차 례

역기구학 및 동역학 소프트웨어 개발 학습모듈의 개요

학습모듈의 목표

로봇을 작업 목적지에 이동시키기 위하여 기구학적 모델을 설계하고, 역기구학 해를 구하고, 소프트웨어를 구현할 수 있으며, 로봇 기구를 구성하는 요소 및 구조 부품을 모델링하여 동역학 모델을 설계하고 소프트웨어를 구현할 수 있다.

선수학습

해당사항 없음

학습모듈의 내용체계

학습	학습 내용	NCS 능력단위 요소	
		코드번호	요소 명칭
1. 기구학적 모델링 설계하기	1-1. 기구학적 모델링 설계	1903080316_16v2.1	기구학적 모델링 설계하기
2. 역기구학 소프트웨어 구현하기	2-1. 역기구학 소프트웨어 구현	1903080316_16v2.2	역기구학 소프트웨어 구현하기
3. 동역학적 모델링 설계하기	3-1. 동역학적 모델링 설계	1903080317_16v2.1	동역학적 모델링 설계하기
4. 동역학 소프트웨어 구현하기	4-1. 동역학 소프트웨어 구현	1903080317_16v2.2	동역학 소프트웨어 구현하기

핵심 용어

모델링, 기구학, 역기구학, 동역학 모델링

1-1. 기구학적 모델링 설계

학습 목표

- 역기구학 파라미터 값을 도출할 수 있다.
- 역기구학 해 구하는 여러 가지 방법 중 적절한 방법을 선정할 수 있다.
- 선정된 방법에 따라 역기구학 해를 구할 수 있다.
- 구한 역기구학 해가 정확한지 검증할 수 있다.

필요 지식 /

1 로봇 기구학 및 역기구학

1. 로봇 순기구학(forward kinematics)

로봇 관절의 각도나 동작으로부터 끝점의 위치나 동작을 구하는 것을 순기구학이라 한다. 즉, 순기구학은 각 관절의 값이 정해지면 끝점의 위치를 어떻게 구할 것인가 하는 것이다. 간단한 예로 아래와 같이 평면에서 2개의 관절을 갖는 매니퓰레이터를 생각해 보자. 먼저 로봇 관절의 각도가 만드는 관절좌표계와 로봇의 베이스에 원점을 두는 직교좌표계(기준좌표계)를 정의한다. 2자유도의 로봇에서 2개의 관절 각도를 갖는 경우, 이를 (θ_1, θ_2)라고 하면 로봇의 끝점의 좌표 (x, y)를 계산할 수 있다. 즉, 순기구학은 관절각도로부터 로봇 끝점의 위치를 구하는 공식을 만드는 방법이다.

[그림 1-1]과 같은 2관절 매니퓰레이터(manipulator)에서 링크의 길이를 l_1, l_2라 하고, 링크가 이루는 각도를 각각 θ_1, θ_2라고 하면, 끝점의 좌표 x, y는 기하학적으로 유도해서 다음과 같은 기구학식으로 나타낼 수 있다.

$$x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \quad \langle 1\text{-}1 \rangle$$

$$y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

위의 식을 시간 t로 미분하면 즉, 아래와 같이 편미분하면 다음과 같이 끝점의 속도식을 구할 수 있다.

$$\frac{dx}{dt}=\frac{\partial x}{\partial \theta_1}\frac{\partial \theta_1}{\partial t}+\frac{\partial x}{\partial \theta_2}\frac{\partial \theta_2}{\partial t} \qquad \langle 1\text{-}2 \rangle$$

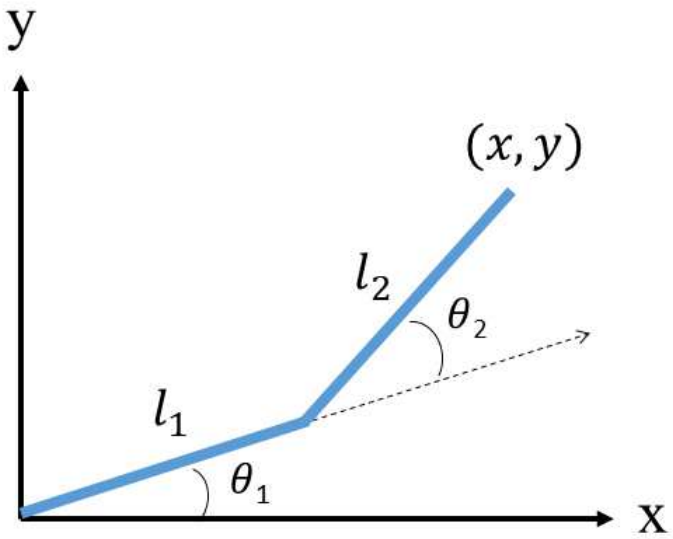
$$\frac{dy}{dt}=\frac{\partial y}{\partial \theta_1}\frac{\partial \theta_1}{\partial t}+\frac{\partial y}{\partial \theta_2}\frac{\partial \theta_2}{\partial t}$$

$$\dot{x}=\{-l_1\sin\theta_1-l_2\sin(\theta_1+\theta_2)\}\dot{\theta}_1+\{-l_2\sin(\theta_1+\theta_2)\}\dot{\theta}_2 \qquad \langle 1\text{-}3 \rangle$$

$$\dot{y}=\{l_1\cos\theta_1+l_2\cos(\theta_1+\theta_2)\}\dot{\theta}_1+\{l_2\cos(\theta_1+\theta_2)\}\dot{\theta}_2$$

위의 식을 행렬식으로 표현하면 다음과 같다. 여기에서 J를 자코비안 행렬이라고 부른다.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}=\begin{bmatrix}-l_1\sin\theta_1-l_2\sin(\theta_1+\theta_2) & -l_2\sin(\theta_1+\theta_2)\\ l_1\cos\theta_1+l_2\cos(\theta_1+\theta_2) & l_2\cos(\theta_1+\theta_2)\end{bmatrix}\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{bmatrix}=J\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{bmatrix} \qquad \langle 1\text{-}4 \rangle$$



[그림 1-1] 2관절 매니퓰레이터의 순기구학

이상과 같은 방법으로 [그림 1-2]와 같은 3관절 매니퓰레이터의 기구학식과 속도식을 구하면 다음과 같다.

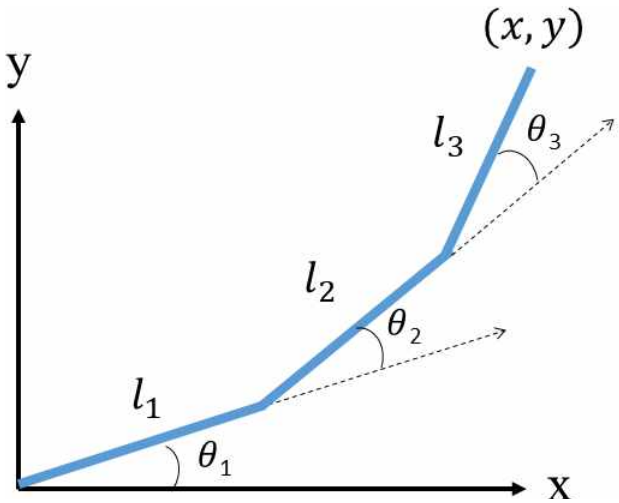
$$x=l_1\cos\theta_1+l_2\cos(\theta_1+\theta_2)+l_3\cos(\theta_1+\theta_2+\theta_3) \qquad \langle 1\text{-}5 \rangle$$

$$y=l_1\sin\theta_1+l_2\sin(\theta_1+\theta_2)+l_3\sin(\theta_1+\theta_2+\theta_3)$$

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}=\begin{bmatrix}-l_1\sin\theta_1-l_2\sin(\theta_1+\theta_2)-l_3\sin(\theta_1+\theta_2+\theta_3)\\ l_1\cos\theta_1+l_2\cos(\theta_1+\theta_2)+l_3\cos(\theta_1+\theta_2+\theta_3)\end{bmatrix} \quad \langle 1\text{-}6\rangle$$

$$\begin{bmatrix}-l_2\sin(\theta_1+\theta_2)-l_3\sin(\theta_1+\theta_2+\theta_3) & -l_3\sin(\theta_1+\theta_2+\theta_3)\\ l_2\cos(\theta_1+\theta_2)+l_2\cos(\theta_1+\theta_2+\theta_3) & l_3\cos(\theta_1+\theta_2+\theta_3)\end{bmatrix}\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{bmatrix}=J\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{bmatrix} \quad \langle 1\text{-}7\rangle$$

여기서 자코비안 행렬 J는 2×3 행렬이 되는데, 이는 여유자유도를 갖는 매니퓰레이터의 특징이다. 끝점의 위치는 2개의 변수이고 매니퓰레이터에서 조정 가능한 변수는 3개이므로 1개 더 여유가 있다. 그 결과 끝점의 위치를 변화시킬 수 있는 방법은 관절 각도 3개의 조합에 의해 수없이 많이 존재한다.



[그림 1-2] 3관절 매니퓰레이터의 순기구학

2. 로봇 역기구학(inverse kinematics)

끝점의 위치나 동작으로부터 각 관절의 위치나 각도를 구하는 것을 역기구학이라 한다. 일반적으로 역기구학의 해는 수식으로 직접 구하기가 곤란한 경우가 많고 해가 여러 개 존재하기도 한다. 그 때에는 서로 인접하는 관절의 배치에 주의하면서 컴퓨터를 활용해서 근사 계산을 반복한다.

[그림 1-3]과 같은 2관절 매니퓰레이터(manipulator)에서 링크의 길이를 l_1, l_2라 하고, 끝점의 좌표를 x, y라 하면, 링크가 이루는 각도 θ_1, θ_2는 기하학적으로 다음과 같이 유도할 수 있다. 끝점의 좌표 x, y가 주어지면 원점으로부터의 거리를 d라고 정의하고 피타고라스 정리를 이용하여 다음과 같이 구할 수 있다.

$$d=\sqrt{x^2+y^2} \quad \langle 1\text{-}8\rangle$$

코사인 법칙에 의해 l_1, l_2, d를 알기에 링크 간의 각도 ϕ를 다음과 같이 구할 수 있다.

$$d^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos\phi \qquad \langle 1\text{-}9 \rangle$$

$$\cos\phi = \frac{l_1^2 + l_2^2 - d^2}{2l_1 l_2}$$

관절 간의 각도 θ를 이용하면, θ_2를 다음과 같이 구할 수 있다.

$$\cos\phi = \cos(180 - \theta_2) = -\cos\theta_2 \qquad \langle 1\text{-}10 \rangle$$

$$\cos\theta_2 = \frac{d^2 - l_1^2 - l_2^2}{2l_1 l_2}$$

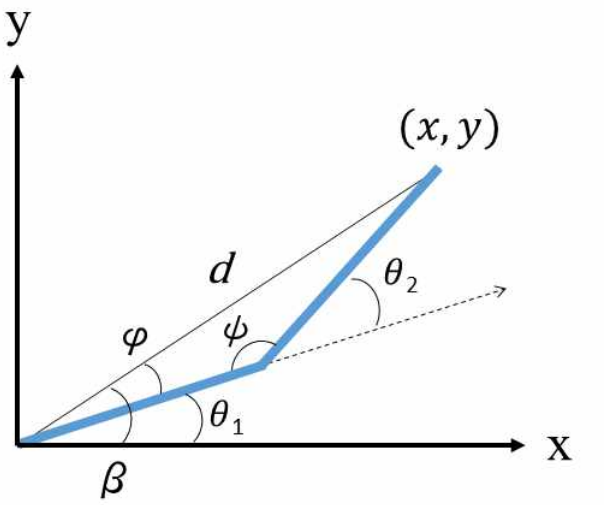
$$\theta_2 = \cos^{-1}\frac{d^2 - l_1^2 - l_2^2}{2l_1 l_2}$$

코사인 법칙에 의해 l_1, l_2, d를 알기에 l_1, d 간의 각도 ψ를 다음과 같이 구할 수 있다. 2관절 매니퓰레이터에서 역기구학의 해는 링크의 위치에 따라 두 가지 해가 존재한다.

$$l_2^2 = d^2 + l_1^2 - 2dl_1 \cos\psi \qquad \langle 1\text{-}11 \rangle$$

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

$$\theta_1 = \beta \pm \psi = \tan^{-1}\frac{y}{x} - \cos^{-1}\frac{d^2 + l_1^2 - l_2^2}{2dl_1} \quad \begin{pmatrix} + : elbow\,up \\ - : elbow\,down \end{pmatrix}$$



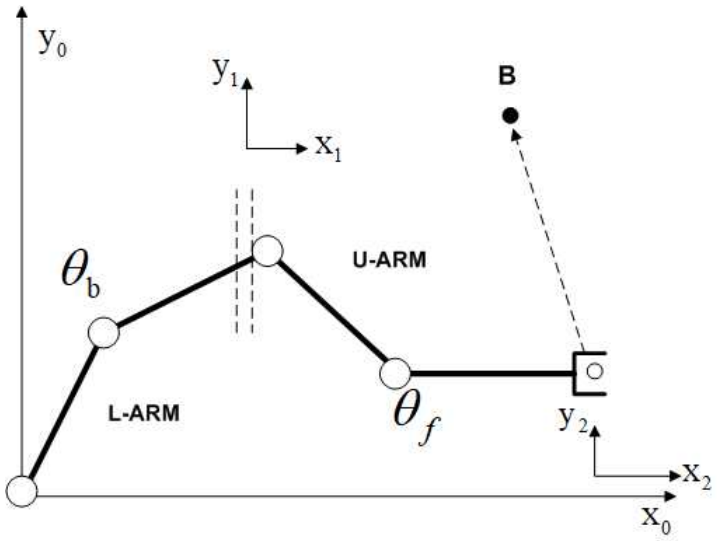
[그림 1-3] 2관절 매니퓰레이터의 역기구학

2 협동로봇의 기구학

앞 절에서는 평면에서 운동하는 2관절의 매니퓰레이터를 예로 설명하였으나, 실제 작업환경에서는 3차원 공간에서 6개의 운동성분에 의하여 임의의 일이 수행될 수 있으므로, 6축 혹은 6자유도를 갖는 로봇은 그 자체로서 한 가지 일을 수행하는 목적을 달성하는 것만을 고려하게 된다. 그러나 로봇이 7축 이상의 잉여 자유도를 갖는 경우, 또는 여러 개의 로봇이 한 가지 일을 수행하는 경우를 생각해 보자.

1. 직렬형 잉여자유도의 로봇 순기구학(forward kinematics of redundant manipulator)

문제를 간단화하기 위하여 평면 작업만을 고려하기로 하자. [그림 1-4]에서 보듯이 평면상의 임의의 점으로 로봇을 이동하기 위해서는 2자유도만 있으면 된다. 이때 4자유도를 갖는 로봇을 상부(U-ARM)와 하부(L-ARM)로 나누어 생각하면, 2개의 U-ARM 및 L-ARM의 협동에 의하여 X-Y 평면상에 임의의 점으로 이동할 수 있음을 알 수 있다.



[그림 1-4] 직렬형 협동제어

그림에서 A→B로 이동하는 운동을 $\dot{X}$으로 표시하면 다음과 같이 정의된다.

$$\begin{aligned}\dot{X} &= J_b\dot{\theta}_b + J_f\dot{\theta}_f \\ &= \begin{bmatrix} J_b & J_f \end{bmatrix}\begin{bmatrix}\dot{\theta}_b \\ \dot{\theta}_f\end{bmatrix} \\ &= J\dot{\theta}\end{aligned} \qquad \langle 1\text{-}12 \rangle$$

${}^{o}\dot{X}_1$ 및 ${}^{1}\dot{X}_2$ 국부적으로 발생되는 운동으로 다음과 같이 표시된다.

$$ {}^{o}\dot{X}_1 = \begin{bmatrix} {}^{o}v_1 \\ {}^{o}w_1 \end{bmatrix} = \begin{bmatrix} {}^{o}J_1^{v} \\ {}^{o}J_1^{w} \end{bmatrix} \begin{bmatrix} \dot{\theta}_b \end{bmatrix} \qquad \langle 1\text{-}13 \rangle $$

$$ {}^{1}\dot{X}_2 = \begin{bmatrix} {}^{1}v_2 \\ {}^{1}w_2 \end{bmatrix} = \begin{bmatrix} {}^{1}J_2^{v} \\ {}^{1}J_2^{w} \end{bmatrix} \begin{bmatrix} \dot{\theta}_f \end{bmatrix} \qquad \langle 1\text{-}14 \rangle $$

이러한 국부적인 운동 End-effector에서 운동 $\dot{X}$로 변환되기 위해서는 좌표계 {1} 및 좌표계 {2} 사이의 벡터 P 및 회전행렬 ${}^{o}R_1$이 필요하다.
이에 의해 직렬형 잉여 다관절로봇의 운동은 다음과 같이 구할 수 있다.

$$ \dot{X} = \begin{bmatrix} {}^{0}v_1 + {}^{0}w_1 \times P + {}^{0}R_1\,{}^{1}v_2 \\ {}^{0}w_1 + {}^{0}R_1\,{}^{1}w_2 \end{bmatrix} \qquad \langle 1\text{-}15 \rangle $$

$$ = \begin{bmatrix} {}^{0}J_1^{v}\theta_b + {}^{0}J_1^{w}\theta_b \times P + {}^{0}R_1\,{}^{0}J_2^{v}\theta_f \\ {}^{0}J_1^{v}\theta_b + {}^{0}R_1^{v}\,{}^{1}J_2^{w}\theta_f \end{bmatrix} $$

$$ \triangleq \; J_b\dot{\theta}_b + J_f\dot{\theta}_f $$

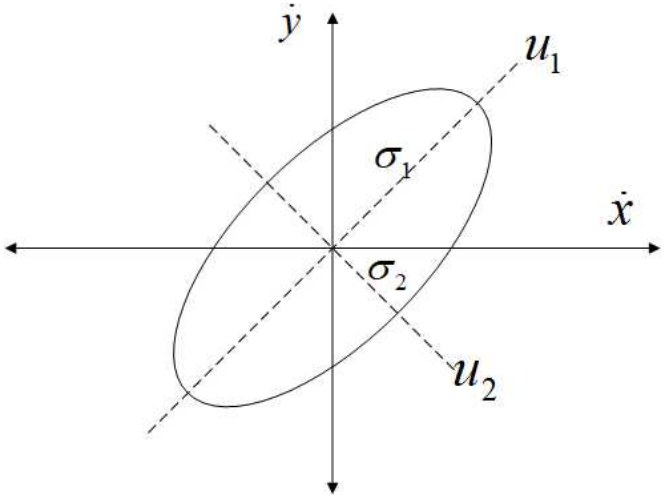
주어진 일이 $\dot{X}_d$로 정해져 있을 때, 이를 만족시키는 $\dot{\theta}_b$ 및 $\dot{\theta}_f$를 구하는 것을 생각해보자. 이를 역으로 해석하면, $\dot{\theta}_b$ (L-ARM운동) 및 $\dot{\theta}_f$ (U-ARM운동)의 협동에 의하여 $\dot{X}_d$를 발생시키는 문제로 해석할 수 있다.
$\dot{X}_d$를 $J_b\dot{\theta}_b$와 $J_f\dot{\theta}_f$로 이루는 문제를 해석하기 위해 $\dot{X} = J\dot{\theta}$의 모양에서 $\|\dot{\theta}\| = 1$의 운동에 의해 $\dot{X}$는 얼마만큼 발생될 수 있는가를 관찰한다.

$$ \dot{\theta}^{T}\dot{\theta} = \|\dot{\theta}\|^2 \qquad \langle 1\text{-}16 \rangle $$

$$ 1 = (J^{-1}\dot{X})^{T}J^{-1}\dot{X} $$

$$ 1 = \dot{X}^{T}(JJ^{T})^{-1}\dot{X} $$

식 <1-16>은 $(JJ^{T})^{-1}$값이 주어진 θ에 대해 상수 값으로 될 때 하나의 타원식으로 표현할 수 있다.
$\dot{X} = \begin{bmatrix} \dot{x}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$로 표시할 때 식 <1-16>은 아래의 그림과 같이 표시된다.



[그림 1-5] 직렬형 협동제어

[그림 1-5]의 $u_1, \sigma_1, u_2, \sigma_2$ 는 자코비안, J의 특이치 분해에 의해 얻어진다. 즉,

$$J = USV^T \qquad \langle 1\text{-}17 \rangle$$

로 SVD에 의해 구해지며 $U = [u_1, \cdots, u_n]$, $S = diag[\sigma_1 , \cdots , \sigma_n]$으로 구해진다.

[그림 1-5]를 보면서 해석하면 $\|\dot{\theta}\| = 1$의 운동에 의해 직교공간에서 u_1방향으로는 σ_1만큼, u_2방향으로는 σ_2만큼 운동이 발생될 수 있다는 것을 의미한다.

$\dot{X}_d$를 $J_b\dot{\theta}_b$와 $J_f\dot{\theta}_f$으로 나눠서 표현할 수 있는데, 다음과 같은 과정을 거쳐서 표시되며 결국 K_1 및 K_2를 최적으로 설정하는 문제로 귀결된다.

$$J_b = U_1 S_1 V_1^T$$

$$J_f = U_2 S_2 V_2^T$$

$$\dot{X}_d = U_1 K_1 + U_2 K_2 \qquad \langle 1\text{-}18 \rangle$$

문제의 간략화를 위해 $K_1 = \begin{bmatrix} k_{11} \\ k_{12} \end{bmatrix}$, $K_2 = \begin{bmatrix} k_{21} \\ k_{22} \end{bmatrix}$로 표현할 수 있고, $U_1 = [u_{11} \;\; u_{12}]$, $U_2 = [u_{21} \;\; u_{22}]$로 간주할 수 있다.

$\dot{X}_d$의 2차원 벡터를 실현하기 위해서는 일반적으로 2개의 벡터만 있으면 될 것이다. 그러

나 앞에서 기술한 것과 같이 $U_{11}, U_{12}, U_{21}, U_{22}$ 의 4개의 벡터를 가지고 있다. 따라서 최적의 K_1, K_2를 정하기 위해 비용함수를 아래와 같이 정의한다.

$$G = \frac{\alpha}{2}\sum_{i=1}^{2}(k_{1i}/\sigma_{1i})^2 + \frac{(1-\alpha)}{2}\sum_{i=1}^{2}(k_{2i}/\sigma_{2i})^2 \qquad \langle 1\text{-}19 \rangle$$

σ값이 큰 방향으로 큰 k 값을 배정하는 것이 유리하다는 원칙에 입각하고 있음을 알 수 있다.
또 α값$(0 \leqq \alpha \leqq 1)$의 배정에 의하여 U-ARM에 일을 많이 줄 것인가 B-ARM에 일을 많이 배정할 것인가를 결정해 준다.
식 <1-19>는 다시 다음 식으로 표기할 수 있다.

$$G \triangleq \frac{1}{2}[K_1^T \quad K_2^T]\, diag[W_{11}, W_{12}, W_{13}, W_{14}]\begin{bmatrix}K_1\\K_2\end{bmatrix} \qquad \langle 1\text{-}20 \rangle$$

$W_{11} = \alpha/\sigma_{11},\ W_{12} = \alpha/\sigma_{12},\ W_{21} = (1-\alpha)/\sigma_{21},\ W_{22} = (1-\alpha)/\sigma_{22}$ 로 된다.
여기서, 식 <1-18>은 다시 다음과 같이 표기된다.

$$\dot{X} = UK \qquad \langle 1\text{-}21 \rangle$$

단, $U = [u_1 \quad u_2],\ K = [K_1^T \quad K_2^T]$
최적화 문제로서 다음과 같이 Lagrangian함수를 정의해 두자.

$$L(K) = G(K) + \lambda^T F(K) \qquad \langle 1\text{-}22 \rangle$$

단, $F(K) = \dot{X} - UK$
식 <1-22>에서 $\frac{\partial L}{\partial K} = 0$를 만족하는 조건을 구하기 위해 다음 식을 풀면 된다.

$$U^T\lambda = WK \qquad \langle 1\text{-}23 \rangle$$

$$\left(\frac{\partial G(K)}{\partial K} = WK, \frac{\lambda^T F(K)}{\partial K} = U^T\lambda\right)$$

식<1-23>에서 차원은 $U=[u_{11},u_{12},u_{21},u_{22}]$로 2×4의 차원을 갖고 있으므로 U^T는 4×2, W 는 4×4, λ 는 2×1, K는 4×1 이다. 식 <1-23>을 따라서 2개의 독립된 식을 찾아서 분할하면 다음과 같다.

$$U_m^T = W_m K \qquad \text{<1-24>}$$

$$\overline{U_m}^T \lambda = \overline{W_m} K \qquad \text{<1-25>}$$

식 <1-24>에서 $\lambda = \overline{U_m}^T W_m K^T$ 로 구해지며, 이를 식 <1-25>에 대입하면 다음 식이 얻어진다.

$$\overline{U_m}^T \overline{U_m}^T W_m K = \overline{W_m} K \qquad \text{<1-26>}$$

식 <1-26>과 $\dot{X}= UK$ 을 사용하면 K는 최적의 고유값을 얻을 수 있다.
이상에서 $\dot{X}_d$ 를 $K_1^T U_1$ 및 $K_2^T U_2$로 최적 분할하였다. 이에 대응하는 Joint공간에서의 속도 즉, $\dot{\theta}_f$ 및 $\dot{\theta}_b$는 다음과 같이 구해진다.

$$U_1 K_1 = U_1 S_1 V_1^T \dot{\theta}_f \ (= J_f \dot{\theta}_f) \qquad \text{<1-27>}$$

$$U_2 K_2 = U_2 S_2 V_2^T \dot{\theta}_b \ (= J_b \dot{\theta}_b) \qquad \text{<1-28>}$$

식 <1-27>, <1-28>에서 다음과 같을 구할 수 있다.($VV^T = I$)

$$\dot{\theta}_{f_{optimal}} = V_1 S_1^T K_1 \qquad \text{<1-29>}$$

$$\dot{\theta}_{b_{optimal}} = V_2 S_2^T K_2 \qquad \text{<1-30>}$$

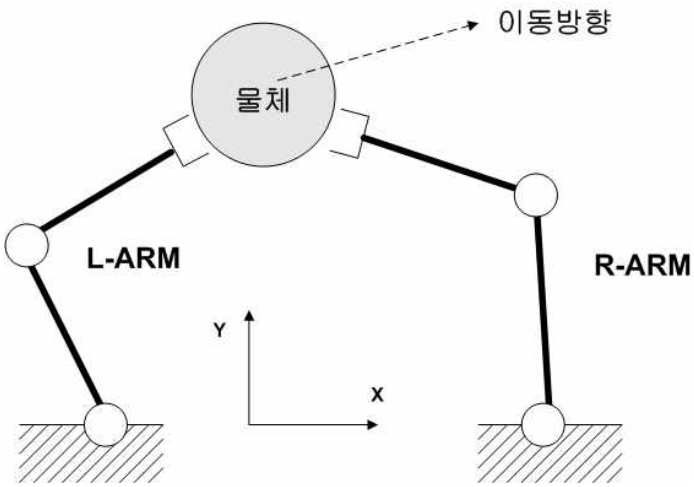
결국, 평면상의 한 점 A로부터 B로 로봇의 End-effector를 옮기는 문제는 비용함수를 최소화하는 방향으로 일을 배정하여 2개의 로봇 차원에서 보면 2개의 직렬로 연결된 로봇이 협동하여 수행하게 되는 것이다.

2. 병렬로봇의 협동제어(forward kinematics of parallel manipulator)

병렬로 2개의 로봇이 작업하는 경우는 두 가지 경우로 다시 나누어 생각할 수 있다.

(1) 기구학적 제약 협동(동역학적인 측면에서는 협조협동)

이는 2개의 로봇의 End-effector가 하나의 무거운 물체를 이동시키는 경우로 아래 그림으로 이해할 수 있다.



[그림 1-6] 제약 협동제어

무거운 물체를 움직임에 있어서 상호 협조를 고려하여 이동하기 쉬운 방향을 찾아서 옮기는 것이 바람직할 것이다. 예를 들어 가상적으로 적군의 로켓이 아군의 이동 물체를 파괴하고자 한다고 가정했을 때, 즉각적으로 회피하기 위하여 어떤 한 방향으로 움직여서 피해야 할 것이다. 이 경우 2개의 로봇이 모두 움직이기 용이한 방향으로 움직여야 할 것이다.

L-ARM의 운동과 R-ARM의 운동은 각각 다음과 식으로 나타낼 수 있다.

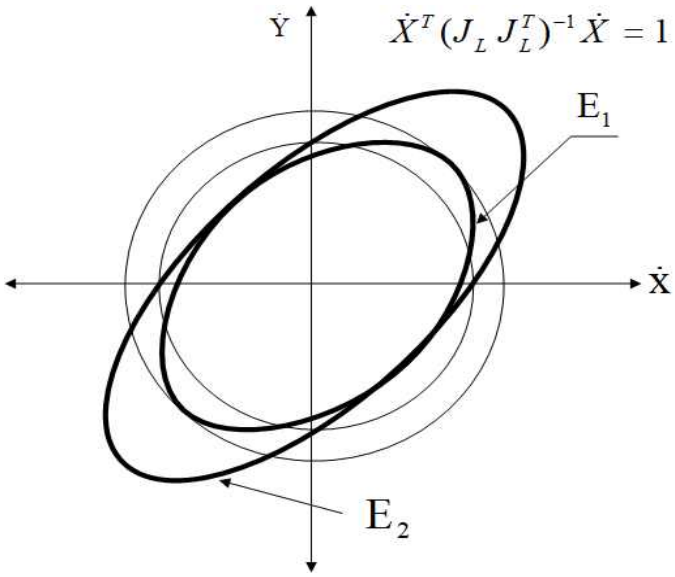
$$\dot{X}_L = J_L\dot{\theta}_L \qquad \langle 1\text{-}31 \rangle$$

$$\dot{X}_R = J_R\dot{\theta}_R$$

이 각각의 로봇에 대한 방향조작성 타원(manipulability ellipsoid) 즉, $\|\dot{\theta}_L\| = 1$, $\|\dot{\theta}\| = 1$에 의해 구해지는 타원 2개를 한 평면에 그려보면 다음과 같다.

그림에서 2개의 타원의 교차 영역을 원 타원의 주축을 사용하여, E_1 및 E_2로 근사화할 수 있고, 그 중 면적이 작은 것을 제약타원으로 사용하여 제약타원의 장축 방향으로 이동시켜주는 것이 바람직할 것이다.

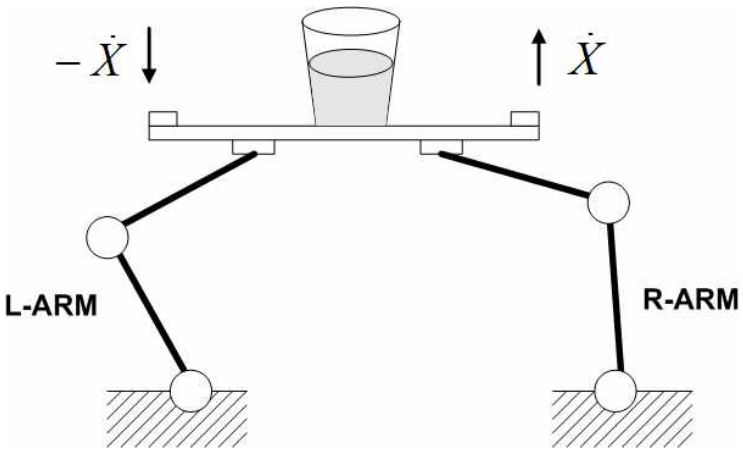
이 문제를 삽입(Insertion) 문제로 돌리면 특정 방향으로 움직이기 위해 로봇의 초기 자세를 최적으로 만들어주는 것으로 해석할 수 있다.



[그림 1-7] 제약 타원

(2) 기구학적 협조협동

이는 2개의 로봇이 하나의 목적을 달성하면서 일을 분담하는 경우이다. 아래 [그림 1-8]으로 이해할 수 있다.



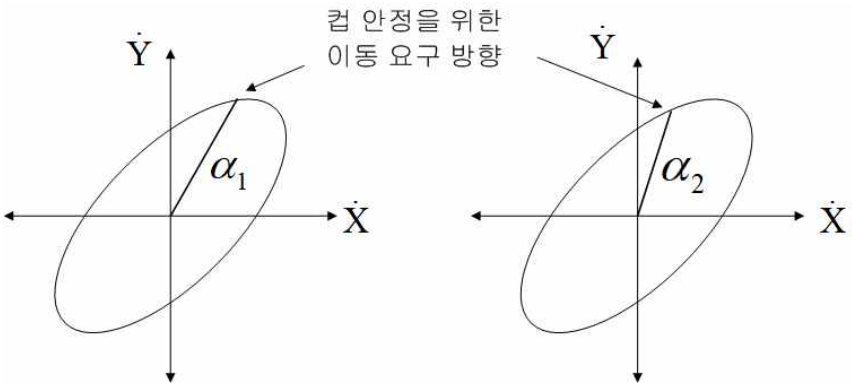
[그림 1-8] 협조 협동제어

위의 그림에서 L-ARM을 내리는 것과 R-ARM을 올리는 것 모두가 물 컵을 안정하게 만드는 목적을 달성할 수 있을 것이다. 이를 위하여 로봇의 운동 $\dot{X}$을 L-ARM 및 R-ARM에 분해해 줄 수 있다.

$$\dot{X}_R = J_R\dot{\theta}_R \qquad \langle 1\text{-}32 \rangle$$

$$\dot{X}_L = J_L\dot{\theta}_L$$

로 표시된 두 로봇의 단위 Joint공간 운동에 대한 직교 좌표 공간운동은 역시 타원으로 표시된다.



(a) L-ARM M 타원 (b) R-ARM M 타원

[그림 1-9] 방향 조작성 타원에 의한 분배

이 α_1 및 α_2의 길이에 따라 $\dot{X}$을 분배하여 지정해 주면 될 것이다.

수행 내용 / 로봇의 운동 궤적 그리기

재료・자료

- 요구사양서
- 로봇 운동학 해석 보고서
- 로봇 하드웨어 매뉴얼
- 로봇 동역학 분석 보고서
- 로봇 작업 시나리오

기기(장비 · 공구)

- 컴퓨터, 프린터
- 로봇 시뮬레이션 프로그램(MATLAB 등)

안전 · 유의 사항

- 로봇 시뮬레이션을 수행할 때에는 범용적으로 사용되는 소프트웨어를 선택해야 한다.

수행 순서

1 2관절 매니퓰레이터의 끝점을 계산한다.

1. 2관절 매니퓰레이터의 끝점의 위치에 관한 순기구학 식의 유도 과정을 이해한다.

$$x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$
$$y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \qquad \langle 1\text{-}1 \rangle$$

2. MATLAB을 이용하여 끝점의 위치를 파악할 수 있는 프로그램을 작성한다.

(1) [그림 1-1]과 같이 링크의 길이가 각각 $l_1 = 0.5$m, $l_2 = 0.4$m 인 2관절 매니퓰레이터에서, θ_1을 45°로 고정시킨 상태에서 θ_2가 -30°일 때 매니퓰레이터의 끝단의 위치를 계산해 본다.

(2) 2관절 매니퓰레이터의 운동에 관한 MATLAB 프로그램은 다음과 같다.

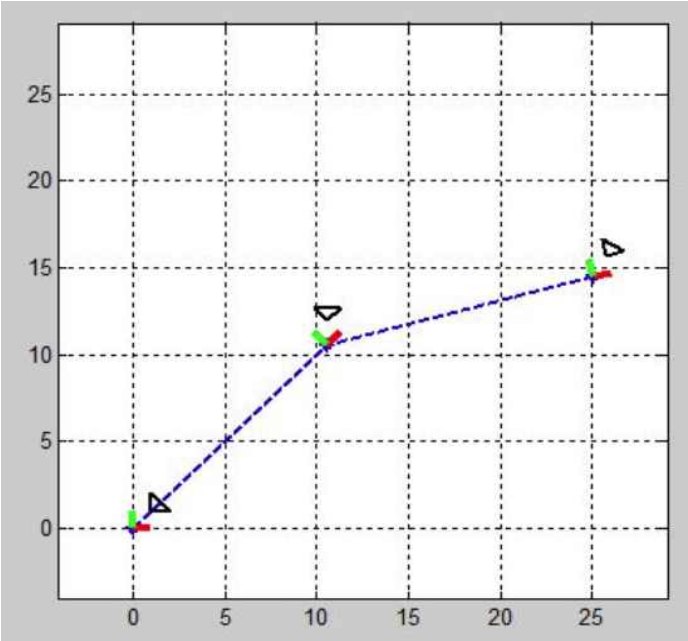
```
theta1 = 45*pi/180;
theta2 = -30*pi/180;

l1 = 0.5;
l2 = 0.4;

RotZ = @(theta) [cos(theta) -sin(theta) 0 0; sin(theta) cos(theta) 0 0; 0 0 1 0 ; 0 0 0 1];
Trans = @(x, y, z) [1 0 0 x; 0 1 0 y; 0 0 1 z; 0 0 0 1];

T_0_1 = RotZ(theta1)*Trans(a1, 0, 0);
T_1_2 = RotZ(theta2)*Trans(a2, 0, 0);
T_total = T_0_1*T_1_2;

y0 = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
y0_1 = T_0_1 * y0;
y0_2 = T_total * y0;
```

(3) 2관절 매니퓰레이터의 운동에 관한 MATLAB 프로그램은 다음과 같다.

```
theta1 = 45*pi/180;
theta2 = -30*pi/180;

l1 = 0.5;
l2 = 0.4;

% DH Table
%       |  theta    d      a      alpha
% ---------------------------------
%     1 |  theta1   0      a1       0
%     2 |  theta2   0      a2       0

RotZ = @(theta) [cos(theta) -sin(theta) 0 0; sin(theta) cos(theta) 0 0; 0 0 1 0 ; 0 0 0 1];
Trans = @(x, y, z) [1 0 0 x; 0 1 0 y; 0 0 1 z; 0 0 0 1];

T_0_1 = RotZ(thetl1)*Trans(l1, 0, 0);
T_1_2 = RotZ(thetl2)*Trans(l2, 0, 0);
T_total = T_0_1*T_1_2;

y0 = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
y0_1 = T_0_1 * y0;
y0_2 = T_total * y0;

originX = [y0(1,4), y0_1(1,4), y0_2(1,4)];
originY = [y0(2,4), y0_1(2,4), y0_2(2,4)];

plot(originX, originY, '*--', 'linewidth', 2);
axis([min([originX, originY])-4, max([originX, originY])+4, min([originX, originY])-4,
max([originX, originY])+4])
grid on
axis square

axisY0X = y0(1, 1:2);
axisY0_1X = y0_1(1, 1:2);
axisY0_2X = y0_2(1, 1:2);

axisY0Y = y0(2, 1:2);
axisY0_1Y = y0_1(2, 1:2);
axisY0_2Y = y0_2(2, 1:2);

line([originX(1), axisY0X(1)], [originY(1), axisY0Y(1)], 'color', 'r', 'linewidth', 3);
line([originX(1), axisY0X(2)], [originY(1), axisY0Y(2)], 'color', 'g', 'linewidth', 3);

line([originX(2), originX(2)+axisY0_1X(1)], [originY(2), originY(2)+axisY0_1Y(1)], 'color', 'r',
'linewidth', 3);
line([originX(2), originX(2)+axisY0_1X(2)], [originY(2), originY(2)+axisY0_1Y(2)], 'color', 'g',
'linewidth', 3);

line([originX(3), originX(3)+axisY0_2X(1)], [originY(3), originY(3)+axisY0_2Y(1)], 'color', 'r',
'linewidth', 3);
```

```
line([originX(3), originX(3)+axisY0_2X(2)], [originY(3), originY(3)+axisY0_2Y(2)], 'color', 'g', 'linewidth', 3);

xPol = [1, 2, 1, 1];
yPol = [1, 1, 2, 1];
pol0 = [xPol; yPol; [0 0 0 0]; [1 1 1 1]];
pol0_1 = T_0_1*pol0;
pol0_2 = T_total*pol0;

line(xPol, yPol, 'color', 'k', 'linewidth', 2);
line(pol0_1(1,:), pol0_1(2,:), 'color', 'k', 'linewidth', 2);
line(pol0_2(1,:), pol0_2(2,:), 'color', 'k', 'linewidth', 2);
```

3. MATLAB 시뮬레이션 결과가 [그림 1-10]과 같이 나오는 것을 확인한다.



[그림 1-10] 2관절 매니퓰레이터 끝단의 위치

2-1. 역기구학 소프트웨어 구현

학습 목표

- 검증된 역기구학 해에 따라 역기구학 소프트웨어를 구현할 수 있다.
- 기존 구현된 역기구학 소프트웨어를 활용할 수 있다.
- 구현된 역기구학 소프트웨어의 시험 검증을 할 수 있다.

필요 지식 /

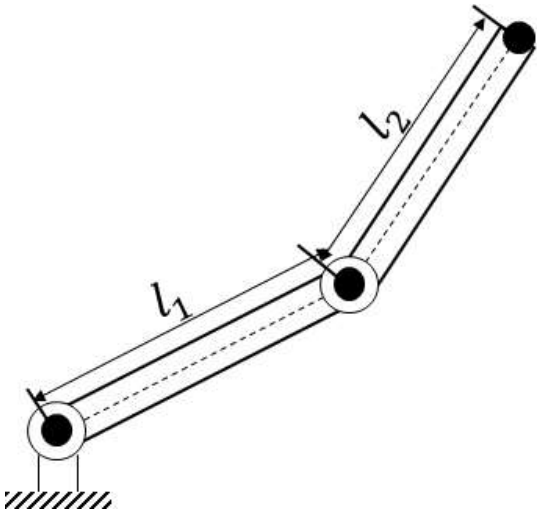
① 관절각의 역기구학

2링크 매니퓰레이터 관절의 각도를 θ_1과 θ_2로 정의할 때, 로봇의 작용점의 위치를 직교좌표 공간상에서 나타내는 것을 순기구학(forward kinematics)이라 부르고 다음과 같이 표시된다.

$$X = l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) \qquad \langle 2\text{-}1 \rangle$$
$$Y = l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2)$$

또 역으로 X, Y의 값을 알 때, θ_1과 θ_2의 값을 구하는 것을 역기구학(inverse kinematics)라 부르고 식 <2-1>을 풀면 다음과 같이 구해진다.

$$\theta_1 = \tan^{-1}\left(\frac{Y}{X}\right) - \tan^{-1}\left(\frac{l_2\sin\theta_2}{l_1+l_2\cos\theta_2}\right) \qquad \langle 2\text{-}2 \rangle$$
$$\theta_2 = \cos^{-1}\left(\frac{X^2+Y^2-l_1^2-l_2^2}{2l_1l_2}\right)$$



[그림 2-1] 2자유도 planar로봇

2 운동변환의 좌표계

D-H 표시법에 의하여 좌표계를 설정하고 두 좌표계 사이의 동차행렬을 구하는 방법을 살펴보았다. 이제 하나의 로봇이 주어져 있을 때, 그 로봇의 기저 좌표계(고정된 기준 좌표계, 사용자 편의상 설정된 cartesian 좌표계임)와 로봇의 작용점 간의 동차행렬을 구하는 방법을 정리하면 다음과 같다.

1. D-H 표시법

(1) Step 1

임의의 좌표계 {0}를 기준으로 설정한다. 이때 z_0는 첫 번째 관절의 회전축 방향으로 설정한다.

(2) Step 2

두 번째 관절 부위에 좌표계 {1}을 설정한다. 이때 D-H조건을 만족하도록 x_1을 설정한다. z_1은 두 번째 관절의 회전축 방향으로 설정한다.

(3) Step 3

step 2를 (n-1)번째 관절 부위까지 진행한 후, 마지막 작용점에 (n)번째 좌표계를 설정한다. 이때 D-H조건을 만족하도록 x_n을 설정하고, z_n은 제약이 없으나 일반적으로 z_{n-1}과 같은 방향으로 설정하는 것이 편리하다.

(4) Step 4

좌표계 {0}→{1}→{2}…{n}에 대한 동차행렬을 구하기 위하여 θ_i , d_i , a_i , α_i의 값을 구하여 link parameter table로 정리한다.

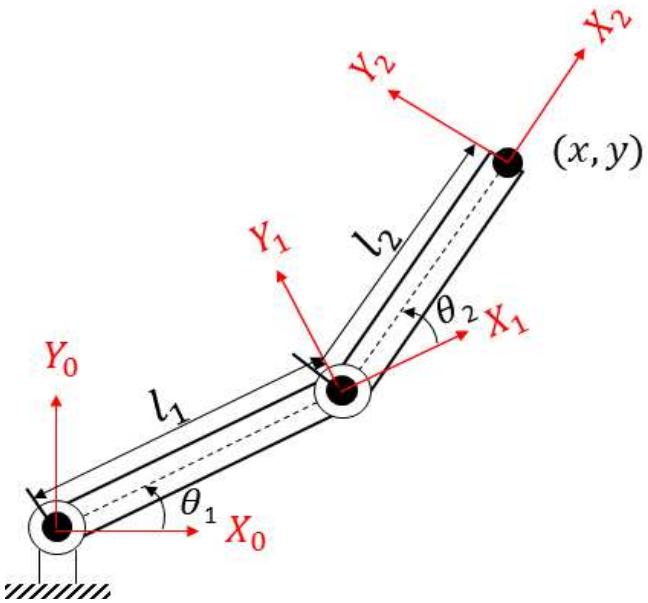
<표 2-1> Link Parameter Table

관절(link)	θ_i	d_i	a_i	α_i
1	θ_1	d_1	a_1	α_1
2	θ_2	d_2	a_2	α_2
⋮	⋮	⋮	⋮	⋮
n	θ_n	d_n	a_n	α_n

(5) Step 5

${}^0H_n$을 구하기 위해 ${}^0H_n = {}^0H_1\,{}^1H_2 \cdots {}^{n-1}H_n$ 의 성질을 이용하여 각각의 곱으로 ${}^0H_n$을 구한다. ${}^0H_n$을 구했다는 것은 작용점 기준으로 설정된 벡터를, 기준 좌표계 기준으로 나타낼 수 있다는 것을 의미한다. 작용점에서의 운동 관계는 학습 3. 동역학적 모델링 설계하기, 학습 4. 동역학 소프트웨어 구현하기에 제시하였다.

2. Planar 로봇의 D-H변환



[그림 2-2] 2자유도 planar 로봇의 좌표 설정

좌표계 {0}은 로봇의 기저에 움직이지 않는 위치에 설정한다. z_0는 관절1의 회전축을 따라 설정하였다. 다음으로 x_1의 설정은 관절 2의 부분에 하며 x_1의 연장선이 z_0와 수직으로 만나도록(DH조건) 설정하고, z_1은 관절 2의 회전축 방향으로 설정한다. y_1은 오른나사의 법칙에 따라 z_1에서 x_1으로 회전할 때 나사가 위로 나오게 되는 원리에 의해 자동으로

설정된다. x_2는 작용점에 그 연장선이 z_1과 수직으로 만나도록 설정하였다. z_2는 편의상 z_1과 동일한 방향으로 설정한다. link parameter table을 구해 보면 <표 2-2>와 같이 나타낼 수 있으며 이를 통하여 ${}^0H_2$를 구할 수 있다.

<표 2-2> Link Parameter Table

관절(link)	θ_i	d_i	l_i	α_i
1	θ_1^*	0	l_1	0
2	θ_2^*	0	l_2	0

3 동차행렬(homogeneous transformations)

링크가 2개이므로 2개의 행이 되며, 사실 한 평면에 있기 때문에 마지막 열의 alpha는 모두 0이 된다. 당연히 z축 중심의 회전인 theta는 그림에서 정의한 대로 θ_1, θ_2로 표현한다. z축 방향의 이동이 없으므로 d도 모두 0이 된다.

$$T_1^0 = Rot(\theta_1)T(a_1,0,0) \qquad \langle 2\text{-}3 \rangle$$

$$T_1^0 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & l_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & l_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

0번 베이스 좌표계에서 1번 링크로 변환하는 행렬은, 위의 DH표에서 보면 z축 중심의 회전(theta1)과 x축 방향으로의 a1만큼의 이동이 된다.

$$T_2^1 = Rot(\theta_2)T(a_2,0,0) \qquad \langle 2\text{-}4 \rangle$$

$$T_1^0 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & l_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & l_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1번 링크에서 2번 링크로의 이동도 동일하게 표현할 수 있다. 최종적으로 MATLAB에서 계산을 하기 위하여 다음과 같은 형식으로 표현할 수 있다.

$$T_2^0 = T_1^0 T_2^1 \qquad \langle 2\text{-}5 \rangle$$

$$T_2^0 = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & l_2\cos(\theta_1+\theta_2)+l_1\cos(\theta_1) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & l_2\sin(\theta_1+\theta_2)+l_1\sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4 동차행렬에 의한 역방향 기구학

1. 역기구학의 동차행렬

좌변과 우변을 적절히 전개해서 thetal, theta2를 얻어내는 것이 이번 예제의 역기구학의 목표가 된다.

$$Rot(\theta_1) = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \langle 2\text{-}6 \rangle$$

$$[Rot(\theta_1)]^{-1} = [Rot(\theta_1)]^T = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \langle 2\text{-}7 \rangle$$

우선 우변의 최좌측행렬인 z축 중심으로 thetal만큼 회전하는 행렬을 우변에서 소거한다. 여기서 회전행렬의 역행렬은 전치(transpose)행렬과 같다.

$$\begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = Rot(\theta_1)\,T(l_1,0,0)\,Rot(\theta_2)\,T(l_2,0,0) \qquad \langle 2\text{-}8 \rangle$$

$$\begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = T(l_1,0,0)\,Rot(\theta_2)\,T(l_2,0,0) \qquad \langle 2\text{-}9 \rangle$$

이제 그 역행렬을 양변에 곱하고, 전개하여 정리하면 다음과 같이 표현 가능하다.

$$\begin{bmatrix} n_x\cos\theta_1+n_y\sin\theta_1 & o_x\cos\theta_1+o_y\sin\theta_1 & a_x\cos\theta_1+a_y\sin\theta_1 & P_x\cos\theta_1+P_y\sin\theta_1 \\ n_y\cos\theta_1-n_x\sin\theta_1 & o_x\cos\theta_1-o_y\sin\theta_1 & a_x\cos\theta_1-a_y\sin\theta_1 & P_x\cos\theta_1-P_y\sin\theta_1 \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \langle 2\text{-}10 \rangle$$

$$= \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_1+a_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & a_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

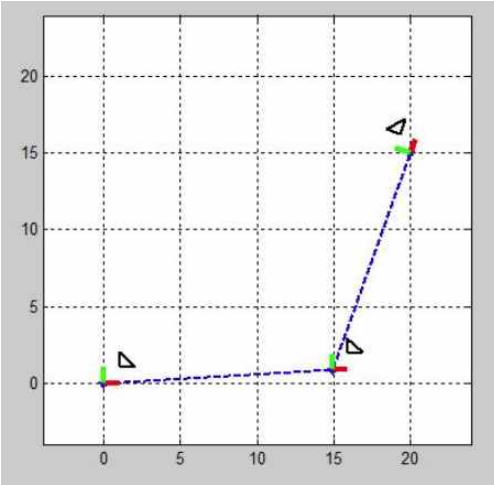
2. 역기구학 동차행렬의 프로그래밍

MATLAB상에서 동차행렬을 이용하여 끝점의 위치를 파악할 수 있는 프로그램을 작성한다. [그림 2-1]과 같이 링크의 길이가 각각 $l_1 = 1.5\text{m}$, $l_2 = 1.5\text{m}$ 인 2관절 매니퓰레이터에서 x점의 좌표가 20, y점 좌표가 15일 때 매니퓰레이터의 링크1, 링크2의 각도를 계산한다.

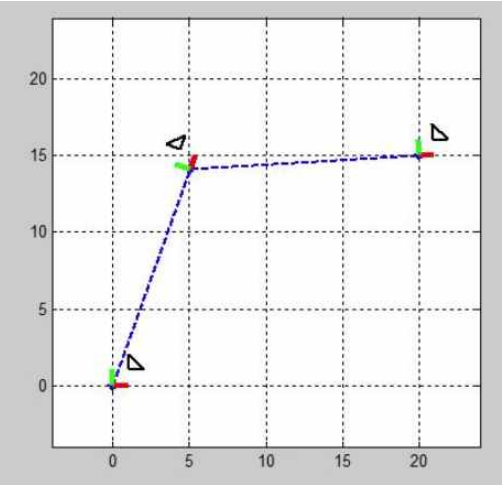
```
syms th1 th2 nx ny nz ox oy oz ax ay az Px Py Pz a1 a2

RotZ1 = [cos(th1) -sin(th1) 0 0; sin(th1) cos(th1) 0 0; 0 0 1 0; 0 0 0 1]
RotZ2 = [cos(th2) -sin(th2) 0 0; sin(th2) cos(th2) 0 0; 0 0 1 0; 0 0 0 1]

TransA1 = [1 0 0 l1; 0 1 0 0; 0 0 1 0; 0 0 0 1]
TransA2 = [1 0 0 l2; 0 1 0 0; 0 0 1 0; 0 0 0 1]

T_0_2 = RotZ1*TransA1*RotZ2*TransA2
TT = [nx ox ax Px; ny oy ay Py; nz oz az Pz; 0 0 0 1]

% Left Step 1
LS1 = simplify(inv(RotZ1)*TT)

% Right Step 1
RS1 = simplify(inv(RotZ1)*T_0_2)

%%
l1 = 15;
l2 = 15;
Px = 20;
Py = 15;
th2 = [2*atan((((a1+a2)^2-(Px^2+Py^2))/(Px^2+Py^2-(a1-a2)^2))^0.5)
       -2*atan((((a1+a2)^2-(Px^2+Py^2))/(Px^2+Py^2-(a1-a2)^2))^0.5)]
th1 = atan2(Py,Px) - atan(a2*sin(th2)./(a1+a2*cos(th2)))

%%
% DH Table
%       ¦ theta   d     a     alpha
% --------------------------------
```

```
%     1 ¦  theta1   0     a1       0
%     2 ¦  theta2   0     a2       0

RotZ = @(theta) [cos(theta) -sin(theta) 0 0; sin(theta) cos(theta) 0 0; 0 0 1 0 ; 0 0 0 1];
Trans = @(x, y, z) [1 0 0 x; 0 1 0 y; 0 0 1 z; 0 0 0 1];

for i=1:2
    theta1 = th1(i);
    theta2 = th2(i);

    T_0_1 = RotZ(theta1)*Trans(a1, 0, 0);
    T_1_2 = RotZ(theta2)*Trans(a2, 0, 0);
    T_total = T_0_1*T_1_2;

    y0 = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
    y0_1 = T_0_1 * y0;
    y0_2 = T_total * y0;

    originX = [y0(1,4), y0_1(1,4), y0_2(1,4)];
    originY = [y0(2,4), y0_1(2,4), y0_2(2,4)];

    figure
    plot(originX, originY, '*--', 'linewidth', 2);
    axis([min([originX, originY])-4, max([originX, originY])+4, min([originX, originY])-4,
    max([originX, originY])+4])
    grid on
    axis square

    axisY0X = y0(1, 1:2);
    axisY0_1X = y0_1(1, 1:2);
    axisY0_2X = y0_2(1, 1:2);

    axisY0Y = y0(2, 1:2);
    axisY0_1Y = y0_1(2, 1:2);
    axisY0_2Y = y0_2(2, 1:2);

    line([originX(1), axisY0X(1)], [originY(1), axisY0Y(1)], 'color', 'r', 'linewidth', 3);
    line([originX(1), axisY0X(2)], [originY(1), axisY0Y(2)], 'color', 'g', 'linewidth', 3);

    line([originX(2), originX(2)+axisY0_1X(1)], [originY(2), originY(2)+axisY0_1Y(1)], 'color', 'r',
    'linewidth', 3);
    line([originX(2), originX(2)+axisY0_1X(2)], [originY(2), originY(2)+axisY0_1Y(2)], 'color', 'g',
    'linewidth', 3);
    line([originX(3), originX(3)+axisY0_2X(1)], [originY(3), originY(3)+axisY0_2Y(1)], 'color', 'r',
    'linewidth', 3);
```

```
    line([originX(3), originX(3)+axisY0_2X(2)], [originY(3), originY(3)+axisY0_2Y(2)], 'color', 'g',
    'linewidth', 3);

    xPol = [1, 2, 1, 1];
    yPol = [1, 1, 2, 1];
    pol0 = [xPol; yPol; [0 0 0 0]; [1 1 1 1]];
    pol0_1 = T_0_1*pol0;
    pol0_2 = T_total*pol0;

    line(xPol, yPol, 'color', 'k', 'linewidth', 2);
    line(pol0_1(1,:), pol0_1(2,:), 'color', 'k', 'linewidth', 2);
    line(pol0_2(1,:), pol0_2(2,:), 'color', 'k', 'linewidth', 2);
end
```



[그림 2-3] $\theta > 0$ 인 경우의 매니퓰레이터 자세



[그림 2-4] $\theta < 0$ 인 경우의 매니퓰레이터 자세

수행 내용 / 역기구학 소프트웨어 구현하기

재료 · 자료

- 요구사양서
- 로봇 운동학 해석 보고서
- 로봇 하드웨어 매뉴얼
- 로봇 동역학 분석 보고서
- 로봇 작업 시나리오

기기(장비 · 공구)

- 컴퓨터, 프린터
- 로봇 시뮬레이션 프로그램(MATLAB 등)

안전 · 유의 사항

- 로봇 시뮬레이션을 수행할 때에는 범용적으로 사용되는 소프트웨어를 선택해야 한다.

수행 순서

1 2관절 매니퓰레이터의 끝점의 연속 동작에 대한 움직임을 시뮬레이션 한다.

1. 2관절 매니퓰레이터의 끝점의 위치에 관한 순기구학 식의 유도 과정을 이해한다.

$$x = l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2)$$
$$y = l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2)$$

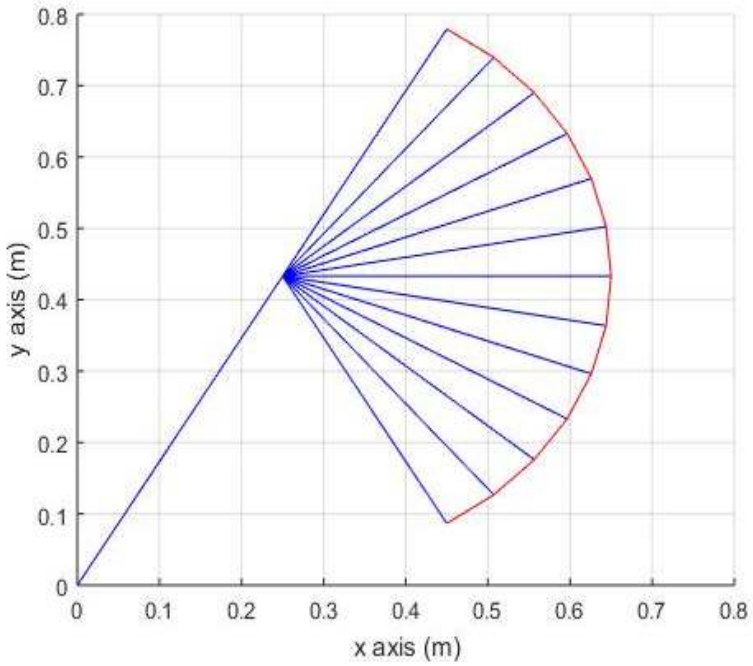
2. MATLAB을 이용하여 끝점의 위치를 파악할 수 있는 프로그램을 작성한다.

(1) [그림 2-1]과 같이 링크의 길이가 각각 0.5m, 0.4m인 2관절 매니퓰레이터에서 θ_1을 60°로 고정시킨 상태로 θ_2를 0°에서 -120°까지 10°씩 변경시킬 때의 매니퓰레이터의 궤적을 그려 본다.

(2) 2관절 매니퓰레이터의 운동에 관한 MATLAB 프로그램(θ_2의 운동)은 다음과 같다.

```
l1=0.5;
l2=0.4;
```

```
angle1=60;
angle2=[0:-10:-60];

theta1=angle1*pi/180;
theta2=angle2*pi/180;
res_x=[];
res_y=[];
for i=theta1
    for j=theta2
        x1=l1*cos(i);
        y1=l1*sin(i);
        x2=l1*cos(i)+l2*cos(i+j);
        y2=l1*sin(i)+l2*sin(i+j);
        hold on
        plot([0,x1,x2],[0,y1,y2],'b')
        res_x=[res_x,x2];
        res_y=[res_y,y2];
    end
end
plot(res_x,res_y,'r');
hold off
```

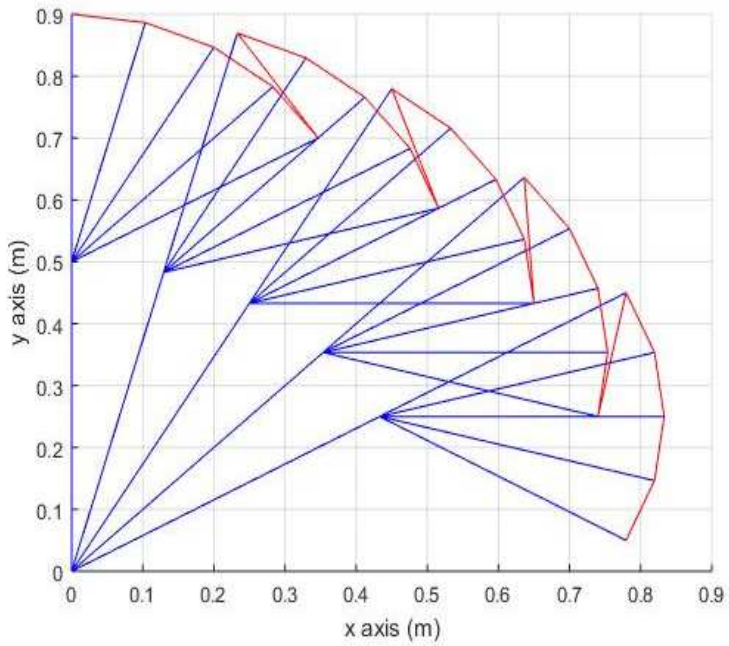


[그림 2-5] 2관절 매니퓰레이터의 운동 궤적(θ_2의 운동)

(3) 2관절 매니퓰레이터에서 θ_1을 90°에서 30°까지, θ_2를 0°에서 -60°까지 15°씩 변경시킬 때의 매니퓰레이터의 궤적을 그려 본다. 이 경우, θ_1과 θ_2의 16개 조합(즉, (90°, 0°)부터 (30°, -60°)에 대한 매니퓰레이터의 궤적을 그리면 된다.

(4) 2관절 매니퓰레이터의 운동에 관한 MATLAB 프로그램(θ_1 및 θ_2의 운동)은 다음과 같다.

```
l1=0.5;
l2=0.4;
angle1=[90:-15:30];
angle2=[0:-15:-60];

theta1=angle1*pi/180;
theta2=angle2*pi/180;
res_x=[];
res_y=[];
for i=theta1
    for j=theta2
        x1=l1*cos(i);
        y1=l1*sin(i);
        x2=l1*cos(i)+l2*cos(i+j);
        y2=l1*sin(i)+l2*sin(i+j);
        hold on
        plot([0,x1,x2],[0,y1,y2],'b')
        res_x=[res_x,x2];
        res_y=[res_y,y2];
    end
end
plot(res_x,res_y,'r');
hold off
```



[그림 2-6] 2관절 매니퓰레이터의 운동 궤적(θ_1 및 θ_2의 운동)

3-1. 동역학적 모델링 설계

학습 목표

- 동역학 모델링을 위한 파라미터를 도출할 수 있다.
- 동역학 모델에 적용할 수 있는 알고리즘을 선정할 수 있다.
- 선정된 방법에 따라 동역학적 해를 구할 수 있다.
- 구한 동역학적 해가 정확한지 검증할 수 있다.

필요 지식 /

1 로봇 동역학 개념

1. 로봇 모델링 과정

로봇을 설계하고 프로그램을 통한 시뮬레이션을 하려면 실제 로봇을 대상으로 다양한 변수 값들을 직접 측정하거나 모델링 과정을 거쳐야 한다. 하지만 로봇의 크기나 가격 등의 문제로 실제 로봇에의 적용이 어려울 경우가 있다. 이러한 경우에 로봇동역학(dynamics)은 실제 로봇이 움직이거나 프로그램상에서 시뮬레이션으로 행동을 미리 알아보기 위해서 로봇을 모델링하는 것이다.

먼저 기구학을 통하여 로봇 조인트의 움직임을 직교좌표 공간에서 나타낼 수 있다. 하지만 이러한 로봇의 움직임은 실제 로봇의 움직임을 고려한 것이 아니라, 기구학의 수식적인 관계를 나타낸 것이다.

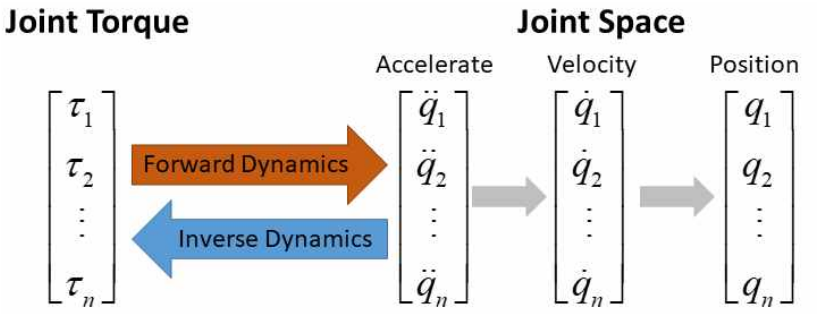
실제 로봇의 움직임을 나타내기 위해서는 각 조인트에 적용되는 힘이나 토크 값을 구해야 한다. 각 조인트의 토크 값은 다양한 힘으로 구성되어 있는데, 이 값을 구하면 토크 값에 의한 로봇의 움직임을 동적으로 나타낼 수 있게 된다.

2. 로봇 운동학과 동역학 관계

로봇 매니퓰레이터 제어를 위한 제어 흐름은 kinematics → dynamics → control의 순서로 진행된다. 위 식의 오른쪽 항은 로봇 매니퓰레이터의 관절각과 공간상의 위치의 관계를 표시한다. 오른쪽의 joint space와 cartesian space를 왔다 갔다 하는 forward/inverse

kinematics 관계를 표현할 수 있다.

dynamics는 매니퓰레이터의 힘을 표현하기 위한 관계식으로 표현되지만 kinematics에서는 힘(토크)은 전혀 고려하지 않고 관절 각도와 위치 관계를 다루고 있다.

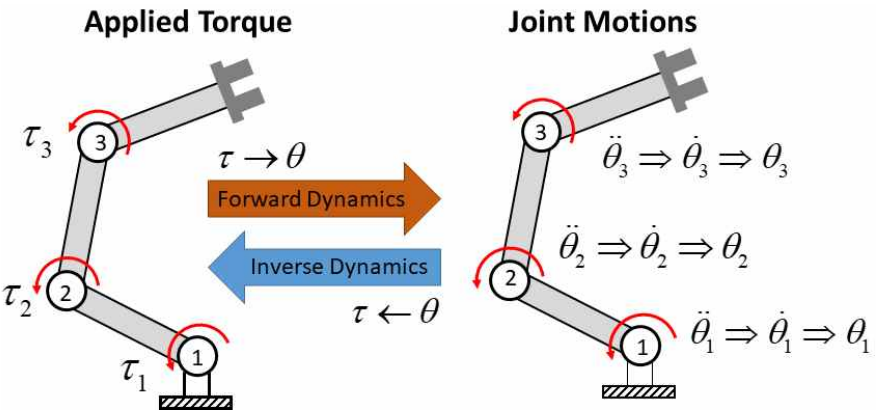


[그림 3-1] 힘(토크)과 관절의 동작 관계

동역학도 kinematics와 마찬가지로 순방향 동역학(forward dynamics)과 역방향 동역학(inverse dynamics)으로 구분할 수 있다.

순방향 동역학은 "매니퓰레이터 관절(joint)에 힘(또는 토크)을 이렇게 주었더니 관절이 이렇게 행동하더라." 를 푸는 문제이고, 반대로 역방향 동역학은 "관절이 이렇게 행동하게 하려면 힘을 어떻게 주어야하지?" 와 같이 힘과 관절의 관계를 다룬다.

control을 제외한 kinematics → dynamics까지만 보면 원하는 cartesian space상의 end-effector의 위치, 자세 등으로부터 inverse kinematics를 통해 관절의 position, velocity 등을 알아낸 후 inverse dynamics로 힘(토크)를 계산하여 각 관절에 가해 준다.



[그림 3-2] 매니퓰레이터의 동역학적 관계

2 로봇 동역학식

1. 로봇 동역학식의 유도 방법

로봇의 동역학식을 유도하는 데는 두 가지 방법이 있다. 하나는 뉴턴-오일러(NEWTON-EULER) 방식이고 다른 하나는 오일러-랑그라지(EULER-LAGRANGIAN) 방식이다.

(1) 뉴턴-오일러(NEWTON-EULER) 방식

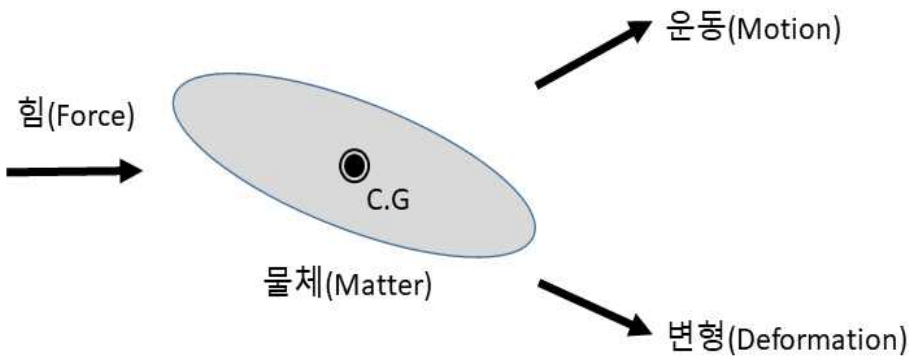
이 방식은 동적 시스템을 뉴턴의 두 번째 법칙을 직접적으로 적용하여 각 링크의 좌표에서 링크의 힘과 모멘트로 나타내는 것이다. 따라서 이 방식은 힘의 균형을 기반으로 한 접근 방식으로 동역학 방정식의 유도에 효율적이며, 시뮬레이션과 컴퓨터 계산을 할 때 편리하다.

(2) 오일러-랑그라지(EULER-LAGRANGIAN) 방식

시스템의 동적 특성을 일과 에너지의 개념으로 나타낸 것으로 뉴턴-오일러 방식보다 간단하다. 에너지를 기반으로 한 동역학적 모델접근 방법으로, 비교적 간단한 운동에서 로봇의 운동에 작용하는 여러 가지 변수에 의한 효과를 이해하는 데 유용하며 일반좌표에 근거한다.

2. 로봇 동역학

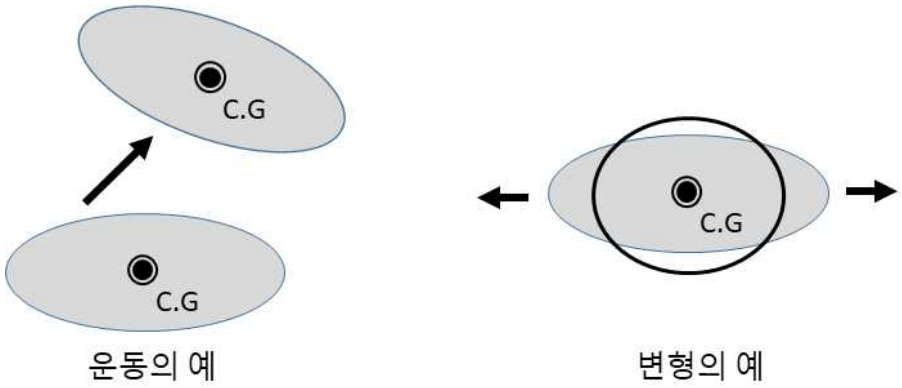
동역학이란 역학(mechanics)의 한 분야로서 물체가 힘을 받아 운동을 일으킬 때, 그 현상을 분석하는 학문이다. 여기서 역학이란 힘, 물체, 운동과 변형 이렇게 3요소 간의 관계를 다루는 물리학의 가장 근본적인 분야를 말한다.



[그림 3-3] 역학의 3요소

힘에는 중력이나 전자기력과 같이 물체의 영역 전체에 작용하는 몸체력(body force)과 마찰력이나 압력과 같이 물체의 경계에 작용하는 접촉력(traction force)이 있다.

운동은 시간에 따른 몸체의 위치 및 자세의 변화를 의미하며, 변형이란 몸체상의 두 점간의 상대적 거리의 변화를 의미한다. 일반적으로는 운동과 변형이 섞여서 일어나며, 이 경우 둘을 구분하기가 곤란하여 변형과 운동을 하나로 취급하여 해석하기도 한다.



[그림 3-4] 운동의 변형

수행 내용 / 동역학적 모델링 설계하기

재료 · 자료

- 요구사양서
- 로봇 운동학 해석 보고서
- 로봇 하드웨어 매뉴얼
- 로봇 동역학 분석 보고서
- 로봇 작업 시나리오

기기(장비 · 공구)

- 컴퓨터, 프린터
- 로봇 시뮬레이션 프로그램(MATLAB 등)

안전 · 유의 사항

- 로봇 시뮬레이션을 수행할 때에는 범용적으로 사용되는 소프트웨어를 선택해야 한다.

수행 순서

1 동역학 표현 방법을 파악한다.

Robot dynamics를 구하기 위한 방법에는 크게 작용반작용의 원리를 이용한 뉴턴-오일러 방법(N-E method)과 energy 관점에서 풀어나가는 오일러-랑그라지 방법(E-L method)이 있다. 랑그라지 방법을 이용하면 완벽한 형태의 closed form의 운동방정식을 유도할 수 있지만, 식을 유도하는 데에 많은 시간이 들고 연산량이 많다는 단점이 있다. 여기에선 뉴턴-오일러 방법을 소개하도록 한다.

뉴턴-오일러 방법을 사용하게 되면 계산을 위해 유도된 식이 컴퓨터로 프로그래밍하게에 적합한 형태를 갖추고 있게 된다(ODE를 직접 풀지 않고 for문을 사용하면 된다).

랑그라지 방법보다 뉴턴-오일러 방법이 상대적으로 연산량이 적으므로 실시간 inverse dynamics를 위한 토크 계산에 뉴턴-오일러 방법이 매우 유리하다.

2 뉴턴 오일러 동역학(NEWTON EULER dynamics) 방법을 파악한다.

1. 회전력(모멘트), n

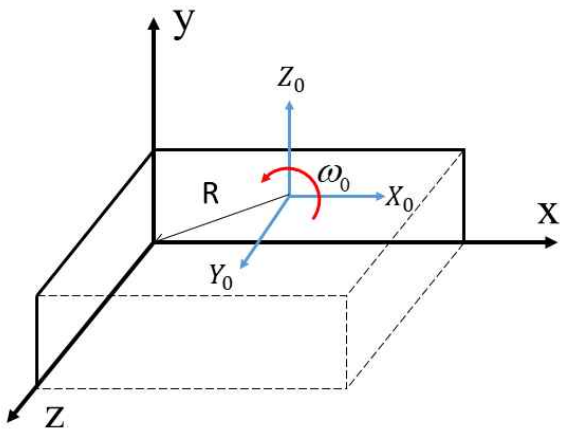
뉴턴의 운동 제2법칙은 선형운동에 대해서 아래 관계식으로 표시된다.

$$\mathrm{F} = ma \qquad \langle 3\text{-}1 \rangle$$

이를 회전운동에 적용하면

$$\frac{d(I_0 \boldsymbol{\omega}_0)}{dt} = n_0 \qquad \langle 3\text{-}2 \rangle$$

의 관계식으로 표시되며 I_0는 inertial frame(이는 질량 중심에 선정된 frame을 의미한다), $\boldsymbol{\omega}_0$는 회전속도, n_0는 그 물체에 가해진 모멘트의 합을 의미한다. 주목할 사항은 관성 모멘트는 물체의 위치에 따라 달라질 수 있다는 것이다.



[그림 3-5] 운동 물체의 모멘트와 좌표관계

물체가 가지고 있는 회전 운동에너지는 어떤 frame에 대해서 표시하든지 동일하므로 아래의 관계식을 만족하게 된다.

$$\omega^{T}_{I_0\omega_0} = \omega^{T} I\omega 0 \qquad \langle 3\text{-}3 \rangle$$

[그림 3-5]에서 frame $\{x,\ y,\ z\}$으로 부터 frame $\{x_0,\ y_0,\ z_0\}$로의 회전행렬을 ${}^{o}R$로 정의하면(${}^{o}R^{T} \triangleq R$로 표시하자) 아래 관계식으로 표현된다.

$$\omega = R^{T}\omega_0 \qquad \langle 3\text{-}4 \rangle$$

이 식을 식 <3-3>에 대입하면 식 <3-5>와 같이 표현할 수 있다.

$$\omega^{T}_{I_0\omega_0} = \omega^{T}_{RIR^{T}\omega_0} 0 \qquad \langle 3\text{-}5 \rangle$$

따라서 $I_0 = RIR^{T}$의 관계식이 성립한다.

회전 운동방정식을 frame$\{x,\ y,\ z\}$에 대해 기술하면 아래와 같이 유도된다.

$$\begin{aligned} n &= \frac{d(I\omega)}{dt} \\ &= \dot{I\omega} + \omega \times (I\omega) \end{aligned} \qquad \langle 3\text{-}6 \rangle$$

이때 $\omega \times (I\omega)$를 gyroscope 항이라 부른다.

이 식의 유도 과정을 살펴보면 다음과 같다.

Angular momentum을 inertial frame에 대해 기술하면 $\mathbf{h}_0 = \mathrm{I}_0\omega_0$ 으로 표현되며, 이를 물체의 특정 부분에 설정된 frame(일반적으로 joint 부분에 설정할 것이다)으로 회전행렬 R을

사용하여 변환하면 다음과 같다.

$$\begin{aligned} \mathbf{h}_0 &= I_0\omega_0 \\ &= RIR^T R\omega \\ &= RI\omega \end{aligned} \qquad \langle 3\text{-}7 \rangle$$

따라서 $\frac{d\mathbf{h}_0}{dt} = n_0$ 의 관계식으로부터 $n_0 = RI\dot{\omega} + \dot{R}I\omega$ 으로 되며 이때 주목할 사항은 inertial frame에 대해서는 I가 상수라는 사실을 사용하기 위하여 어떤 joint에서 요구되는 τ를 직접 구하지 않고 inertial frame으로 표시되는 τ_0를 구한 후 회전 행렬을 사용하여 joint frame으로 옮기는 것이다. 좀 더 구체적으로 보면,

$$\dot{R} = S(\omega_0)\mathrm{R} \qquad \langle 3\text{-}8 \rangle$$

의 관계를 사용하면 $n_0 = RI\dot{\omega} + S(\omega)\mathrm{RI}\omega$ 로 된다.

최종적으로 $\tau = R^T\tau_0$의 관계에 의하여 아래와 같이 표현된다.

$$\begin{aligned} n &= R^T RI\dot{\omega} + R^T S(\omega_0)RI\omega \\ &= I\dot{\omega} + S(\omega)I\omega \\ &= I\dot{\omega} + \omega \times I\omega \end{aligned} \qquad \langle 3\text{-}9 \rangle$$

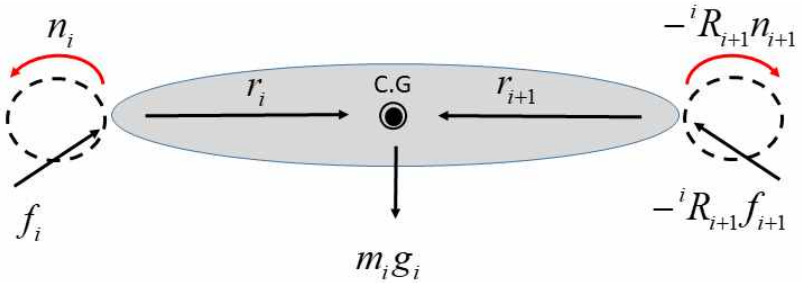
2. Force Balance 방정식

$R^T S(\omega_0)\mathrm{R} = \mathrm{S}(\omega)$ 에 대한 구체적인 표현은 아래와 같다.

$$\begin{aligned} RS(\omega_0)R^T\omega_x &= R(\omega_0 \times R^T\omega_x) \\ &= (R\omega_0) \times (R\,R^T\omega_x) \\ &= (R\omega_0) \times \omega_x \\ &= S(R\omega_0)\omega_x \end{aligned} \qquad \langle 3\text{-}10 \rangle$$

따라서 $RS(\omega_0)\mathrm{R}^{\mathrm{T}} = \mathrm{S}(\omega)$ 이며, $\omega = R\omega_0$ 인 것은 이미 정의된 바대로다.

이제 NEWTON-EULER dynamic 방정식들은 force balance 방정식과 moment balance 방정식에 의하여 유도될 수 있음을 본다.



[그림 3-6] i번째 링크에서의 force 및 torque.

먼저 force balance 방정식은 다음과 같이 표현된다.

$$\mathbf{f}_i - {}^{i}\mathrm{R}_{i+1}\mathbf{f}_{i+1} + \mathrm{m}_i\mathbf{g}_i = \mathrm{m}_i\mathbf{a}_{c,i} \qquad \langle 3\text{-}11 \rangle$$

이 식에서 $\mathbf{a}_{c,i}$는 질량중심이 가지고 있는 가속도를 나타낸다.

모멘트(moment) 방정식은 좀 더 복잡하지만, 중력에 의한 회전력은 포함되지 않을 것을 짐작할 수 있으며, 아래와 같이 표현할 수 있다.

$$n_i - {}^iR_{i+1}n_{i+1} + \mathbf{f}_i \times \mathbf{r}_i - {}^{i}\mathrm{R}_{i+1}\mathbf{f}_{i+1} \times \mathbf{r}_{i+1} = \mathrm{I}_i\boldsymbol{\alpha}_i + \boldsymbol{\omega}_i \times (I_i\boldsymbol{\omega}_i) \qquad \langle 3\text{-}12 \rangle$$

이때 좌변은 외부에서 가해진 모든 moment의 합이며 우변은 이로 인한 회전운동 방정식이다. 식 <3-11>, <3-12>를 다시 정렬하면 다음과 같이 표현할 수 있다.

$$\mathbf{f}_i = \mathrm{m}_i\mathbf{a}_{c,i} - \mathrm{m}_i\mathbf{g}_i + {}^{i}\mathrm{R}_{i+1}\mathbf{f}_{i+1} \qquad \langle 3\text{-}13 \rangle$$

$$n_i = I_i\boldsymbol{\alpha}_i + \boldsymbol{\omega}_i \times (\mathrm{I}_i\boldsymbol{\omega}_i) + {}^{i}\mathrm{R}_{i+1}\mathrm{n}_{i+1} - \mathbf{f}_i \times \mathbf{r}_i + {}^{i}\mathrm{R}_{i+1}\mathbf{f}_{i+1} \times \mathbf{r}_{i+1} \qquad \langle 3\text{-}14 \rangle$$

식 <3-13>과 식 <3-14>에서 $i+1$번째 joint가 바로 end-effector라고 보면 $f_{i+1} = 0$, $\eta_{i+1} = 0$이 되어 로봇의 주어진 기구학적 parameter들인 $m_i, {}^iR_{i+1}, \mathbf{r}_i, \mathbf{r}_{i+1}$값들이 주어져 있고 $g_i = [0,\ 0,\ -9.8m/\sec^2]^T$로 주어져 있으며, 운동에 관련된 변수들의 값 $\boldsymbol{\omega}_i, \dot{\boldsymbol{\omega}}_i, \mathbf{a}_{c,i}$가 주어져 있으며, i번째 joint에서 필요한 힘 및 모멘트 값, $\mathbf{f}_i$ 및 n_i가 계산된다는 것을 알 수 있다(inward iteration). n_i가 구해졌을 때 Denavit-Hatenberg constraint에 의해 i joint를 위한 z축을 i joint의 회전방향으로 잡는다고 가정하면 $\tau_i = n_i^T[0\ 0\ 1]^T$로 구해진다. i번째 joint가 prismatic joint이며, 이 joint의 운동방향으로 z축이 할당되어 있다면 $\boldsymbol{\tau}_i = \boldsymbol{\tau}_i^T[0\ 0\ 1]^T$로 구해진다.

이제 운동관계를 기술하기 위하여, 기준 frame을 {0}으로 가정한다.

$$^{0}\boldsymbol{\omega}_{i} = {}^{0}\boldsymbol{\omega}_{i+1} + \mathbf{z}_{i+1}\dot{q}_{i} \qquad \langle 3\text{-}15 \rangle$$

위 식은 의미적으로 i번째 frame의 회전운동을 i번째까지의 회전운동에 i번째 frame 자체의 회전운동을 더한 것이다.

$\boldsymbol{\omega}_{i}$를 i번째 frame을 기준으로 한 $\boldsymbol{\omega}_{i}$로 표기하면 $^{0}R_{i}\boldsymbol{\omega}_{i} = {}^{0}\mathrm{R}_{i-1}\boldsymbol{\omega}_{i-1} + \mathbf{z}_{i-1}\dot{q}_{i}$에서 다음과 같이 표현된다(단, $\mathbf{b}_{i} = {}^{0}\mathrm{R}_{\mathbf{z}_{i-1}}^{\mathrm{T}}{}_{i}$).

$$\begin{aligned}\boldsymbol{\omega}_{i} &= (^{0}R_{i})^{T\,0}R_{i-1}\boldsymbol{\omega}_{i-1} + {}^{0}R_{\mathbf{z}_{i+1}}^{T}\dot{q}_{i}{}^{i} \\ &= {}^{i}R_{i-1}\boldsymbol{\omega}_{i-1} + \mathbf{b}_{i}\dot{q}_{i}\end{aligned} \qquad \langle 3\text{-}16 \rangle$$

이 관계식을 이용하면, base frame으로부터 시작하여 즉, frame{0}으로부터 1, 2, … i 번째까지 frame에 해당되는 각속도를 구할 수 있다(outward iteration).

3. 각가속도

각가속도 $\boldsymbol{\alpha}_{i}$를 구해보도록 하자. $\boldsymbol{\alpha}_{i} = {}^{i}\mathrm{R}_{0}{}^{0}\dot{\boldsymbol{\omega}}_{i}$로 표시되는데 여기서 $\boldsymbol{\alpha}_{i}$는 단지 i번째 frame에서 발생되는 회전운동 속도의 변화량이 아니라는 것이다. 즉, $\boldsymbol{\alpha}_{i} \neq \dot{\boldsymbol{\omega}}_{i}$ 이 $\boldsymbol{\alpha}_{i}$를 구하기 위하여 식 <3-15>에서 양변에 미분을 하면 다음과 같이 주어진다.

$$^{0}\dot{\boldsymbol{\omega}}_{i} = {}^{0}\dot{\boldsymbol{\omega}}_{i-1} + \mathbf{z}_{i-1}\ddot{q}_{i} + {}^{0}\boldsymbol{\omega}_{i} \times \mathbf{z}_{i-1}\dot{q}_{i} \qquad \langle 3\text{-}17 \rangle$$

이때 i frame에 대해 기술하면 아래와 같이 표시된다(outward Iteration).

$$\boldsymbol{\alpha}_{i} = {}^{i}\mathrm{R}_{i-1}\boldsymbol{\alpha}_{i-1} + \mathbf{b}_{i}\ddot{q}_{i} + \boldsymbol{\omega}_{i} \times \mathbf{b}_{i}\dot{q}_{i} \qquad \langle 3\text{-}18 \rangle$$

4. 선형속도 및 가속도

이제 선형속도와 가속도 항들을 구해보도록 하자.

link i의 질량 중심에서의 선형속도는 다음과 같이 표현된다.

$$^{0}\dot{\mathbf{v}}_{c,i} = {}^{0}\dot{\mathbf{v}}_{e,i-1} + {}^{0}\dot{\boldsymbol{\omega}}_{i} \times {}^{0}\mathbf{r}_{i} + {}^{0}\boldsymbol{\omega}_{i} \times {}^{0}\dot{\mathbf{r}}_{i} \qquad \langle 3\text{-}19 \rangle$$

양변을 미분하면서 $\frac{d}{dt}(^{0}R_{i}\mathbf{r}_{i}) = \mathrm{S}(^{0}\boldsymbol{\omega}_{i})^{0}\mathrm{R}_{i}\mathbf{r}_{i} = {}^{0}\boldsymbol{\omega}_{i} \times {}^{0}\mathrm{R}_{i}\mathbf{r}_{i} = {}^{0}\boldsymbol{\omega}_{i} \times {}^{0}\mathbf{r}_{i}$ 을 이용하면 다음과 같이 표현된다(여기서 $\mathbf{a}_{c,i} = {}^{i}\mathrm{R}_{0}{}^{0}\mathbf{a}_{c,i}$ 로 구해진다).

$$^{0}\mathbf{a}_{c,i} = {}^{0}\mathbf{a}_{e,i-1} + {}^{0}\dot{\boldsymbol{\omega}}_{i} \times {}^{0}\mathbf{r}_{i} + {}^{0}\boldsymbol{\omega}_{i} \times (^{0}\boldsymbol{\omega}_{i} \times {}^{0}\mathbf{r}_{i}) \qquad \langle 3\text{-}20 \rangle$$

따라서 $\mathbf{a}_{c,i} = {}^{i}\mathrm{R}_{i-1}\mathbf{a}_{e,i-1} + \dot{\boldsymbol{\omega}}_i \times \mathbf{r}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{r}_i)$ 로 얻어진다.

End-effector에서 acceleration을 구하기 위해서는 $\mathbf{r}_i$를 질량 중심까지가 아니고 i link의 끝점까지로 연장한 것으로만 대입하면 된다. 이를 $\mathbf{r}_{i,c}$로 표시한다. 그러면 다음과 같은 식으로 표현할 수 있다.

$$\mathbf{a}_{e,i} = {}^{i}\mathrm{R}_{i-1}\mathbf{a}_{e,i-1} + \dot{\boldsymbol{\omega}}_i \times \mathbf{r}_{i,e} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{r}_{i,e}) \qquad \langle 3\text{-}21 \rangle$$

지금까지 구한 수식들에 의하여 식 <3-13>과 식 <3-14>로 주어진 NEWTON-EULER formulation은 완성된다. 이를 실제적으로 구하는 측면에서 정리하면 다음과 같다.

(1) Outward Iteration.: $0 \rightarrow n$

주어진 $\boldsymbol{\omega}_0 = 0$, $\boldsymbol{\alpha}_0 = 0$, $\mathbf{a}_{c,0} = 0$, $\mathbf{a}_{e,0} = 0$ 로부터(이는 고정된 base frame에 대한 값으로 당연하다)

$$\begin{aligned}
\boldsymbol{\omega}_i &= {}^{i}R_{i-1}\boldsymbol{\omega}_{i-1} + \mathbf{b}_i\dot{q}_i \\
\boldsymbol{\alpha}_i &= {}^{i}R_{i-1}\boldsymbol{\alpha}_{i-1} + \mathbf{b}_i\ddot{q}_i + \boldsymbol{\omega}_i \times \mathbf{b}_i\dot{q}_i \\
\mathbf{a}_{e,i} &= {}^{i}R_{i-1}\mathbf{a}_{e,i-1} + \dot{\boldsymbol{\omega}}_i \times \mathbf{r}_{i,e} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{r}_{i,e}) \\
\mathbf{a}_{c,i} &= {}^{i}R_{i-1}\mathbf{a}_{e,i-1} + \dot{\boldsymbol{\omega}}_i \times \mathbf{r}_i + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{r}_i)
\end{aligned} \qquad \langle 3\text{-}22 \rangle$$

위 식을 사용하여 $0 \sim n$에 해당되는 $\boldsymbol{\omega}_i$,$\boldsymbol{\alpha}_i$,$\mathbf{a}_{c,i}$를 차례로 오름차순으로 모두 정한다.

(2) Inward Iteration: $n \rightarrow 0$

주어진 $\mathbf{f}_{n+1} = 0$,$\mathbf{n}_{n+1} = 0$ 로부터 아래 식을 사용하여 $n \sim 0$에 해당되는 $\mathbf{f}_i$ 및 n_i의 값을 내림차순으로 모두 정한다.

$$\mathbf{f}_i = \mathrm{m}_i\mathbf{a}_{c,i} - \mathrm{m}_i\mathbf{g}_i + {}^{i}\mathrm{R}_{n+1}\mathbf{f}_{i+1} \qquad \langle 3\text{-}23 \rangle$$

$$n_i = I_i\mathbf{a}_i + \boldsymbol{\omega}_i \times (\mathrm{I}_i\boldsymbol{\omega}_i) + {}^{i}\mathrm{R}_{i+1}\mathrm{n}_{i+1} - \mathbf{f}_i \times \mathbf{r}_i + {}^{i}\mathrm{R}_{i+1}\mathbf{f}_{i+1} \times \mathbf{r}_{i+1} \qquad \langle 3\text{-}24 \rangle$$

앞에서 설명하였듯이 $\mathbf{f}_i$ 및 n_i가 모두 구해지면 joint의 종류에 상관없이 필요한 힘 혹은 토크 값이 구해진다.

학습 4 동역학 소프트웨어 구현하기

4-1. 동역학 소프트웨어 구현

학습 목표

- 검증된 동역학 해에 따라 동역학 소프트웨어를 구현할 수 있다.
- 기존 구현된 동역학 소프트웨어를 활용할 수 있다.
- 구현된 동역학 소프트웨어의 시험 검증을 할 수 있다.

필요 지식 /

1 동역학 시뮬레이션

학습 3에서는 시스템의 동역학 모델을 얻기 위해 라그랑지 방정식을 찾는 법에 대해 설명했다. 이번에는 매니퓰레이터의 관절을 구성하는 모터에 대한 동역학 시뮬레이션을 위한 모델링을 해보자. 식 <4-1>에서 모터가 내는 힘(F)은 전류(i)에 토크상수(K_t)를 곱하고 기어비(n)를 곱하면 된다. 그러나 마찰력이 힘의 발생을 방해하기 때문에 마찰계수를 고려해야 한다.

$$F = nK_t i + f_m \dot{\theta} \qquad \langle 4\text{-}1 \rangle$$

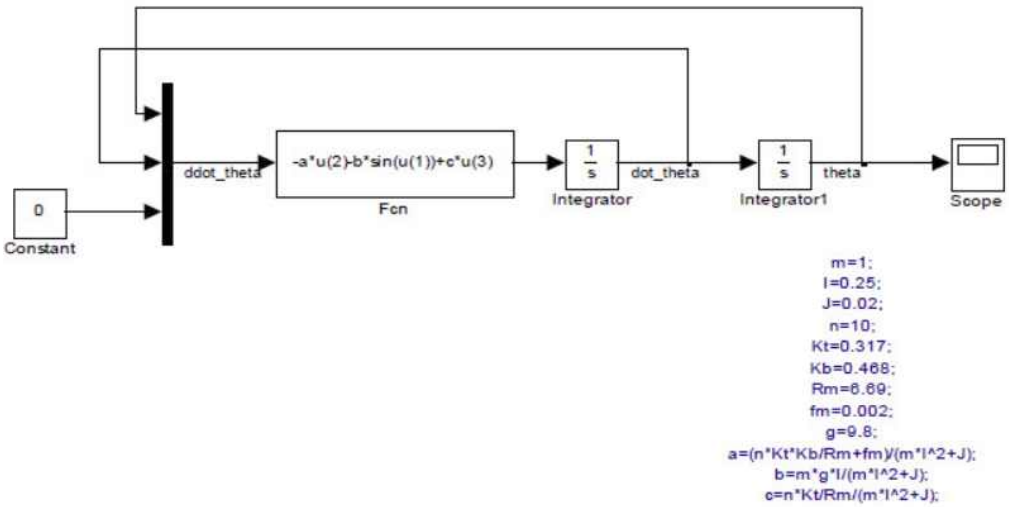
식 <4-2>는 모터의 전기적 힘에 의해 회전하게 되는, 전류와 전압의 관계를 표현한 식이다.

$$Li + R_m i = \nu + K_b \dot{\theta} \qquad \langle 4\text{-}2 \rangle$$

라그랑지 방정식에 DC 모터로 구성된 매니퓰레이터의 한 개의 관절에 대한 방정식까지 넣어서 구축한 비선형 모델을 다음과 같이 표현할 수 있다.

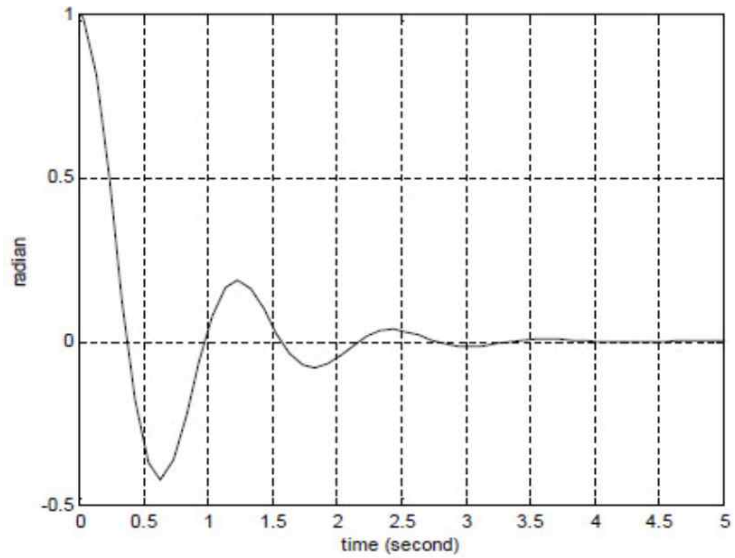
$$\ddot{\theta} = -\frac{1}{ml^2 + J}\left(\frac{nK_t K_b}{R_m} + f_m\right)\dot{\theta} - \frac{mgl}{ml^2 + J}\sin\theta + \frac{nK_t}{R_m(ml^2 + J)}\nu \qquad \langle 4\text{-}3 \rangle$$

MATLAB/Simulink를 이용하여 아래와 같이 시뮬레이션 제어 블록도를 구성할 수 있다.



[그림 4-1] 모터 동역학에 따른 제어 블록

Integrator1번에 초기치를 1라디안으로 설정하고, 제어입력을 '0'으로 만들면 아래와 같이 자유운동하는 그래프를 얻을 수 있다.

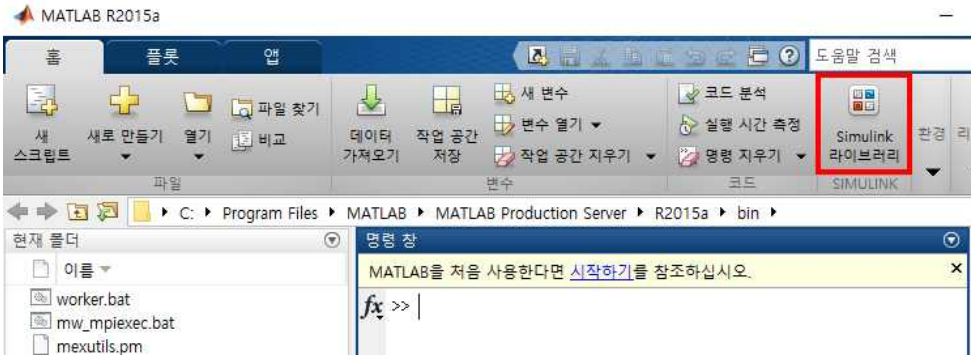


[그림 4-2] 관절의 회전에 따른 출력

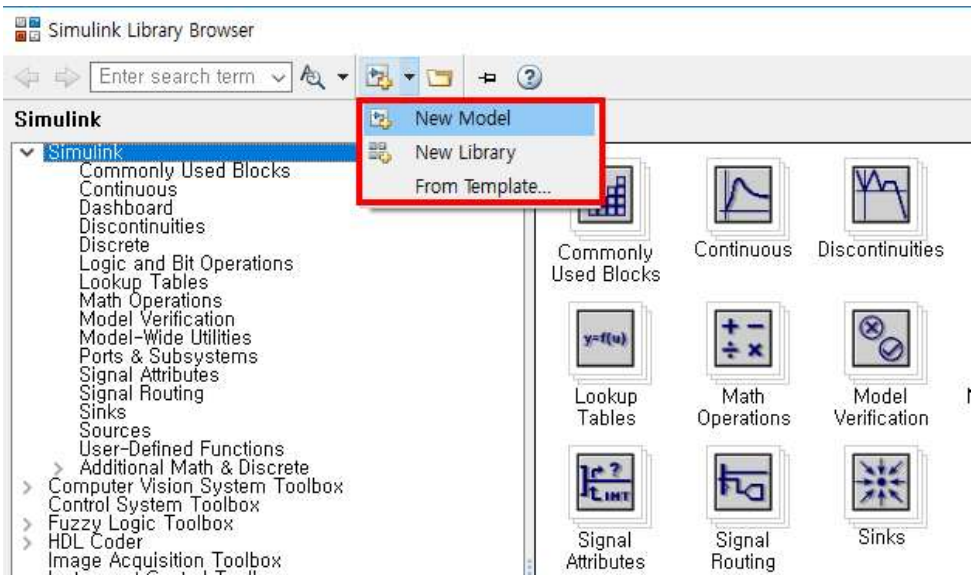
2 동역학 소프트웨어 환경 설정

simulink를 실행하는 방법은 command window에서 "Simulink" 라고 입력하거나 혹은 위에 표시된 아이콘을 누른다. 그러면 홈 탭의 simulink library 메뉴를 확인할 수 있다.

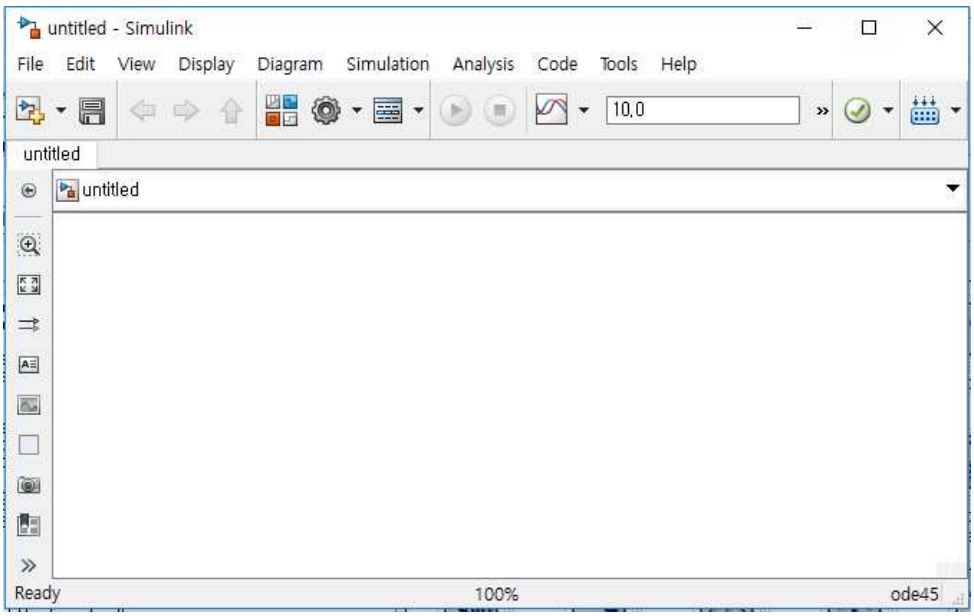
위에 표시된 simulink library browser 창에서 "new model" 열기를 클릭하면 [그림 4-5]와 같이 작업 가능한 창이 나타난다.



[그림 4-3] Simulink의 명령창



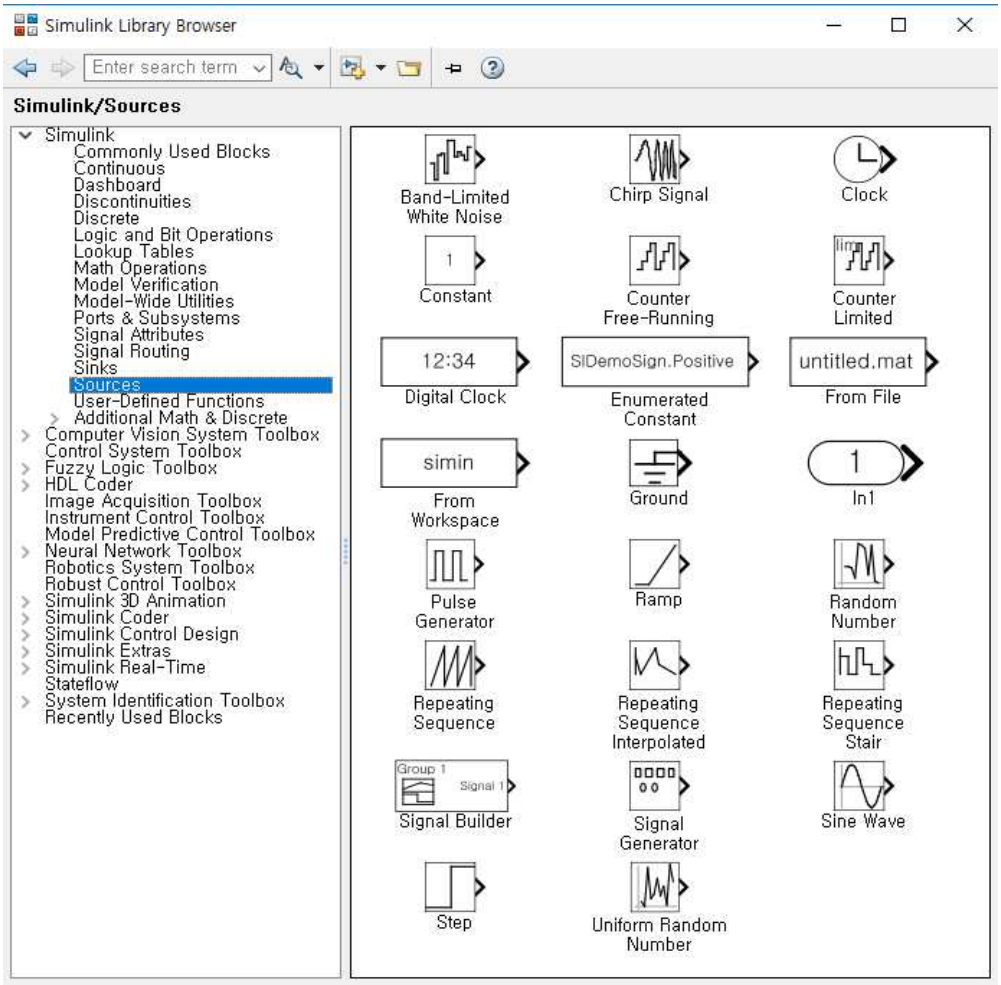
[그림 4-4] 명령창의 브라우저와 라이브러리



[그림 4-5] Simulink 작업창

1. Sources

sources 폴더에는 출력 단자만 갖는 블록들이 [그림 4-6]과 같이 모여 있는데, 시스템에 입력하고자 하는 신호를 선택하도록 한다.



[그림 4-6] Simulink의 다양한 Sources

(1) Clock

시간을 출력하는데 몇 분 몇 초 형식이 아니라 초 단위로 나타낸다.

(2) From File

mat 파일로 저장된 블록에서 데이터를 가져와 일정 시간간격으로 출력한다.

(3) Signal Builder

sin과 같이 수학적으로 간단히 수식화 할 수 없는 신호를 GUI 기반으로 편집할 수 있다.

(4) Chirp Signal

통상 system identification에서 시스템에 가하는 입력 신호로 많이 사용되며 저주파부터 고주파까지의 정현파신호를 출력할 수 있다.

(5) Constant

상수를 출력한다.

(6) From Workspace

workspace상에서 저장되어 있는 변수를 일정 시간 간격에 맞춰 출력한다.

(7) In1

subsystem 등을 만들 때 subsystem 내에서 입력을 정의할 경우 사용한다.

(8) Signal Generator

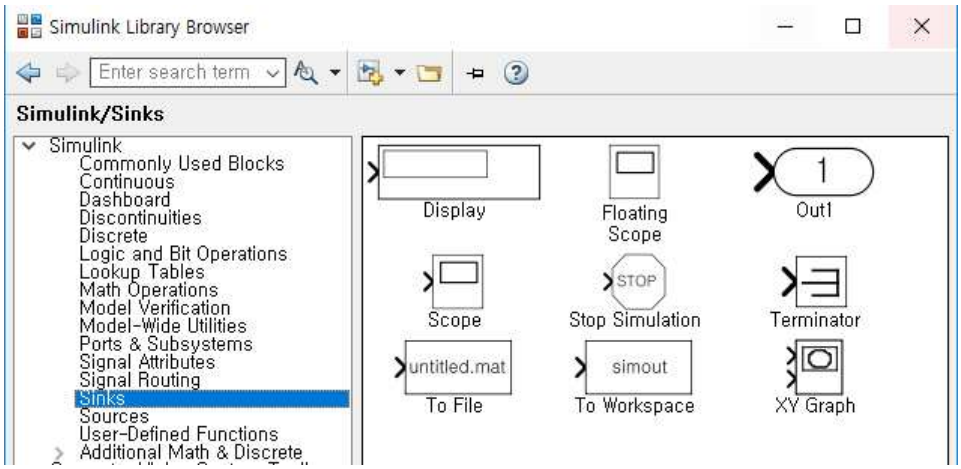
matlab상의 function generator라고 보면 된다. 다양한 신호를 생성할 수 있다.

(9) Step

원하는 시간에 step 입력을 가지도록 설정할 수 있다.

2. Sink

데이터의 출력을 어떤 형태로 표현 혹은 저장할 것인지를 정해주는 폴더이다.



[그림 4-7] Simulink의 데이터 출력 및 저장 형태 블록

(1) Out1

위의 In1처럼 subsystem 내에서 사용하는 것이다.

(2) To File

입력받은 데이터들을 mat 파일 형태로 바로 저장해 준다.

(3) XY Graph

x축 데이터와 y축 데이터를 받아 그림을 그려 준다.

(4) Scope

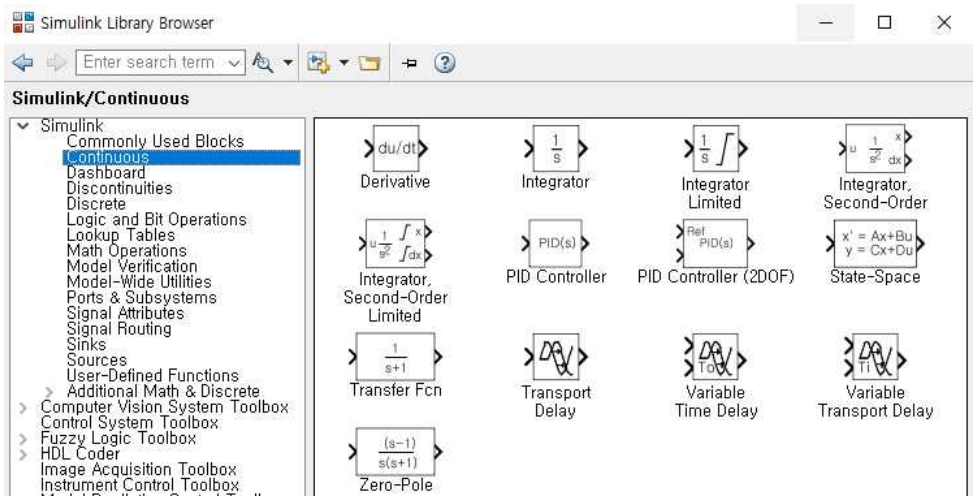
시뮬레이션 시간을 x축으로 가지고, 주어진 신호를 y축 신호로 하여 그림을 출력해 준다.

(5) To Workspace

입력된 데이터들을 workspace에 저장해 준다.

3. Continuous & Discontinuities

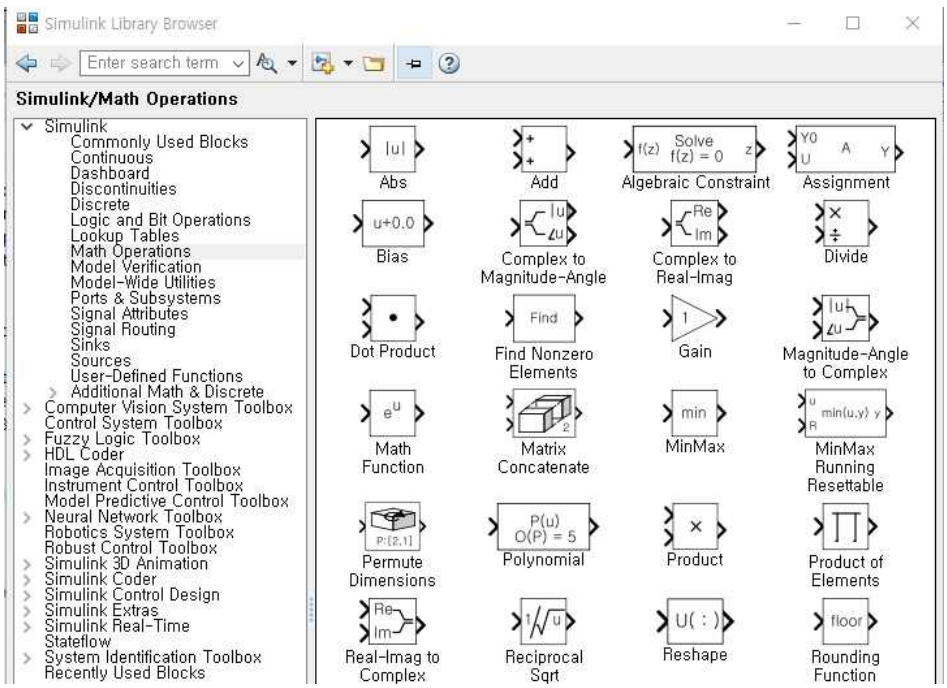
continuous 폴더에서는 주로 두 가지가 자주 사용된다. 바로 미분(derivative)과 적분(integrator) 블록이다. 적분 블록에는 초기치를 지정할 수 있도록 되어 있다.



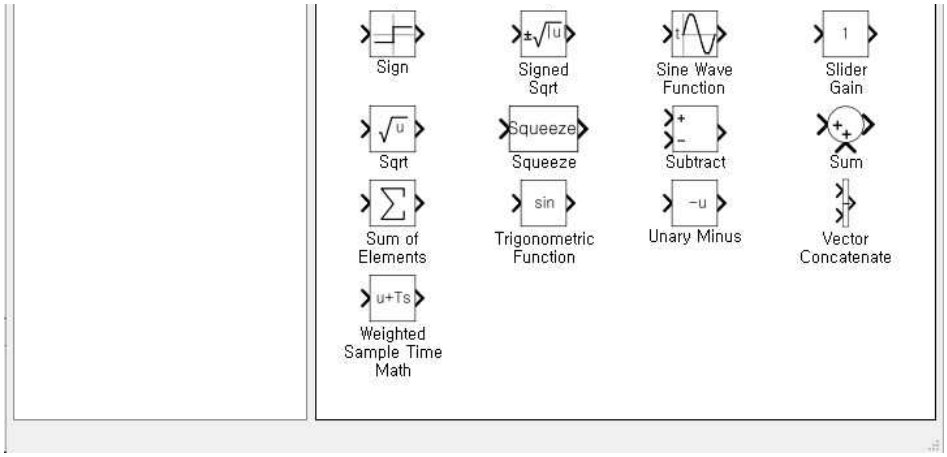
[그림 4-8] Simulink의 Continuous & Discontinuities 블록

4. Math Operation

여러 가지 수학 함수 중 블록으로 복잡하게 꾸미기 보다는, 주로 function 블록이나 embedded function 블록을 m-file로 자주 작성하므로 이 폴더에서는 몇 가지만 사용한다.



[그림 4-9] Simulink의 Math Operation 블록



[그림 4-10] Simulink의 Function과 Sum 블록

(1) Gain

신호에 특정한 값을 이용해 실수배를 수행한다.

(2) Product

두 개 이상의 신호를 곱한다.

(3) Sign

경계값(0)을 기준으로 1과 -1의 값을 가지도록 한다.

(4) Sine

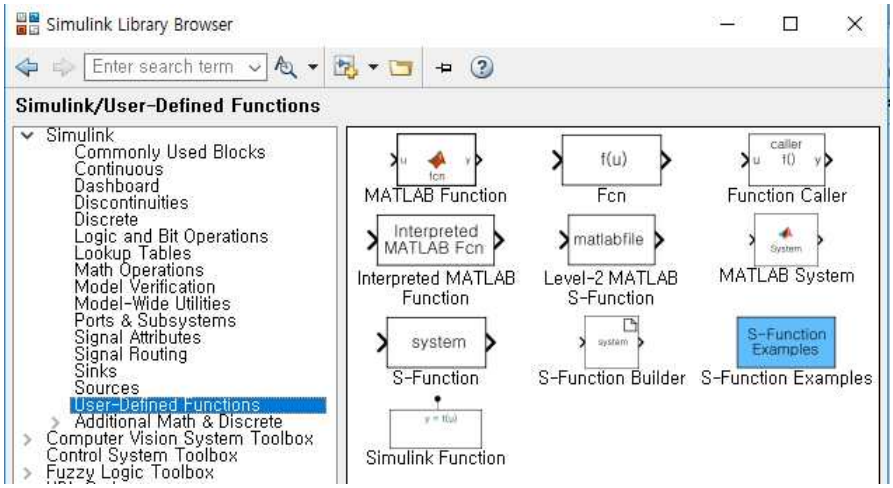
Wave Function: 정현파 신호를 만든다.

(5) Sum

두 개 이상의 신호를 더한다.

5. User-Defined Function

사용자의 사용 목적에 따른 function문을 작성할 수 있다.



[그림 4-11] Simulink의 사용자 정의 함수 블록

(1) Fcn

mux를 통해 다수의 입력을 받아 그것을 이용해서 적절한 출력을 가지도록 한 줄의 함수를 작성할 수 있다.

(2) MATLAB Fcn

matlab의 command window에서 실행할 수 있는 명령이라면 이 블록에 입력해서 그 함수의 출력을 사용할 수 있게 해준다.

(3) Embedded Matlab Function

Fcn블록은 한 줄의 명령어만 사용해서 복잡한 코드의 구현이 어려울 때가 있는데, 이 때 MATLAB Fcn을 사용할 수 있다.

수행 내용 / 동역학 소프트웨어 구현하기

재료 · 자료

- 요구사양서
- 로봇 운동학 해석 보고서
- 로봇 하드웨어 매뉴얼
- 로봇 동역학 분석 보고서
- 로봇 작업 시나리오

기기(장비 · 공구)

- 컴퓨터, 프린터
- 로봇 시뮬레이션 프로그램(MATLAB 등)

안전 · 유의 사항

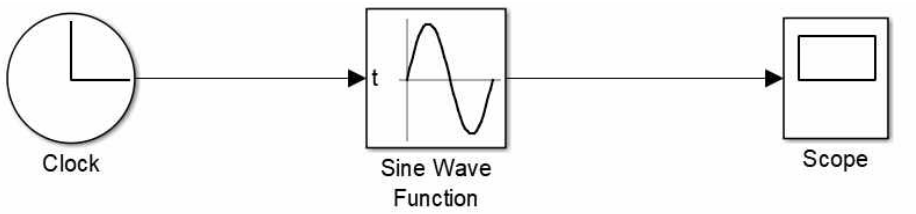
- 로봇 시뮬레이션을 수행할 때에는 범용적으로 사용되는 소프트웨어를 선택해야 한다.

수행 순서

1 시뮬레이션의 기초 파악하기

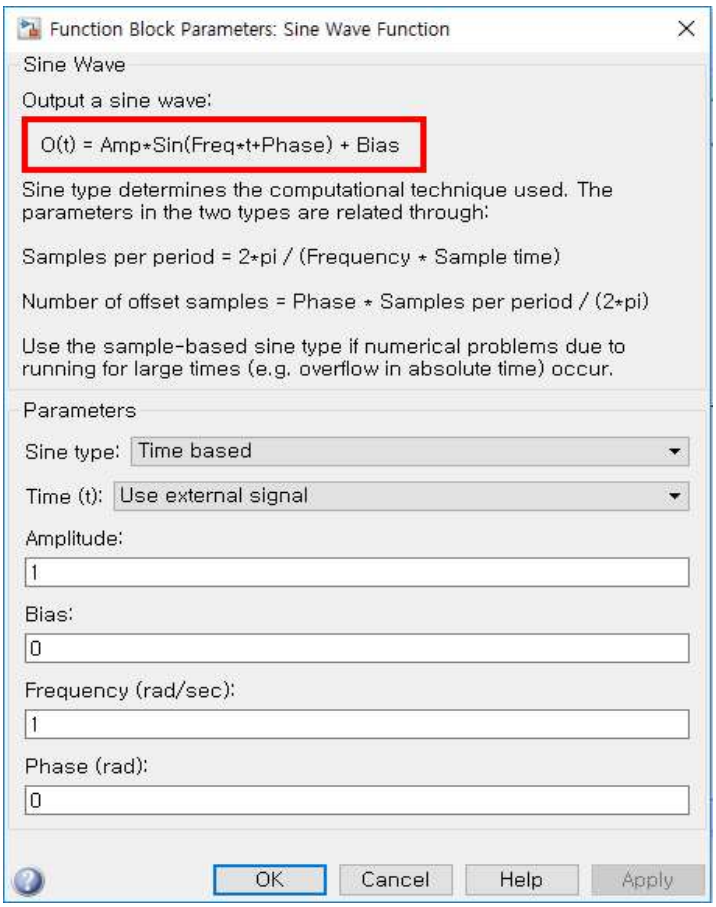
다음과 같은 sine 함수를 출력하는 예제를 통해 simulink의 사용해 본다.

우선 함수를 구성하기 위하여 작업창 untitled.mdl을 만들고 그 안에 블록을 [그림 4-12]와 같이 구성해 보도록 한다. library에서 마우스로 왼쪽 버튼으로 블록을 누르고 새로 열린 작업창에 가져다 놓는다. 왼쪽 버튼을 누르고 선을 그어 주거나, clock 블록을 마우스 왼쪽 버튼을 누르고 키보드에서 Ctrl키를 누른 상태로 다음 블록을 마우스 오른쪽 버튼을 눌러가면서 클릭하여 [그림 4-12]와 같은 제어 흐름선을 연결해 본다.

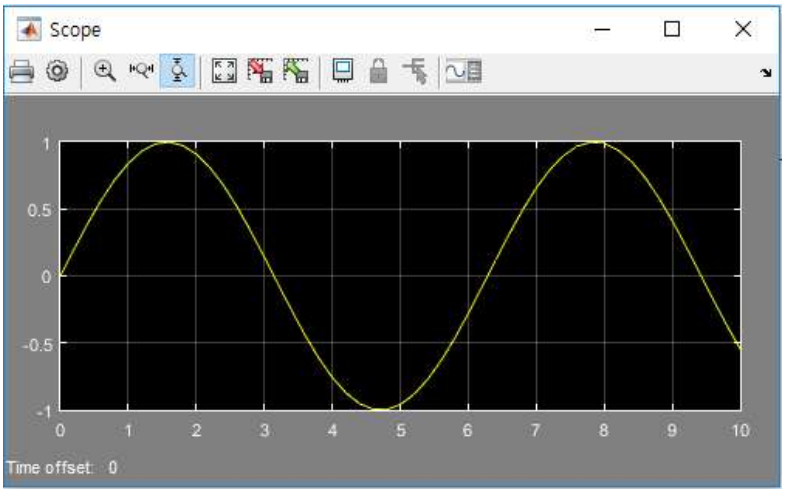


[그림 4-12] Sine 함수 출력 Simulink 함수

[그림 4-13]과 같이 sine wave function을 더블 클릭하면 위의 대화창이 나타나고 표시된 부분을 보면 신호폭, 주파수, 위상 등을 설정할 수 있다. 본 대화창에서 진폭, 바이어스, 주파수, 위상에 대한 파라미터 값을 설정하고 ok 버튼을 눌러 저장 후 작업창의 상단에 플레이 버튼으로 시뮬레이션을 수행하게 된다. 이때 scope창을 더블클릭하면 [그림 4-14]와 같은 결과를 확인할 수 있다.



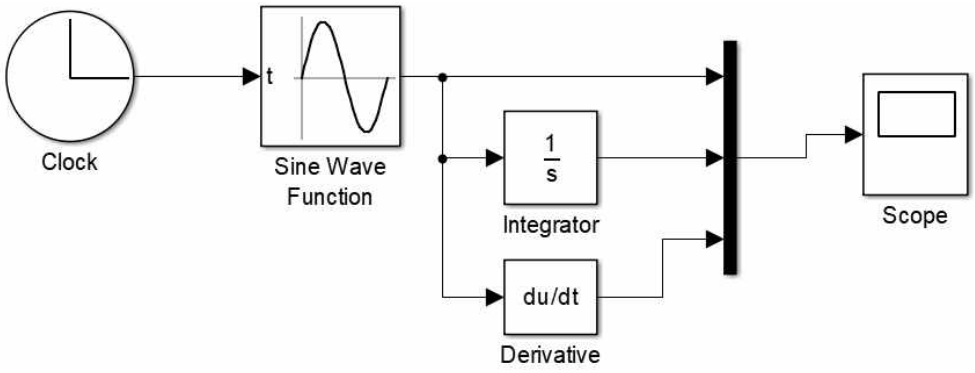
[그림 4-13] Sine 함수에 대한 파라미터 설정



[그림 4-14] Sine 파 시뮬레이션 결과

2 시뮬레이션 확장하기

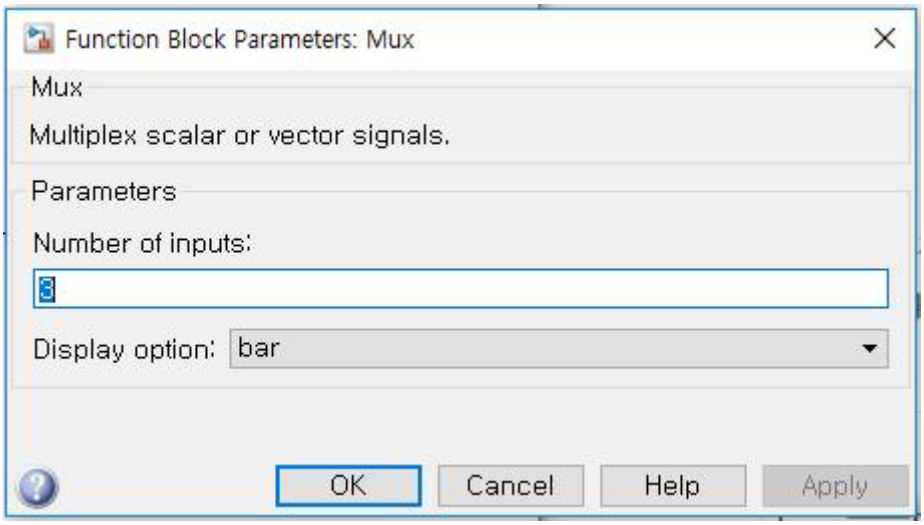
첫 번째 예제를 확장한 시스템에 대한 시뮬레이션을 해보도록 한다. 아래 그림과 같이 입력신호에 대한 미분과 적분으로 원신호와 같이 세 개의 신호를 한 scope창에서 출력해 보도록 한다. 이때 mux라는 블록을 사용하여 적용해 본다.



[그림 4-15] 미분 및 적분변수를 포함 Sine 함수 출력 Simulink 함수

각 명령에 대한 블록을 library에서 가져와 더블클릭하면 아래와 같이 나타난다. 여기서 세 갈래의 신호선으로 나누는 방법은 마우스 오른쪽으로 우선 한선을 드래그하여 연결한 뒤, 왼쪽으로 분개해서 연결한다.

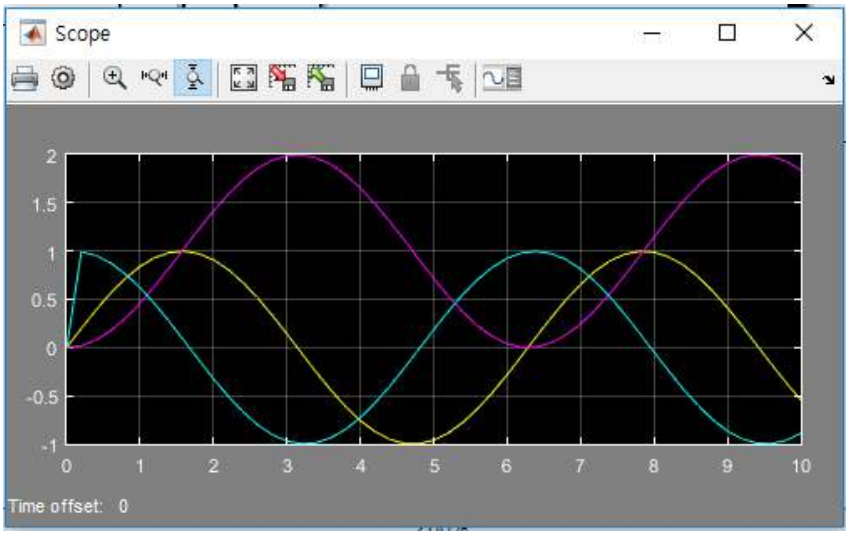
신호의 결합을 위해서 Signal Routing Library에 있는 Mux블록을 이용하여 선을 모으도록 한다. 이때 입력 신호가 세 개이므로 Mux 블록의 파라미터 창에서 Number of inputs 항을 이용하여 “3“이라고 입력하고 시뮬레이션을 동작시켜 본다.



[그림 4-16] Mux 블록의 파라미터 설정

mux 블록으로 scope 블록에 두 개 이상의 신호를 넣은 경우 청록색이 제일 위의 신호, 그 다음이 노랑, 그 다음이 핑크색으로 표현됨을 확인한다.

scope의 axes 영역에서 마우스 오른쪽 버튼을 클릭하여 axes properties에서 Y-min, Y-max를 설정할 수 있다.



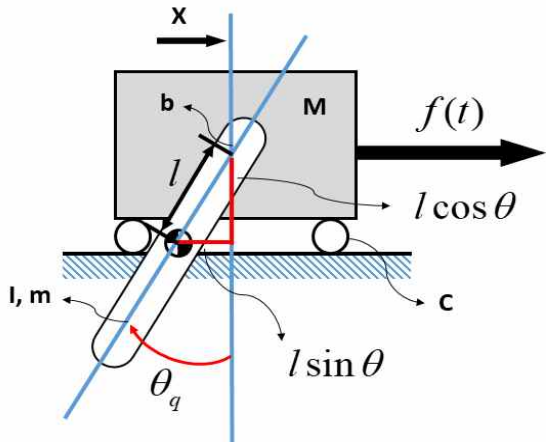
[그림 4-17] Sine파의 미분, 적분 시뮬레이션 결과

3 동역학 시뮬레이션을 실행하기

1. 카트형 진자시스템의 시뮬레이션

본 절에서는 시스템(지면에서 수평으로 이동하는 대차)을 역학적으로 해석하여 미분방정식의 형태로 표현한 동역학 방정식을 이용한 시뮬레이션을 구현해 보도록 한다. 실제로는 복잡한 형태를 가지게 되지만, 동역학에 대한 결과식을 이용하여 MATLAB에서 구현한 시

뮬레이션을 실행한다. 아래 시스템은 카트형 진자시스템의 좌표계를 나타낸다.



[그림 4-18] 카트형 진자시스템의 좌표계

이때 사용한 동역학 방정식은 다음과 같다.

$$\begin{aligned}(M+m)\ddot{x}+c\dot{x}-ml\ddot{\theta}&=f(t)\\(ml^2+I)\theta-ml\ddot{x}+mgl\theta+f_\theta\dot{\theta}&=0\end{aligned} \qquad \langle 4\text{-}4\rangle$$

위 식을 그대로 simulink로 꾸미면, algebraric loop가 있다는 경고문을 보게 된다. 왜냐하면 자기 자신을 참조($\ddot{x}$, $\ddot{\theta}$)하기 때문인데, 그래서 아래와 같이 매트릭스 형태로 최고차 미분항으로 정리할 필요가 있다.

$$\begin{bmatrix}M+m & -ml\\ -ml & ml^2+I\end{bmatrix}\begin{bmatrix}\ddot{x}\\ \ddot{\theta}\end{bmatrix}=\begin{bmatrix}-c\dot{x}+F\\ -mgl\theta-f_\theta\end{bmatrix} \qquad \langle 4\text{-}5\rangle$$

$$\begin{bmatrix}\ddot{x}\\ \ddot{\theta}\end{bmatrix}=\begin{bmatrix}M+m & -ml\\ -ml & ml^2+I\end{bmatrix}^{-1}\begin{bmatrix}-c\dot{x}+F\\ -mgl\theta-f_\theta\end{bmatrix}$$

식 <4-5>을 간단히 행렬로 표현하고, 역행렬을 이용해서 다음과 같이 정리할 수 있다.

$$\begin{bmatrix}\ddot{x}\\ \ddot{\theta}\end{bmatrix}=\frac{1}{(M+m)(ml^2+I)+m^2l^2}\begin{bmatrix}ml^2+I & ml\\ ml & M+m\end{bmatrix}\begin{bmatrix}-c\dot{x}+F\\ -mgl\theta-f_\theta\end{bmatrix} \qquad \langle 4\text{-}6\rangle$$

그리고 우측 항의 역행렬을 정리하면 다음과 같이 표현할 수 있다.

$$\begin{bmatrix}\ddot{x}\\ \ddot{\theta}\end{bmatrix}=\frac{1}{(M+m)(ml^2+I)+m^2l^2}\begin{bmatrix}(ml^2+I)(-c\dot{x}+F)+ml(-mgl\theta-f_\theta\dot{\theta})\\ ml(-c\dot{x}+F)+(M+m)(-mgl\theta-f_\theta\dot{\theta})\end{bmatrix} \quad \langle 4\text{-}7\rangle$$

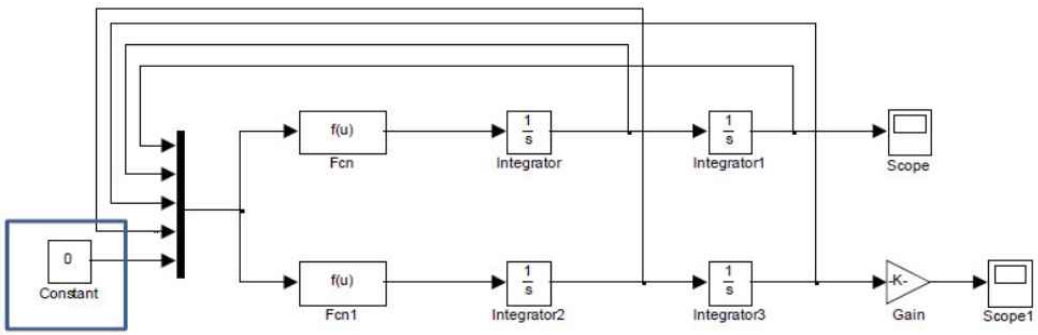
이때 시스템에서 갖고 있는 상수 값 및 파라미터를 식 <4-8>와 같이 설정하여 정리하면 식 <4-9>과 같이 표현된다.

$$M=5, m=1, l=1, g=9.8, c=0.01, f_\theta=0.01, I=0.02 \quad \langle 4\text{-}8\rangle$$

$$\begin{bmatrix}\ddot{x}\\ \ddot{\theta}\end{bmatrix}=\frac{1}{6.012}\begin{bmatrix}-0.0102\dot{x}-9.8\theta-0.01\dot{\theta}+1.02F\\ -0.01\dot{x}-58.8\theta-0.06\dot{\theta}+F\end{bmatrix} \quad \langle 4\text{-}9\rangle$$

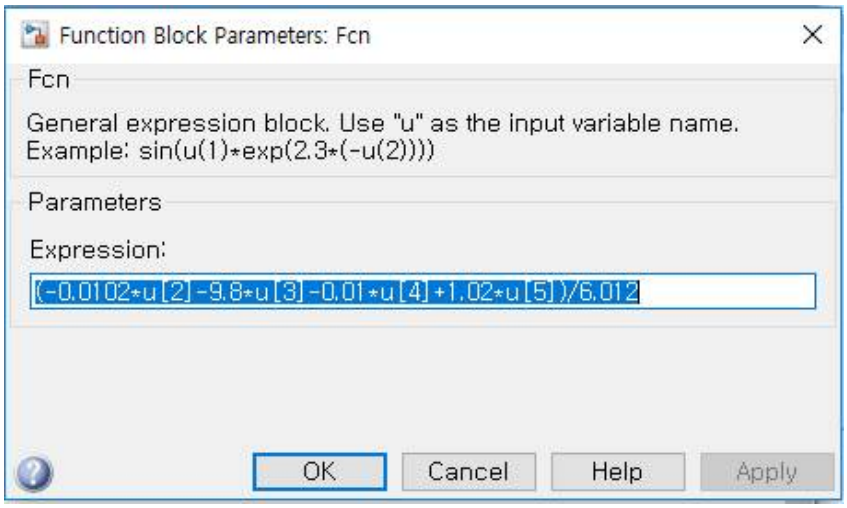
2. Simulink로 동역학

제어 블록 라이브러리를 이용하여 소스를 아래와 나열한다. 위에 표시된 부분은 제어입력 F인데, 지금은 제어입력을 구현하지 않으므로 '0'을 인가하면 된다.



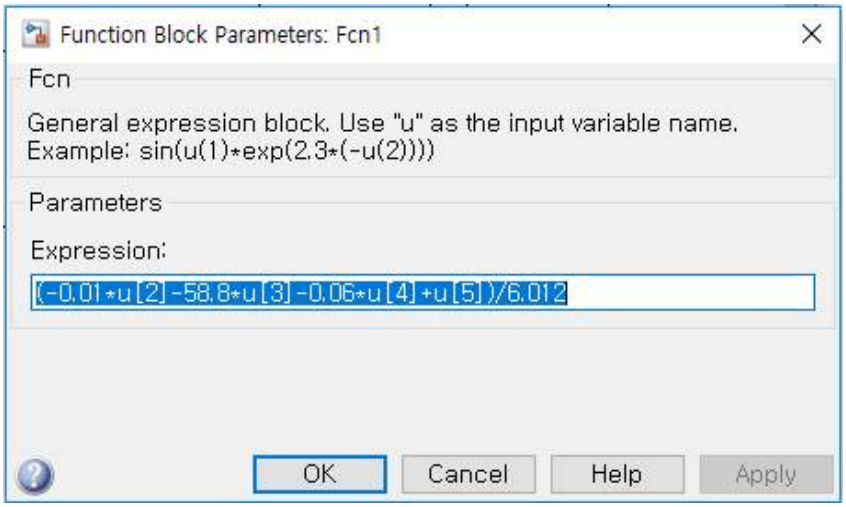
[그림 4-19] 카트형 진자시스템의 제어블록

위쪽 블록 행은 $\ddot{x}$ 함수이며, Fcn 블록에 대해서는 아래와 같이 값을 넣어주면 된다.



[그림 4-20] 변수 x에 대한 입력함수

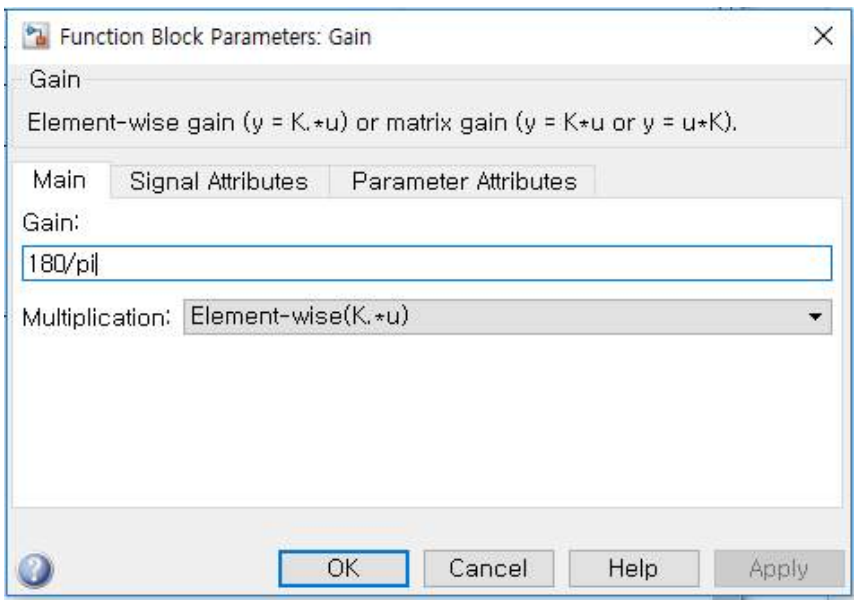
아래 블록 행은 $\ddot{\theta}$ 함수이며, Fcn1 블록에 대해서는 아래와 같이 값을 넣어주면 된다.



[그림 4-21] 변수 θ에 대한 입력함수

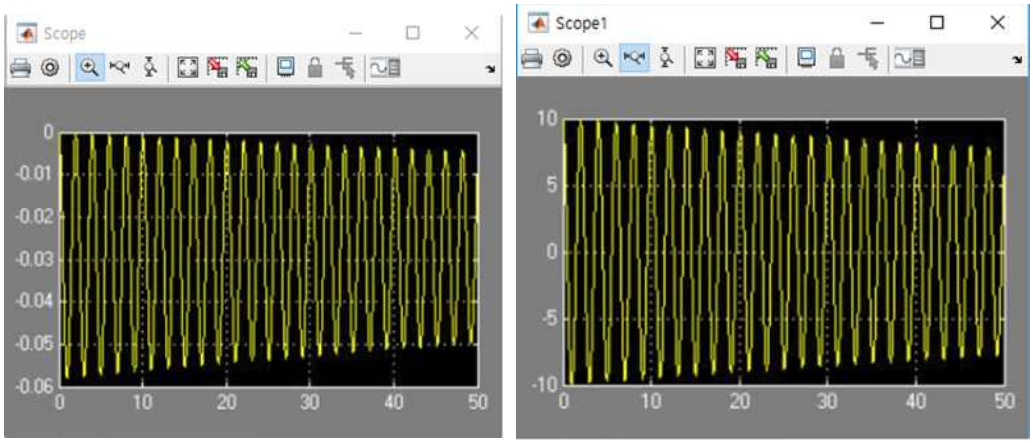
simulink 내부에서는 라디안 단위로 계산하기 때문에 10도의 초기값의 형태인 "10*pi/180" 의 값으로 integrator3에 입력한다.

그리고 gain은 라디안으로 표시되는 각도를 scope에서 도 단위로 확인할 수 있도록 변환하여 아래와 같이 입력한다.



[그림 4-22] 게인값 조정 입력창

시뮬레이션을 실행하면 다음과 같은 결과를 확인할 수 있다.



(a) x값에 대한 변화 (b) θ값에 대한 변화

[그림 4-23] 카트형 진자시스템의 시뮬레이션 결과