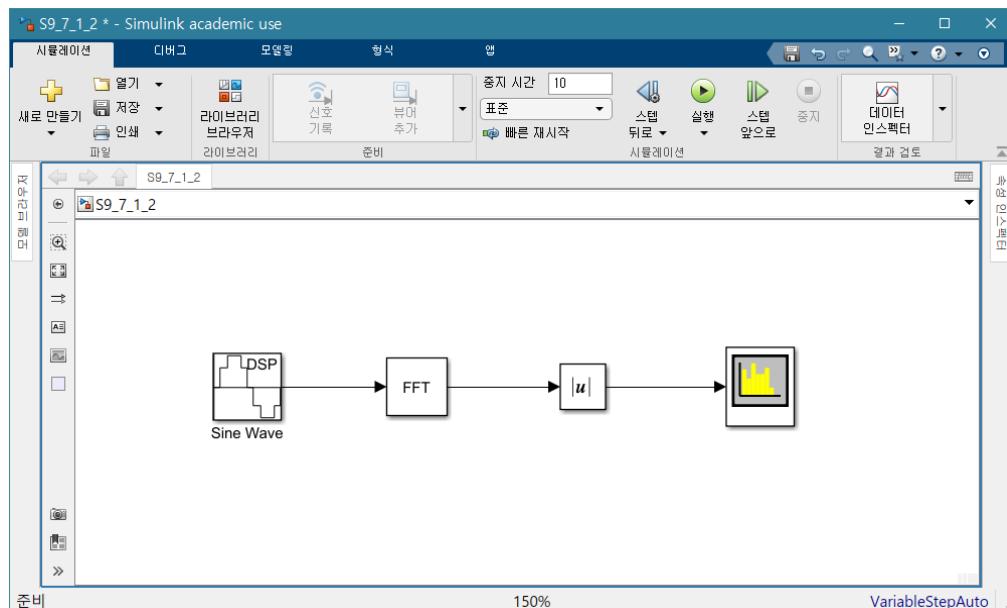
- ✓ 중지 시간 : 10

▶ 출력 파형



2) FFT 를 이용한 스펙트럼 관찰

▶ 블록 모델



- ✓ DSP System Toolbox – Sources – Sine Wave
- ✓ DSP System Toolbox – Transforms – FFT
- ✓ Simulink – Math Operations – Abs
- ✓ DSP System Toolbox – Sinks – Spectrum Analyzer

▶ 블록 파라미터

- ✓ Sine Wave
- ✗ 9.7.1절 1)과 동일

✓ FFT

✗ 수정 사항 없음

✓ Spectrum Analyzer

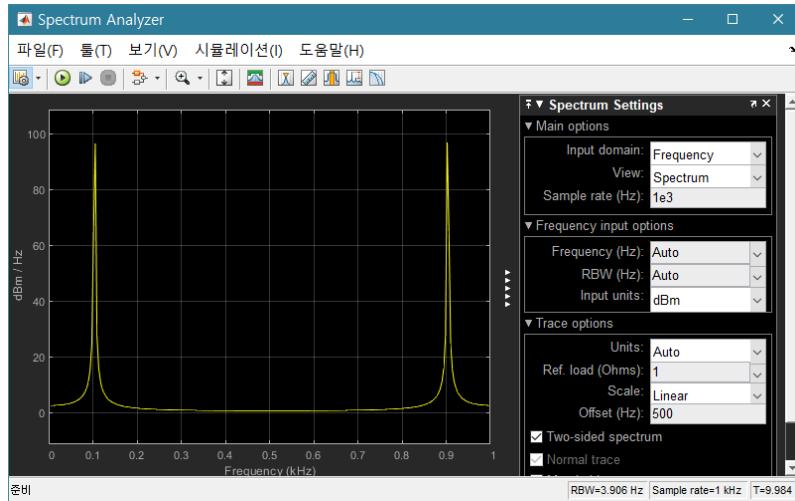
✗ Input domain : Frequency

✗ Offset (Hz) : 500

▶ 시뮬레이션 구성 파라미터

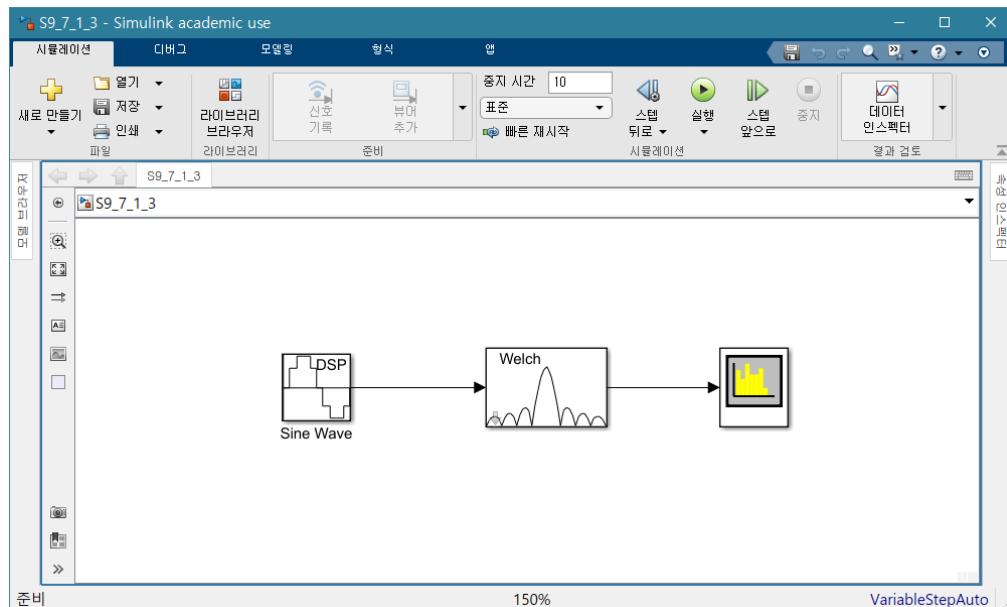
✓ 중지 시간 : 10

▶ 출력 파형



3) 스펙트럼 추정 알고리즘을 이용한 스펙트럼 관찰

▶ 블록 모델



✓ DSP System Toolbox – Sources – Sine Wave

- ✓ DSP System Toolbox – Estimation – Power Spectrum Estimation – Periodogram

- ✓ DSP System Toolbox – Sinks – Spectrum Analyzer

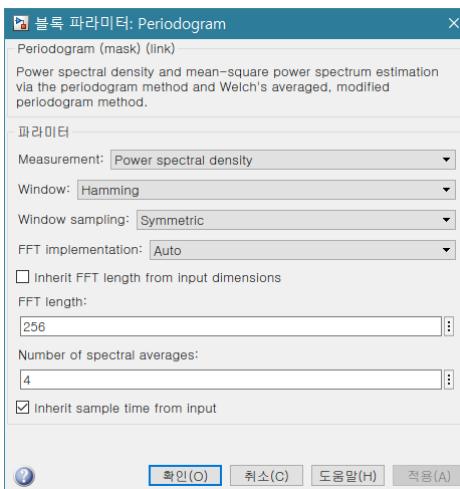
 블록 파라미터

- ✓ Sine Wave

✗ 9.7.1절 1)과 동일

- ✓ Periodogram

✗ FFT length : 256



- ✓ Spectrum Analyzer

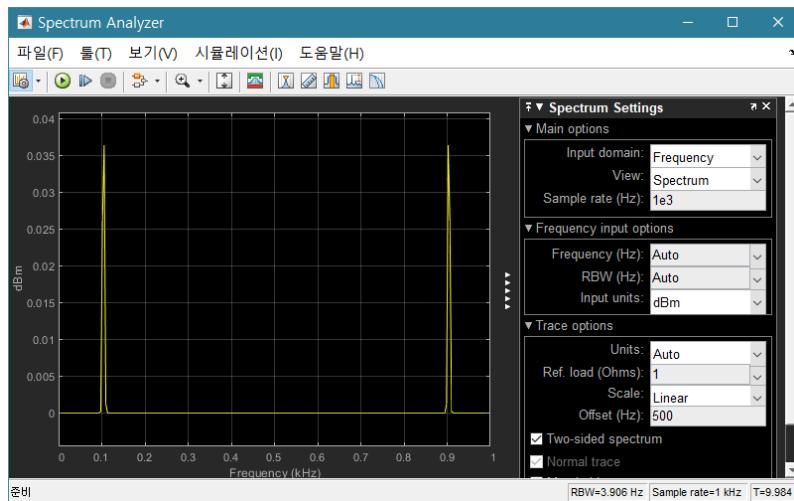
✗ Input domain : Frequency

✗ Offset (Hz) : 500

 시뮬레이션 구성 파라미터

- ✓ 중지 시간 : 10

 출력 파형



9.7.2 다중 정현파 신호의 스펙트럼

$$x_1(t) = A_1 \sin(2\pi F_1 t), \quad A_1 = 1, F_1 = 100 \text{ Hz}$$

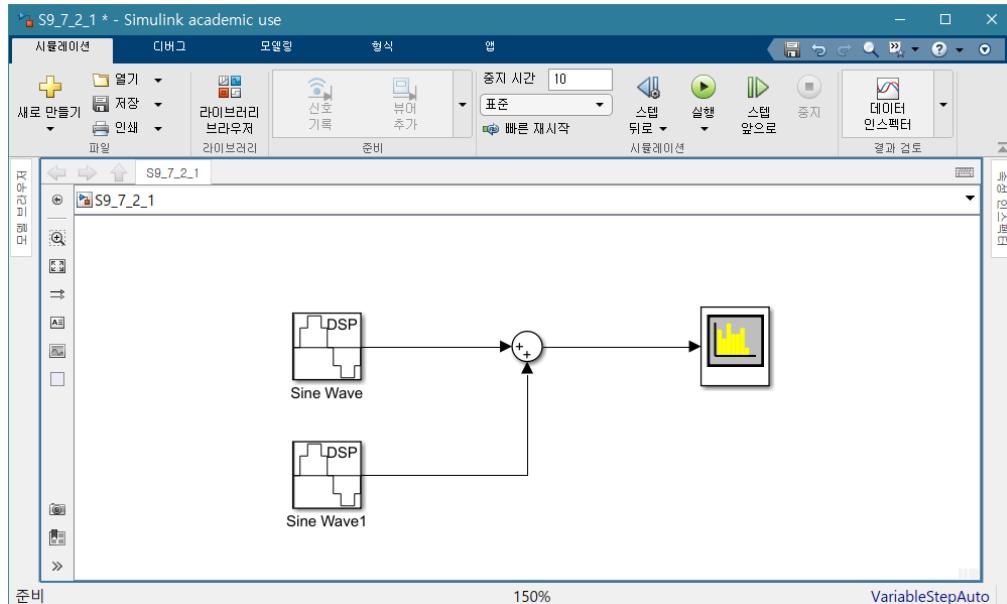
$$x_2(t) = A_2 \sin(2\pi F_2 t), \quad A_2 = 1, F_2 = 200 \text{ Hz}$$

$$x(t) = x_1(t) + x_2(t)$$

$$x[n] = A_1 \sin\left(\frac{2\pi F_1}{F_s} n\right) + A_2 \sin\left(\frac{2\pi F_2}{F_s} n\right)$$

1) 스펙트럼 분석기를 이용한 스펙트럼 관찰

블록 모델



- ✓ DSP System Toolbox – Sources – Sine Wave
- ✓ Simulink – Math Operation – Sum
- ✓ DSP System Toolbox – Sinks – Spectrum Analyzer

▶ 블록 파라미터

✓ Sine Wave

- ✗ Main – Amplitude : 1
- ✗ Main – Frequency : 100
- ✗ Main – Sample time : 1/1000
- ✗ Main – Samples per frame : 256

✓ Sine Wave 1

- ✗ Main – Amplitude : 1
- ✗ Main – Frequency : 200
- ✗ Main – Sample time : 1/1000
- ✗ Main – Samples per frame : 256

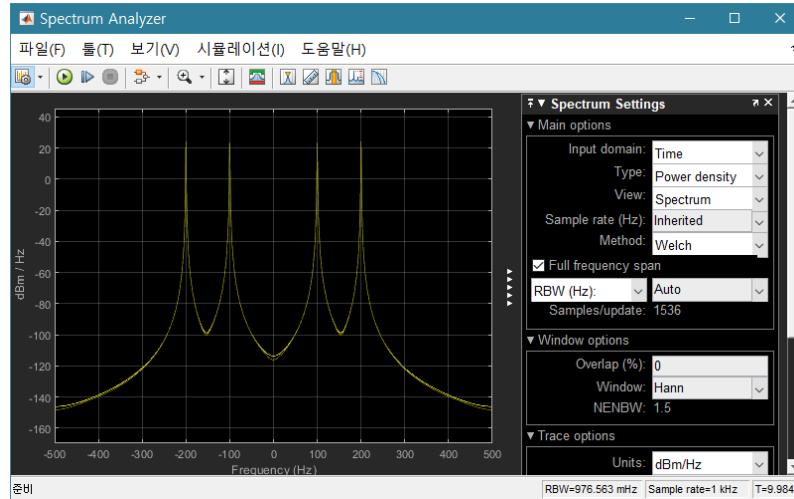
✓ Spectrum Analyzer

- ✗ Input domain : Frequency
- ✗ Offset (Hz) : 500

▶ 시뮬레이션 구성 파라미터

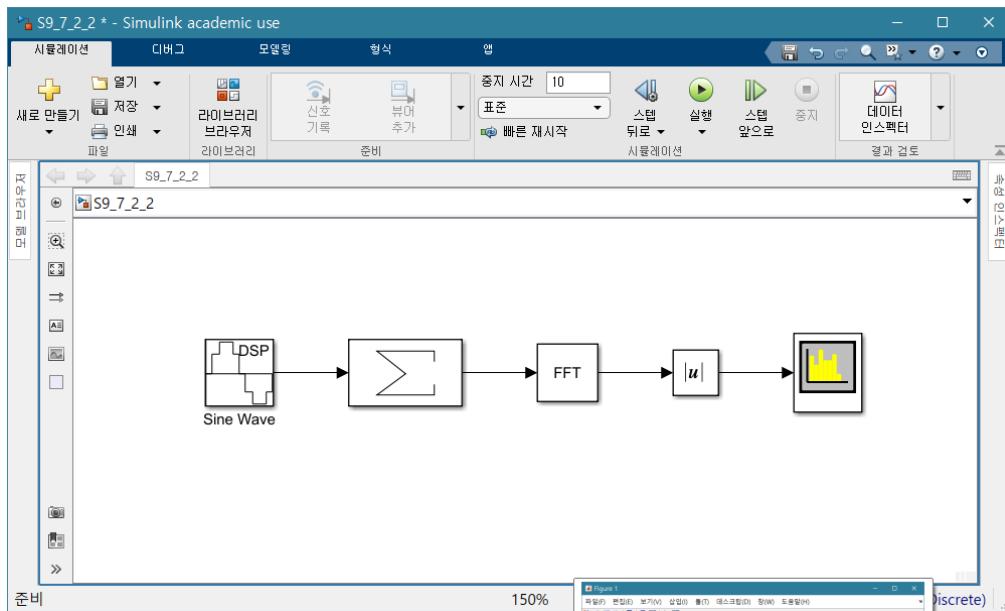
✓ 중지 시간 : 10

▶ 출력 파형



2) FFT를 이용한 스펙트럼 관찰

▶ 블록 모델



- ✓ DSP System Toolbox – Sources – Sine Wave
- ✓ DSP System Toolbox – Math Functions – Matrices and Linear Algebra – Matrix Operations – Matrix Sum
- ✓ DSP System Toolbox – Transforms – FFT
- ✓ Simulink – Math Operations – Abs
- ✓ DSP System Toolbox – Math Functions – math Operations – dB Conversion
- ✓ DSP System Toolbox – Sinks – Array Plot

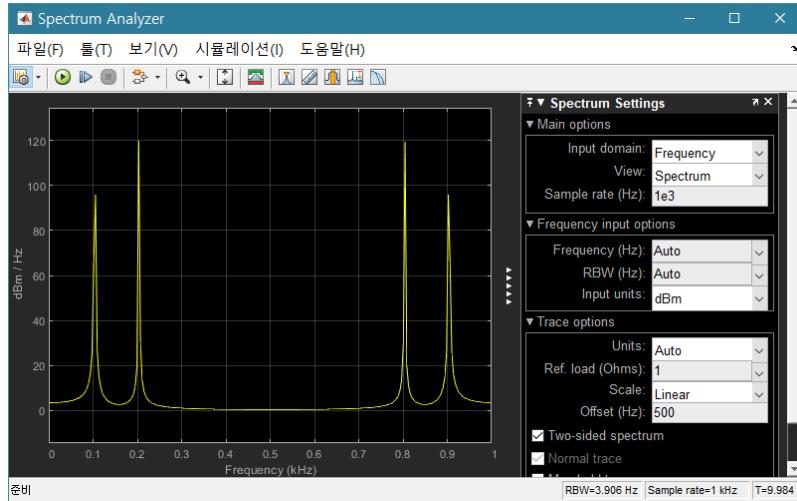
▶ 블록 파라미터

- ✓ Sine Wave
 - ✗ Main – Amplitude : [1 1]
 - ✗ Main – Frequency : [100 200]
 - ✗ Main – Sample time : 1/1000
 - ✗ Main – Samples per frame : 256
- ✓ Matrix Sum
 - ✗ 합 대상 : 지정된 차원
 - ✗ 차원 : 2
- ✓ Spectrum Analyzer
 - ✗ Input domain : Frequency
 - ✗ Offset (Hz) : 500

✚ 시뮬레이션 구성 파라미터

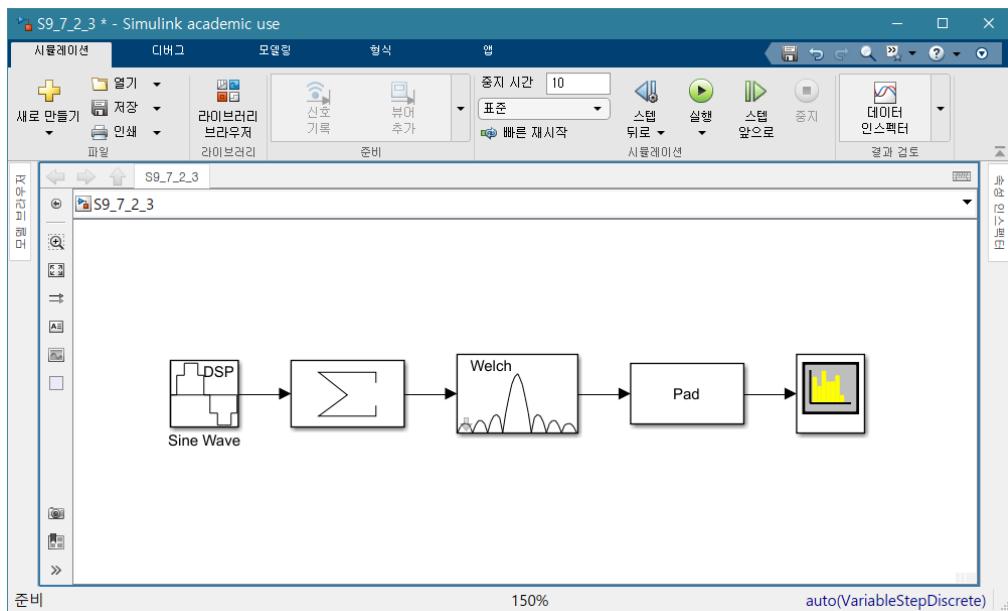
- ✓ 중지 시간 : 10

✚ 출력 파형



3) 스펙트럼 추정 알고리즘을 이용한 스펙트럼 관찰

✚ 블록 모델



- ✓ DSP System Toolbox – Sources – Sine Wave
- ✓ DSP System Toolbox – Math Functions – Matrices and Linear Algebra – Matrix Operations – Matrix Sum
- ✓ DSP System Toolbox – Estimation – Power Spectrum Estimation – Periodogram
- ✓ DSP System Toolbox – Signal Operations – Pad

- ✓ DSP System Toolbox – Sinks – Spectrum Analyzer

▶ 블록 파라미터

- ✓ Sine Wave

✗ 9.7.2절 2)와 동일

- ✓ Matrix Sum

✗ Sum over : Specified dimension

✗ Dimension : 2

- ✓ Periodogram

✗ FFT length : 256

- ✓ Pad

✗ Pad over : Columns

✗ Column size : 128 (frame의 절반)

- ✓ Spectrum Analyzer

✗ Input domain : Frequency

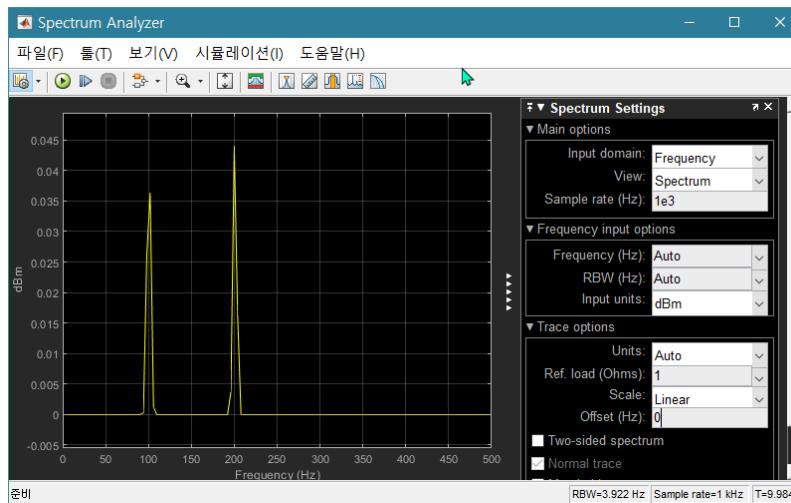
✗ Offset (Hz) : 0

✗ Two-sided spectrum : 체크 해제

▶ 시뮬레이션 구성 파라미터

- ✓ 중지 시간 : 10

▶ 출력 파형



9.7.3 잡음이 있는 다중 정현파 신호의 스펙트럼

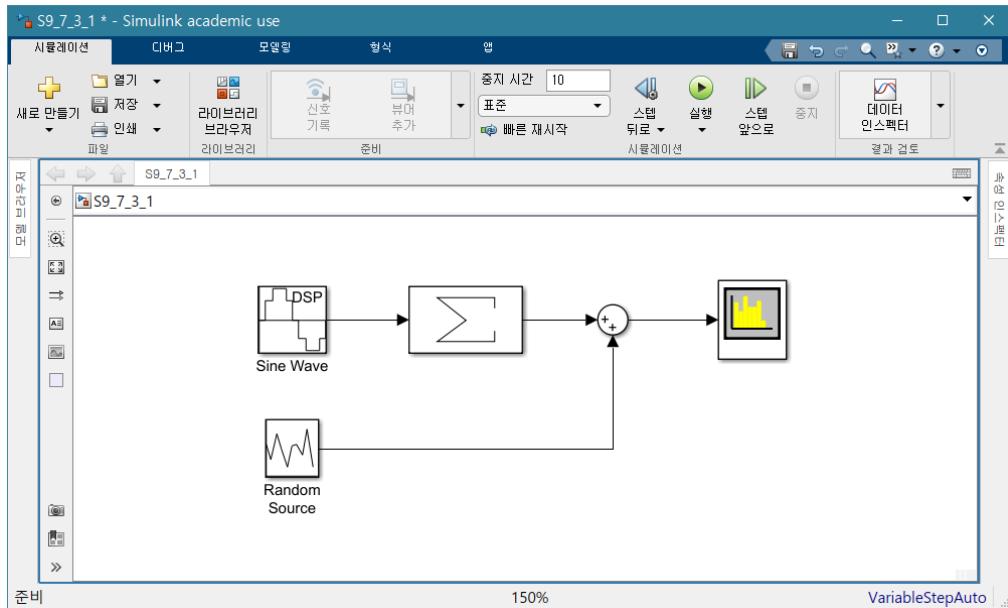
$$x_1(t) = A_1 \sin(2\pi F_1 t), \quad A_1 = 1, F_1 = 100 \text{ Hz}$$

$$x_2(t) = A_2 \sin(2\pi F_2 t), \quad A_2 = 1, F_2 = 200 \text{ Hz}$$

$$x(t) = x_1(t) + x_2(t) + w_c(t), \quad \sigma_w^2 = 0.1$$

$$x[n] = A_1 \sin\left(\frac{2\pi F_1}{F_s} n\right) + A_2 \sin\left(\frac{2\pi F_2}{F_s} n\right) + w[n]$$

1) 스펙트럼 분석기를 이용한 스펙트럼 관찰

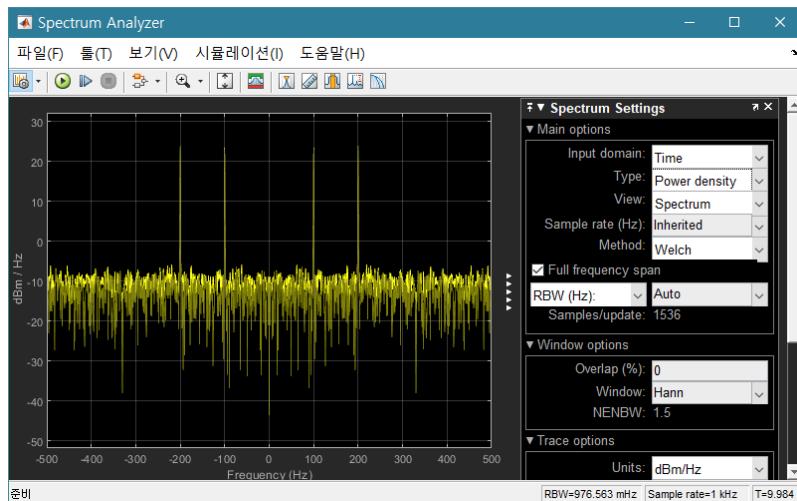


- ✓ DSP System Toolbox – Sources – Sine Wave
- ✓ DSP System Toolbox – Sources – Random Source
- ✓ DSP System Toolbox – Math Functions – Matrices and Linear Algebra – Matrix Operations – Matrix Sum
- ✓ Simulink – Math Operation – Sum
- ✓ DSP System Toolbox – Sinks – Spectrum Analyzer

▶ 블록 파라미터

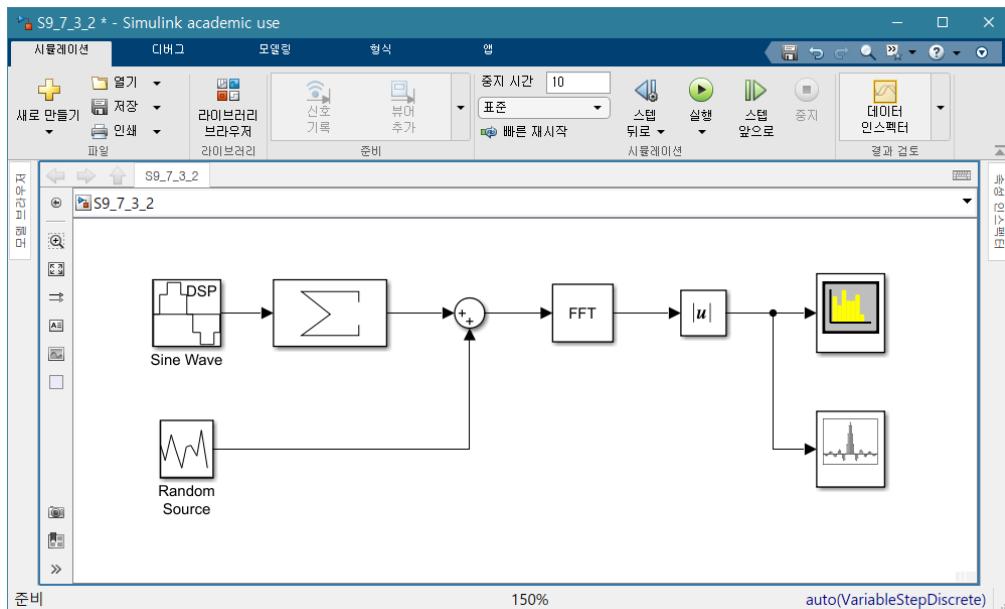
- ✓ Sine Wave
 - ✗ 9.7.2절 2)와 동일
- ✓ Random Source
 - ✗ Source Type : Gaussian
 - ✗ 평균 : 0
 - ✗ 분산 : 0.1

- ✖ 샘플 시간 : 1/1000
- ✖ Samples per frame : 256
- ✓ Matrix Sum
 - ✖ 합 대상 : 지정된 차원
 - ✖ 차원 : 2
- ✓ Spectrum Analyzer
 - ✖ Main options - Input domain : Time
 - ✖ Main options – Type : Power density
 - ✖ Main options – View : Spectrum
 - ✖ Main options – Method : Welch
- ➊ 시뮬레이션 구성 파라미터
 - ✓ 중지 시간 : 2
- ➋ 출력 파형



2) FFT 를 이용한 스펙트럼 관찰

- ➌ 블록 모델



- ✓ DSP System Toolbox – Sources – Sine Wave
- ✓ DSP System Toolbox – Sources – Random Source
- ✓ DSP System Toolbox – Math Functions – Matrices and Linear Algebra – Matrix Operations – Matrix Sum
- ✓ Simulink – Math Operation – Sum
- ✓ DSP System Toolbox – Transforms – Magnitude FFT
- ✓ DSP System Toolbox – Sinks – Spectrum Analyzer
- ✓ DSP System Toolbox – Sinks – Array Plot

▶ 블록 파라미터

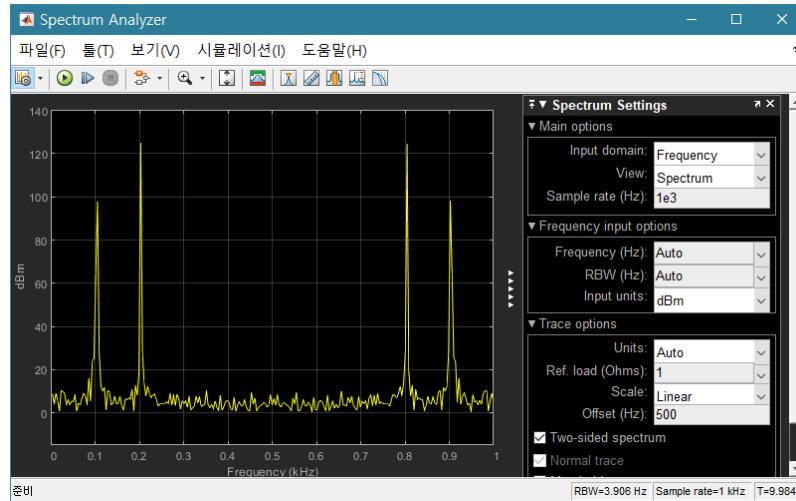
- ✓ Sine Wave
 - ✗ 9.7.3절 1)과 동일
- ✓ Random Source
 - ✗ 9.7.3절 1)과 동일
- ✓ Matrix Sum
 - ✗ 9.7.3절 1)과 동일
- ✓ Spectrum Analyzer
 - ✗ Input domain : Frequency

▶ 시뮬레이션 구성 파라미터

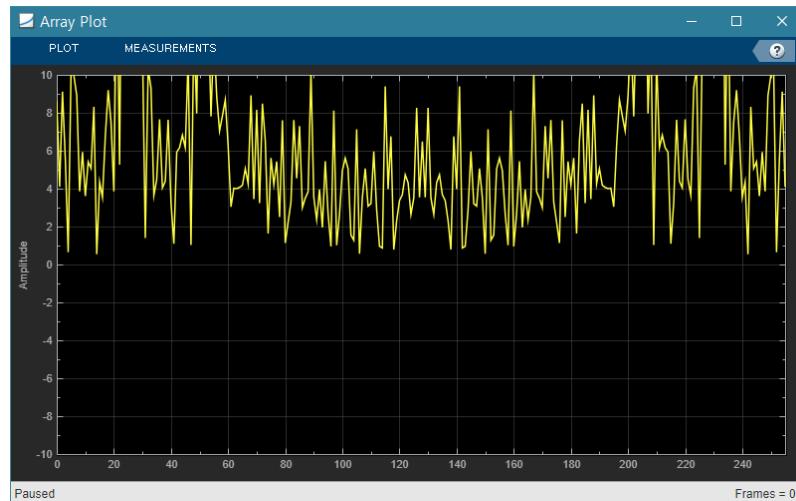
- ✓ 중지 시간 : 10

✖ 출력 파형

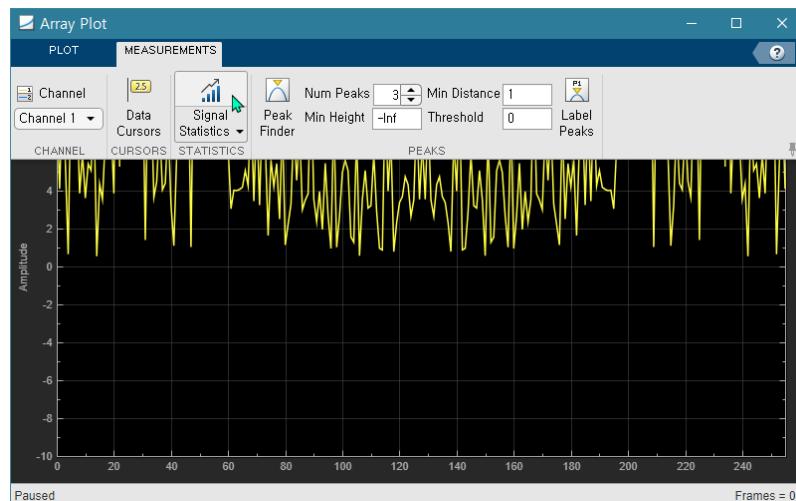
✓ Spectrum Analyzer

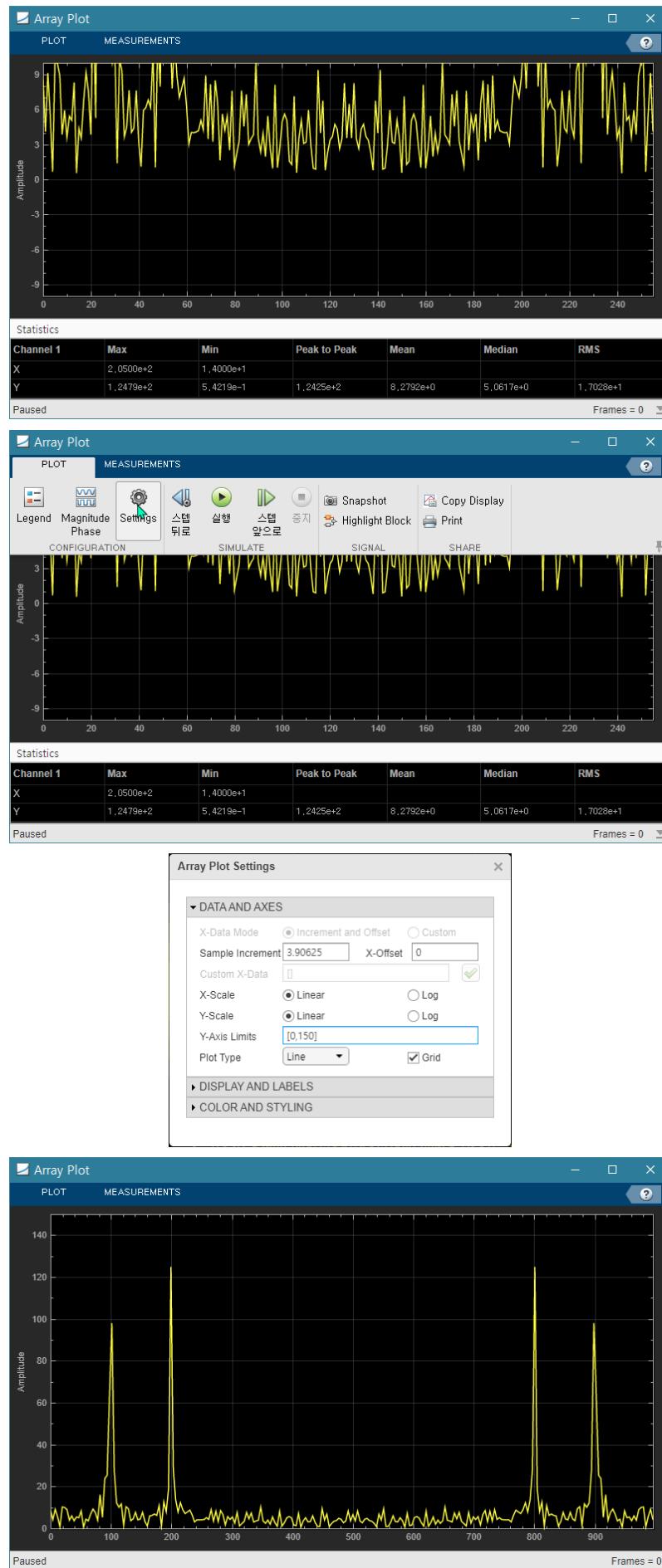


✓ Array Plot



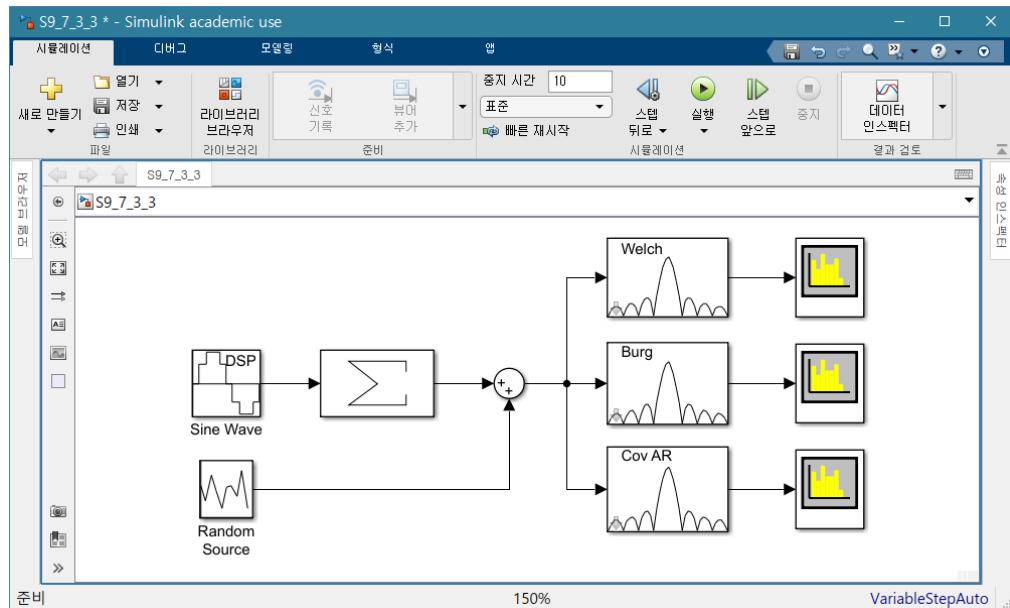
✖ 수정 방법





3) 스펙트럼 추정 알고리즘을 이용한 스펙트럼 관찰

▶ 블록 모델

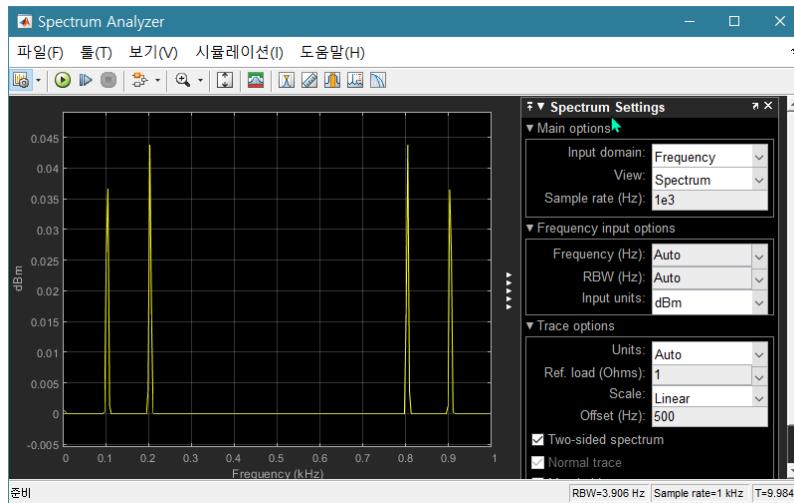


- ✓ DSP System Toolbox – Sources – Sine Wave
- ✓ DSP System Toolbox – Sources – Random Source
- ✓ DSP System Toolbox – Math Functions – Matrices and Linear Algebra – Matrix Operations – Matrix Sum
- ✓ Simulink – Math Operation – Sum
- ✓ DSP System Toolbox – Estimation – Power Spectrum Estimation – Periodogram
- ✓ DSP System Toolbox – Estimation – Power Spectrum Estimation – Burg Method
- ✓ DSP System Toolbox – Estimation – Power Spectrum Estimation – Covariance Method
- ✓ DSP System Toolbox – Sinks – Spectrum Analyzer

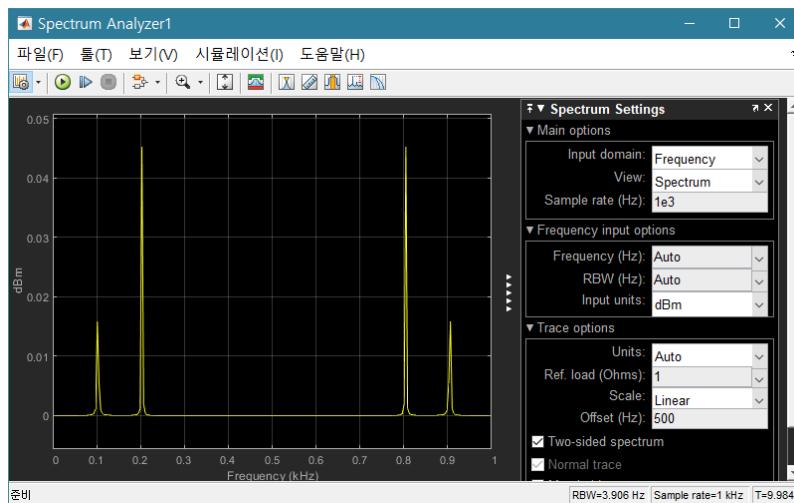
▶ 블록 파라미터

- ✓ Sine Wave
 - ✗ 9.7.3절 1)과 동일
- ✓ Random Source
 - ✗ 9.7.3절 1)과 동일

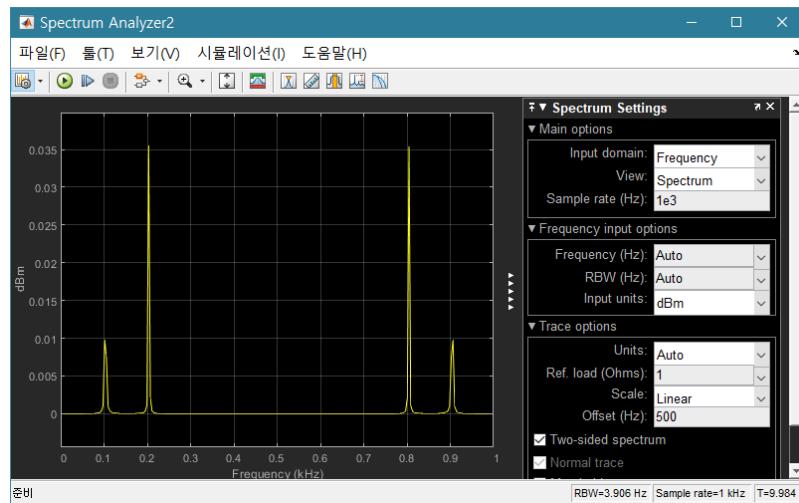
- ✓ Matrix Sum
 - ✗ 9.7.3절 1)과 동일
- ✓ Spectrum Analyzer
 - ✗ Input domain : Frequency
- ✚ 시뮬레이션 구성 파라미터
 - ✓ 증지 시간 : 10
- ✚ 출력 파형
 - ✓ Periodogram



- ✓ Burg Method



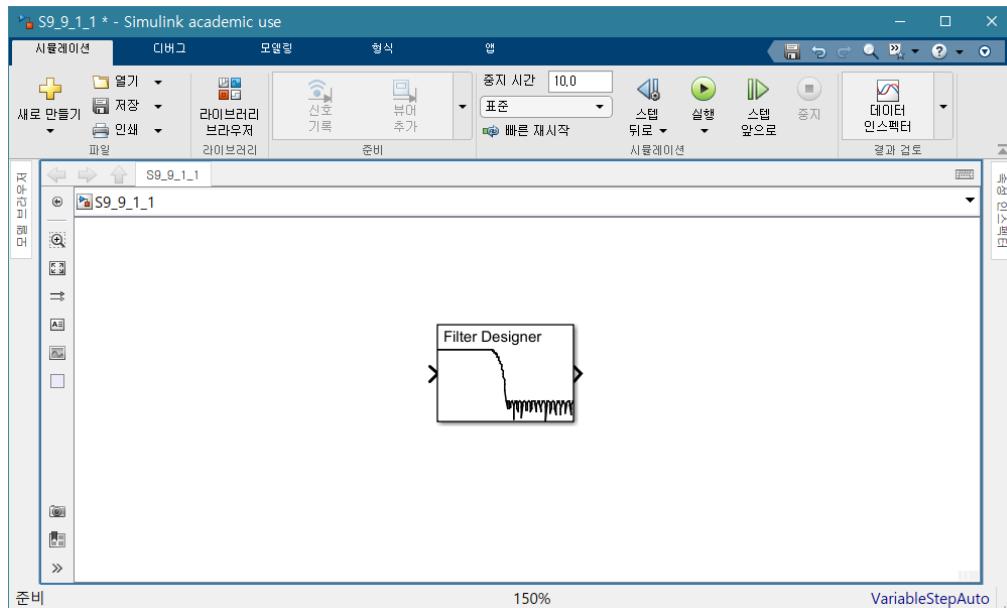
- ✓ Covariance Method



9.8 필터 설계와 신호의 필터링

9.8.1 필터 설계

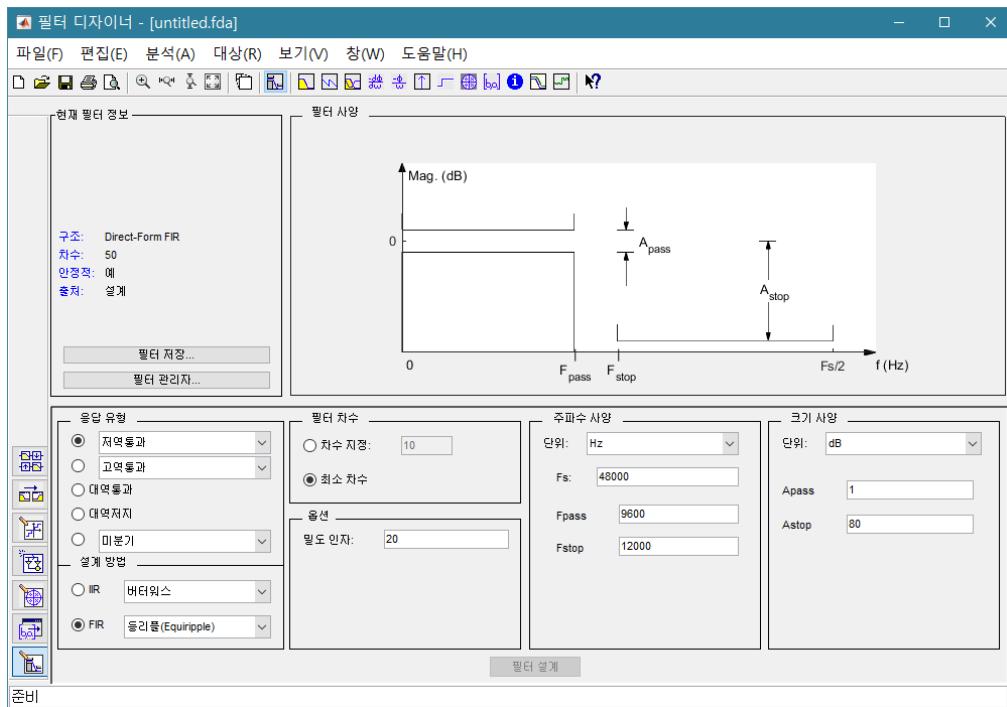
블록 모델



- ✓ DSP System Toolbox – Filtering – Filter Implementation – Digital Filter Design

블록 파라미터

- ✓ Digital Filter Design



- ✓ 7 장의 filterDesigner 실행 결과와 같다.

9.8.2 주어진 시스템에 의한 신호의 필터링

▣ 입력 신호

- ✓ 평균이 0이고 분산이 1인 가우시안 백색 잡음

▣ 이산 시스템

$$\begin{aligned} H(z) &= \frac{0.05634(1+z^{-1})(1-1.0166z^{-1}+z^{-2})}{(1-0.683z^{-1})(1-1.4461z^{-1}+0.7957z^{-2})} \\ &= 0.05634 \frac{1+z^{-1}}{1-0.683z^{-1}} \frac{1-1.0166z^{-1}+z^{-2}}{1-1.4461z^{-1}+0.7957z^{-2}} \end{aligned}$$

- ✓ 전달 함수를 3개의 전달 함수로 분리하여 처리한다.

$$H(z) = GH_1(z)H_2(z)$$

$$G = 0.05634, \quad H_1(z) = \frac{1+z^{-1}}{1-0.683z^{-1}}, \quad H_2(z) = \frac{1-1.0166z^{-1}+z^{-2}}{1-1.4461z^{-1}+0.7957z^{-2}}$$

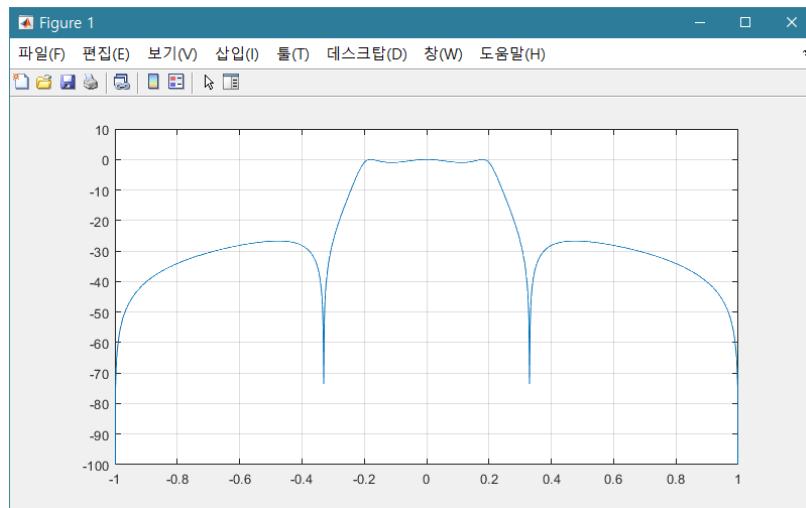
- ✓ 이산 시스템의 주파수 특성

```
% Example of simple LPF
clc; clear;

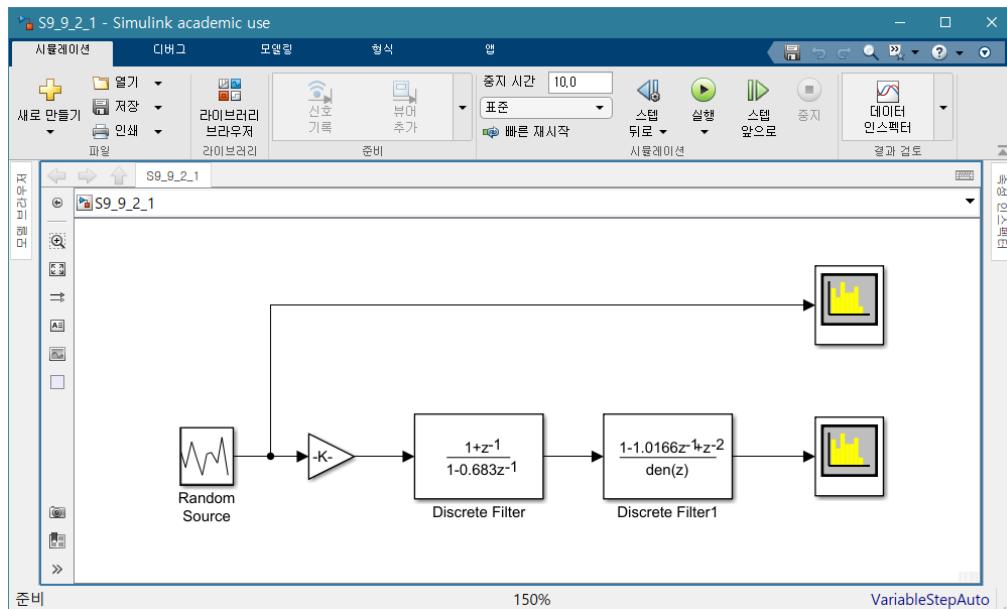
% Transfer function
G = 0.05634;
b1 = [1 1]; a1 = [1 -0.683];
b2 = [1 -1.0166 1]; a2 = [1 -1.4461 0.7957];
b = G*conv(b1,b2)
a = conv(a1,a2)

% Frequency response
w = linspace(-pi,pi,1025);
H = freqz(b,a,w);
Hm = abs(H);
Hm dB = 20*log10(Hm);

figure(1)
plot(w/pi,Hm dB)
grid on
ylim([-100,10])
```



▶ 블록 모델



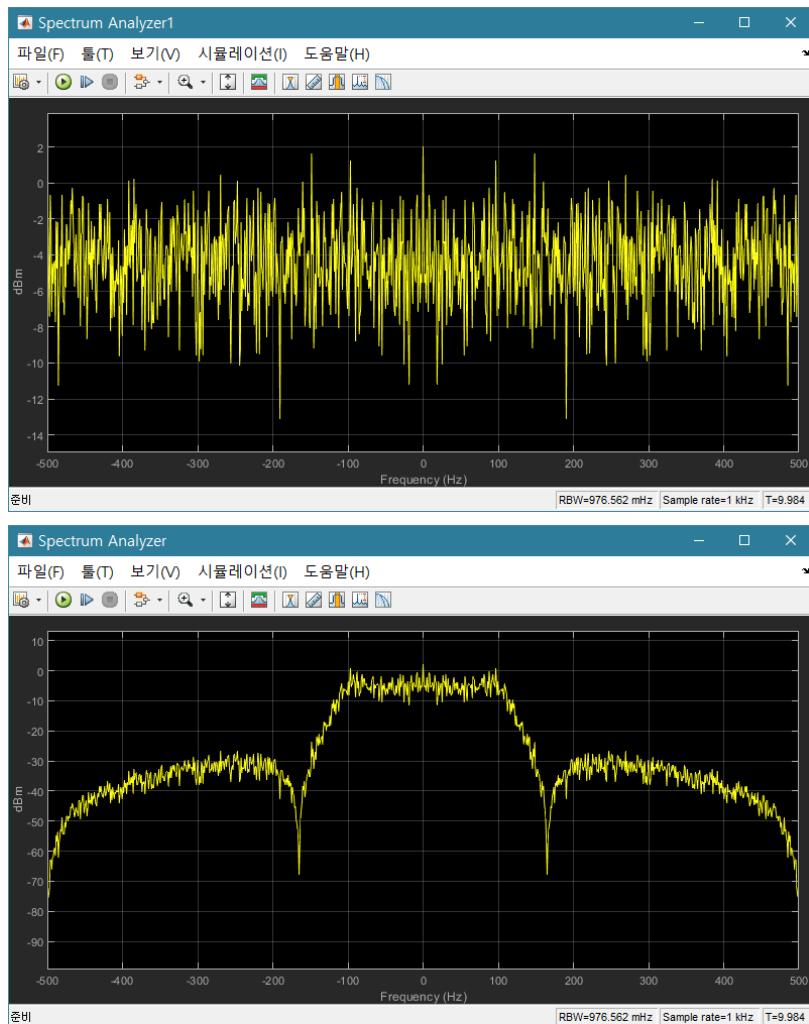
- ✓ DSP System Toolbox – Source – Random Source
- ✓ Simulink – Math Operations – Gain
- ✓ DSP System Toolbox – Filtering – Filter Implementations – Discrete Filter
- ✓ DSP System Toolbox – Sink – Spectrum Analyzer

▶ 블록 파라미터

- ✓ Random Source
 - ✗ Source Type : Gaussian
 - ✗ Mean : 0
 - ✗ variance : 0.1
 - ✗ Sample time : 1/1000

- ✗ Samples per frame : 256
- ✓ Gain
 - ✗ 이득 : 0.05634
- ✓ Discrete Filter
 - ✗ 분자 : [1 1]
 - ✗ 분모 : [1 -0.683]
- ✓ Discrete Filter 1
 - ✗ 분자 : [1 -1.0166 1]
 - ✗ 분모 : [1 -1.4461 0.7957]
- ✓ Spectrum Analyzer

출력 파형



- ✓ 백색 잡음의 스펙트럼이 flat 하므로 시스템을 통과한 유색 잡음의 스펙트럼은 시스템의 주파수 특성과 같게 된다.

9.8.3 사양이 주어진 디지털 필터에 의한 신호의 필터링

▣ 입력 신호

- ✓ 크기가 1이고 주파수가 500 Hz, 2 kHz 인 사인파의 합
- ✓ 샘플링 주파수 : 8 kHz

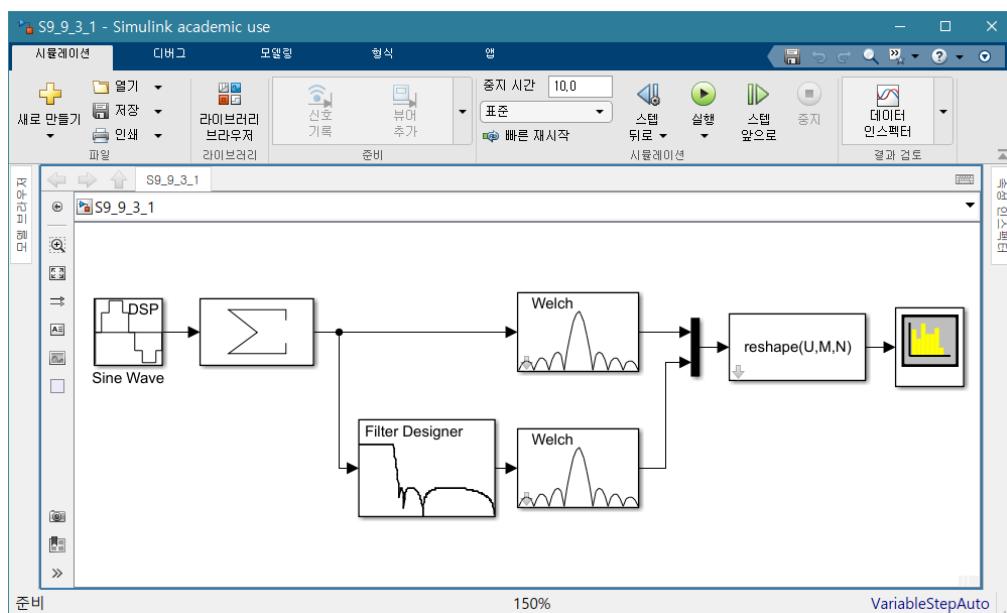
▣ 디지털 필터

- ✓ IIR 타원 필터
- ✓ 통과 대역 주파수 : $F_p = 1 \text{ kHz}$
- ✓ 저지 대역 주파수 : $F_s = 1.2 \text{ kHz}$
- ✓ 통과 대역 감쇄 : $A_p = 1 \text{ dB}$
- ✓ 저지 대역 감쇄 : $A_s = 60 \text{ dB}$
- ✓ 샘플링 주파수 : $F_{\text{samp}} = 8 \text{ kHz}$

▣ 스펙트럼 추정 방법

- ✓ Periodogram

▣ 블록 모델

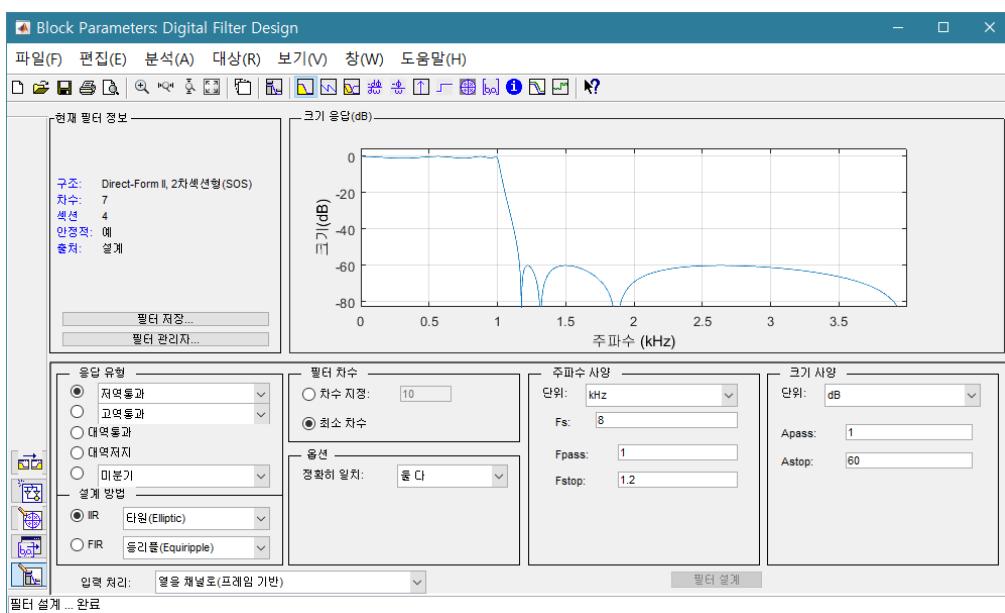


- ✓ DSP System Toolbox – Sources – Sine Wave
- ✓ DSP System Toolbox – Matrices and Linear Algebra – Matrix Operations – Matrix Sum
- ✓ DSP System Toolbox – Estimation – Power Spectrum Estimation – Periodogram

- ✓ DSP System Toolbox – Filtering – Filter Implementations – Digital Filter Design
- ✓ Simulink – Signal Routing – Mux
- ✓ DSP System Toolbox – Signal Management – Signal Attributes – Convert 1-D to 2-D
- ✓ DSP System Toolbox – Sinks – Spectrum Analyzer

블록 파라미터

- ✓ Sine Wave
 - ✗ Main – 진폭 : 1
 - ✗ Main – 주파수 : [500 2000]
 - ✗ Main – 샘플 시간 : 1/8000
 - ✗ Main – Samples per frame : 256
- ✓ Matrix Sum
 - ✗ 합 대상 : 지정된 차원
 - ✗ 차원 : 2
- ✓ Digital filter Design
 - ✗ 응답 유형 : 저역통과
 - ✗ 설계 방법 : IIR – 타원(Elliptic)
 - ✗ 주파수 사양 – 단위 : kHz, Fs : 8, Fpass : 1, Fstop : 1.2
 - ✗ 크기 사양 – 단위 : dB, Apass : 1, Astop : 60

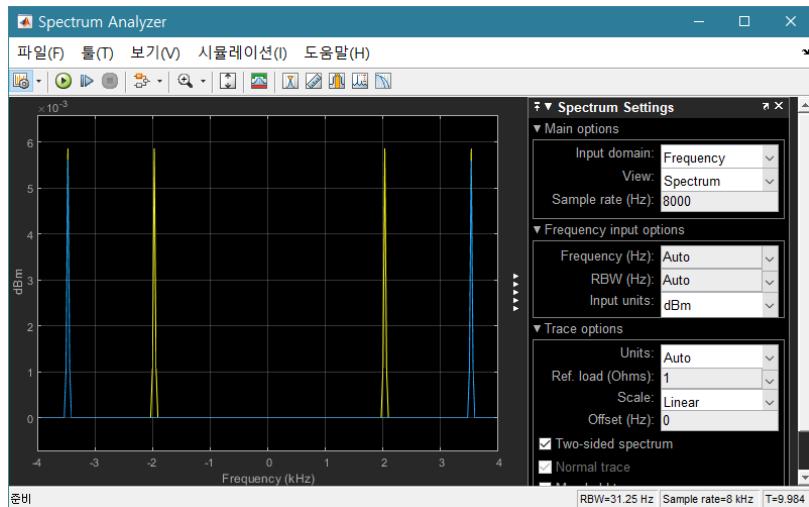


- ✓ Convert 1-D to 2-D
 - ✗ Number of output rows : 256
 - ✗ Number of output columns : 2
- ✓ Spectrum Analyzer
 - ✗ 아래 그림 참고

▶ 시뮬레이션 구성 파라미터

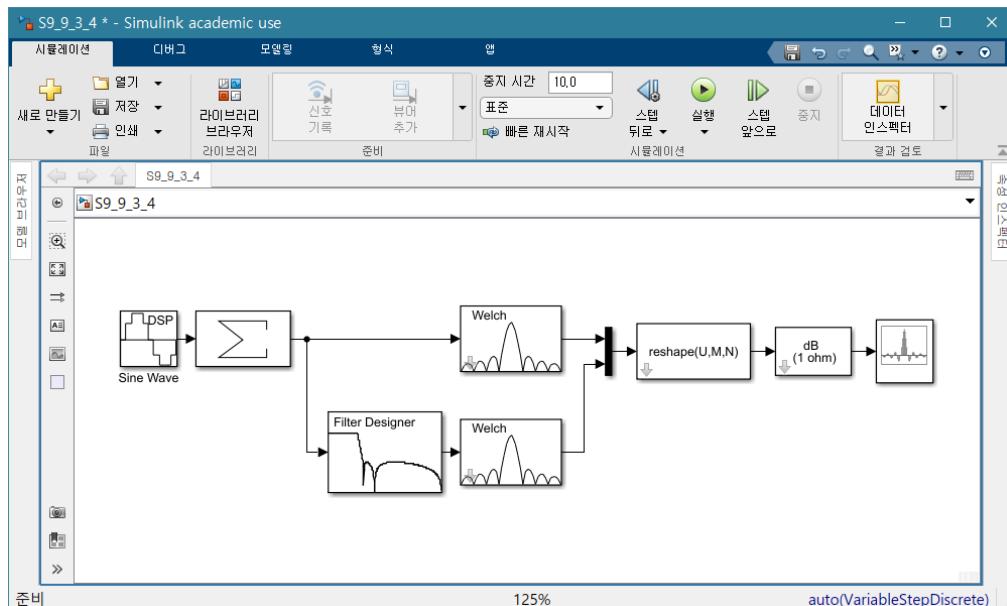
- ✓ Solver – 중지 시간 : 10

▶ 출력 파형



▶ Array Plot 으로 보는 스펙트럼

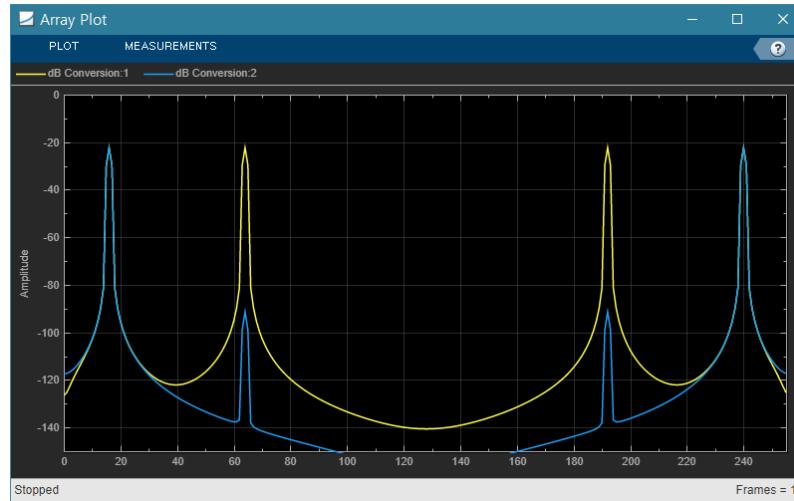
- ✓ 블록 모델



- ✗ Periodogram 모델에서 power spectral density를 구했기 때문에 dB 모델

의 파라미터에서 Input signal은 Power를 선택한다.

✓ 출력 파형



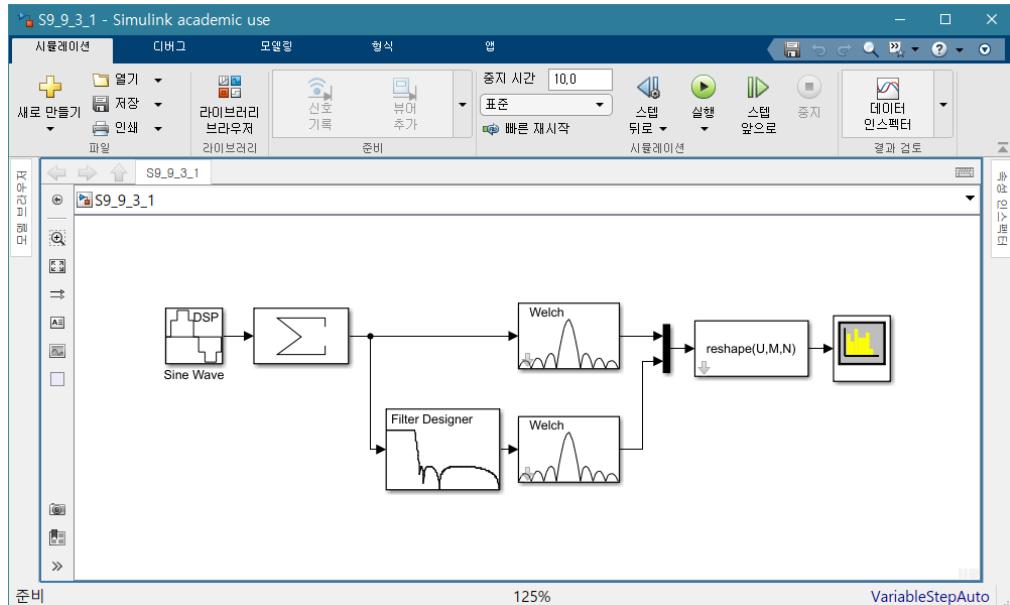
- ✗ 통과 대역에 있는 신호(500 Hz 성분)는 그대로 통과되었지만 저지 대역에 있는 신호(2 kHz 성분)은 60 dB 이상 감소되었다. (필터 통과 전후의 2 kHz 성분의 스펙트럼을 비교해 보면 -20 dB, -90 dB로 70 dB이 감소했다.)
- ✗ Digital filter design 블록의 크기(dB)를 보면 2 kHz에서 감쇄율이 -70 dB로 Simulink 결과와 잘 맞음을 알 수 있다.

9.9 Subsystem

- ✚ 몇 개의 블록을 모아 하나의 subsystem을 만들 수 있다.
- ✓ 한 개의 블록으로 표시된다.

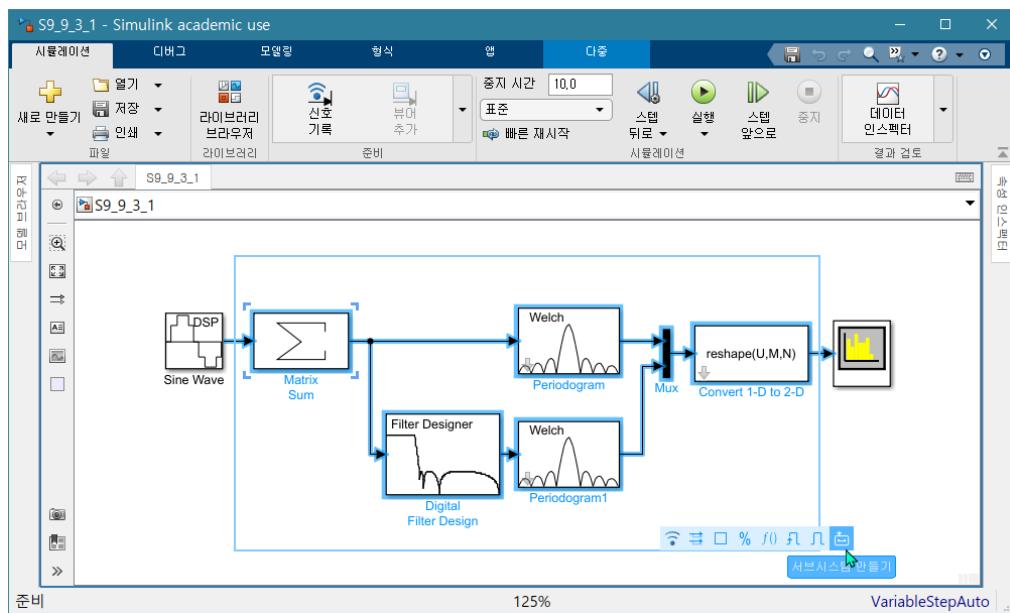
9.9.1 Subsystem 1

9.8.3 절 블록 모델

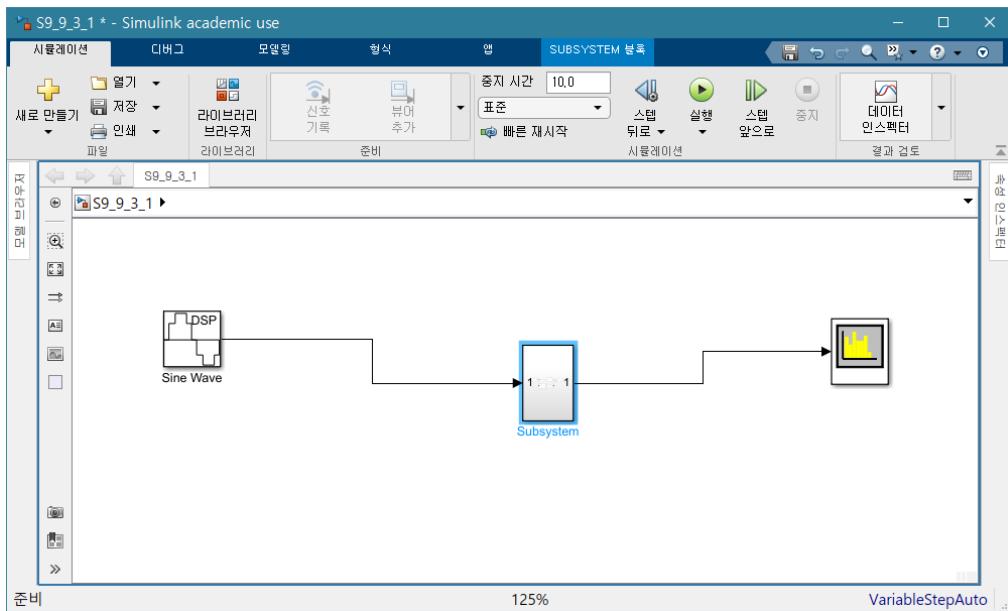


블록 지정

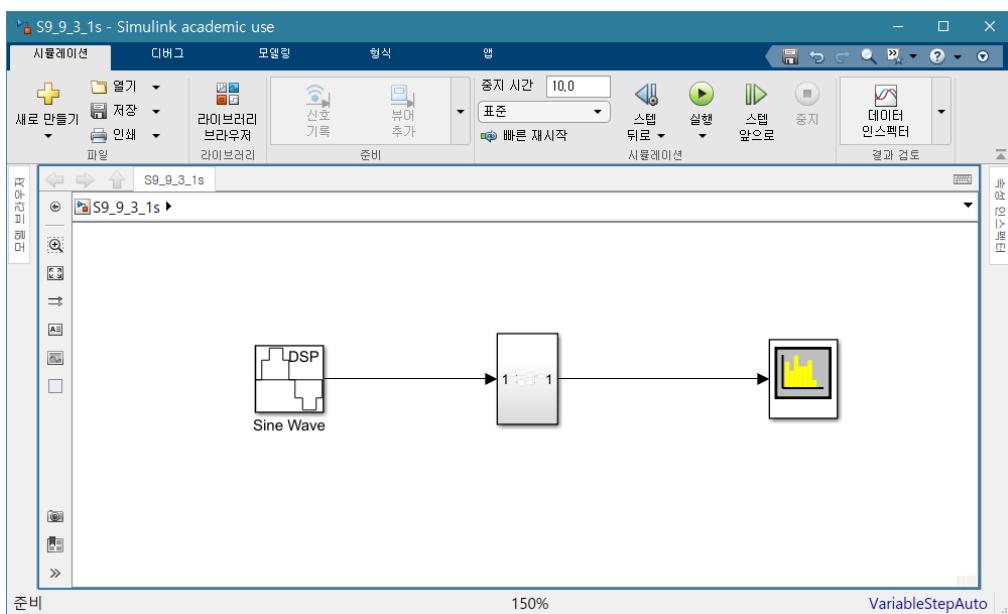
- ✓ 마우스를 드래그하여 영역을 지정한 후 서브시스템 만들기를 클릭한다.
(단축키 : 영역을 지정한 후 Ctrl-g)



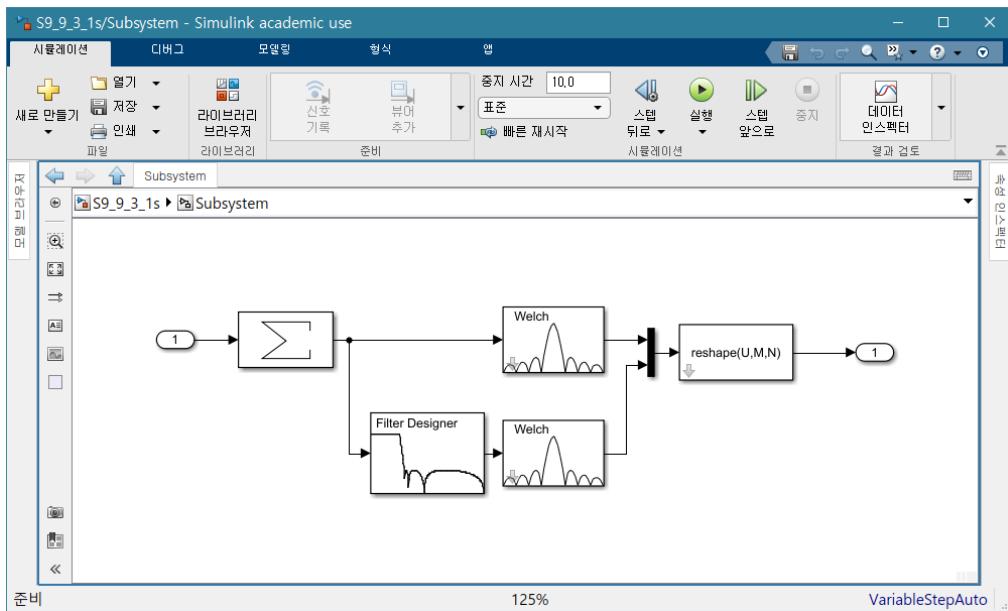
▣ 간략화된 블록 모델



▣ 최종 블록 모델

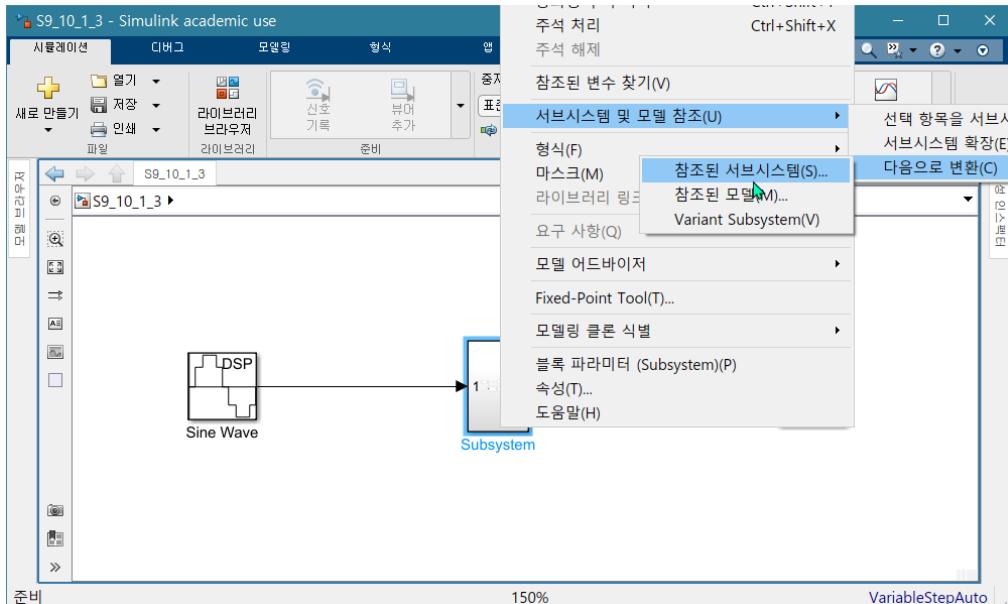


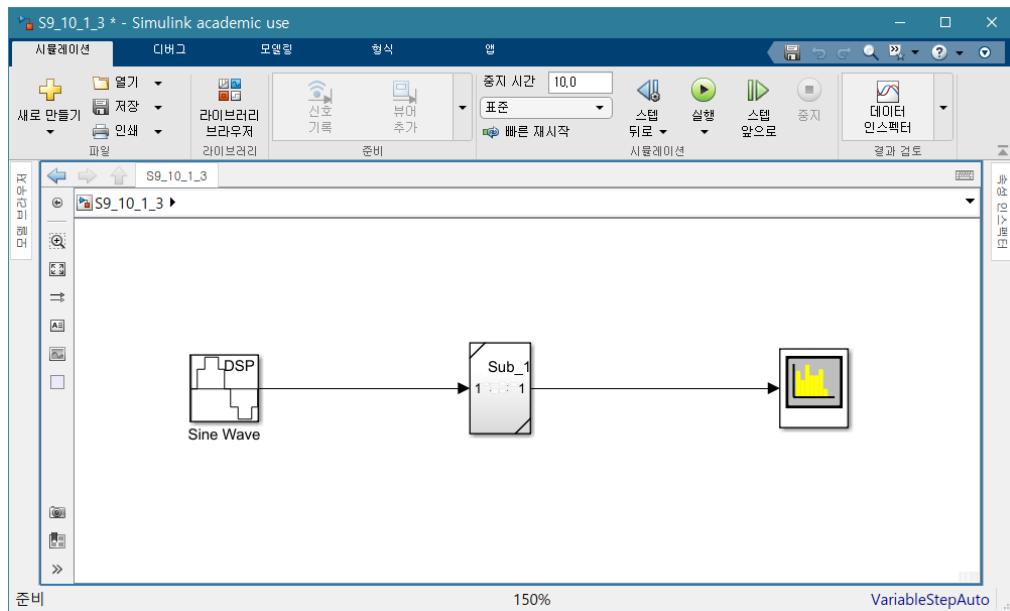
▣ 서브 시스템 클릭



▣ 참조된 서브 시스템 만들기

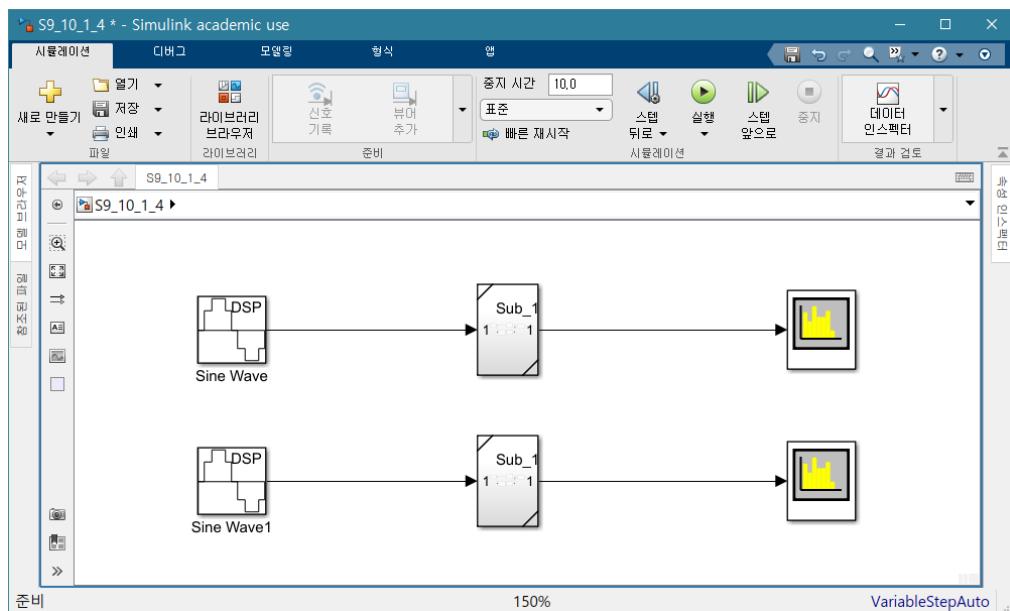
- ✓ 서브 시스템에서 마우스 오른쪽 버튼을 클릭하여 서브 시스템 및 모델 참조 – 다음으로 변환 – 참조된 서브 시스템을 선택한다.





- ✓ 서브 시스템 블록에 삼각형이 생기고 별도의 파일로 저장된다.

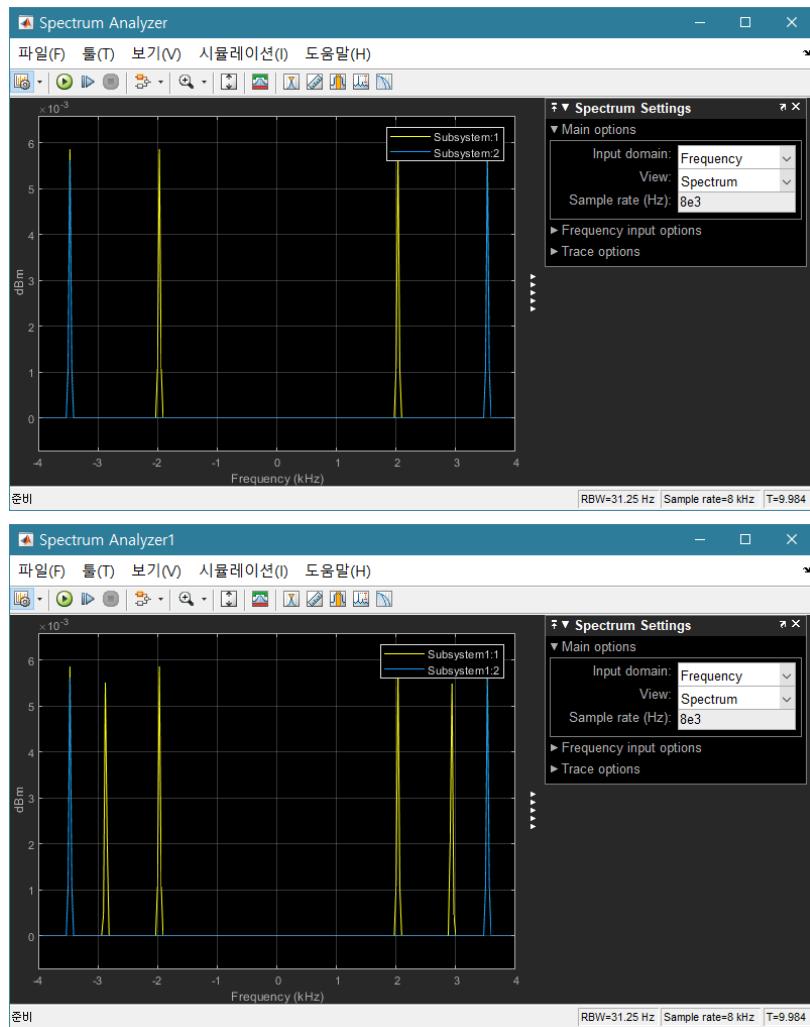
✚ 2 개의 주파수 성분과 3 개의 주파수 성분 신호의 스펙트럼 비교



✚ 블록 파라미터

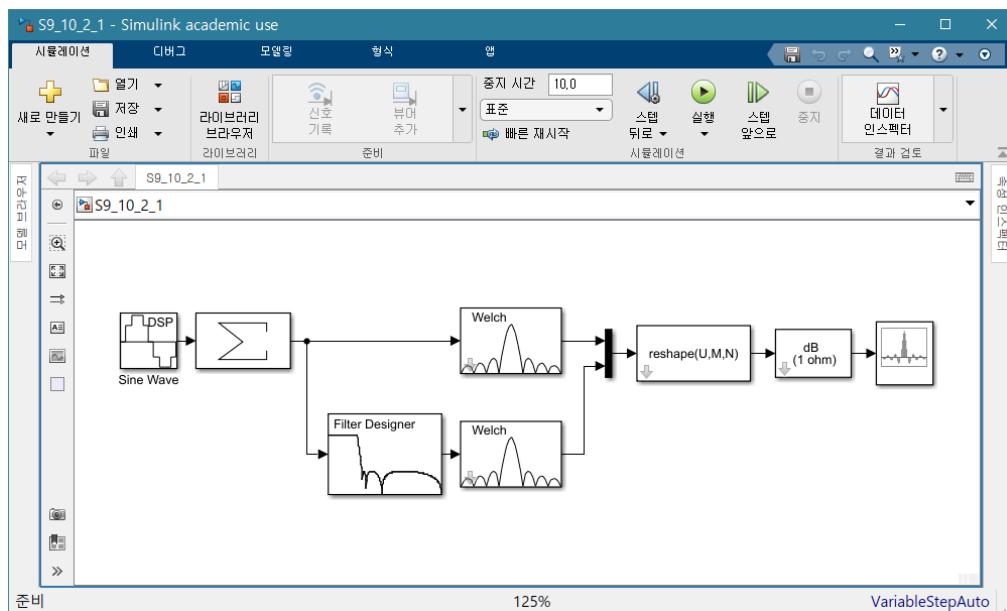
- ✓ Sine Wave, Sine Wave 1
- ✗ Main – 진폭 : 1
- ✗ Main – 주파수 : [500 2000], [5000 1100 2000]
- ✗ Main – 샘플 시간 : 1/8000
- ✗ Main – Samples per frame : 256

✚ 출력 파형

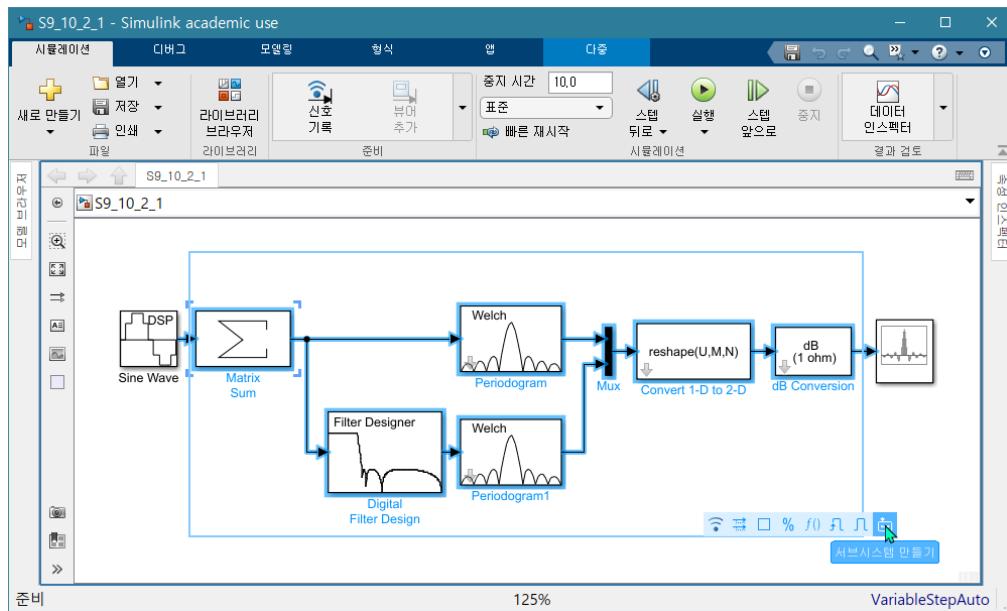


9.10.2 Subsystem 2

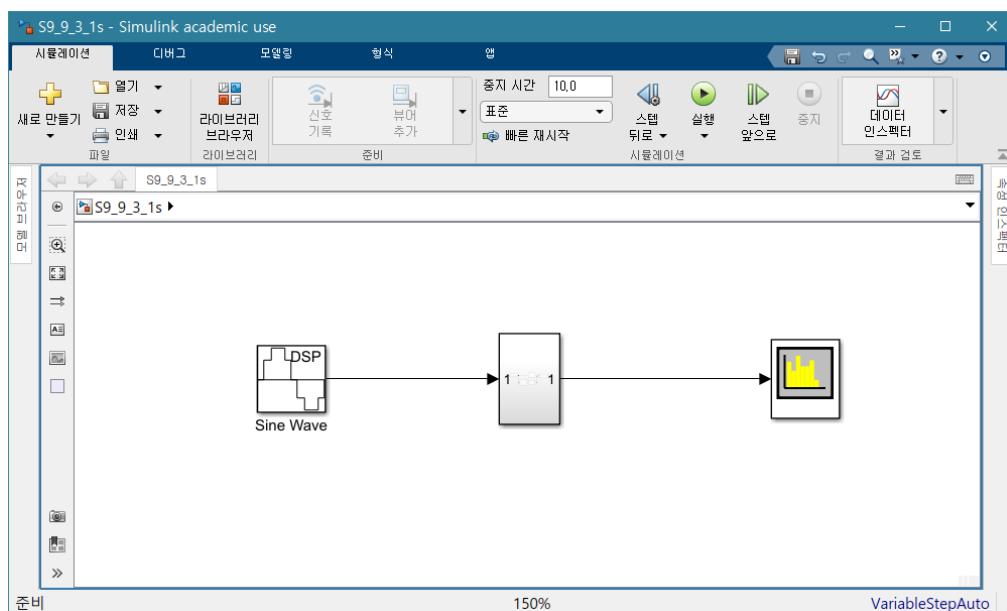
9.9.3 절 블록 모델



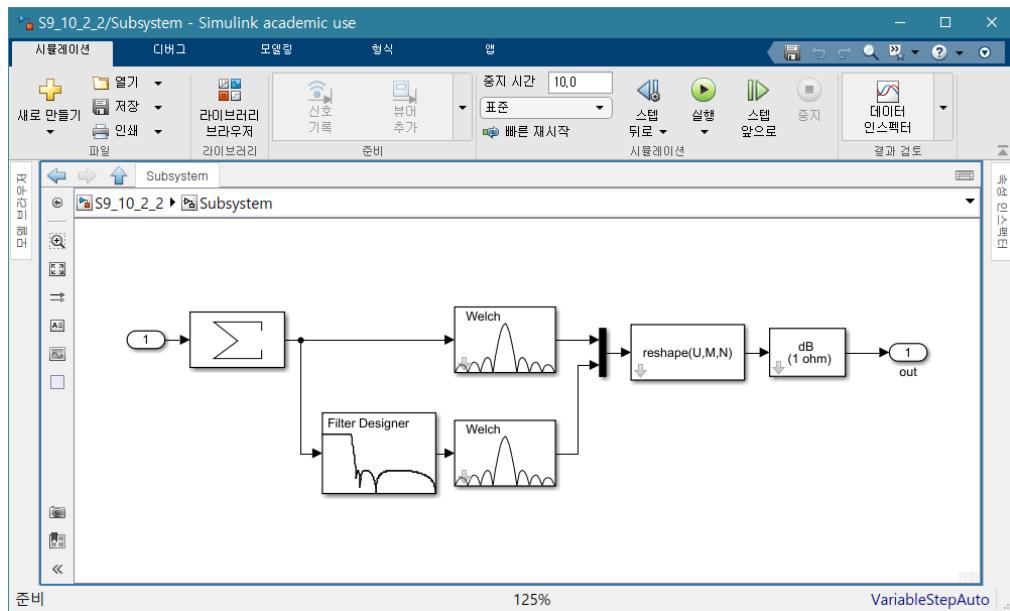
▶ 블록 지정



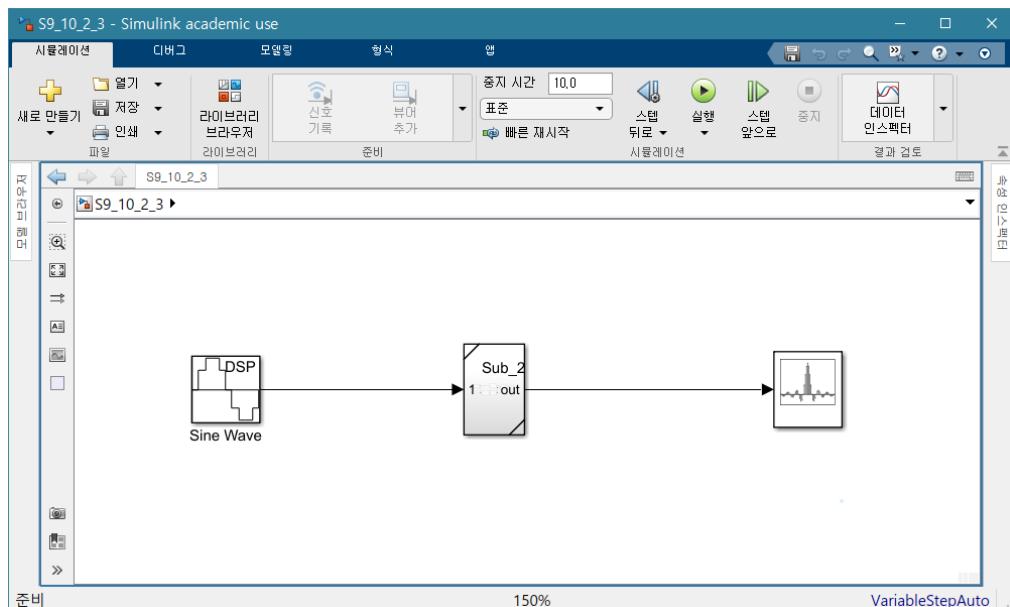
▶ 간략화된 블록 모델



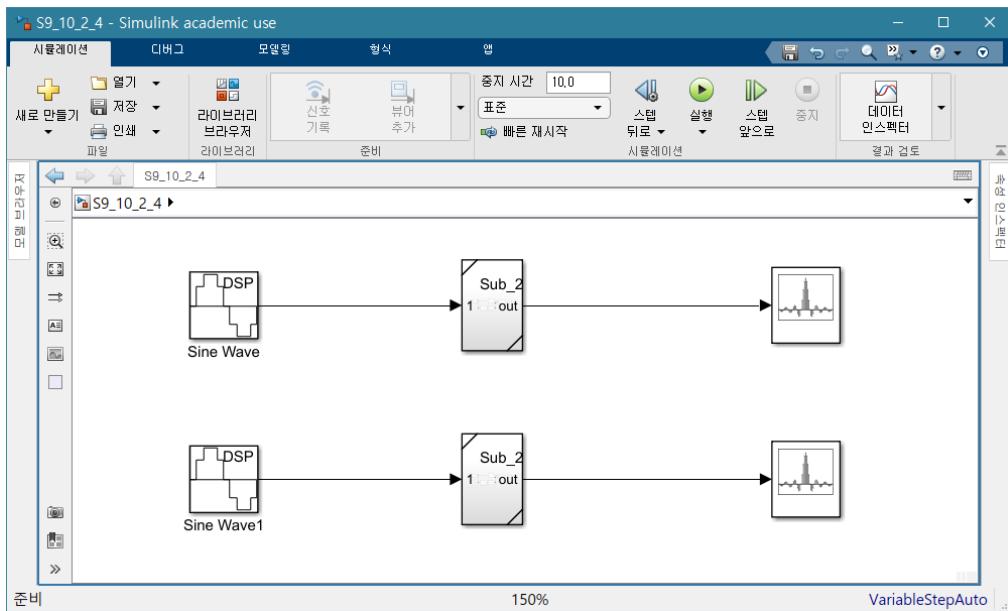
▶ 서브 시스템 클릭



▣ 참조된 서브 시스템 만들기



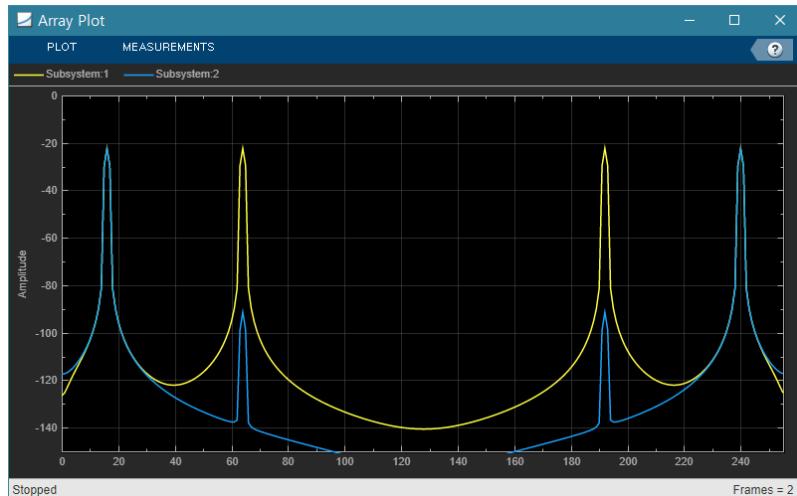
▣ 2 개의 주파수 성분과 3 개의 주파수 성분 신호의 스펙트럼 비교

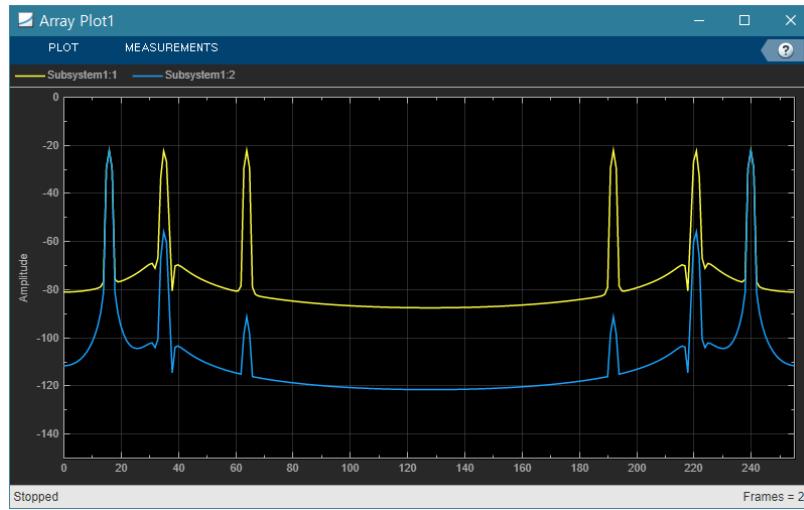


▶ 블록 파라미터

- ✓ Sine Wave, Sine Wave 1
- ✗ Main – 진폭 : 1
- ✗ Main – 주파수 : [500 2000], [5000 1100 2000]
- ✗ Main – 샘플 시간 : 1/8000
- ✗ Main – Samples per frame : 256

▶ 출력 파형





- ✓ 통과 대역의 신호 성분 (500 Hz)은 그대로이다.
- ✓ 저지 대역의 신호 성분 (2 kHz)은 60 dB 이상(70 dB) 감소하였다.
- ✓ 천이 대역의 신호 성분 (1100 Hz)은 40 dB 감소하였다.

9.10 연습 문제

- ▶ 문제 1. 크기가 1이고 주파수가 100 Hz 인 사인파, 평균이 0이고 분산이 0.5인 가우시안 백색 잡음, 두 신호의 합을 3 채널로 스코프로 관찰하라. 단 샘플링 주파수는 1 kHz 이다.
- ▶ 문제 2. 크기가 1이고 주파수가 100 Hz , 200 Hz 인 사인파와 평균이 0이고 분산이 0.5인 가우시안 백색 잡음의 합을 구하는 subsystem 을 만들고 이를 이용하여 이 신호의 스펙트럼을 여러 가지 스펙트럼 추정 방법으로 관찰하라. 단 샘플링 주파수는 1 kHz 이다.
- ▶ 문제 3. 문제 2에서 주어진 신호를 통과 대역 주파수가 130 Hz , 저지 대역 주파수가 170 Hz , 통과 대역 감쇄가 3 dB , 저지 대역 감쇄가 60 dB 인 타원 필터에 통과시켰다. 필터 전후의 신호의 스펙트럼을 관찰하라.

10. 디지털 신호 처리 문제

10.1 신호와 그래프

10.2 연속 시간 시스템의 주파수 응답

10.3 이산 시간 시스템의 주파수 응답

10.4 라플라스 변환

10.5 z 변환

10.6 디지털 시스템의 구조

10.7 아날로그 필터의 설계

10.8 디지털 필터의 설계

10.9 스펙트럼 추정

10.10 Decode of DTMF Signal

10.1 신호와 그래프

▣ 예제 1-1 : 연속 시간 신호의 그래프

- ✓ 다음과 같이 주어지는 연속 시간 신호가 있다. 단 ω 는 임의로 주어라.

$$x(t) = 2 \sin \omega t$$

- ✓ 2 주기 동안 나타내라.

- ✓ Symbolic math 를 이용하여 2 주기 동안 나타내라.

▣ 예제 1-2 : 이산 시간 신호의 그래프

- ✓ 다음과 같이 주어지는 이산 시간 신호가 있다.

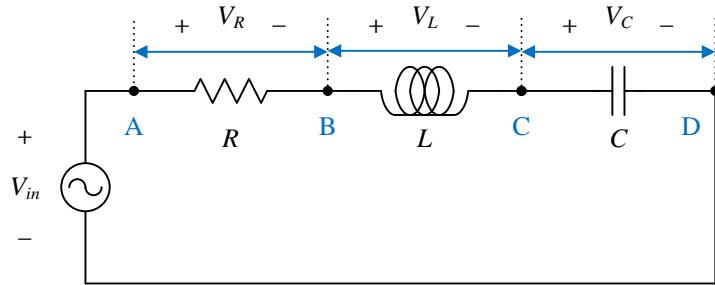
$$x[n] = 3\left(\frac{1}{2}\right)^n u[n]$$

- ✓ 구간 $0 \leq n \leq 10$ 동안 나타내라.

10.2 연속 시간 시스템의 주파수 응답

▣ 예제 2-1 : 연속 시간 시스템의 주파수 응답

- ✓ 다음과 같은 직렬 공진 회로가 있다. $R=100\Omega$, $H=1\text{mH}$, $C=0.1\mu\text{F}$ 이다.



- ✓ 전압 이득 $A_V(s) = V_R(s)/V_{in}(s)$ 과 공진 주파수 ω_0 를 구하라.
- ✓ 전압 이득 $A_V(s)$ 의 크기 응답과 크기 응답(dB)을 $\omega_0/100 \leq \omega \leq 100\omega_0$ 동안 그려라. ω_0 는 공진 주파수이다.
- ✓ 전압 이득 $A_V(s)$ 의 크기 응답 대역폭을 구하라.

10.3 이산 시간 시스템의 주파수 응답

▣ 예제 3-1 : 이산 시간 시스템의 주파수 응답

- ✓ 다음과 같이 차분 방정식이 주어지는 이산 시간 시스템이 있다.

$$y[n] - \frac{3}{4}y[n-1] + \frac{1}{8}y[n-2] = x[n]$$

- ✓ 크기 응답, 위상 응답, 군 지연을 아래 구간 동안 그려라.
- ✓ 구간 $0 \leq \omega \leq \pi$ 동안 그려라.
- ✓ 구간 $-\pi \leq \omega \leq \pi$ 동안 그려라.
- ✓ 극점, 영점을 그려라.
- ✓ 임펄스 응답 $h[n]$ 을 그려라.

▣ 예제 3-2 : 이산 시간 시스템의 주파수 응답

- ✓ 다음과 같이 전달함수가 주어지는 이산 시스템이 있다.

$$H(z) = \frac{(1-1.5z^{-1}-z^{-2})(1+0.9z^{-1})}{(1-0.8z^{-1})(1+0.7jz^{-1})(1-0.7jz^{-1})}$$

- ✓ 앞의 예제를 반복하라.

10.4 라플라스 변환

▶ 예제 4-1 : 라플라스 변환

- ✓ 다음과 같은 신호를 symbolic math 를 이용하여 라플라스 변환하라.

$$x(t) = 2 \exp[-2t]u(t) - 3 \exp[-3t]u(t)$$

▶ 예제 4-2 : 라플라스 역 변환

- ✓ 다음과 같은 함수를 라플라스 역 변환하고자 한다.

$$X(s) = \frac{2s^2 - 5s + 9}{s^3 - 3s + 2}$$

- ✓ 유수(residue) 정리를 이용하여 역 변환하라.
- ✓ Symbolic math 를 이용하여 역 변환하라.

10.5 z 변환

▣ 예제 5-1 : z 변환

- ✓ 다음과 같은 신호를 z 변환하라.

$$x[n] = \left(\frac{1}{2}\right)^n u[n] + 2^n u[n]$$

▣ 예제 5-2 : z 역 변환

- ✓ 다음과 같은 함수를 z 역 변환하고자 한다.

$$X(z) = \frac{-1 + 7z^{-1}}{1 - \frac{3}{2}z^{-1} - z^{-2}}$$

- ✓ 유수(residue) 정리를 이용하여 역 변환하라.
- ✓ Symbolic math 를 이용하여 역 변환하라.

10.6 디지털 시스템의 구조

▣ 예제 6-1 : 종속 결합 구조

- ✓ 다음과 같이 주어지는 시스템이 있다.

$$H(z) = \frac{\left(1 + \left(1 - \frac{j}{2}\right)z^{-1}\right)\left(1 + \left(1 + \frac{j}{2}\right)z^{-1}\right)}{\left(1 + \frac{j}{2}z^{-1}\right)\left(1 - \frac{j}{2}z^{-1}\right)\left(1 + \frac{1}{2}z^{-1}\right)\left(1 - \frac{1}{2}z^{-1}\right)}$$

- ✓ 2 차 종속 결합(cascade)형으로 실현하라.

10.7 아날로그 필터의 설계

▣ 아날로그 LPF 사양

$$f_p = 1 \text{ kHz}, f_s = 1.5 \text{ kHz}, R_p = 1 \text{ dB}, R_s = 80 \text{ dB}$$

▣ 예제 7-1 : 버터워스 아날로그 LPF 의 설계

- ✓ 버터워스 필터로 설계하고 주파수 응답(크기, dB 크기)을 그려라.

▣ 예제 7-2 : 체비세프 1 형 아날로그 LPF 의 설계

- ✓ 체비세프 1 형 필터로 설계하고 주파수 응답(크기, dB 크기)을 그려라.

▣ 예제 7-3 : 체비세프 2 형 아날로그 LPF 의 설계

- ✓ 체비세프 2 형 필터로 설계하고 주파수 응답(크기, dB 크기)을 그려라.

▣ 예제 7-4 : 타원 아날로그 LPF 의 설계

- ✓ 타원 필터로 설계하고 주파수 응답(크기, dB 크기)을 그려라.

10.8 디지털 필터의 설계

▣ 디지털 BPF 사양

$$\omega_{p1} = 0.4\pi, \omega_{p2} = 0.5\pi, \omega_{s1} = 0.3\pi, \omega_{s2} = 0.6\pi$$

$$\delta_p = 0.01, \delta_{s1} = 0.001, \delta_{s2} = 0.0001$$

▣ 예제 8-1 : 버터워스 디지털 BPF 의 설계

- ✓ 버터워스 필터로 설계하고 주파수 응답(크기, dB 크기)을 그려라.

▣ 예제 8-2 : 체비세프 1 형 디지털 BPF 의 설계

- ✓ 체비세프 1 필터로 설계하고 주파수 응답(크기, dB 크기)을 그려라.

▣ 예제 8-3 : 체비세프 2 형 디지털 BPF 의 설계

- ✓ 체비세프 2 필터로 설계하고 주파수 응답(크기, dB 크기)을 그려라.

▣ 예제 8-4 : 타원 디지털 BPF 의 설계

- ✓ 타원 필터로 설계하고 주파수 응답(크기, dB 크기)을 그려라.

▣ 예제 8-5 : 카이저 창을 이용한 디지털 FIR BPF 필터

- ✓ 카이저 창을 이용한 FIR 디지털 필터를 설계하고 주파수 응답(크기, dB 크기)을 그려라.

▣ 예제 8-6 : 등 리플 최적 디지털 FIR BPF 필터

- ✓ Parks-McClellan 방법을 이용한 최적 FIR 필터를 설계하고 주파수 응답(크기, dB 크기)을 그려라.

10.9 스펙트럼 추정

▣ 신호 :

- ✓ 크기가 2이고 주파수가 1kHz인 사인파와 크기가 2이고 주파수가 2kHz인 사인파에 평균이 0이고 분산이 0.1인 가우시안 백색 잡음을 섞은 신호를 10kHz로 샘플링하여 1024 개의 신호를 얻었다.

▣ 예제 9-1 :

- ✓ FFT 를 사용하여 스펙트럼을 구하고 나타내는 프로그램을 작성하라.
- ✓ periodogram 함수를 이용하여 전력 스펙트럼 밀도(PSD)를 구하고 나타내는 프로그램을 작성하라.

▣ 예제 9-2 : Simulink 사용

- ✓ 스펙트럼 분석기를 사용하여 신호의 스펙트럼을 직접 구하라.
- ✓ FFT 블록과 스펙트럼 분석기를 사용하여 신호의 스펙트럼을 나타내라.
- ✓ periodogram 블록과 스펙트럼 분석기를 사용하여 신호의 스펙트럼을 나타내라.
- ✓ Burg method 블록과 스펙트럼 분석기를 사용하여 신호의 스펙트럼을 나타내라.

10.10 Decode of DTMF signal

▣ 예제 10-1 : Decode of DTMF signal

- ✓ DTMF 주파수

	1209 Hz	1336kHz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

- ✓ 파일 이름 : dtmf_sig11.wav, dtmf_sig41.wav
- ✓ 스펙트럼 추정을 이용하여 주어진 DTMF 신호에 해당하는 숫자(문자)를 구하라.