Advanced Robotics

-Computer Problem Set 6-

2018. 06. 08.

미래융합기술학과

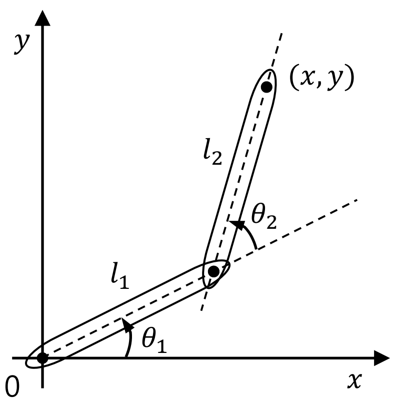
20187087 조은기

Content

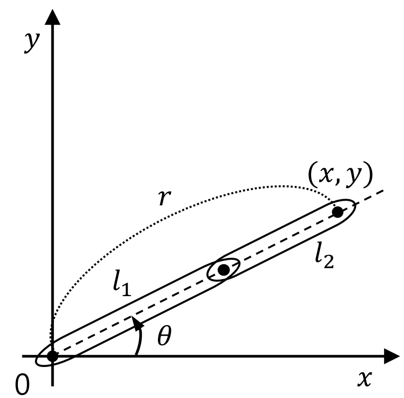
1. Introduction

① 프로젝트 목적

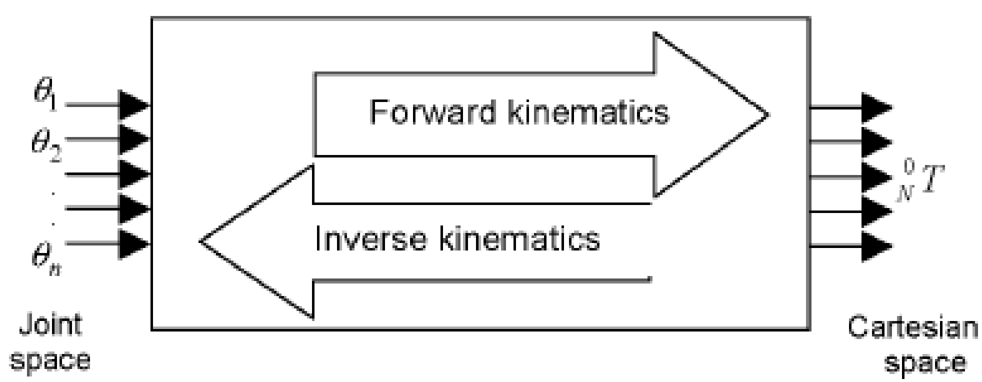
- Forward Kinematics, Inverse Kinematics, Jacobian, Runge-Kutta법 이해
- 2 DOF Manipulator 시스템 설계 및 Simulation



< $\theta_1 - \theta_2$ Planar Robot >



< $\theta - r$ Planar Robot >

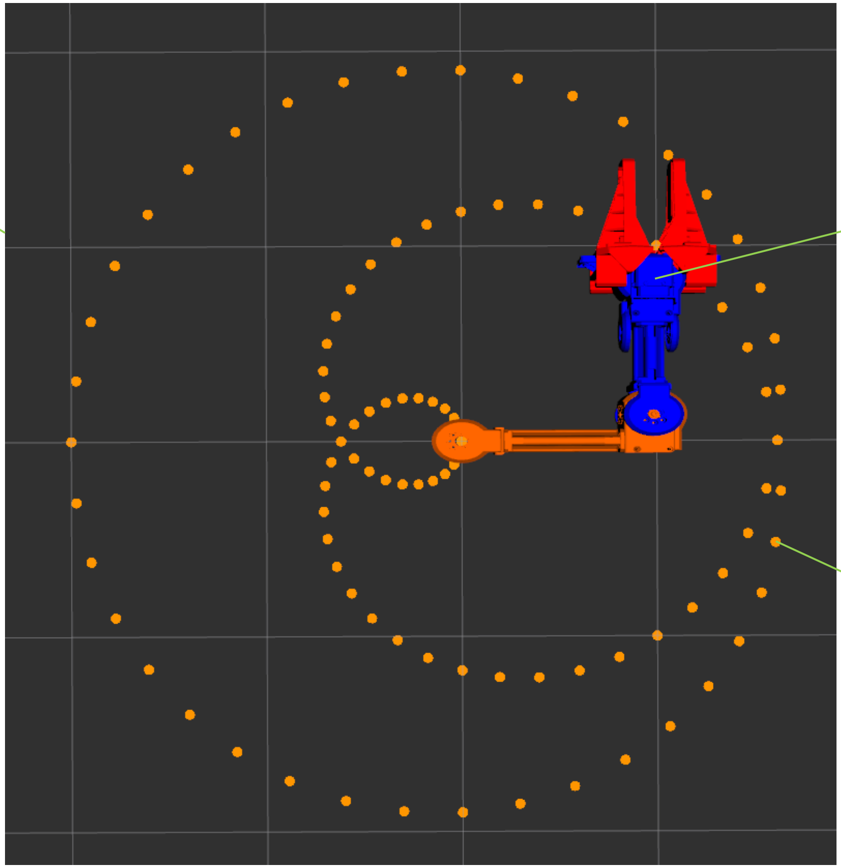


1. Introduction

② 사용 Tool 및 라이브러리

- Ubuntu
- GCC Compiler
- ROS (Robot Operating System)
- PCL (Point Cloud Library)

Rviz (ROS)



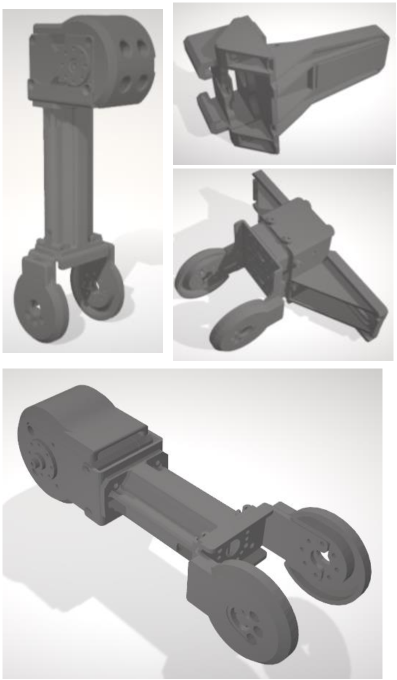
3D Model file

point cloud

2. Algorithm

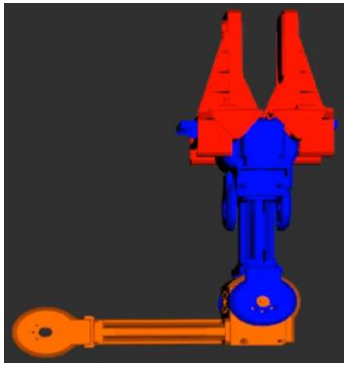
① Manipulator 설계

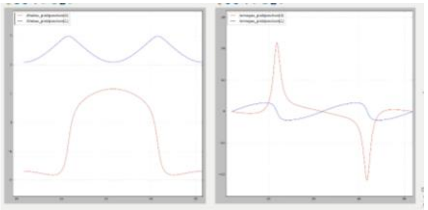
Manipulator 3D Modeling (create .stl files) → **Joint / Link Design** (ROS URDF) → **Visualization** (ROS Rviz, rqt_plot)



```xml
<link name="pan_link">
    <visual>
        <geometry>
            <cylinder length="0.4" radius="0.04"/>
        </geometry>
        <origin rpy="0 0 0" xyz="0 0 0.09"/>
        <material name="red">
            <color rgba="0 0 1 1"/>
        </material>
    </visual>

    <collision>
        <geometry>
            <cylinder length="0.4" radius="0.06"/>
        </geometry>
        <origin rpy="0 0 0" xyz="0 0 0.09"/>
    </collision>

    <inertial>
        <mass value="1"/>
        <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
    </inertial>
</link>

<joint name="tilt_joint" type="revolute">
```

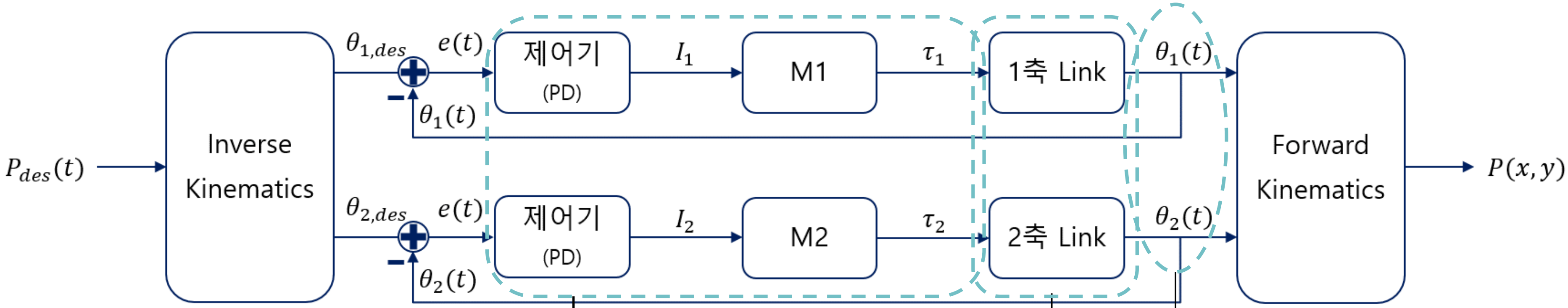
```xml
<joint name="tilt_joint" type="revolute">
    <parent link="pan_link"/>
    <child link="tilt_link"/>
    <origin xyz="0 0 0.2"/>
    <axis xyz="0 1 0" />
    <limit effort="300" velocity="0.1" lower="-4.71239" upper="-1.570796"/>
    <dynamics damping="50" friction="1"/>
</joint>
```





2. Algorithm

② 시스템 수식



$$\theta_2 = \mathrm{atan2}(\sin\theta_2, \cos\theta_2)$$

$$\theta_1 = \mathrm{atan2}(\sin\theta_1, \cos\theta_1)$$

$$\tau = K_p e(t) + K_D \frac{de(t)}{dt}$$

$$P_x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$

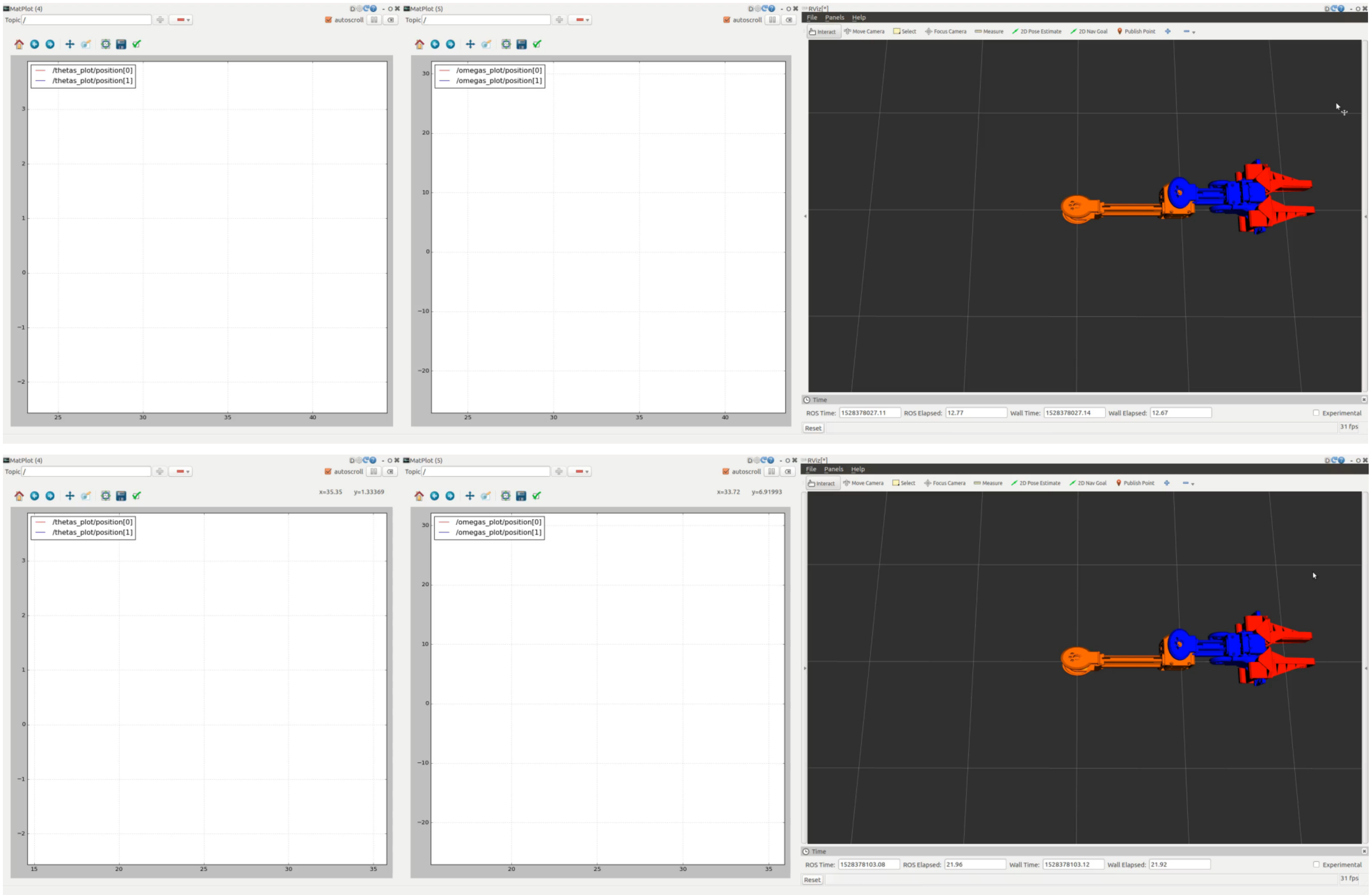
$$P_y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

$$\begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} = \underbrace{\begin{bmatrix} -\sin(\theta_1) - \sin(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) \\ \cos(\theta_1) + cos(\theta_1 + \theta_2) & cos(\theta_1 + \theta_2) \end{bmatrix}^{-1}}_{J} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

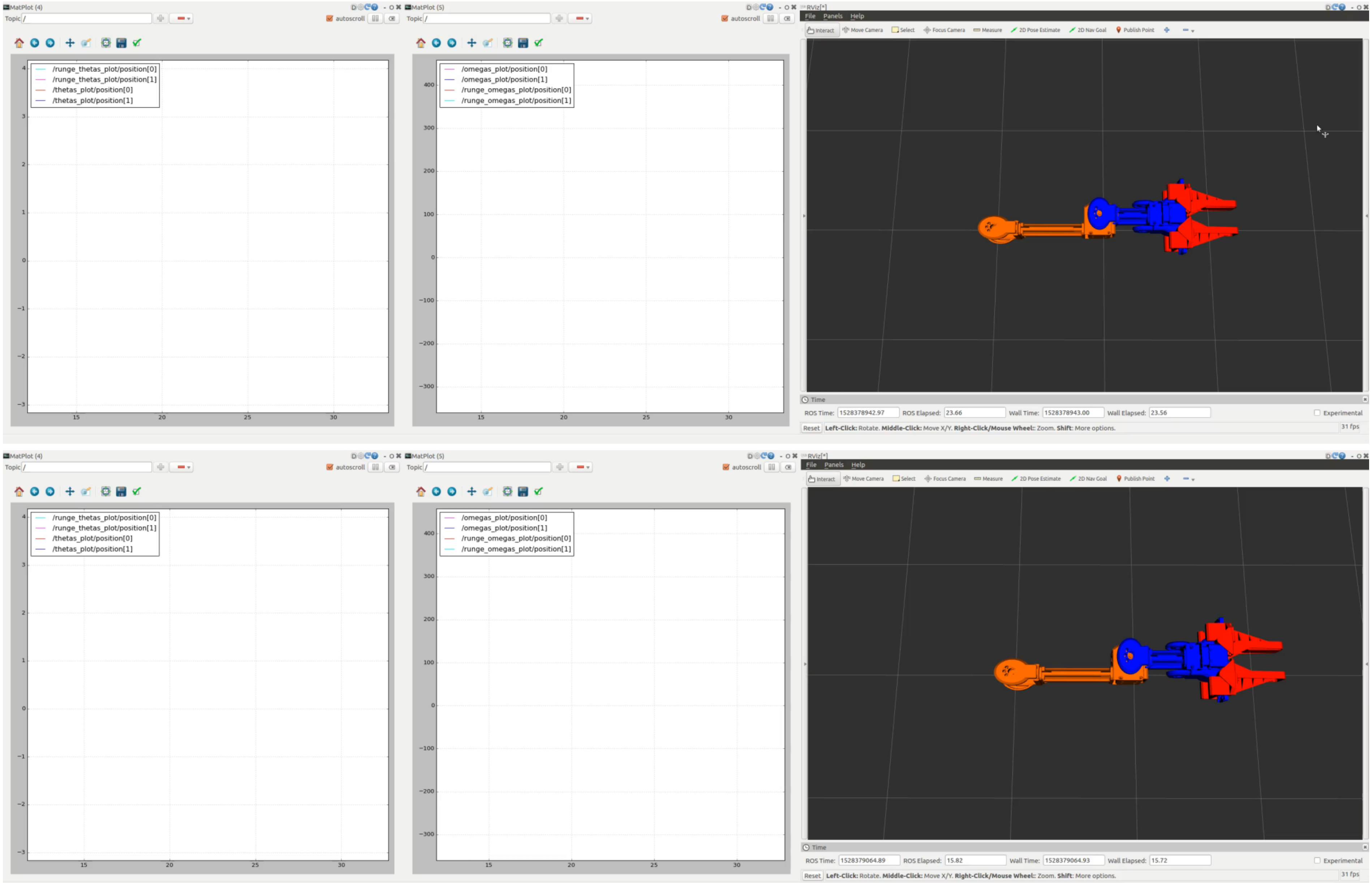
$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$$

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \\ \dot{X}_4 \end{bmatrix} = \begin{bmatrix} X_2 \\ \frac{3}{4}[K_{p1}(\theta_{1des} - \theta_1) + K_{D1}(\omega_{1des} - \omega_1)] \\ X_4 \\ \frac{15}{4}[K_{p2}(\theta_{2des} - \theta_2) + K_{D2}(\omega_{2des} - \omega_2)] \end{bmatrix}$$

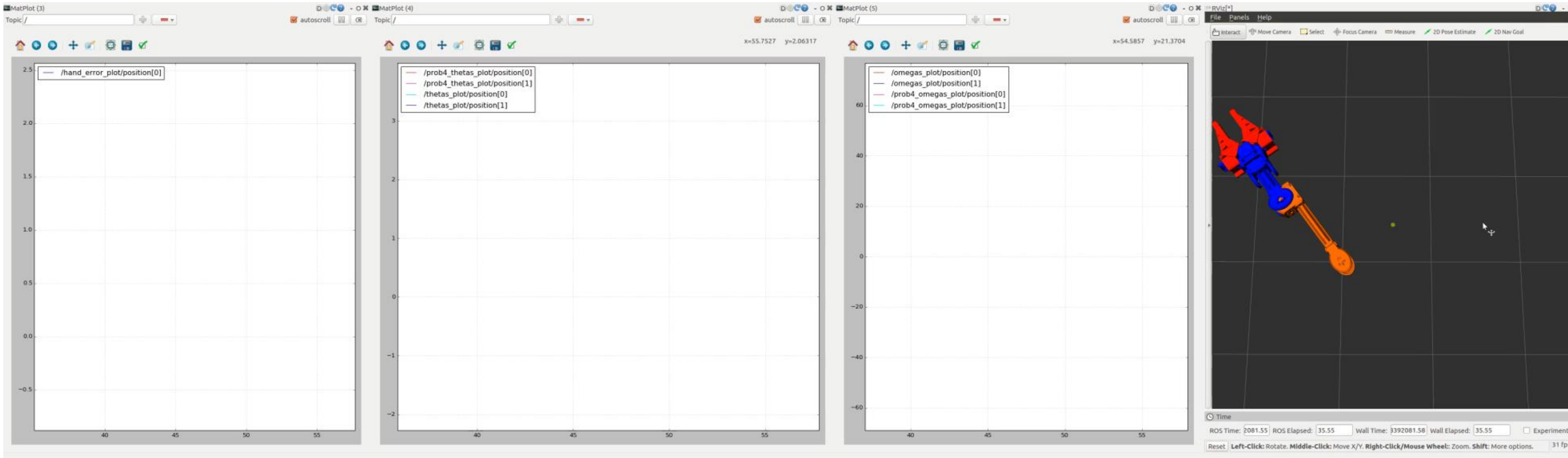
3. Demo _Problem 1

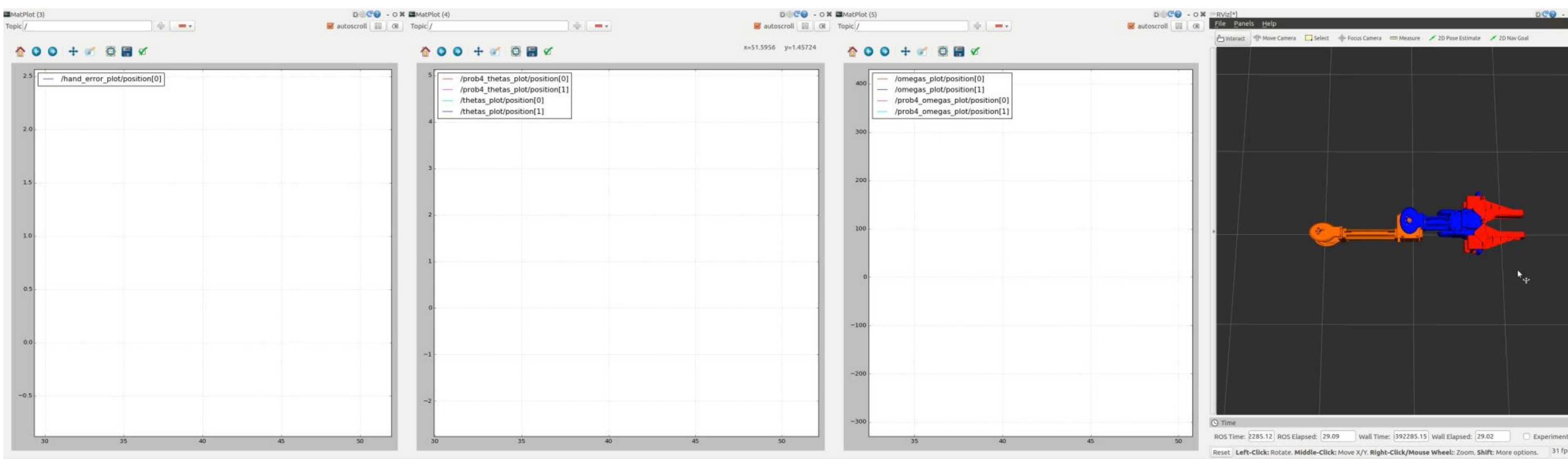


3. Demo _Problem 3



3. Demo _Problem 4





4. Source Analysis

```
// Problem 1          // 궤적 그리기 + Inverse Kinematics + Jacobian(omega1,2)
// 궤적 그리기
float x_ = (0.7 * cos(2 * PI * f * time_)) + 0.1;
float y_ = (-0.7 * cos(2 * PI * f * time_)) + 0.1;

x[j] = x_;
y[j] = y_;

pt.x = x[j], pt.y = y[j], pt.z = 0;
pt.r = 255, pt.g = 150, pt.b = 0;
position_cluster1.push_back(pt);

// Inverse Kinematics
float c2 = (pow(x[j], 2) + pow(y[j], 2) - 2) / 2;
float s2 = sqrt(1 - pow(c2, 2));

if(atan2(s2, c2) > 0 && atan2(s2, c2) < PI)
    joint_position[1] = atan2(s2, c2);
else
    joint_position[1] = -atan2(s2, c2);

float c1 = ((1 + c2) * x[j] + s2 * y[j]) / (pow((1 + c2), 2) + pow(s2, 2));
float s1 = (-1 * s2 * x[j] + (1 + c2) * y[j]) / (pow((1 + c2), 2) + pow(s2, 2));
joint_position[0] = atan2(s1, c1);
// Jacobian
double J[2][2] = { { (-L1 * sin(joint_position[0]) - L2 * sin(joint_position[0] + joint_position[1])), (-L2 * sin(joint_position[0] + joint_position[1])) },
           { (L1 * cos(joint_position[0]) + L2 * cos(joint_position[0] + joint_position[1])), (L2 * cos(joint_position[0] + joint_position[1])) } };
double(*J_inverse)[2] = inverse(J);

float dx = -0.7 * sin(2 * PI * f * time_) * 2 * PI * f;
float dy = 0.7 * sin(2 * PI * f * time_) * 2 * PI * f;

d_theta[0] = J_inverse[0][0] * dx + J_inverse[0][1] * dy;
d_theta[1] = J_inverse[1][0] * dx + J_inverse[1][1] * dy;
```

$$\cos\theta_2 = \frac{x^2 + y^2 - (l_1^2 + l_2^2)}{2 l_1 l_2}$$

$$\sin\theta_2 = \pm\sqrt{1 - \cos^2\theta_2}$$

$$\theta_2 = \mathrm{atan2}(\sin\theta_2, \cos\theta_2)$$

$$\cos\theta_1 = \frac{(l_1 + l_2 c_2)x + l_2 s_2 y}{(l_1 + l_2 c_2)^2 + (l_2 s_2)^2}$$

$$\sin\theta_1 = \frac{-l_2 s_2 x + (l_1 + l_2 c_2)y}{(l_1 + l_2 c_2)^2 + (l_2 s_2)^2}$$

$$\theta_1 = \mathrm{atan2}(\sin\theta_1, \cos\theta_1)$$

$$J = \begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix}$$

$$\Delta\theta = J^{-1} \cdot \Delta P$$

4. Source Analysis

```
double(*inverse(double a[2][2]))[2]{
    static double result[2][2];
    double f = a[0][0] * a[1][1] - a[0][1] * a[1][0];

    result[0][0] = a[1][1] / f;
    result[0][1] = -1 * a[0][1] / f;
    result[1][0] = -1 * a[1][0] / f;
    result[1][1] = a[0][0] / f;

    return result;
}
```

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

```
// Problem 2
float Kp1 = pow((16 * PI), 2) * 4 / 3;
float Kd1 = 2 * 16 * PI * 4 / 3;
float Kp2 = pow((16 * PI), 2) * 4 / 15;
float Kd2 = 2 * 16 * PI * 4 / 15;
```

$$\tau = K_p e(t) + K_D \frac{de(t)}{dt}$$

4. Source Analysis

```
// Problem 3        // Runge-Kutta + Forward Kinematics
float tau1 = Kp1 * (joint_position[0] - theta1_runge) + Kd1 * (d_theta[0] - omega1_runge);
float tau2 = (Kp2 * (joint_position[1] - theta2_runge)) + (Kd2 * (d_theta[1] - omega2_runge));

float X1 = joint_position[0];
float X2 = d_theta[0];
float X3 = joint_position[1];
float X4 = d_theta[1];

float X1_dot = X2;
float X2_dot = 0.75 * tau1;
float X3_dot = X4;
float X4_dot = 3.75 * tau2;

float K1 = 0.01 * omega1_runge;
float L1 = 0.01 * X2_dot;
float B1 = 0.01 * omega2_runge;
float C1 = 0.01 * X4_dot;

float K2 = 0.01 * (omega1_runge + L1);
float B2 = 0.01 * (omega2_runge + C1);
float L2 = 0.01 * (X2_dot);
float C2 = 0.01 * (X4_dot);

theta1_runge += (K1 + K2) * 0.5;
theta2_runge += (B1 + B2) * 0.5;
omega1_runge += (L1 + L2) * 0.5;
omega2_runge += (C1 + C2) * 0.5;

tau1 = Kp1 * (joint_position[0] - theta1_runge) + Kd1 * (d_theta[0] - omega1_runge);
tau2 = (Kp2 * (joint_position[1] - theta2_runge)) + (Kd2 * (d_theta[1] - omega2_runge));
float X2_dot_2 = 0.75 * tau1;
float X4_dot_2 = 3.75 * tau2;

L2 = 0.01 * (X2_dot_2);
C2 = 0.01 * (X4_dot_2);

omega1_runge += (L1 + L2) / 2;
omega2_runge += (C1 + C2) / 2;
```

$$\begin{bmatrix}\theta_1\\ \theta_2\\ \theta_3\\ \theta_4\end{bmatrix} = \begin{bmatrix}X_1\\ X_2\\ X_3\\ X_4\end{bmatrix}$$

$$\begin{bmatrix}\dot{X}_1\\ \dot{X}_2\\ \dot{X}_3\\ \dot{X}_4\end{bmatrix} = \begin{bmatrix} X_2 \\ \frac{3}{4}[K_{p1}(\theta_{1des}-\theta_1)+K_{D1}(\omega_{1des}-\omega_1)] \\ X_4 \\ \frac{15}{4}[K_{p2}(\theta_{2des}-\theta_2)+K_{D2}(\omega_{2des}-\omega_2)]\end{bmatrix}$$

4. Source Analysis

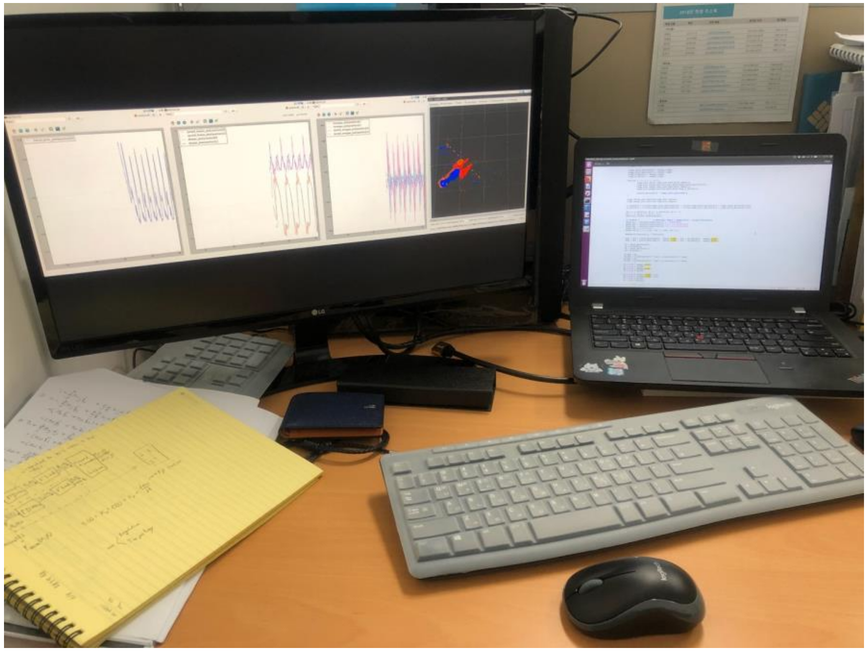
```
// Problem 4        // Nonlinear Robot + Runge-Kutta + Forward Kinematics
float H11 = cos(joint_position[1]) + 1.6666666666;
float H12 = cos(joint_position[1]) * 0.5 + 0.3333333333;
float H21 = cos(joint_position[1]) * 0.5 + 0.3333333333;
float H22 = 0.3333333333;
double H[2][2] = { { H11, H12 }, { H21, H22 } };

double(*H_inverse)[2] = inverse(H);

tau1 = Kp1 * (joint_position[0] - theta1_prob4) + Kd1 * (d_theta[0] - omega1_prob4);
tau2 = Kp2 * (joint_position[1] - theta2_prob4) + Kd2 * (d_theta[1] - omega2_prob4);

X1 = joint_position[0];
X2 = d_theta[0];
X3 = joint_position[1];
X4 = d_theta[1];

X1_dot = X2;
X2_dot = H_inverse[0][0] * tau1 + H_inverse[0][1] * tau2;
X3_dot = X4;
X4_dot = H_inverse[1][0] * tau1 + H_inverse[1][1] * tau2;

K1 = 0.01 * omega1_prob4;
L1 = 0.01 * X2_dot;
B1 = 0.01 * omega2_prob4;
C1 = 0.01 * X4_dot;

K2 = 0.01 * (omega1_prob4 + L1);
B2 = 0.01 * (omega2_prob4 + C1);
L2 = 0.01 * X2_dot;
C2 = 0.01 * X4_dot;

theta1_prob4 += (K1 + K2) * 0.5;
theta2_prob4 += (B1 + B2) * 0.5;
omega1_prob4 += (L1 + L2) * 0.5;
omega2_prob4 += (C1 + C2) * 0.5;

tau1 = Kp1 * (joint_position[0] - theta1_prob4) + Kd1 * (d_theta[0] - omega1_prob4);
tau2 = Kp2 * (joint_position[1] - theta2_prob4) + Kd2 * (d_theta[1] - omega2_prob4);

X2_dot_2 = H_inverse[0][0] * tau1 + H_inverse[0][1] * tau2;
X4_dot_2 = H_inverse[1][0] * tau1 + H_inverse[1][1] * tau2;

L2 = 0.01 * (X2_dot_2);
C2 = 0.01 * (X4_dot_2);

omega1_prob4 += (L1 + L2) * 0.5;
omega2_prob4 += (C1 + C2) * 0.5;
```

$$\begin{bmatrix}\theta_1\\ \theta_2\\ \theta_3\\ \theta_4\end{bmatrix} = \begin{bmatrix}X_1\\ X_2\\ X_3\\ X_4\end{bmatrix}$$

$$\begin{bmatrix}\dot{X}_1\\ \dot{X}_2\\ \dot{X}_3\\ \dot{X}_4\end{bmatrix} = \begin{bmatrix} X_2 \\ \frac{3}{4}[K_{p1}(\theta_{1des}-\theta_1)+K_{D1}(\omega_{1des}-\omega_1)] \\ X_4 \\ \frac{15}{4}[K_{p2}(\theta_{2des}-\theta_2)+K_{D2}(\omega_{2des}-\omega_2)]\end{bmatrix}$$

5. Impression

실제 질량과 중력을 생각하여 Manipulator 설계하고 PD제어기를 이용하여 제어하는 것이 앞서 했던 프로젝트들 보다 훨씬 어려웠고 데이터를 시각적으로 확인할 때마다 성취감을 느낄 수 있었습니다.



Q & A

감사합니다.