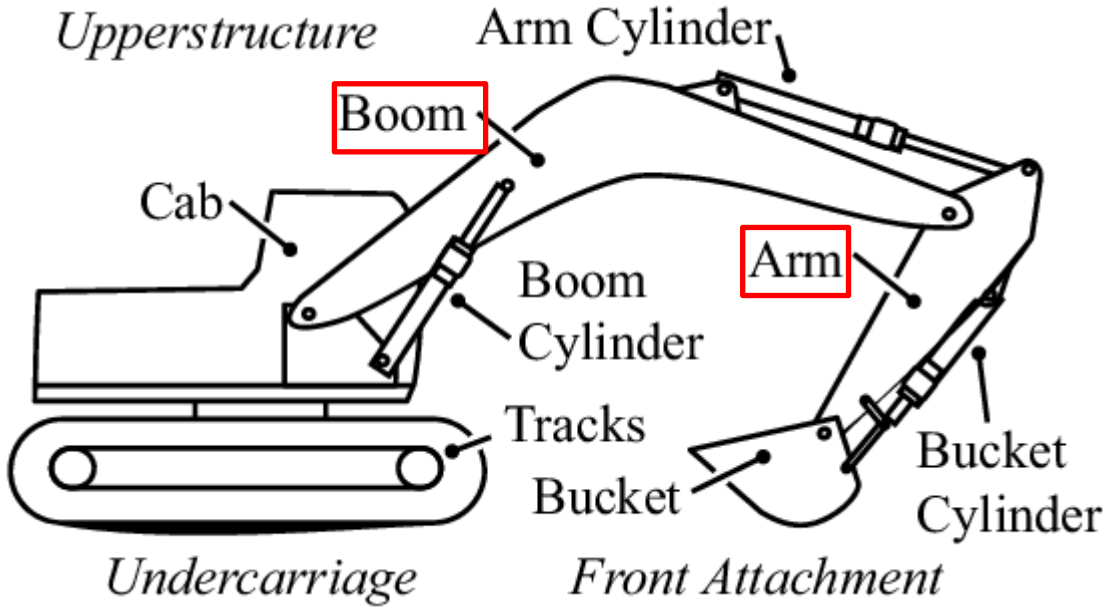
We Arm스트롱

2020092560 장경민
2018015996 백종욱

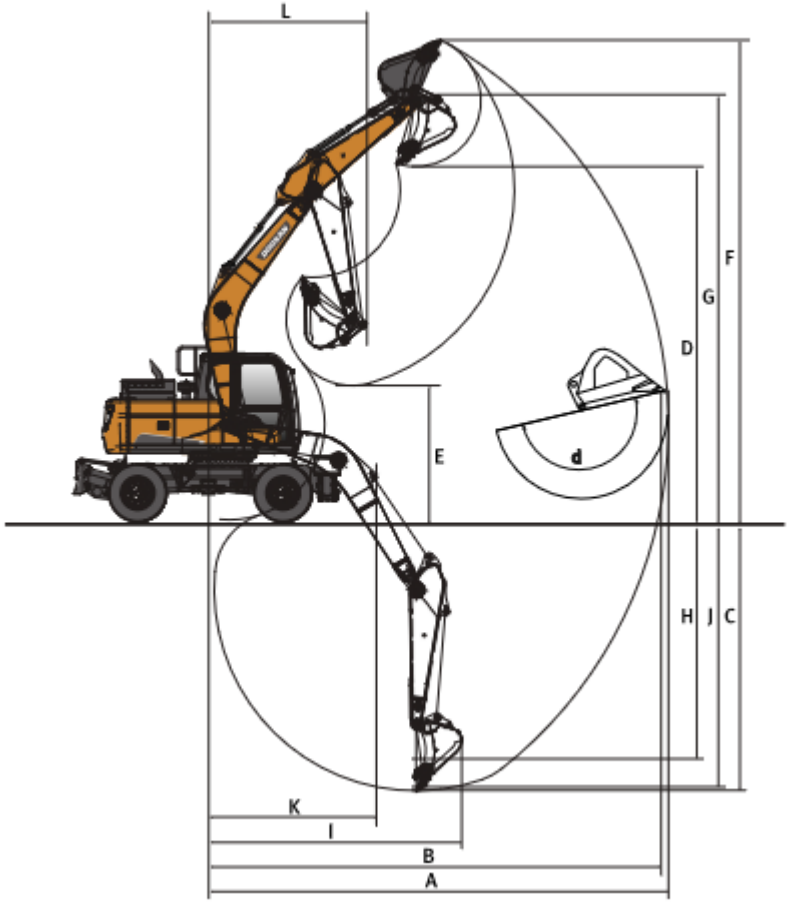
굴삭기 소개

Doosan DX140W-5

붐(Boom): 4.4m & 1500kg, 암(Arm): 2.2m & 700kg



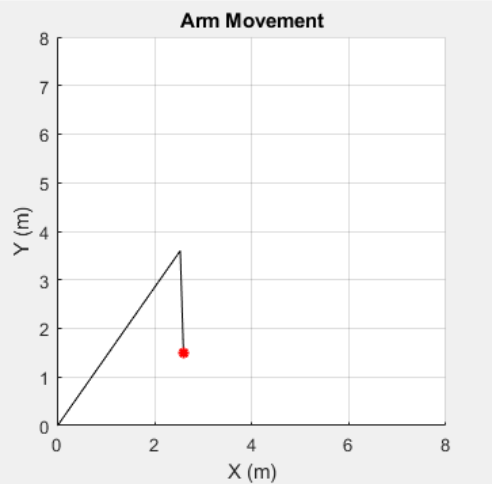
굴삭기 상황 설명



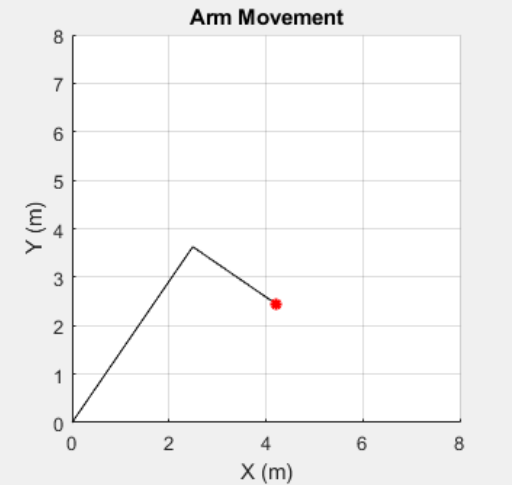
- (1)부터 (3)까지의 상황에 따라 **토크, 응력**이 어떻게 달라질까?
- 그리고 굴삭기의 **안전성 평가**를 해보자.

MATLAB, Simulink와 COMSOL을 이용해 설명해보자.

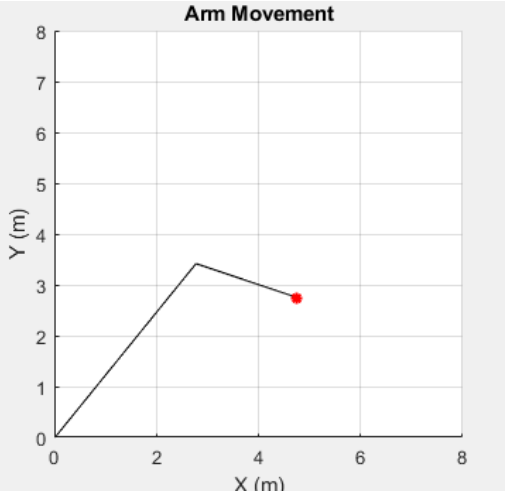
(1) **예각**



(2) **직각**

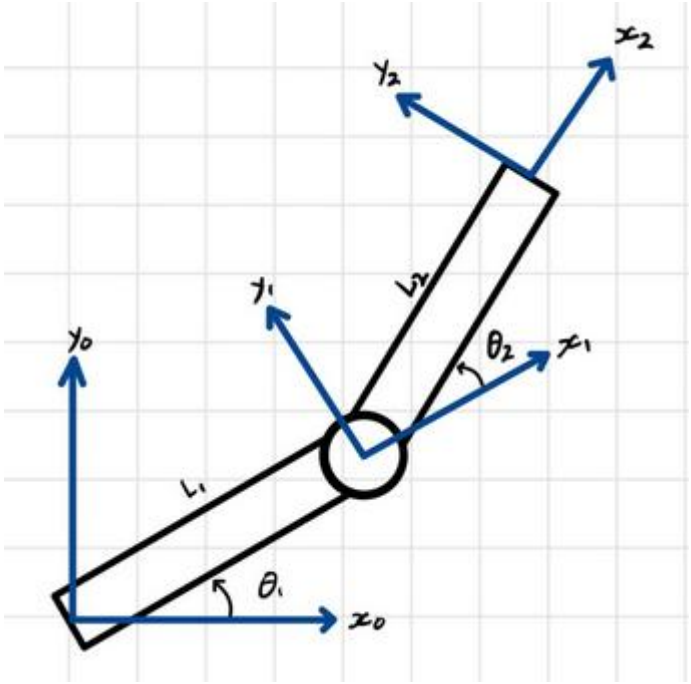


(3) **둔각**



순기구학 (Forward Kinematics)

- 각 조인트 공간에서 각이 주어졌을 때, 직교공간에서 링크의 끝의 위치를 구하는 것

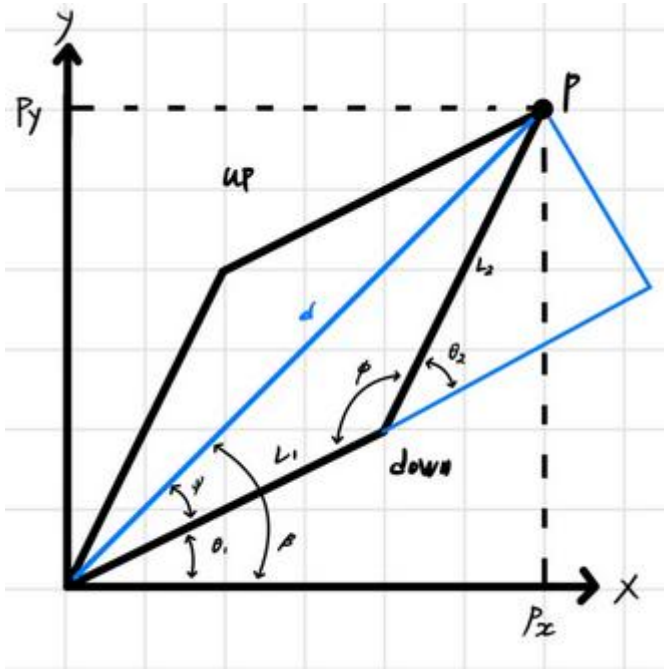


기하학적 방법으로 접근

$$x = L_1 cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$
$$y = L_1 sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

역기구학 (Inverse Kinematics)

- 직교공간에서 링크의 끝의 위치가 주어졌을 때, 조인트 공간에서 각의 값을 구하는 것



기하학적 방법으로 접근

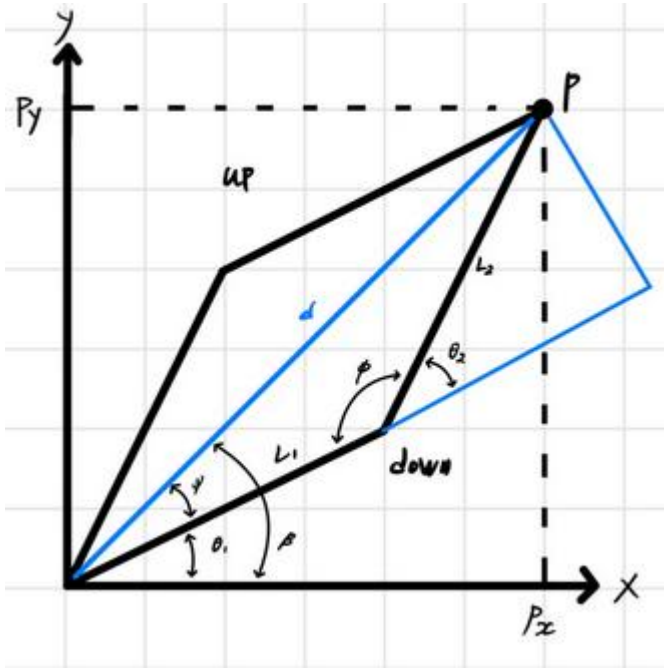
$$\mathrm{d} = \sqrt{P_x^2 + P_y^2} \qquad d^2 = l_1^2 + l_2^2 - 2l_1 l_2 cos\varphi$$

$$cos\varphi = \frac{l_1^2+l_2^2-d^2}{2l_1 l_2} = \cos(180° - \theta_2) \qquad \cos\theta_2 = -\left(\frac{l_1^2+l_2^2-d^2}{2l_1 l_2}\right)$$

$$\sin\theta_2 = \pm\sqrt{1 - cos^2\theta_2} \qquad \therefore \theta_2 = tan^{-1}\left(\frac{sin\theta_2}{cos\theta_2}\right)$$

역기구학 (Inverse Kinematics)

- 직교공간에서 링크의 끝의 위치가 주어졌을 때, 조인트 공간에서 각의 값을 구하는 것



기하학적 방법으로 접근

$$l_2^2 = d^2 + l_1^2 - 2dl_1 cos\psi$$

$$cos\psi = \frac{l_1^2 + d^2 - l_2^2}{2l_1 d} \qquad \therefore \psi = cos^{-1}\left(\frac{l_1^2 + d^2 - l_2^2}{2l_1 d}\right)$$

$$\beta = tan^{-1}\left(\frac{P_y}{P_x}\right) \qquad \therefore \theta_1 = \beta \pm \psi$$

결론

$$\therefore \theta_1 = \beta \pm \psi, \quad \theta_2 = tan^{-1}\left(\mp\frac{\sqrt{1-cos^2\theta_2}}{cos\theta_2}\right)$$

-> 굴삭기는 up의 상황만 존재.

라그랑주 방식 – 동역학 방정식

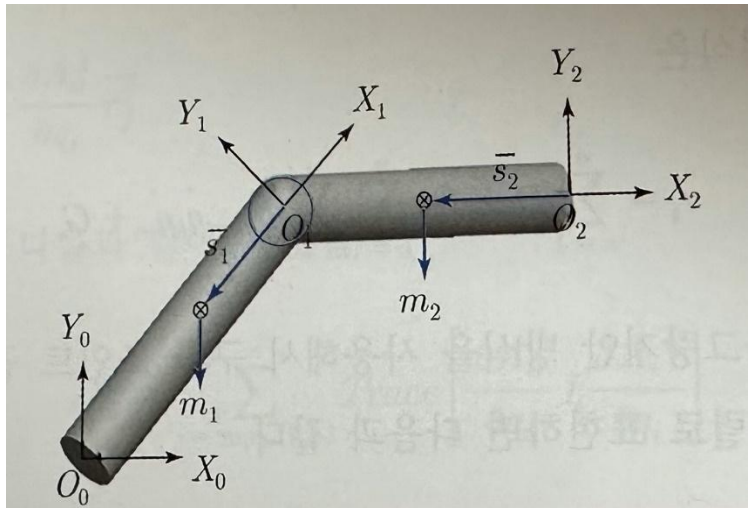
$$\boldsymbol{\tau = H(q)\ddot{q} + C(q,\dot{q}) + G(q)}$$

1. $\boldsymbol{H(q)}$: **관성 행렬**
   링크의 질량 분포와 관성을 나타냄
2. $\boldsymbol{C(q,\dot{q})}$: **코리올리 및 원심력 행렬**
   관성에 의해 발생하는 속도 의존성 항
3. $\boldsymbol{G(q)}$: **중력 항**
   링크가 중력에 의해 받는 힘
4. $\boldsymbol{\tau}$: **조인트 토크 벡터,** $\boldsymbol{q}$: **각도 벡터**

라그랑주 방식 – 동역학 방정식

$$\tau = H(q)\ddot{q} + C(q,\dot{q}) + G(q)$$

계산 과정은 생략



$$H(q) = \begin{bmatrix} \frac{1}{3}m_1L_1^2 + \frac{1}{3}m_2L_2^2 + m_2L_1^2 + m_2L_1L_2c_2 & \frac{1}{3}m_2L_2^2 + \frac{1}{2}m_2L_1L_2c_2 \\ \frac{1}{3}m_2L_2^2 + \frac{1}{2}m_2L_1L_2c_2 & \frac{1}{3}m_2L_2^2 \end{bmatrix}$$

$$C(q,\dot{q}) = \begin{bmatrix} -m_2L_1L_2s_2\dot{q}_1\dot{q}_2 - \frac{1}{2}m_2L_1L_2s_2\dot{q}_2^2 \\ \frac{1}{2}m_2L_1L_2s_2\dot{q}_1^2 \end{bmatrix}$$

$$G(q) = \begin{bmatrix} \frac{1}{2}m_1L_1gc_1 + \frac{1}{2}m_2L_2gc_{12} + m_2L_1gc_1 \\ \frac{1}{2}m_2L_2gc_{12} \end{bmatrix}$$

참고

$$c_1 = \cos(q_1), c_2 = \cos(q_2)$$
$$s_2 = sin(q_2), c_{12} = \cos(q_1 + q_2)$$

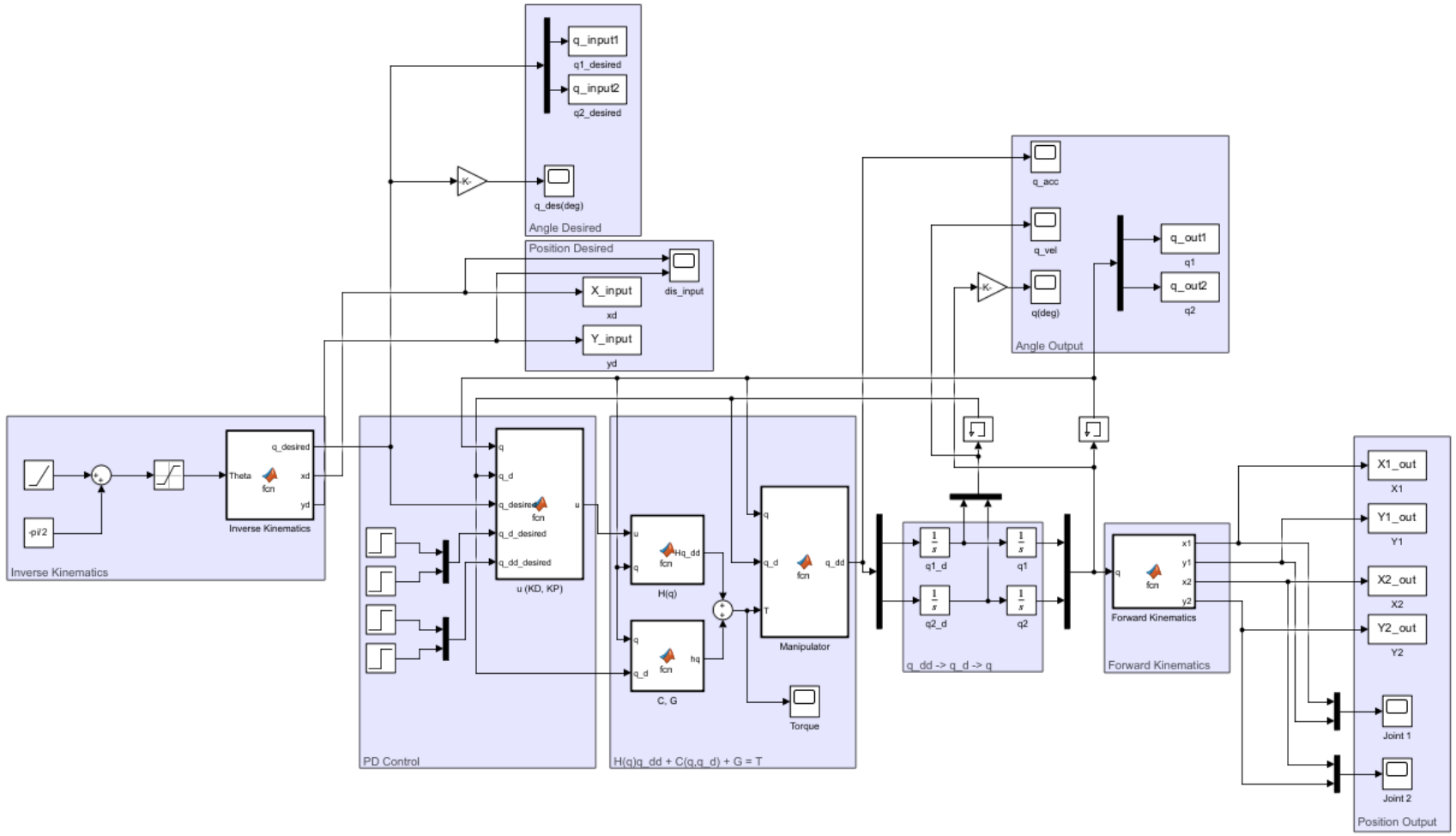
PD Control

$$\tau = H(q)(\ddot{q} - K_p e - K_d \dot{e}) + C(q, \dot{q}) + G(q)$$

$$e = q - q_d, \qquad \dot{e} = \dot{q} - \dot{q}_d$$

- 오차 기반으로 PD 제어를 통해 원하는 각가속도를 구현
- 굴삭기에 적용한 경우: 부드러운 움직임, 작업 효율성 향상, 안정성 강화

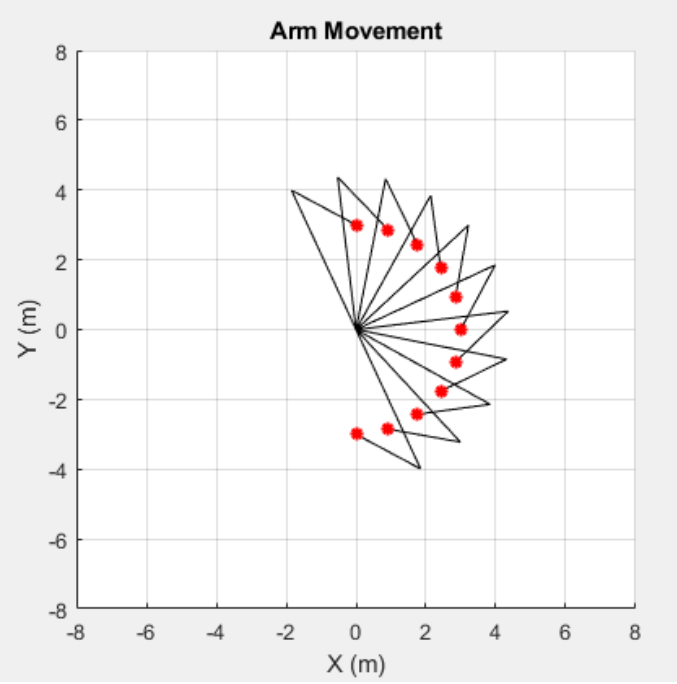
Simulink 구현



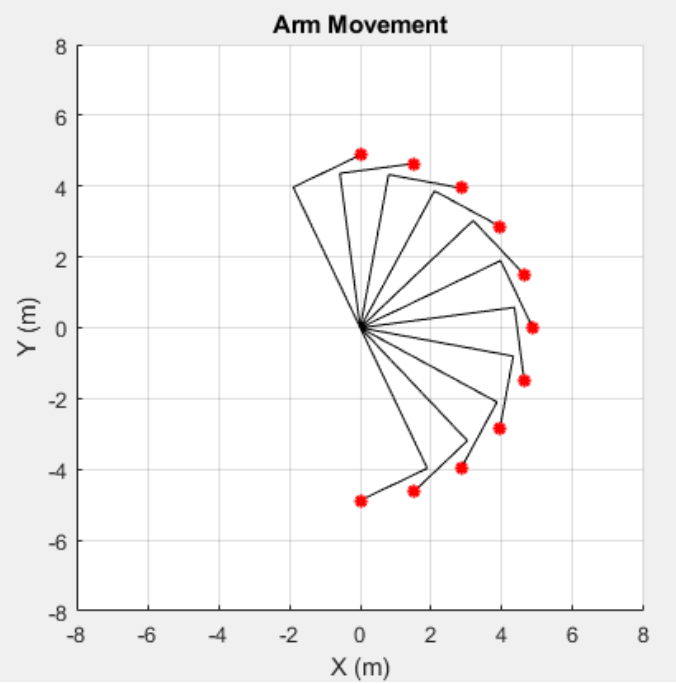
Excavator_Control.slx

MATLAB 구현

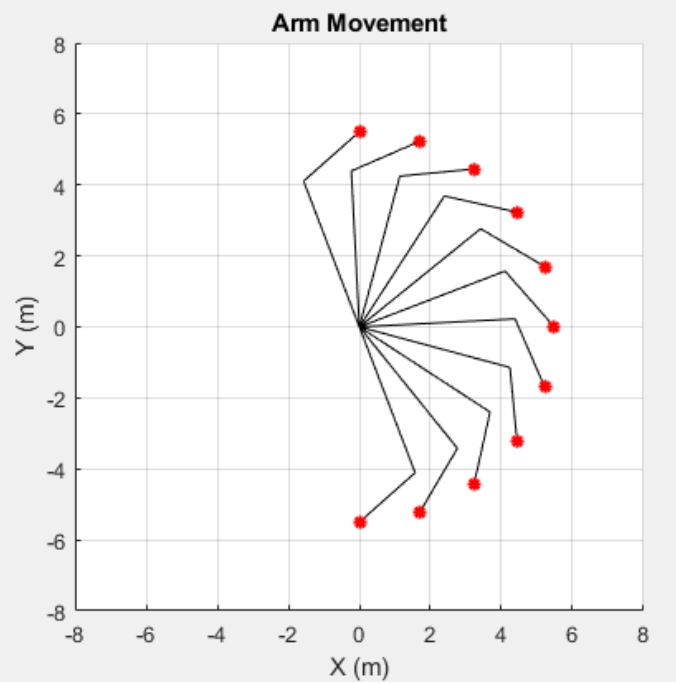
제원표에 따른 굴삭기의 동선을 구현



(1) 예각



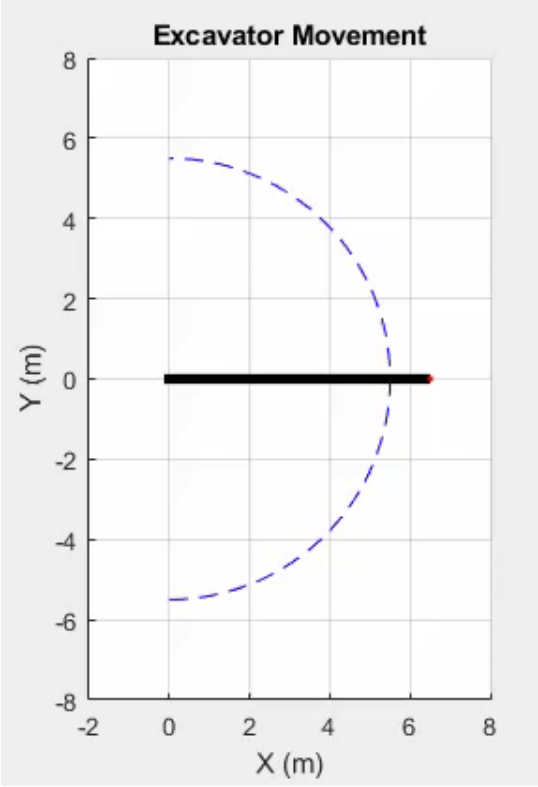
(2) 직각



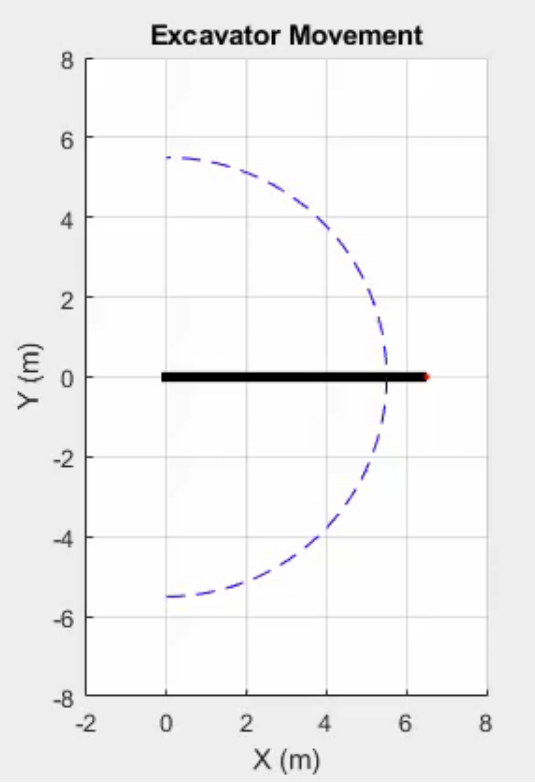
(3) 둔각

MATLAB + Simulink 구현

PD Control 적용 후, 진동이 크게 줄어든 것을 볼 수 있음.



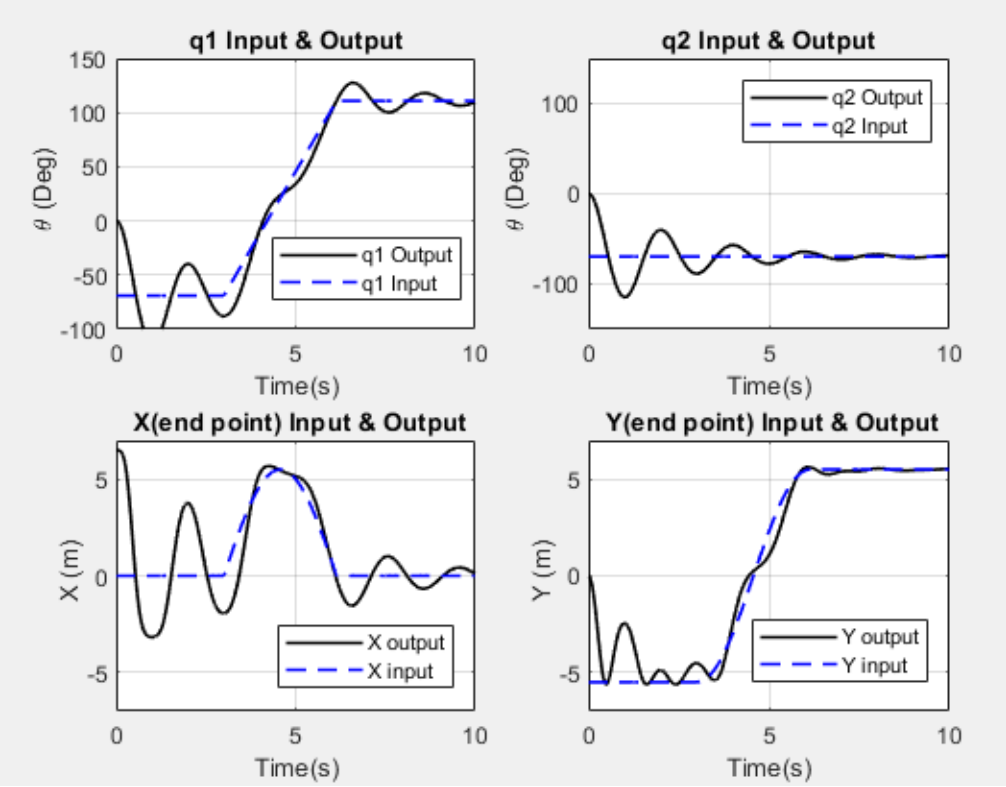
적용 전



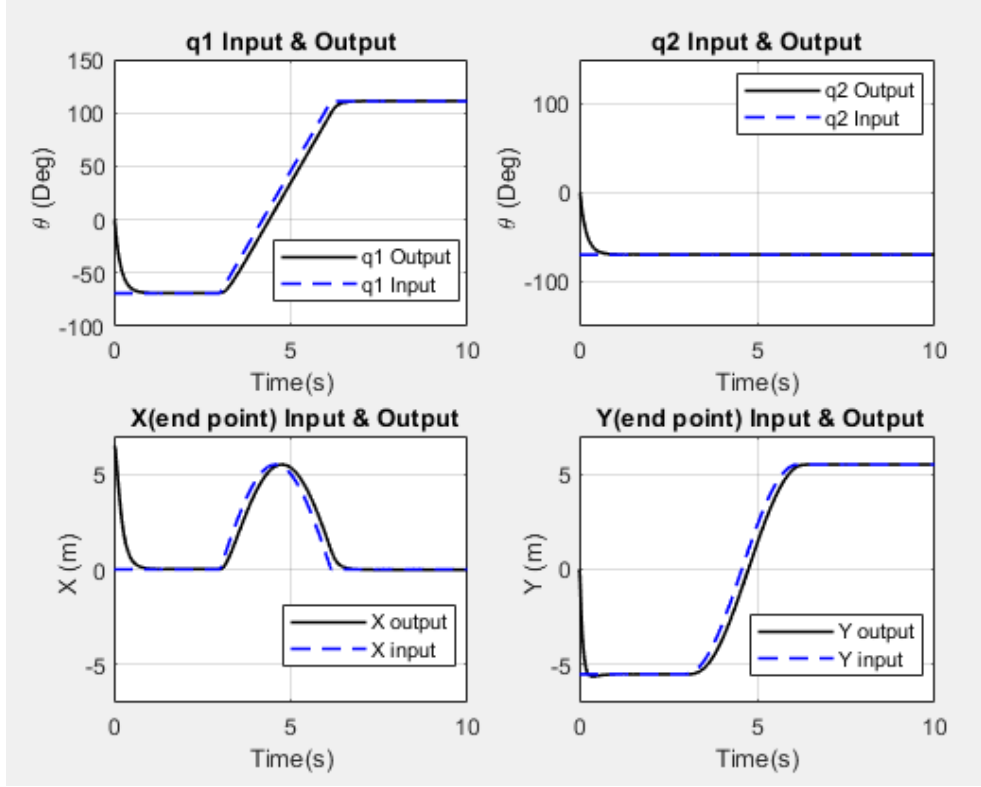
적용 후

MATLAB + Simulink 구현

PD Control 적용 후, 진동이 크게 줄어든 것을 볼 수 있음.



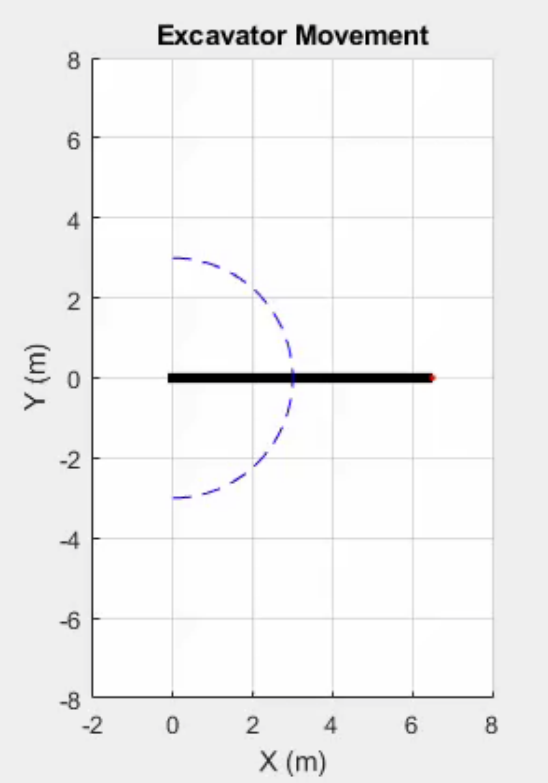
적용 전



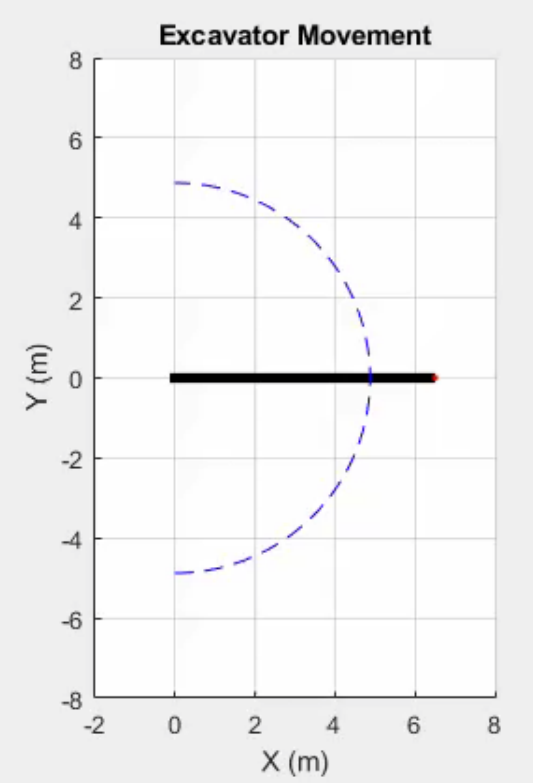
적용 후

MATLAB + Simulink 구현

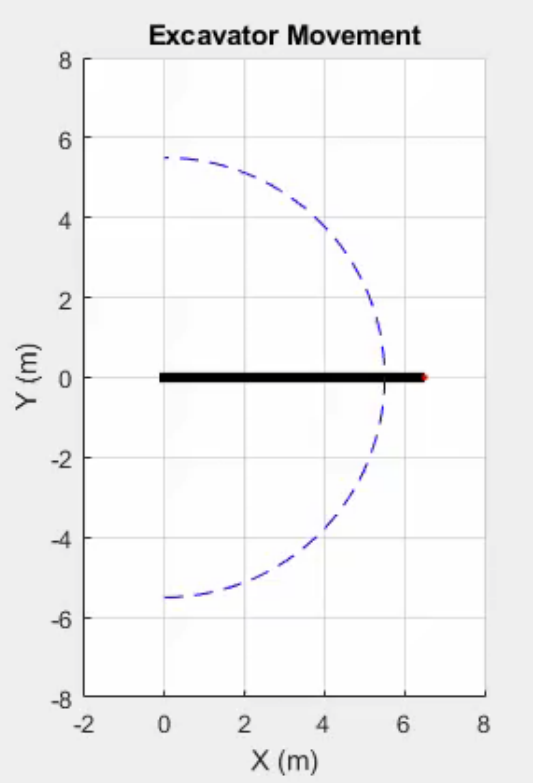
붐과 암이 이동하는 과정을 구현



(1) 예각

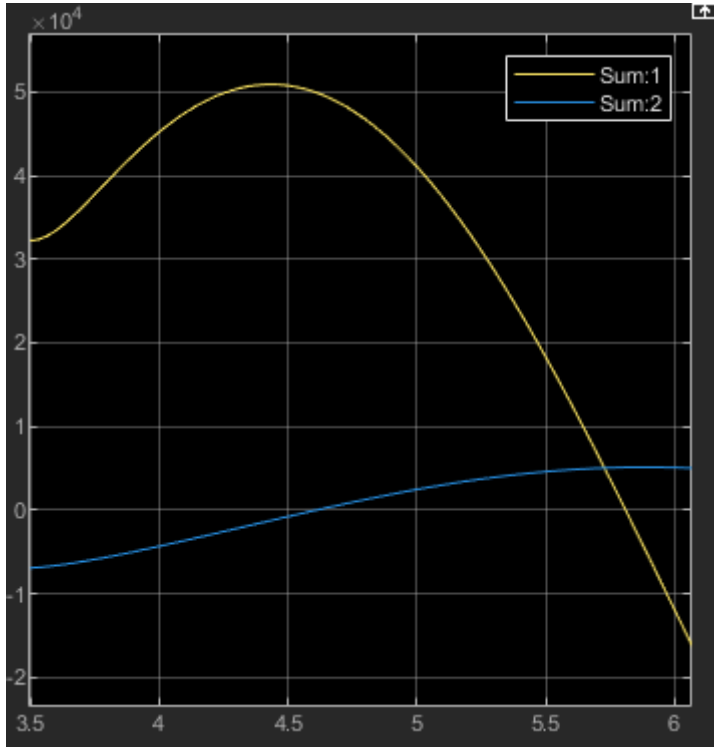


(2) 직각

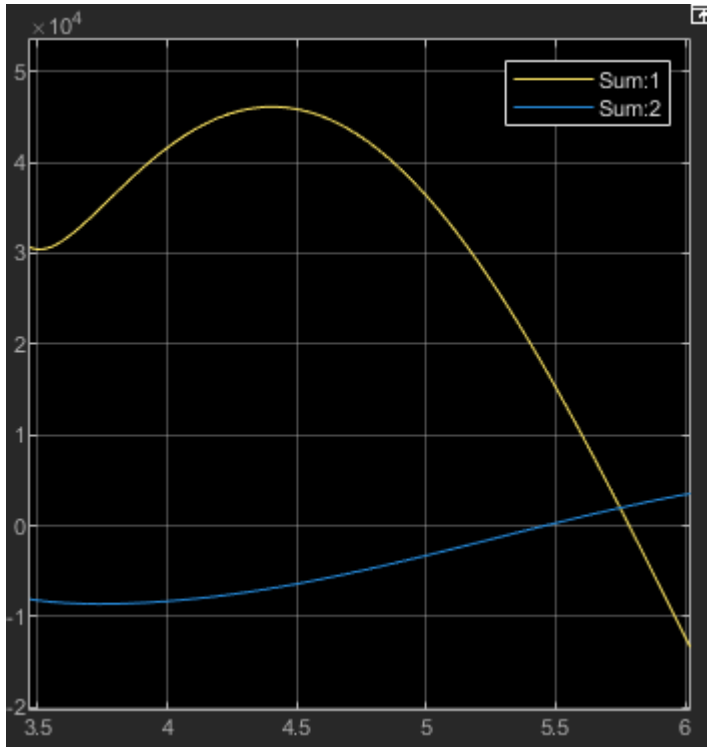


(3) 둔각

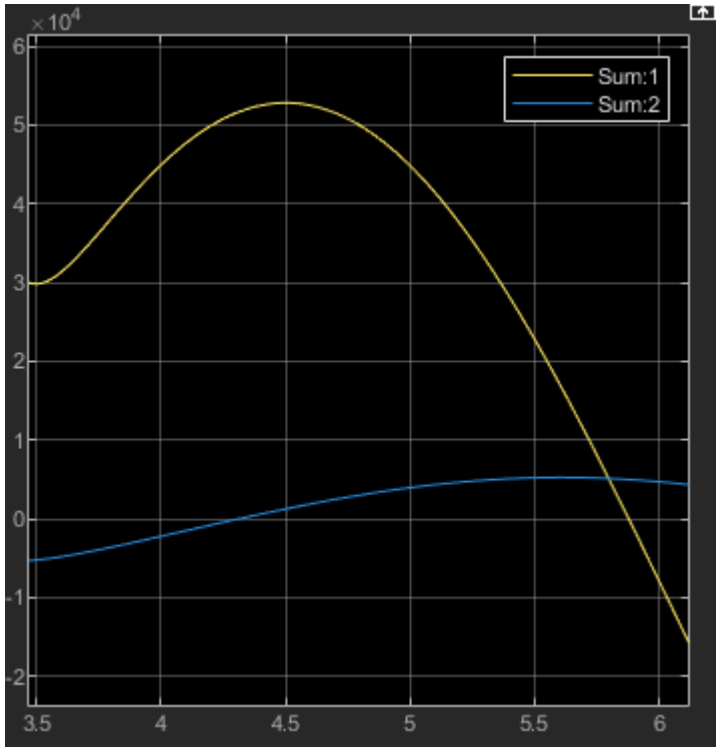
토크 그래프



(1) 예각



(2) 직각

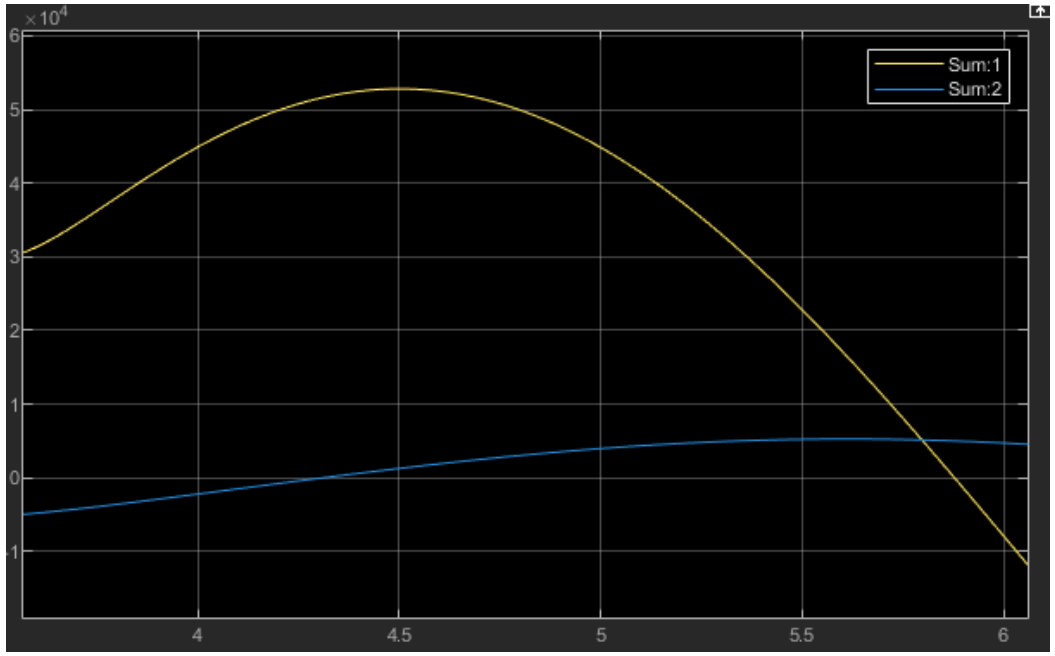


(3) 둔각

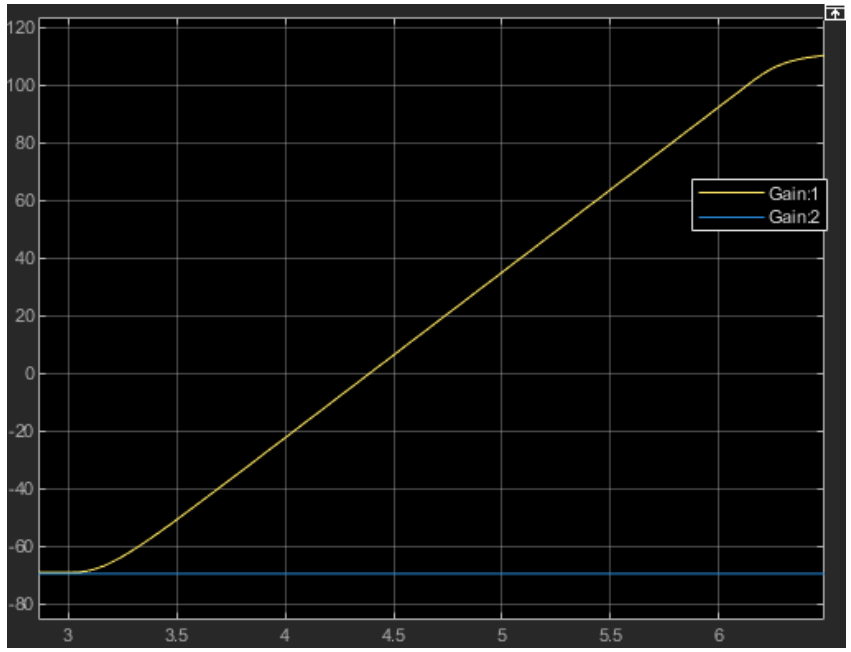
노란색이 관절 1
파란색이 관절 2

Simulink 그래프

토크 그래프와 각 관절의 각도 그래프를 이용해 COMSOL 프로그램에 이용



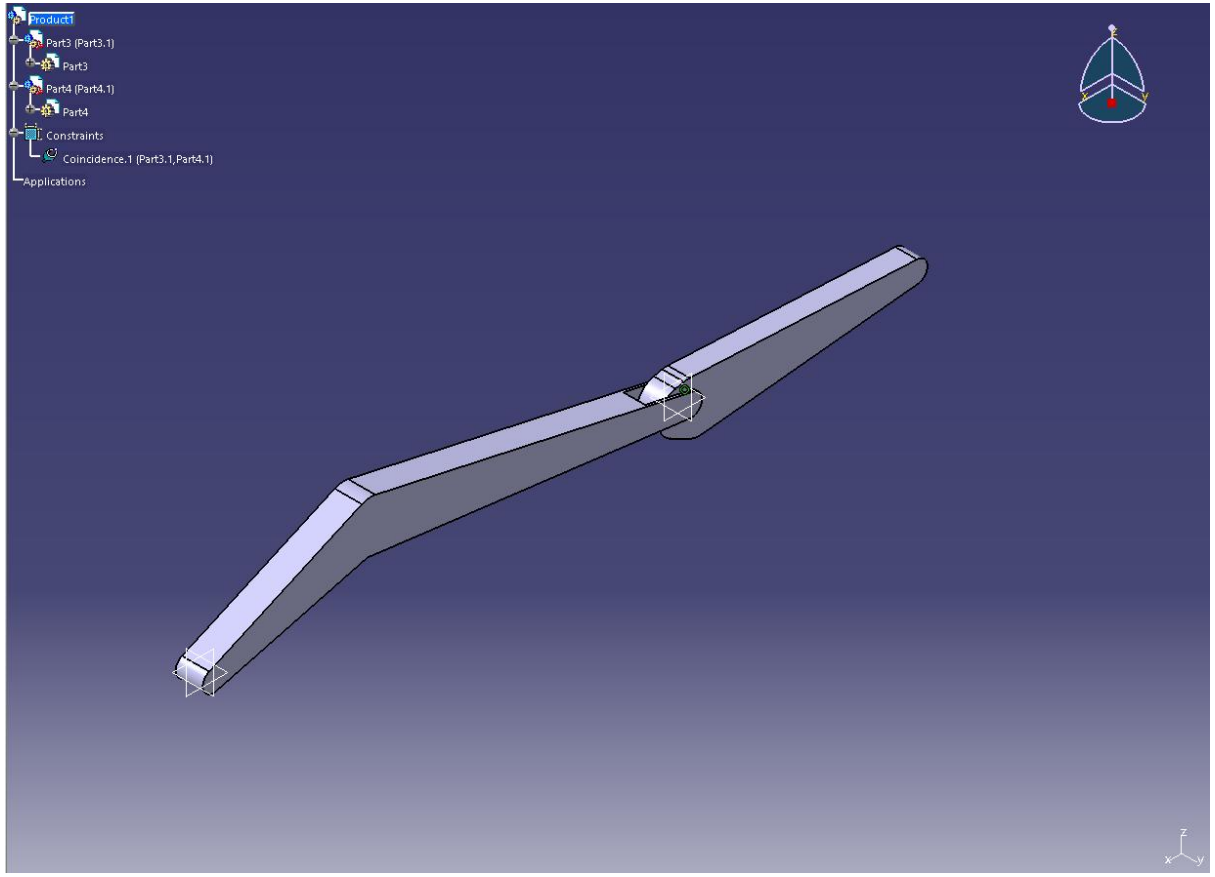
토크 그래프



각 관절의 각도 그래프

노란색이 관절 1
파란색이 관절 2

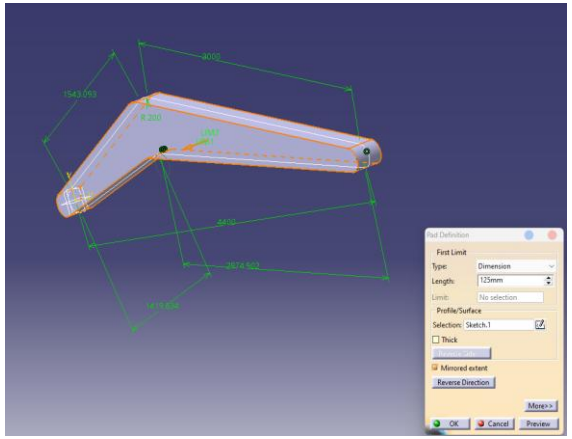
CATIA 모델링

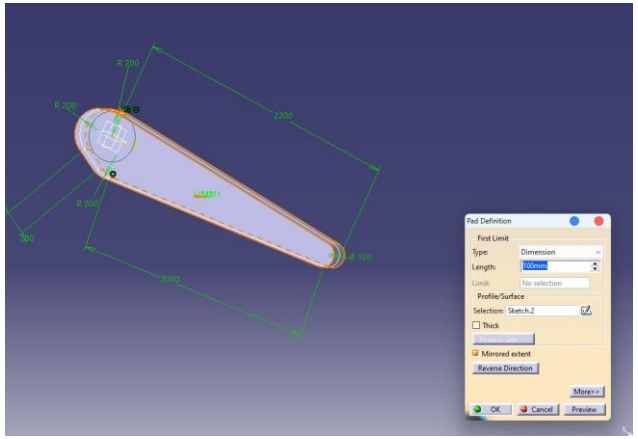


HSLA 550

E (Young's modulus)	ν (Poisson's ratio)	ρ (Density)
200GPa	0.30	$7850kg/m^3$

Boom & Arm Length

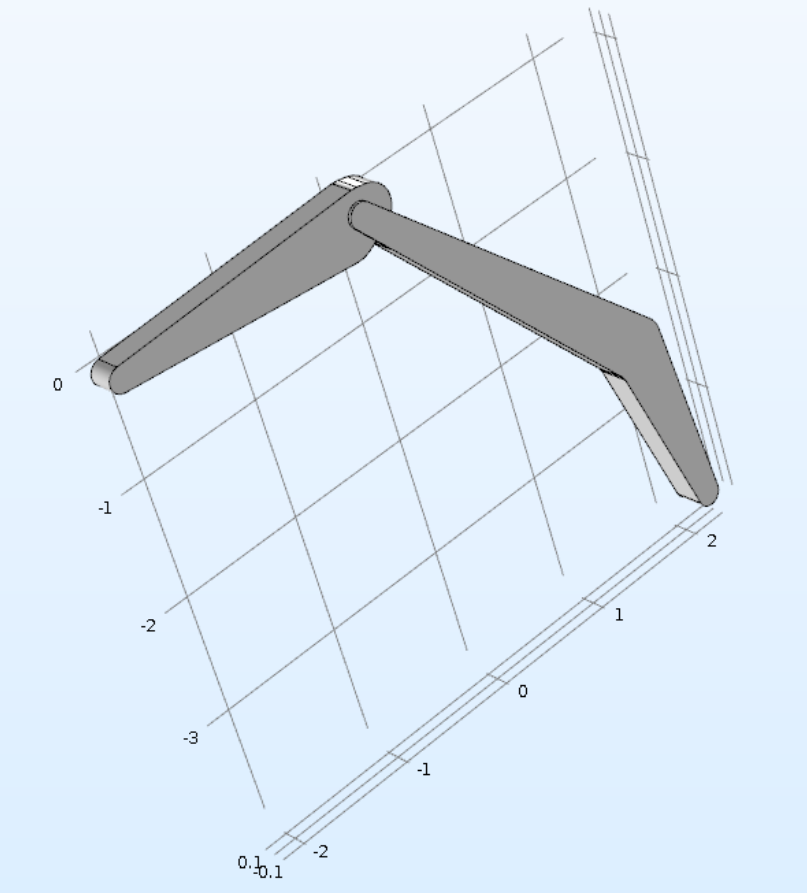




Boom length:4400mm
Thickness:250mm

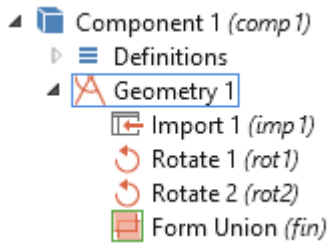
Arm length:2200mm
Thickness: 200mm

COMSOL 해석과정

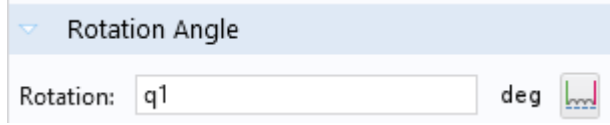


CATIA 파일을 COMSOL로 가져오기

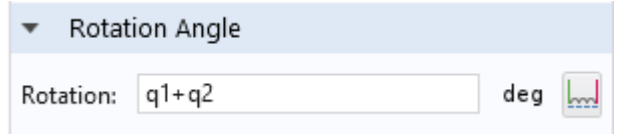
MATLAB, Simulink로부터 q1, q2값을 얻어 Rotate에 입력



Rotate 1

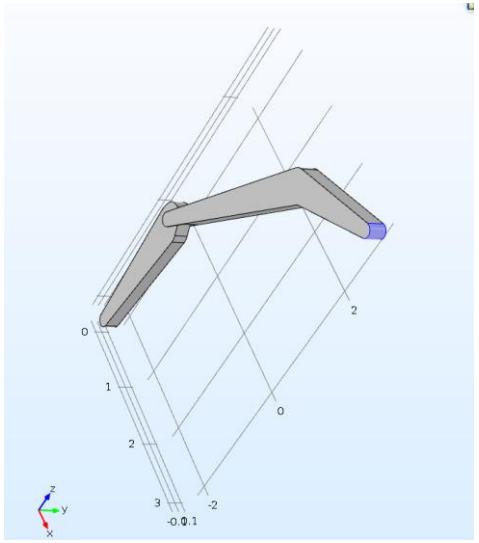


Rotate 2

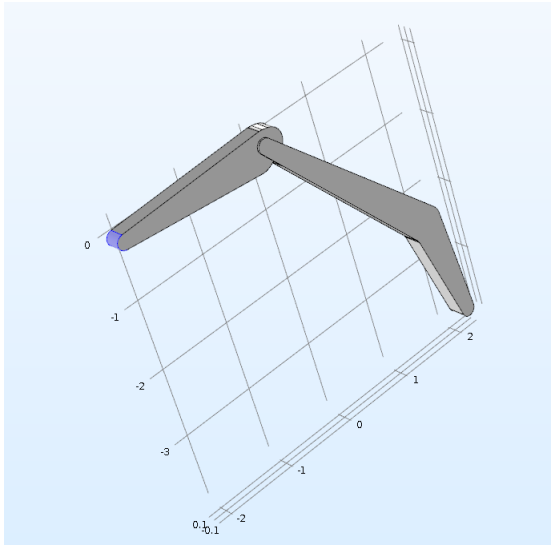


COMSOL 해석과정

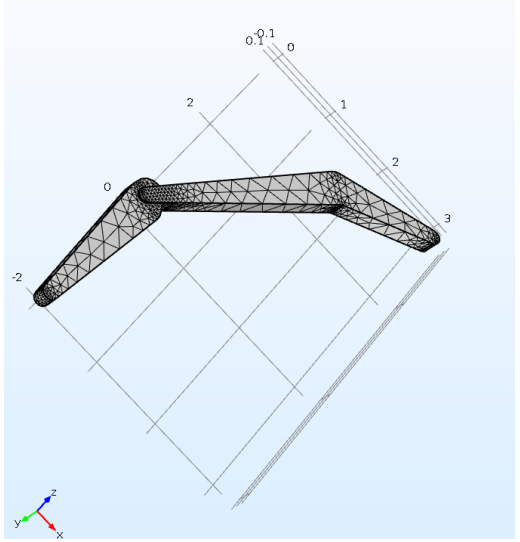
<Fixed constraint>

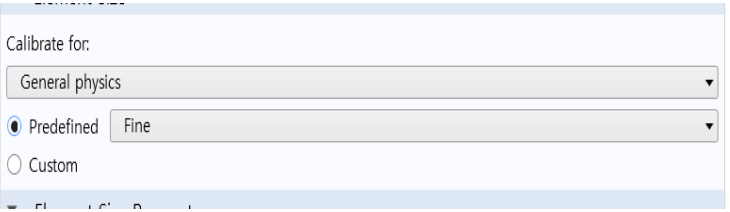


<Boundary load>

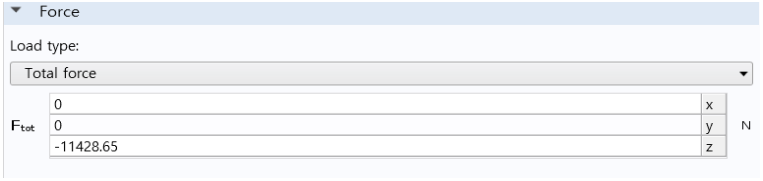


<Mesh>

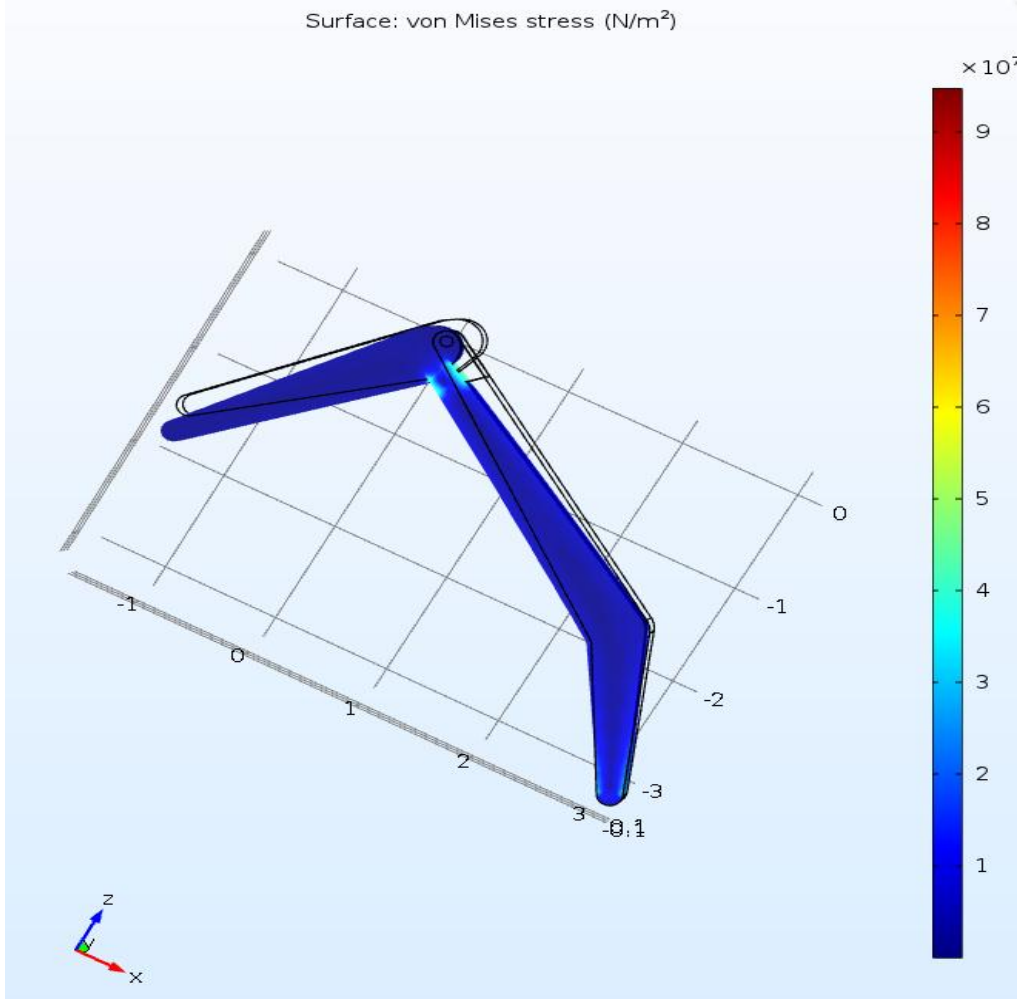




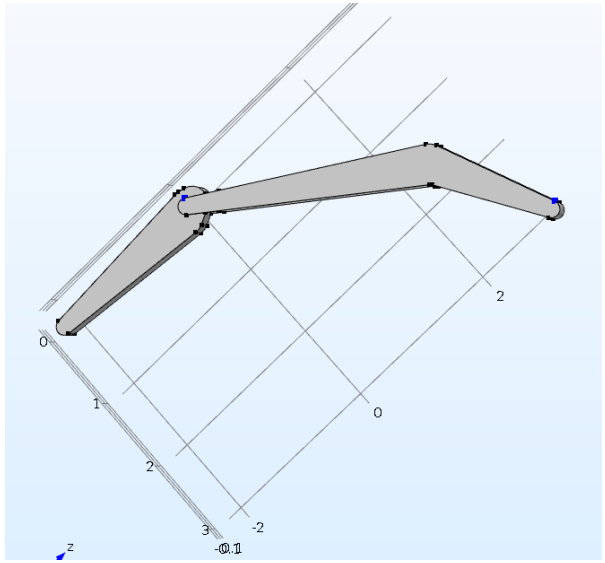
DX140W-5굴삭기 바스켓 부분 (GPT 사용)
모레기준 최대로 담았을 때 중량 사용

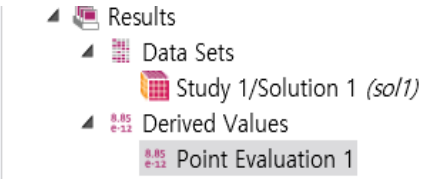


COMSOL 해석과정



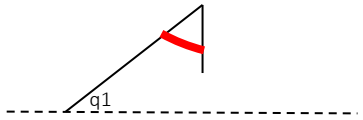
Joint 응력 집중 되는 것을
확인 할 수 있음





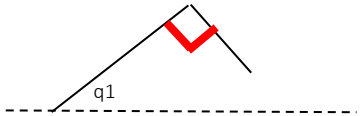
Point evaluation을 사용하여
joint 부분을 확인하기로 결정

COMSOL 해석결과



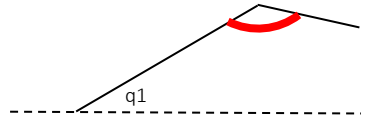
(1) 예각

Time[s]	Joint1 [MPa]	Joint2 [MPa]
3.5	8.96	0.019
4	18.70	0.016
4.9 (T Max)	26.91	0.011
5	36.1	0.003
5.5	22.2	0.005
5.783 (T min)	15.03	0.010



(2) 직각

Time[s]	Joint1 [MPa]	Joint2 [MPa]
3.5	14.7	0.66
4	35.7	0.22
4.896 (T Max)	47.3	0.22
5	47.9	0.69
5.5	36.2	0.97
5.755 (T min)	24.7	1.02

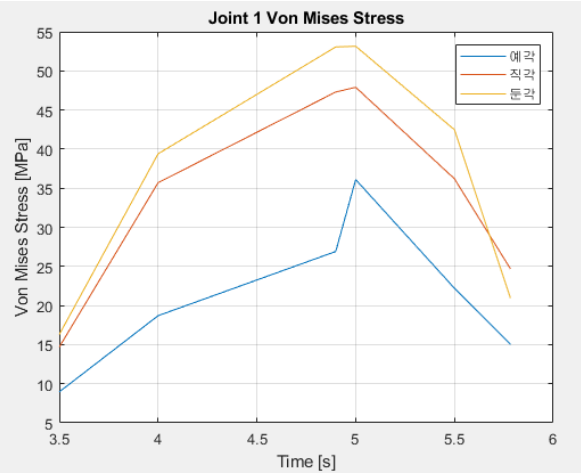


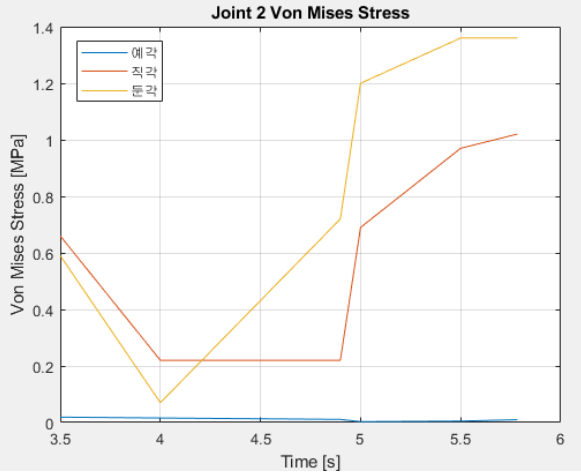
(3) 둔각

Time[s]	Joint1 [MPa]	Joint2 [MPa]
3.5	16.3	0.59
4	39.4	0.07
4.775 (T Max)	53.06	0.72
5	53.15	1.20
5.5	42.47	1.36
5.645 (T Min)	20.94	1.36

해석결과 그래프

Joint1,2시간에 따른 von mises stress 그래프

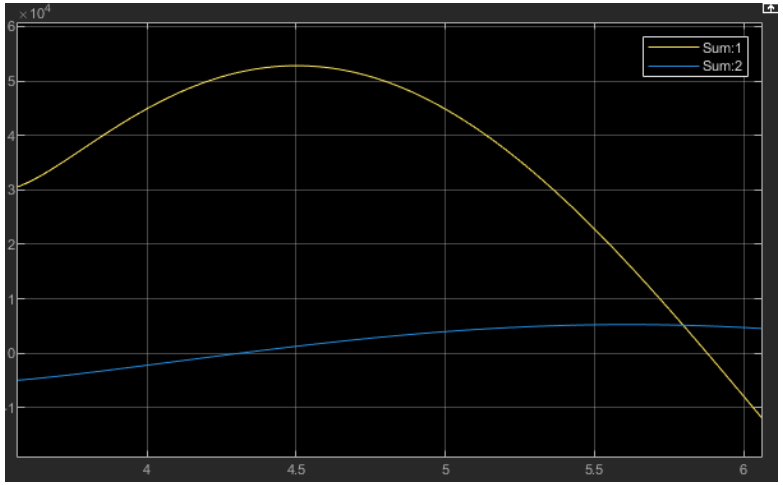




shear stress:

$$\tau = \frac{T \cdot r}{J}$$

Joint1,2 시간에 따른 토크 그래프



Von mises stress:

$$\sigma_v = \sqrt{\sigma_x^2 - \sigma_x \sigma_y + \sigma_y^2 + 3\tau^2}$$

결론

1. 토크증가로 인해 shear stress , principal stress 증가, 그러므로 von mises stress가 토크 그래프와 유사하다.

2. 둔각일 때, 최대 von mises stress가 제일 크다는 것을 알 수 있다. 모멘트 암이 멀어질수록 커진다.

3. 굴삭기 붐과 암의 안정성 평가를 하기 위해 stress가 가장 많이 받고 제일 취약한 부분을 고려 해야 한다.
   Joint부분의 von mises stress가 제일 큰 부분 측정 해야 한다.
   특히, 사이각이 둔각인 상황일 때 고려를 많이 해야 한다.

아쉬웠던 점

1. CATIA로 모델링 시 정확한 제원이 부족하기에 길이와 폭을 간소화하여 비율에 맞게 넣었다.
   따라서, 실제 굴삭기 형태의 붐과 암의 von mises stress를 구하기 어려웠다.

2. 토크가 4.5s일 때 최대지만 von mises stress는 5s일 때 제일 크게 나왔다.
   그 이유는 붐의 형상이 매트랩과 달리 약간 꺾인 형태이므로 그렇게 나타나는 것으로 추정된다.

3. MATLAB과 Simulink를 실행할 때 처음 위치를 조정했음에도 시뮬레이션 과정에서는 조정이 안 된다.
   경로를 따라 이동하는 과정에서의 토크를 구하기 위해 처음 몇 초간 멈추고 움직이도록 설정했다.

참고자료

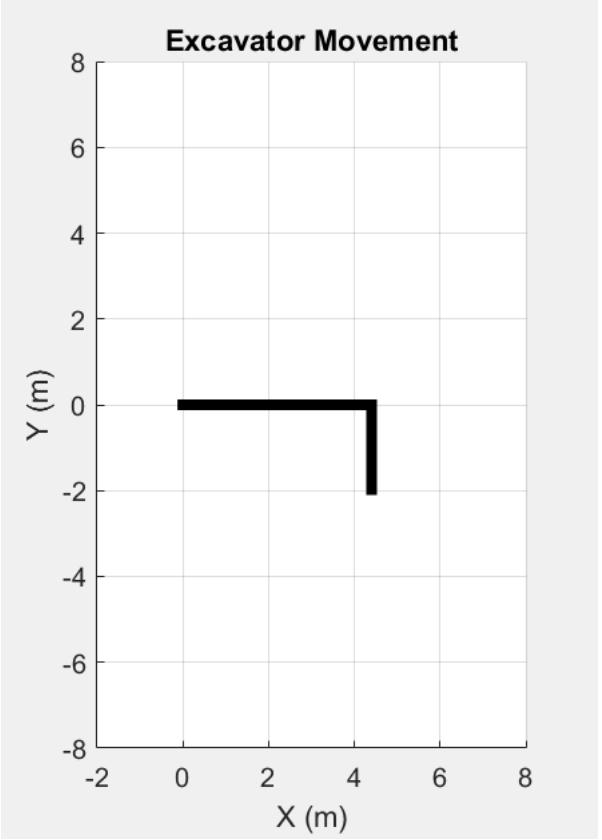
1. 정슬. (2019). 로봇공학. 청문각(교문사).
2. 정슬. (2021). 로봇 시스템 제어. 청문각(교문사).
3. 두산휠굴삭기 DX140W-5 제원.
   1) https://www.ritchiespecs.com/model/doosan-dx140w-5-mobile-excavator
   2) https://blog.naver.com/PostView.nhn?blogId=cetec16&logNo=221088986488
4. 로봇역학 기구학 정리.
   https://m.blog.naver.com/long_bagstrap/223404510524

Q&A

MATLAB 구현

```
%% Excavator 2D planar
figure(1)
axis equal
axis([-2 8 -8 8]);
grid on
hold on
xlabel('X (m)')
ylabel('Y (m)')
title('Excavator Movement');

% Initial Position Of Joints
Joint_1=[0 0];
Joint_2=[4.4 0];
Joint_3=[4.4 -2.1];

% Excavator Arm Settings
Excavator_arm_1=patch('XData',[Joint_1(1) Joint_2(1)],'YData',[Joint_1(2) Joint_2(2)],'LineWidth',4);
Excavator_arm_2=patch('XData',[Joint_2(1) Joint_3(1)],'YData',[Joint_2(2) Joint_3(2)],'LineWidth',4);
```

Excavator_Setttings.m



실행 결과

MATLAB 구현

```
% Set Data to Excavator arms
set(Excavator_arm_1,'xdata',[Joint_1(1) Excavator.X1],'ydata',[Joint_1(2) Excavator.Y1])
set(Excavator_arm_2,'xdata',[Excavator.X1 Excavator.X2],'ydata',[Excavator.Y1 Excavator.Y2])
```

Plot_Excavator.m

- 시뮬레이션 구현하는 동안 실행되는 파일
- 특정 시간에 붐과 암의 위치를 그려준다.

MATLAB 구현

```
%% Excavator Configurtion
Excavator_Settings;
```

```
%% Run Simulation
SimOut = sim('Excavator_Control'); %Run Simulink

figure(1);
Setting Video File
video = VideoWriter('5.5m.mp4', 'MPEG-4');
video.FrameRate = 30;
open(video);

% Simulation Loop
for S = 1 : 1 : size(SimOut,1)
    % Joint 1 Values
    Excavator.X1 = X1_out.signals.values(S);
    Excavator.Y1 = Y1_out.signals.values(S);

    % Joint 2 Values
    Excavator.X2 = X2_out.signals.values(S);
    Excavator.Y2 = Y2_out.signals.values(S);

    % Excavator Plot
    Plot_Excavator

    % Plot path
    plot(X2_out.signals.values(S),Y2_out.signals.values(S),'r.');
    plot(X_input.signals.values,Y_input.signals.values,'--b');
    pause(.001)

    % Save Frame
    frame = getframe(gcf);
    writeVideo(video, frame);
end

close(video);
```

Excavator_Simulation.m

1. 시뮬레이션을 위한 그래프 세팅

2. Simulink 실행

3. Simulink 실행 후, Plot_Excavator를 이용해 시뮬레이션 과정을 그래프로 구현

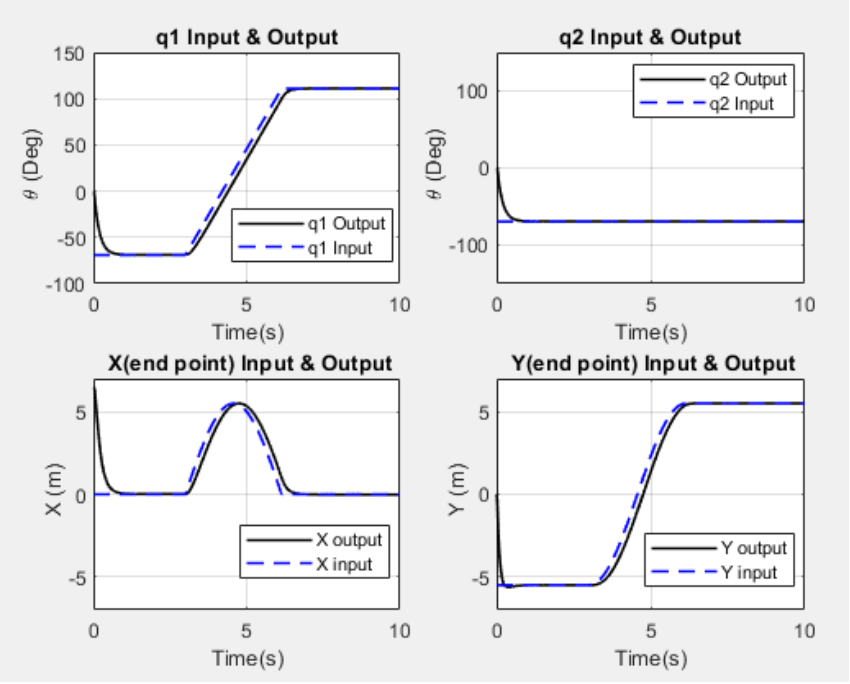
4. 발표 자료를 제작하기 위해 30프레임 mp4파일을 생성

MATLAB 구현

```
%% Plot Data (Comparison Input & Output Signal Value)
figure(2)
subplot(2,2,1);
plot(q_input1.time,rad2deg(q_out1.signals.values),'k',q_input1.time,rad2deg(q_input1.signals.values),'--b','linewidth',1.2)
title('q1 Input & Output');
xlabel('Time(s)');
ylabel(' \theta (Deg)');
axis([0 10 -100 150]);
legend('q1 Output','q1 Input');
grid on

subplot(2,2,2);
plot(q_input2.time,rad2deg(q_out2.signals.values),'k',q_input2.time,rad2deg(q_input2.signals.values),'--b','linewidth',1.2)
title('q2 Input & Output');
xlabel('Time(s)');
ylabel(' \theta (Deg)');
axis([0 10 -150 150]);
legend('q2 Output','q2 Input');
grid on

subplot(2,2,3);
plot(X2_out.time,X2_out.signals.values,'k',X2_out.time,X_input.signals.values,'--b','linewidth',1.2)
title(' X(end point) Input & Output');
xlabel('Time(s)');
ylabel(' X (m)');
axis([0 10 -7 7]);
legend('X output','X input');
grid on

subplot(2,2,4);
plot(Y2_out.time,Y2_out.signals.values,'k',X2_out.time,Y_input.signals.values,'--b','linewidth',1.2)
title(' Y(end point) Input & Output');
xlabel('Time(s)');
ylabel(' Y (m)');
axis([0 10 -7 7]);
legend('Y output','Y input');
grid on
```

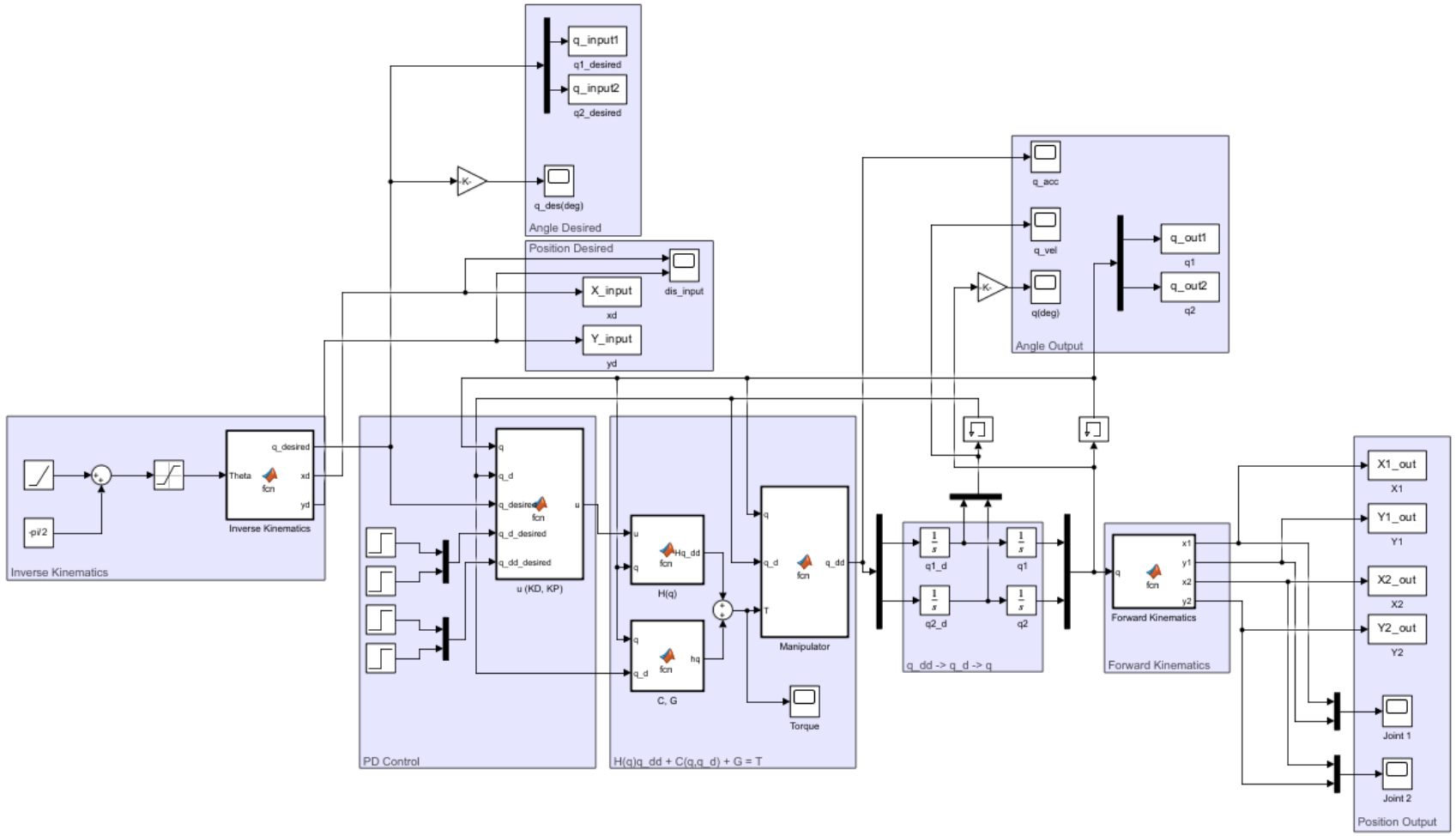
Excavator_Simulation.m

5. PD제어가 제대로 이루어지는지 확인하기 위한 그래프 생성



출력 시 나오는 그래프 예시

Simulink 구현



Excavator_Control.slx

Simulink 구현

- Inverse Kinematics

```
function [q_desired, xd, yd] = fcn(Theta)
%% Parameters
l1 = 4.4; l2 = 2.2;
x0 = 0; y0 = 0;
r = 5.5;

%% Trajectory Design (Circle)
xd = x0 + r*cos(Theta);
yd = y0 + r*sin(Theta);

%% Inverse Kinematic
C = (xd^2 + yd^2 - l1^2 - l2^2) / (2 * l1 * l2);
D = - sqrt(1 - C^2);
q2_desired = atan2(D, C);

q1_part1 = atan2(yd, xd);
q1_part2 = atan2(l2*sin(q2_desired), l1+l2*cos(q2_desired));
q1_desired = q1_part1 - q1_part2;

% Output Theta
q_desired = [q1_desired; q2_desired];
```

1. 끝점의 경로 설정

2. 역기구학으로 각 관절의 각도 구하기

Simulink 구현

- PD Control

```
%% PD Control
function u = fcn(q, q_d, q_desired, q_d_desired, q_dd_desired)
KD = 40;
KP = 200;

q_tilted = q - q_desired;
q_d_tilted = q_d - q_d_desired;

u = q_dd_desired-(KD*(q_d_tilted))-(KP*(q_tilted));
```

$$u = \ddot{q} - K_p e - K_d \dot{e}$$

$$e = q - q_d, \qquad \dot{e} = \dot{q} - \dot{q}_d$$

Simulink 구현

- 동역학 방정식

```
%% H(q)q_dd + C(q,q_d) + G = T
function q_dd = fcn(q, q_d, T)
l1 = 4.4; l2 = 2.2;
m1 = 1500; m2 = 700;
g = 9.81;

H11 = (1/3)*m1*(l1^2)+(1/3)*m2*(l2^2)+m2*(l1^2)+m2*l2*l2*cos(q(2));
H22 = (1/3)*m2*(l2^2)+(1/2)*m2*l1*l2*cos(q(2));
H12 = (1/3)*m2*(l2^2)+(1/2)*m2*l1*l2*cos(q(2));
H21 = (1/3)*m2*(l2^2);

H = [H11 H12; H21 H22];

h1 = - m2*l1*l2*sin(q(2))*q_d(1)*q_d(2) - (1/2)*m2*l1*l2*sin(q(2))*(q_d(2)^2);
h2 = (1/2)*m2*l2*l2*sin(q(2))*(q_d(1)^2);

C = [h1; h2];

g1 = (1/2)*m1*l1*g*cos(q(1)) + (1/2)*m2*l2*g*cos(q(1)+q(2)) + m2*l1*g*cos(q(1));
g2 = (1/2)*m2*l2*g*cos(q(1)+q(2));

G = [g1; g2];

%% Angular Accerlation
q_dd = H \ (T - C - G);
```

$$\boldsymbol{\tau} = \boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q})$$

계산을 통해 $\ddot{\boldsymbol{q}}$의 값을 구할 수 있다.

Simulink 구현

- Forward Kinematics

```
%% Forward Kinematics
function [x1,y1,x2,y2] = fcn(q)
l1 = 4.4; l2 = 2.2;
g = 9.81;

% Joint 2 location
x1 = l1*cos(q(1));
y1 = l1*sin(q(1));

% Final location
x2 = l1*cos(q(1))+l2*cos(q(1)+q(2));
y2 = l1*sin(q(1))+l2*sin(q(1)+q(2));
```

관절 2의 위치

$$x = L_1 cos q_1$$
$$y = L_1 sin q_1$$

끝 점의 위치

$$x = L_1 cos q_1 + L_2 \cos(q_1 + q_2)$$
$$y = L_1 sin q_1 + L_2 \sin(q_1 + q_2)$$

계산을 통해 관절 2와 끝점의 좌표 값을 구할 수 있다.