

항공방산SW산학연구프로젝트

Research Project

int**Ro**-Lab.

청주대학교 무인항공기학과
김상현 교수 (www.intro-lab.com)

공지: 중간평가

- ◆ 과제 주제: Skydio F10 드론 기술 조사
- ◆ 제출 형식: 파워포인트 10슬라이드 분량 (PPT, PDF 모두 제출)
- ◆ 작성 내용
 - 자유롭게 작성하되 전문적인 기술 내용 위주로 작성.
- ◆ 제출 기한: 11월 02일 (월) 오후 11시
- ◆ 제출 방법: 에델바이스 제출
- ◆ 발표 일정: 11월 03일 (화) 정규 수업 시간
- ◆ 발표 시간: 10분 이내 (발표 시간 초과 시 감점, 핵심적인 내용 위주로)
- ◆ 평가 기준: 전문성(100%)

마감 이후 제출: 감점

◆ Introducing Skydio F10 Lightrunner + MegaDock | Autonomous Fixed-Wing Drone



◆ 행렬의 연산의 예

행렬 $\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ 와 $\mathbf{B} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$ 에 대해 다음을 구하라.

(a) $\mathbf{AB}$

(b) $\mathbf{BA}$

(c) $(\mathbf{AB})^T$

(d) $(\mathbf{AB})^{-1}$

(e) $\mathbf{B}^T \mathbf{A}^T$

(f) $\mathbf{B}^{-1} \mathbf{A}^{-1}$

◆ 행렬의 연산의 예

풀이

$$(a) \mathbf{AB} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

$$(b) \mathbf{BA} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 23 & 34 \\ 31 & 46 \end{bmatrix}$$

$$(c) (\mathbf{AB})^T = \left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \right)^T = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}^T = \begin{bmatrix} 19 & 43 \\ 22 & 50 \end{bmatrix}$$

◆ 행렬의 연산의 예

$$\begin{aligned}
 \text{(d) } (\mathbf{AB})^{-1} &= \left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \right)^{-1} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}^{-1} \\
 &= \frac{\text{adj} \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}}{\begin{vmatrix} 19 & 22 \\ 43 & 50 \end{vmatrix}} = \frac{\begin{bmatrix} 50 & -43 \\ -22 & 19 \end{bmatrix}^T}{950 - 946} \\
 &= \frac{\begin{bmatrix} 50 & -22 \\ -43 & 19 \end{bmatrix}}{4} = \begin{bmatrix} 12.5 & -5.5 \\ -10.75 & 4.75 \end{bmatrix}
 \end{aligned}$$

$$\text{(e) } \mathbf{B}^T \mathbf{A}^T = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}^T \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 19 & 43 \\ 22 & 50 \end{bmatrix}$$

◆ 행렬의 연산의 예

$$\begin{aligned}
 \text{(f) } \mathbf{B}^{-1}\mathbf{A}^{-1} &= \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} \\
 &= \frac{\text{adj} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}}{\det \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}} \times \frac{\text{adj} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}}{\det \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}} \\
 &= \frac{\begin{bmatrix} 8 & -7 \\ -6 & 5 \end{bmatrix}^T}{5 \times 8 - 7 \times 6} \times \frac{\begin{bmatrix} 4 & -3 \\ -2 & 1 \end{bmatrix}^T}{1 \times 4 - 3 \times 2} \\
 &= \frac{\begin{bmatrix} 8 & -6 \\ -7 & 5 \end{bmatrix}}{-2} \times \frac{\begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix}}{-2} \\
 &= \begin{bmatrix} -4 & 3 \\ 3.5 & -2.5 \end{bmatrix} \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix} = \begin{bmatrix} 12.5 & -5.5 \\ -10.75 & 4.75 \end{bmatrix}
 \end{aligned}$$

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T = \begin{bmatrix} 19 & 43 \\ 22 & 50 \end{bmatrix}$$

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1} \mathbf{A}^{-1} = \begin{bmatrix} 12.5 & -5.5 \\ -10.75 & 4.75 \end{bmatrix}$$

◆ 행렬식

● 2 x 2 행렬식

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \times a_{22} - a_{12} \times a_{21}$$

● 예시

$$\begin{vmatrix} 3 & 2 \\ 4 & 5 \end{vmatrix} = 3 \times 5 - 4 \times 2 = 7$$

◆ 행렬식

● 3 x 3 행렬식

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12} \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13} \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

● 예시

$$\begin{vmatrix} 2 & 3 & 0 \\ 1 & -5 & 4 \\ 6 & 0 & -2 \end{vmatrix} = 2 \begin{vmatrix} -5 & 4 \\ 0 & -2 \end{vmatrix} - 3 \begin{vmatrix} 1 & 4 \\ 6 & -2 \end{vmatrix} + 0 \begin{vmatrix} 1 & -5 \\ 6 & 0 \end{vmatrix}$$
$$= 2 \times 10 - 3 \times (-26) + 0 \times 30$$
$$= 98$$

◆ 소행렬식

● 예시

$$A = \begin{bmatrix} 2 & 4 & 3 \\ 1 & 5 & -2 \\ 0 & 6 & 4 \end{bmatrix}$$

$$\Delta_{11} = \begin{vmatrix} 5 & -2 \\ 6 & 4 \end{vmatrix} = 32, \quad \Delta_{12} = \begin{vmatrix} 1 & -2 \\ 0 & 4 \end{vmatrix} = 4, \quad \Delta_{13} = \begin{vmatrix} 1 & 5 \\ 0 & 6 \end{vmatrix} = 6$$

$$\Delta_{21} = \begin{vmatrix} 4 & 3 \\ 6 & 4 \end{vmatrix} = -2, \quad \Delta_{22} = \begin{vmatrix} 2 & 3 \\ 0 & 4 \end{vmatrix} = 8 \quad \bullet \bullet \bullet$$

◆ 여인수

● 예시

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & 3 \\ 1 & 5 & -2 \\ 0 & 6 & 4 \end{bmatrix}$$

$$C_{11} = \begin{vmatrix} 5 & -2 \\ 6 & 4 \end{vmatrix} = 32, \quad C_{12} = - \begin{vmatrix} 1 & -2 \\ 0 & 4 \end{vmatrix} = -4, \quad C_{13} = \begin{vmatrix} 1 & 5 \\ 0 & 6 \end{vmatrix} = 6$$

$$C_{21} = - \begin{vmatrix} 4 & 3 \\ 6 & 4 \end{vmatrix} = 2, \quad C_{22} = \begin{vmatrix} 2 & 3 \\ 0 & 4 \end{vmatrix} = 8, \quad \bullet \bullet \bullet$$

◆ 행렬식

● 소행렬식 혹은 여인수 이용

$$A = \begin{bmatrix} 2 & 4 & 3 \\ 1 & 5 & -2 \\ 0 & 6 & 4 \end{bmatrix}$$

$$\Delta_{11} = 32, \Delta_{12} = 4, \Delta_{13} = 6$$

$$C_{11} = 32, C_{12} = -4, C_{13} = 6$$

$$|A| = a_{11} \times \Delta_{11} - a_{12} \times \Delta_{12} + a_{13} \times \Delta_{13}$$

$$= a_{11} \times C_{11} + a_{12} \times C_{12} + a_{13} \times C_{13}$$

$$= 2 \times 32 - 4 \times 4 + 3 \times 6 = 66$$

행렬

◆ 역행렬

● 정의

$$A^{-1}A = AA^{-1} = I$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}$$

◆ 수반행렬

★ 수반행렬 ★

$$\text{adj } A = \begin{bmatrix} C_{11} & C_{12} & C_{13} & \cdots & C_{1n} \\ C_{21} & C_{22} & C_{23} & \cdots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ C_{n1} & C_{n2} & C_{n3} & \cdots & C_{nn} \end{bmatrix}^T$$

◆ 수반행렬의 예

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & -4 \\ 0 & -4 & 2 \\ 1 & -1 & 5 \end{bmatrix}$$

$$C_{11} = \begin{vmatrix} -4 & 2 \\ -1 & 5 \end{vmatrix} = -18$$

$$C_{12} = -\begin{vmatrix} 0 & 2 \\ 1 & 5 \end{vmatrix} = 2$$

$$C_{13} = \begin{vmatrix} 0 & -4 \\ 1 & -1 \end{vmatrix} = 4$$

$$C_{21} = -\begin{vmatrix} 3 & -4 \\ -1 & 5 \end{vmatrix} = -11$$

$$C_{22} = \begin{vmatrix} 2 & -4 \\ 1 & 5 \end{vmatrix} = 14$$

$$C_{23} = -\begin{vmatrix} 2 & 3 \\ 1 & -1 \end{vmatrix} = 5$$

$$C_{31} = \begin{vmatrix} 3 & -4 \\ -4 & 2 \end{vmatrix} = -10$$

$$C_{32} = -\begin{vmatrix} 2 & -4 \\ 0 & 2 \end{vmatrix} = -4$$

$$C_{33} = \begin{vmatrix} 2 & 3 \\ 0 & -4 \end{vmatrix} = -8$$

$$\text{adj } \mathbf{A} = \begin{bmatrix} -18 & 2 & 4 \\ -11 & 14 & 5 \\ -10 & -4 & -8 \end{bmatrix}^T = \begin{bmatrix} -18 & -11 & -10 \\ 2 & 14 & -4 \\ 4 & 5 & -8 \end{bmatrix}$$

◆ 수반행렬을 이용한 역행렬 계산

★ 역행렬 ★

$$A^{-1} = \frac{adj A}{|A|}$$

$$|A| = 0$$

← 특이행렬

◆ 역행렬 계산 예시

행렬 $\mathbf{A} = \begin{bmatrix} -1 & 2 & -3 \\ 2 & 1 & 0 \\ 4 & -2 & 5 \end{bmatrix}$ 의 역행렬을 구하라.

$$\mathbf{A}^{-1} = \frac{\text{adj } \mathbf{A}}{|\mathbf{A}|}$$

풀이

$$C_{11} = \begin{vmatrix} 1 & 0 \\ -2 & 5 \end{vmatrix} = 5$$

$$C_{12} = -\begin{vmatrix} 2 & 0 \\ 4 & 5 \end{vmatrix} = -10$$

$$C_{13} = \begin{vmatrix} 2 & 1 \\ 4 & -2 \end{vmatrix} = -8$$

$$C_{21} = -\begin{vmatrix} 2 & -3 \\ -2 & 5 \end{vmatrix} = -4$$

$$C_{22} = \begin{vmatrix} -1 & -3 \\ 4 & 5 \end{vmatrix} = 7$$

$$C_{23} = -\begin{vmatrix} -1 & 2 \\ 4 & -2 \end{vmatrix} = 6$$

$$C_{31} = \begin{vmatrix} 2 & -3 \\ 1 & 0 \end{vmatrix} = 3$$

$$C_{32} = -\begin{vmatrix} -1 & -3 \\ 2 & 0 \end{vmatrix} = -6$$

$$C_{33} = \begin{vmatrix} -1 & 2 \\ 2 & 1 \end{vmatrix} = -5$$

$$\text{adj } \mathbf{A} = \begin{bmatrix} 5 & -10 & -8 \\ -4 & 7 & 6 \\ 3 & -6 & -5 \end{bmatrix}^T = \begin{bmatrix} 5 & -4 & 3 \\ -10 & 7 & -6 \\ -8 & 6 & -5 \end{bmatrix}$$

◆ 역행렬 계산 예시

$$\text{adj } \mathbf{A} = \begin{bmatrix} 5 & -4 & 3 \\ -10 & 7 & -6 \\ -8 & 6 & -5 \end{bmatrix}$$

$$\begin{aligned} |\mathbf{A}| &= a_{11} C_{11} + a_{12} C_{12} + a_{13} C_{13} \\ &= (-1) \times 5 + 2 \times (-10) + (-3) \times (-8) \\ &= -5 - 20 + 24 = -1 \end{aligned}$$

$$\mathbf{A}^{-1} = \frac{\text{adj } \mathbf{A}}{|\mathbf{A}|} = \frac{\begin{bmatrix} 5 & -4 & 3 \\ -10 & 7 & -6 \\ -8 & 6 & -5 \end{bmatrix}}{-1} = \begin{bmatrix} -5 & 4 & -3 \\ 10 & -7 & 6 \\ 8 & -6 & 5 \end{bmatrix}$$

◆ 파이썬 다운로드

<https://www.intro-lab.com/drive/>

[Download: arduino-1.8.19-windows.zip](#)

[Download: arduino-1.8.19-windows.zip](#)

4. Software: Python with Control

파이썬 코딩: 자동제어 실습용

[Download: thonny-5.0.0-windows-portable-x64_control_2026_10_12.zip](#)

5. Software: image2ico

image 파일(jpg, png)을 아이콘 파일(ico)로 변환하는 프로그램(포터블 버전, Portable Version)

[Download: image2ico_v0.1.0.exe](#)

6. 교재

드론코더 300 교재

[Download: 드론코더300 강의교재_2026_04_22.zip](#)

◆ 코딩

```
1 import numpy as np
2
3 a = np.array([1, 2])
4
5 print(a)|
```

Shell x

```
>>> %Run -c $EDITOR_CONTENT
```

```
[1 2]
```

```
1 import numpy as np
2
3 a = np.array([1, 2])
4
5 print(a)
6
7 A = np.array([[1, 2],
8               [3, 4]])
9
10 print(A)
```

Shell x

```
>>> %Run -c $EDITOR_CONTENT
```

```
[1 2]
[[1 2]
 [3 4]]
```

```
1 import numpy as np
2
3 A = np.array([[1, 2],
4               [3, 4]])
5
6 B = np.array([[3, 4],
7               [5, 6]])
8
9 C = A + B
10
11 print(C)
12
```

Shell ×

```
>>> %Run -c $EDITOR_CONTENT
```

```
[[ 4  6]
 [ 8 10]]
```

```
1 import numpy as np
2
3 A = np.array([[1, 2],
4               [3, 4]])
5
6 B = np.array([[3, 4],
7               [5, 6]])
8
9 C = A * B
10
11 print(C)
12
```

Shell ×

```
>>> %Run -c $EDITOR_CONTENT
```

```
[[ 3  8]
 [15 24]]
```

```
1 import numpy as np
2
3 A = np.array([[1, 2],
4               [3, 4]])
5
6 B = np.array([[3, 4],
7               [5, 6]])
8
9 C = A.dot(B)
10
11 print(C)
12
```

Shell x

```
>>> %Run -c $EDITOR_CONTENT
```

```
[[13 16]
 [29 36]]
```

```
1 import numpy as np
2
3 A = np.array([[1, 2],
4               [3, 4]])
5
6 B = np.array([[3, 4],
7               [5, 6]])
8
9 C = A @ B
10
11 print(C)
12
```

Shell x

```
>>> %Run -c $EDITOR_CONTENT
```

```
[[13 16]
 [29 36]]
```

```
1 import numpy as np
2
3 A = np.array([[1, 2],
4               [3, 4]])
5
6 B = np.array([[3, 4],
7               [5, 6]])
8
9 C = A @ B
10 print(C)
11
12 D = B @ A
13 print(D)
14
```

Shell x

```
[[13 16]
 [29 36]]
[[15 22]
 [23 34]]
```

```
1 import numpy as np
2
3 A = np.array([[1, 2],
4               [3, 4]])
5
6 E = A.transpose()
7
8 print(E)
9
```

hell x

```
·>> %Run -c $EDITOR_CONTENT
```

```
[[1 3]
 [2 4]]
```

```
1 import numpy as np
2
3 A = np.array([[1, 2],
4               [3, 4]])
5
6 E = A.T
7
8 print(E)
9
```

hell x

```
>> %Run -c $EDITOR_CONTENT
```

```
[[1 3]
 [2 4]]
```

```

1 import numpy as np
2
3 A = np.matrix([[1, 2],
4                [3, 4]])
5
6 B = np.matrix([[3, 4],
7                [5, 6]])
8
9 C = A * B
10
11 print(C)
12

```

Shell ×

>>> %Run -c \$EDITOR_CONTENT

```

[[13 16]
 [29 36]]

```

```

1 import numpy as np
2
3 A = np.matrix([[1, 2],
4                [3, 4]])
5
6 E = A.T
7
8 print(E)
9

```

hell ×

>>> %Run -c \$EDITOR_CONTENT

```

[[1 3]
 [2 4]]

```

```

1 import numpy as np
2
3 A = np.matrix([[1, 2],
4                [3, 4]])
5
6 print(A)
7
8 A_det = np.linalg.det(A)
9
10 print(A_det)
11

```

Shell ×

>>> %Run -c \$EDITOR_CONTENT

```

[[1 2]
 [3 4]]
-2.0000000000000004

```

```
1 import numpy
2
3 A = numpy.matrix([[1, 2],
4                   [3, 4]])
5
6 print(A)
7
8 A_det = numpy.linalg.det(A)
9
10 print(A_det)
11
```



```
1 import numpy as np
2 import numpy.linalg as la
3
4 A = np.matrix([[1, 2],
5                [3, 4]])
6
7 print(A)
8
9 A_det = la.det(A)
10
11 print(A_det)
12
```

```
1 from numpy import *
2 from numpy.linalg import *
3
4 A = matrix([[1, 2],
5             [3, 4]])
6
7 print(A)
8
9 A_det = det(A)
10
11 print(A_det)
12
```

Shell x

```
>>> %Run -c $EDITOR_CONTENT
```

```
[[1 2]
 [3 4]]
-2.0000000000000004
```

```

1 import numpy as np
2 import numpy.linalg as la
3
4 A = np.matrix([[1, 2],
5                [3, 4]])
6
7 print(A)
8
9 A_inv = la.inv(A)
10
11 print(A_inv)
12
13 print(A * A_inv)
14

```

Shell ×

```
>>> %Run -c $EDITOR_CONTENT
```

```

[[1 2]
 [3 4]]
[[-2.  1. ]
 [ 1.5 -0.5]]
[[1.0000000e+00 0.0000000e+00]
 [8.8817842e-16 1.0000000e+00]]

```

```

1 import numpy as np
2 import numpy.linalg as la
3
4 A = np.array([[1, 2],
5               [3, 4]])
6
7 print(A)
8
9 A_inv = la.inv(A)
10
11 print(A_inv)
12
13 print(A @ A_inv)
14

```

Shell ×

```
>>> %Run -c $EDITOR_CONTENT
```

```

[[1 2]
 [3 4]]
[[-2.  1. ]
 [ 1.5 -0.5]]
[[1.0000000e+00 0.0000000e+00]
 [8.8817842e-16 1.0000000e+00]]

```

복소수

```
1 z = 1j
2
3 print(z * z)
4
5
```

ell x

```
>> %Run -c $EDITOR_CONTENT
(-1+0j)
```

```
1 z = 2 + 3j
2
3 print(z)
4
5
```

Shell x

```
>>> %Run -c $EDITOR_CONTENT
(2+3j)
```

```
1 a = 2 + 3j
2 b = 4 + 5j
3
4 c = a + b
5 print(c)
6
7 d = a - b
8 print(d)
9
10 e = a * b
11 print(e)
12
```

shell x

```
>>> %Run -c $EDITOR_CONTENT
```

```
(6+8j)
(-2-2j)
(-7+22j)
```

```
1 x = 2
2 y = 3
3
4 z = complex(x, y)
5
6 print(z)
7
```

```
1 x = 2
2 y = 3
3
4 z = complex(x, y)
5
6 print(z)
7 print(z.real)
8 print(z.imag)
9
```

shell x

```
>> %Run -c $EDITOR_CONTENT
```

```
(2+3j)
2.0
3.0
```

```
1 import cmath
2
3 z = 2 + 3j
4
5 c = cmath.polar(z)
6 print(c)
```

Shell x

```
>>> %Run -c $EDITOR_CONTENT
(3.605551275463989, 0.982793723247329)
```

```

1 import cmath
2
3 z = 1 + 0j
4
5 c = cmath.polar(z)
6 print(c)
7
8 print((1**2 + 0**2)**0.5)
9 print(0)

```

hell x

```

>>> %Run -c $EDITOR_CONTENT
(1.0, 0.0)
1.0
0

```

```

1 import cmath
2
3 z = 0 + 1j
4
5 c = cmath.polar(z)
6 print(c)
7
8 print((0**2 + 1**2)**0.5)
9 print(3.141592/2)

```

hell x

```

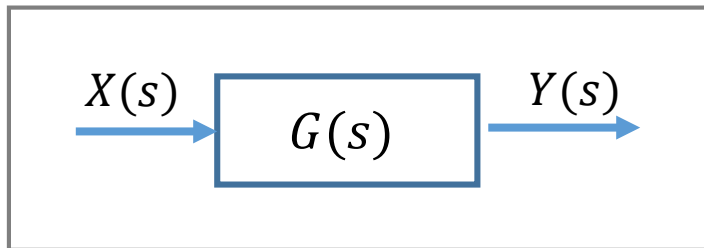
>>> %Run -c $EDITOR_CONTENT
(1.0, 1.5707963267948966)
1.0
1.570796

```

전달함수

◆ 전달함수의 정의

- 입력 라플라스 변환에 대한 출력 라플라스 변환의 비
- 단, 모든 초기 조건들은 0이다.



★ 전달함수의 정의 ★

$$G(s) = \frac{Y(s)}{X(s)}$$

◆ 전달함수의 정의

- 전달함수 $G(s)$ 가 주어진 시스템에 입력 $x(t)$ 를 가했을 때의 출력 $y(t)$

- ① 입력의 라플라스 변환 $X(s)$ 를 구해, 전달함수 $G(s)$ 에 곱하여 출력의 라플라스 변환 $Y(s)$ 를 구한다.

$$Y(s) = G(s) X(s)$$

- ② 구한 $Y(s)$ 를 역변환하여 출력 $y(t)$ 를 얻는다.

$$y(t) = \mathcal{L}^{-1} [Y(s)] = \mathcal{L}^{-1} [G(s) X(s)]$$

◆ 전달함수 구하기

입력과 출력 관계가 다음과 같이 미분방정식으로 표현되는 자동제어시스템의 전달함수를 구하라.

$$\frac{d^2}{dt^2} c(t) + 2 \frac{d}{dt} c(t) + 3 c(t) = 5 \frac{d}{dt} r(t) + 4 r(t)$$

◆ 전달함수 구하기

$$\frac{d^2}{dt^2} c(t) + 2 \frac{d}{dt} c(t) + 3 c(t) = 5 \frac{d}{dt} r(t) + 4 r(t)$$

초기값 0

풀이

$$[\{s^2 C(s) - s c(0) - \dot{c}(0)\} + 2\{s C(s) - c(0)\} + 3 C(s)] = [5\{s R(s) - r(0)\} + 4 R(s)]$$

$$s^2 C(s) + 2s C(s) + 3 C(s) = 5s R(s) + 4 R(s)$$

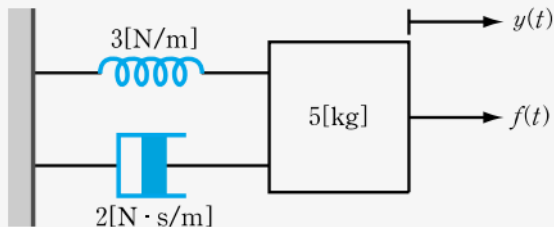
$$(s^2 + 2s + 3) C(s) = (5s + 4) R(s)$$

$$G(s) = \frac{C(s)}{R(s)} = \frac{5s + 4}{s^2 + 2s + 3}$$

◆ 전달함수 구하기

[그림 3-3]과 같이 입력 힘 $f(t)$ 에 대한 변위 $y(t)$ 를 출력으로 하는 기계시스템의 힘의 방정식은 다음과 같다. 이 기계시스템의 전달함수 $G(s)$ 를 구하라.

$$5 \frac{d^2}{dt^2} y(t) + 2 \frac{d}{dt} y(t) + 3 y(t) = f(t)$$



[그림 3-3] [예제 3-2]의 기계시스템 구성

◆ 전달함수 구하기

$$5 \frac{d^2}{dt^2} y(t) + 2 \frac{d}{dt} y(t) + 3 y(t) = f(t)$$

$$5 \{s^2 Y(s) - s y(0) - \dot{y}(0)\} + 2 \{s Y(s) - y(0)\} + 3 Y(s) = F(s)$$

$$5 s^2 Y(s) + 2 s Y(s) + 3 Y(s) = F(s)$$

$$(5 s^2 + 2 s + 3) Y(s) = F(s)$$

$$G(s) = \frac{Y(s)}{F(s)} = \frac{1}{5 s^2 + 2 s + 3}$$

◆ 코딩

전달함수 $H(s) = \frac{2s+1}{s^2+5s+6}$ 을 정의하고 스텝 응답을 구하는 코드

python

```
import control as ct  
import numpy as np
```

분자(num)와 분모(den) 다항식 계수 정의 (s의 내림차순)

```
num = [2, 1]
```

```
den = [1, 5, 6]
```

전달함수 생성

```
sys = ct.tf(num, den)
```

```
print(sys)
```

스텝 응답 계산

```
t, y = ct.step_response(sys)
```

$$H(s) = \frac{2s+1}{s^2+5s+6}$$

```
1 import control as ct
2
3 num = [2, 1]
4 den = [1, 5, 6]
5
6 sys = ct.tf(num, den)
7 print(sys)
```

shell ×

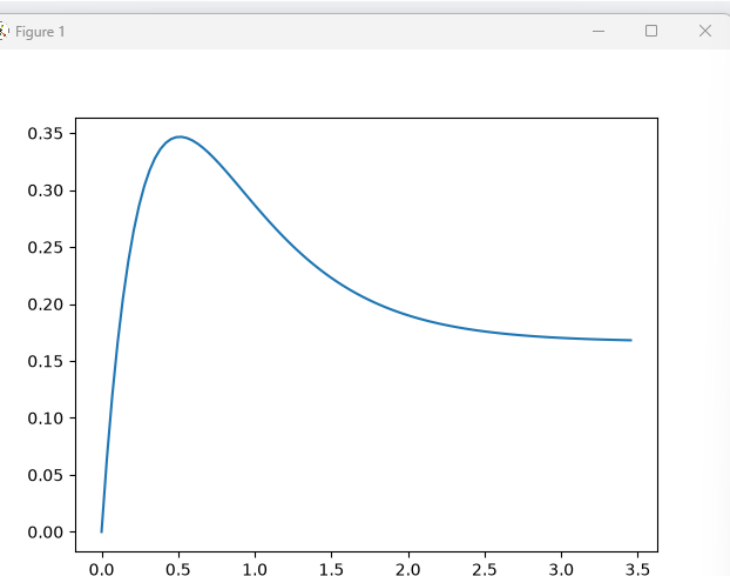
```
>>> %Run -c $EDITOR_CONTENT
```

```
<TransferFunction>: sys[0]
```

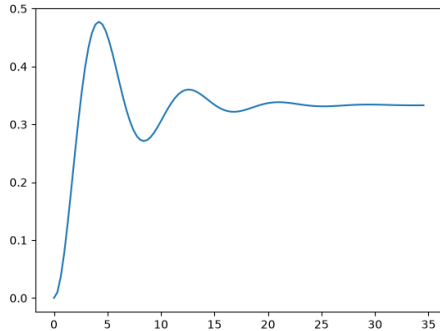
```
Inputs (1): ['u[0]']
```

```
Outputs (1): ['y[0]']
```

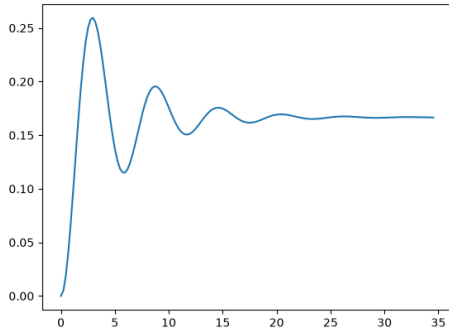
$$\frac{2s + 1}{s^2 + 5s + 6}$$



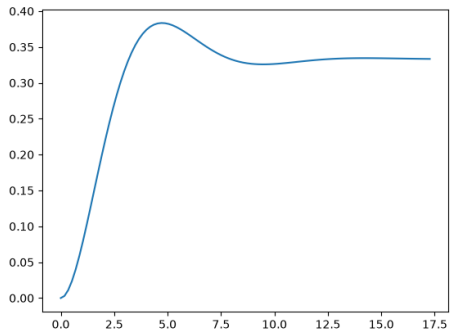
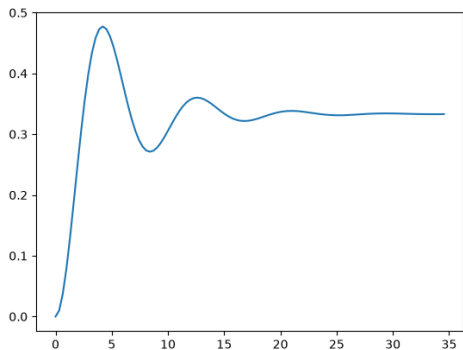
```
<untitled> * x
1 import control as ct
2 import matplotlib.pyplot as plt
3
4 num = [2, 1]
5 den = [1, 5, 6]
6
7 sys = ct.tf(num, den)
8 print(sys)
9
10 t, y = ct.step_response(sys)
11
12 plt.plot(t, y)
13
14 plt.show()
15
```



```
<untitled> * x Resume (F8)
1 import control as ct
2 import matplotlib.pyplot as plt
3
4 num = [1]
5 den = [5, 2, 3]
6
7 sys = ct.tf(num, den)
8 print(sys)
9
10 t, y = ct.step_response(sys)
11
12 plt.plot(t, y)
13
14 plt.show()
15
```



```
1 import control as ct
2 import matplotlib.pyplot as plt
3
4 num = [1]
5 den = [5, 2, 3*2]
6
7 sys = ct.tf(num, den)
8 print(sys)
9
10 t, y = ct.step_response(sys)
11
12 plt.plot(t, y)
13
14 plt.show()
15
```



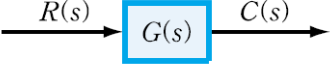
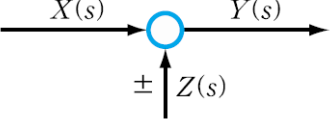
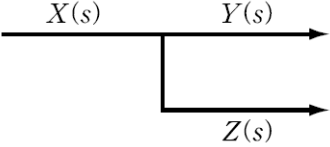
```
<untitled> * x [Resume (F8)]
1 import control as ct
2 import matplotlib.pyplot as plt
3
4 num = [1]
5 den = [5, 2, 3]
6
7 sys = ct.tf(num, den)
8 print(sys)
9
10 t, y = ct.step_response(sys)
11
12 plt.plot(t, y)
13
14 plt.show()
15
```

```
<untitled> * x
1 import control as ct
2 import matplotlib.pyplot as plt
3
4 num = [1]
5 den = [5, 2*2, 3]
6
7 sys = ct.tf(num, den)
8 print(sys)
9
10 t, y = ct.step_response(sys)
11
12 plt.plot(t, y)
13
14 plt.show()
15
```

전달함수

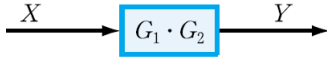
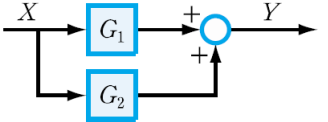
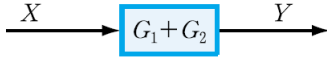
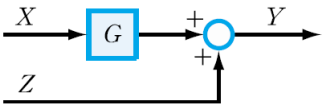
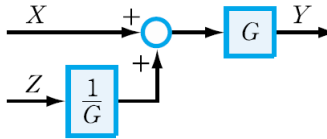
◆ 블록선도

● 표기법

종류	기호	연산
신호		화살표 방향으로만 전달
전달요소		$C(s) = G(s) R(s)$
가산점		$Y(s) = X(s) \pm Z(s)$
인출점		$X(s) = Y(s) = Z(s)$

전달함수

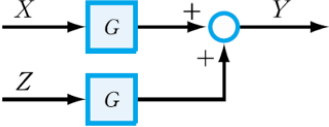
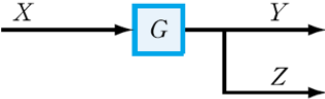
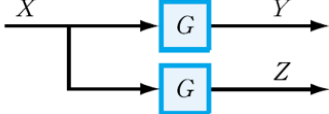
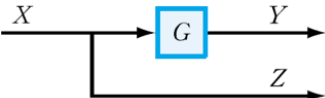
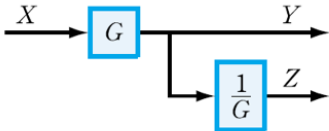
◆ 블록선도 ● 등가변환

번호	블록연산	블록선도	등가 블록선도
1	교환		
2	직렬결합		
3	병렬결합		
4	가산점을 앞으로 이동		

전달함수

◆ 블록선도

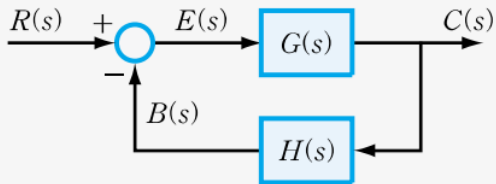
● 등가변환

번호	블록연산	블록선도	등가 블록선도
5	가산점을 뒤로 이동		
6	인출점을 앞으로 이동		
7	인출점을 뒤로 이동		

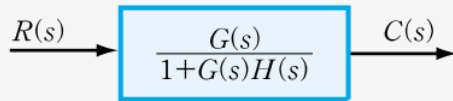
◆ 블록선도

- 등가변환

[그림 3-9(a)]와 같은 폐루프 제어시스템의 등가변환이 [그림 3-9(b)]와 같음을 보여라.



(a) 폐루프 제어시스템의 블록선도



(b) 등가변환 블록선도

[그림 3-9] [예제 3-5]의 등가변환

◆ 블록선도

● 등가변환

$$E(s) = R(s) - B(s)$$

$$C(s) = G(s) E(s)$$

$$B(s) = H(s) C(s)$$

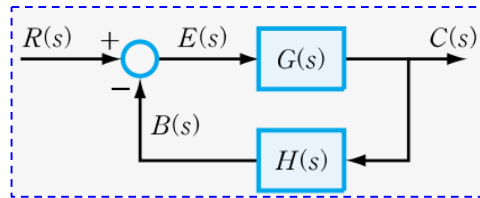
$$E(s) = R(s) - H(s) C(s)$$

$$C(s) = G(s) E(s) = G(s) \{R(s) - H(s) C(s)\}$$

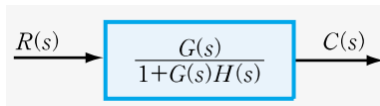
$$C(s) + G(s) H(s) C(s) = G(s) R(s)$$

$$\{1 + G(s) H(s)\} C(s) = G(s) R(s)$$

$$C(s) = \frac{G(s)}{1 + G(s) H(s)} R(s)$$

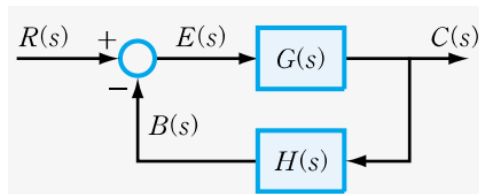


$$M(s) = \frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s) H(s)}$$



전달함수

◆ 블록선도 ● 등가변환



- 전향경로 전달함수(feed forward transfer function) : $G(s)$
- 개루프 전달함수(open-loop transfer function) : $G(s)H(s)$
- 폐루프 전달함수(closed-loop transfer function) : $\frac{G(s)}{1 + G(s)H(s)}$



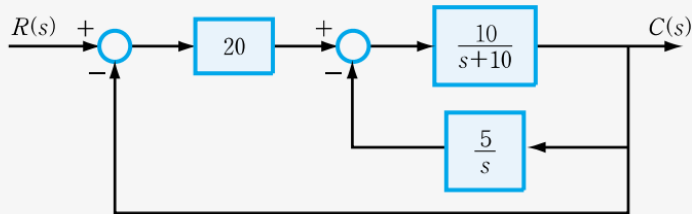
$$M(s) = \frac{G(s)}{1 + G(s)H(s)}$$

[그림 3-9] 단위귀환 제어시스템

전달함수

◆ 블록선도 ● 등가변환

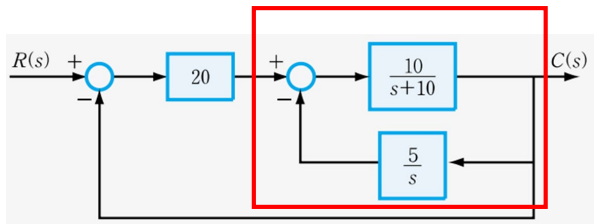
[그림 3-11]과 같이 복잡한 귀환 제어시스템의 전체 폐루프 전달함수를 구하라.



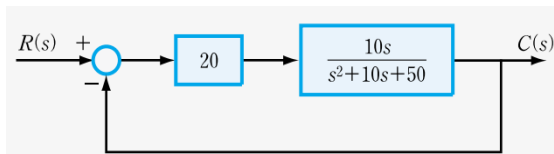
[그림 3-11] [예제 3-6]의 귀환 제어시스템

전달함수

◆ 블록선도 ● 등가변환



$$\frac{\frac{10}{s+10}}{1 + \frac{10}{s+10} \times \frac{5}{s}} = \frac{10s}{s^2 + 10s + 50}$$



[그림 3-12] 간략화된 [그림 3-11]의 귀환 제어시스템

$$\begin{aligned} M(s) &= \frac{\frac{200s}{s^2 + 10s + 50}}{1 + \frac{200s}{s^2 + 10s + 50}} \\ &= \frac{200s}{s^2 + 210s + 50} \end{aligned}$$