80846 - Report - 4th

GU JUN, 6132230056-4

May 16, 2024

Intruoduction

1 Computed torque controller

Problem Statement

According to the class, we are required to complete the
1. Jacobian matrix
2. A block for forward kinematics:

1.1 Simulink Model

See the Figure 1 below. The whole system includes two subsystems: RobotDynamics and Controller.

1.2 Formulas and Source Codes

This part includes the formulas and source codes for the controller with the Jacobian matrix and forward kinematics blocks.

Jacobian matrix

The Jacobian matrix is calculated by the following formula:

$$J = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} \end{bmatrix}$$

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

The source code is shown below:

```
function tau = controller(error, derror, q)
```

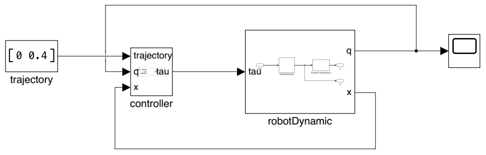

Figure 1: Simulink Model of the whole system

```
lg1 = 0.2;
lg2 = 0.2;

kp = [800, 0;
       0, 800];


kd = [1, 0;
        0, 1];

jacobian = [-lg1 * sin(q(1)) - lg2 * sin(q(1) + q(2)), -lg2 * sin(q(1) + q(2));
    lg1 * cos(q(1)) + lg2 * cos(q(1) + q(2)), lg2 * cos(q(1) + q(2))];


tau = kp * transpose(jacobian) * error - kd * derror;
end
```

Forward Kinematics

Forward Kinematics is calculated by

$$x = l_1 cos(\theta_1) + l_2 cos(\theta_1 + \theta_2) \tag{1}$$

$$y = l_1 sin(\theta_1) + l_2 sin(\theta_1 + \theta_2) \tag{2}$$

The source code is shown below:

```
function x = kinematics(q)

lg1 = 0.2;
lg2 = 0.2;
x = [0;
     0];
x(1) = lg1 * cos(q(1)) + lg2 * cos(q(1) + q(2));
x(2) = lg1 * sin(q(1)) + lg2 * sin(q(1) + q(2));

end
```

1.3 Simulation Results

As a result, I got the following Figure 2.

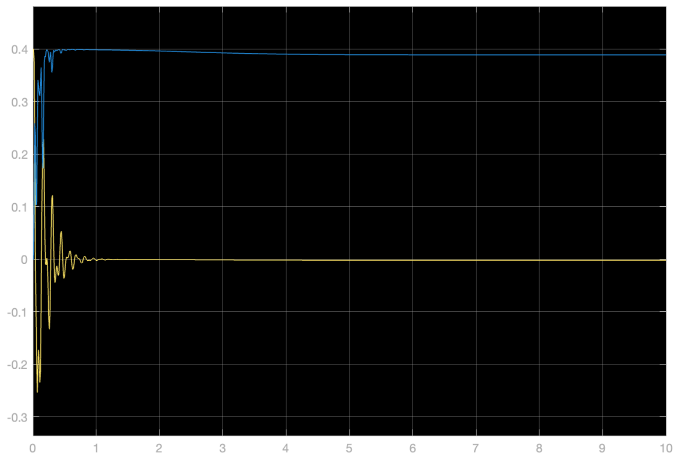

Figure 2: Result Plot with a target of (0, 0.4).

2 Desired task space trajectory controller

Problem Statement

According to the class, we are required to complete a controller that can track a torque function $q_d(t)$. I use the sin function with 0.5π shift on y.

2.1 Simulink Model

See the Figure 3 below.

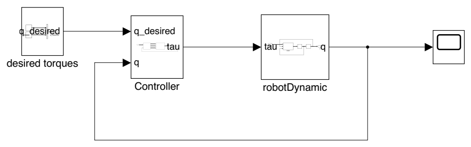

Figure 3: Simulink Model of the system for the Computed torque controller

2.2 Formulas and Source Codes

This part includes the formulas and source codes for the inverse dynamics

Inverse Dynamics

According to the formula below, we need to calculate the τ_d

$$\tau = \tau_d + k_p(q_d - q) + k_v(\dot{q}_d - \dot{q}) \tag{3}$$

To calculate τ_d, I use inverse dynamics.

$$\tau_d = M\ddot{q_d} + V\dot{q_d} + G \tag{4}$$

Where M, V, G have been mentioned in the past lectures.
The source code is shown below:

```
I1 = 0.05;
m1 = 1.5;
lg1 = 0.2;
lr1 = lg1 / 2;

I2 = 0.01;
```

```
m2 = 0.5;
lg2 = 0.2;
lr2 = lg2 / 2;

g = 9.8;

M = [m1 * lr1^2 + I1 + m2 * lg1^2, m2 * lg1 * lr2 * cos(q_desired(2));
        m2 * lg1 * lr2 * cos(q_desired(2)), m2 * lr2^2 + I2];

V = [-m2 * lg1 * lr2 * qd_desired(2)^2 * sin(q_desired(2));
    m2 * lg1 * lr2 * qd_desired(1)^2 * sin(q_desired(2))];

G = [m1 * g * lr1 * cos(q_desired(1)) + m2 * g * lg1 * cos(q_desired(1));
    m2 * g * lr2 * cos(q_desired(1) + q_desired(2));];

tau_d = M * qdd_desired + V + G;
```

Controller

For the controller, there is just a little change.

The source code is shown below:

2.3 Simulation Results

After some tuning of the gains, I got the following Figure 4.

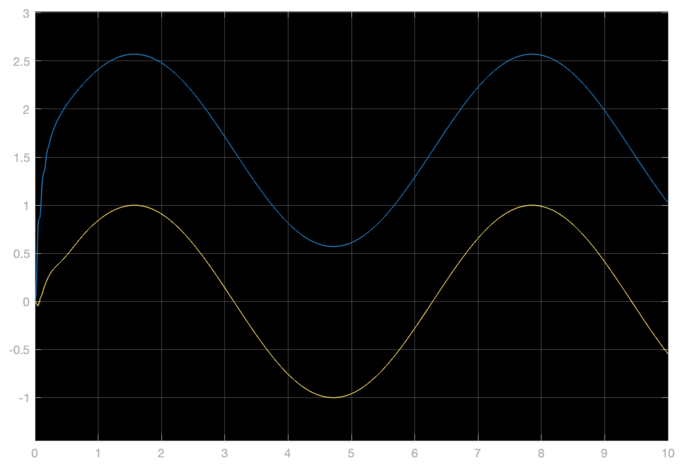

Figure 4: Result Plot, error disappeared after 1 second

3 Explanation

The first thing I need to mention is I found I calculated the M and V matrices wrong in the past simulations, so I fixed the problem and now robot dynamics runs normally.

3.1 Passivity-based task-space controller

For this part, it is important to understand the Jacobian matrix and simple forward kinematics for 2-dof joints. After that, everything goes simple.

3.2 Computed torques controller

For this part, an interesting thing I found is that even if I don't add the inverse dynamic block and modify the controller part, the result is still ok.

The inverse dynamic block is very simple to make, almost the same as the robot dynamic, you just need to change the input and output of the function.