ESE446 Final Project

RRR robotic arm kinetics and dynamics simulation

Wentao Xu
ID:458395

In this project, I simulated a RRR robot arm in different circumstances under given certain conditions by Matlab Simulink.

Dynamics Derivation:

Initial conditions:

$\theta_{1\text{-initial}} = 10°,$

$\theta_{2\text{-initial}} = 20°,$

$\theta_{3\text{-initial}} = 30°,$

L_1= 4m,
L_2=3m,
L_3=2m,
M_1=20kg,
M_2=15kg,
M_3=10kg,
I_1=0.5kg-m^2
I_2=0.2kg-m^2
I_3=0.1kg-m^2

$$I_{C1} = \begin{bmatrix} I_{1xx} & 0 & 0 \\ 0 & I_{1yy} & 0 \\ 0 & 0 & I_{1zz} \end{bmatrix} I_{C2} = \begin{bmatrix} I_{2xx} & 0 & 0 \\ 0 & I_{2yy} & 0 \\ 0 & 0 & I_{2zz} \end{bmatrix} I_{C3} = \begin{bmatrix} I_{3xx} & 0 & 0 \\ 0 & I_{3yy} & 0 \\ 0 & 0 & I_{3zz} \end{bmatrix}$$

Transformation Matrix:

$$^0T_1 = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad ^1T_2 = \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad ^2T_3 = \begin{pmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$^0T_2 = {}^0T_1 \times {}^1T_2$

$^0T_3 = {}^0T_1 \times {}^1T_2 \times {}^2T_3$

$$Jacobian_{v1} = \begin{bmatrix} \frac{\partial P_1}{\partial q_1} & \frac{\partial P_1}{\partial q_2} & \frac{\partial P_1}{\partial q_3} \end{bmatrix}$$

$$Jacobian_{v2} = \begin{bmatrix} \frac{\partial P_2}{\partial q_1} & \frac{\partial P_2}{\partial q_2} & \frac{\partial P_2}{\partial q_3} \end{bmatrix}$$

$$Jacobian_{v1} = \begin{bmatrix} \frac{\partial P_3}{\partial q_1} & \frac{\partial P_3}{\partial q_2} & \frac{\partial P_3}{\partial q_3} \end{bmatrix}$$

$$Jacobian_{w1} = \begin{bmatrix} & 0 & 0 \\ {}^0z_1 & 0 & 0 \\ & 0 & 0 \end{bmatrix}$$

$$Jacobian_{w2} = \begin{bmatrix} & & 0 \\ {}^0z_1 & {}^0z_2 & 0 \\ & & 0 \end{bmatrix}$$

$$Jacobian_{w3} = \begin{bmatrix} {}^0z_1 & {}^0z_2 & {}^0z_3 \\ & & \end{bmatrix}$$

$$M(q) = \sum_{i=1}^{3} (m_i J_{vi}^T J_{vi}^T + J_{wi}^T I_{ci} J_{wi})$$

$$C(q,\dot{q}) = \begin{bmatrix} 0.5m_{111} & m_{122} - 0.5m_{211} & m_{133} - 0.5m_{331} \\ m_{211} - 0.5m_{112} & 0.5m_{222} & m_{233} - 0.5m_{332} \\ m_{311} - 0.5m_{113} & 0.5(m_{323} + m_{332} - m_{233}) & 0.5m_{222} \end{bmatrix}$$

$$B(q,\dot{q}) = \begin{bmatrix} m_{112} & m_{113} & m_{123} + m_{132} - m_{231} \\ m_{212} + m_{221} - m_{122} & m_{213} + m_{231} - m_{132} & m_{223} \\ m_{312} + m_{321} - m_{123} & m_{313} + m_{331} - m_{133} & m_{323} + m_{332} - m_{233} \end{bmatrix}$$

$$Friction(\dot{q}) = c_1 \dot{q} + c_2 sign(\dot{q})$$

$$G(\dot{q}) = -\begin{bmatrix} J_{v1}^T \begin{bmatrix} 0 \\ m_1 g \\ 0 \end{bmatrix} & J_{v2}^T \begin{bmatrix} 0 \\ m_2 g \\ 0 \end{bmatrix} & J_{v3}^T \begin{bmatrix} 0 \\ m_3 g \\ 0 \end{bmatrix} \end{bmatrix}$$

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = \tau$$

$$\ddot{q} = M^{-1}(q)\left[\tau - C(q,\dot{q})\begin{bmatrix} \dot{q}_1^2 \\ \dot{q}_3^2 \\ \dot{q}_3^2 \end{bmatrix} - B(q,\dot{q})\begin{bmatrix} \dot{q}_1 \dot{q}_2 \\ \dot{q}_1 \dot{q}_3 \\ \dot{q}_2 \dot{q}_3 \end{bmatrix} - G(q) - F(\dot{q})\right]$$

Matlab simulation:

Task-1:

No actuator torque, no gravity see figure task1-1

No actuator torque, with gravity see video clip task1-2

Joint-1 actuator torque in order to provide equilibrium, with gravity

See video clip task1-3

Joint-1 and Joint-2 actuator torque in order to provide equilibrium, with gravity

See video clip task1-4

Joint-1, Joint-2, and Joint-3 actuator torque in order to provide equilibrium, with gravity

See video clip task 1-5

Task-2:

Simulation with control partitioning:

I build a closed loop to reach our desired angle with two control gains: $K_v = 2\sqrt{K_p}$ (which is critical damping condition)

$\theta_{\text{1-initial}} = 10°, \quad \theta_{\text{1-desired}} = 45°,$

$\theta_{\text{2-initial}} = 20°, \quad \theta_{\text{2-desired}} = \text{-}30°,$

$\theta_{\text{3-initial}} = 30°, \quad \theta_{\text{3-desired}} = 10°,$

Matlab simulation: See video clip task 2-1

Input a step response (45 degree)at time 0 with varying K_p

Since the matlab simulation time elapses differently based on each simulation condition so from the video we cannot tell the time response accurately, therefore, I only attached one video clip different K_p gives us pretty much the same video while higher K_p with slower simulation time elapses. However, the scope with show us the situation more explicitly so that I include each specific case with their corresponding scope screenshot.

Chart with critical damping condition: $K_v = 2\sqrt{K_p}$ while K_p= 1,10,100,200,500

K_p	1	10	100	200	500
Simulation and scope screenshot	Video clip task2-2 Figure task2Kp=1	Video clip task2-2 Figure task2Kp=10	Video clip task2-2 Figure task2Kp=100	Video clip task2-2 Figure task2Kp=200	Video clip task2-2 Figure task2Kp=500
rise-time	$T_r \cong 3.5$	$T_r \cong 1.1$	$T_r \cong 0.6$	$T_r \cong 0.35$	$T_r \cong 0.16$

I cannot get overshoot just by adjusting K_p, because it is at critical state, we do can get overshoot by adjusting both K_p and K_v, we just need to set $K_v < 2\sqrt{K_p}$ then it becomes underdamping overshoot happens.

Set $K_v = 100, K_p = 2\sqrt{50}$ for the simulation see video clip task2-2overshoot we can clearly see the arm goes over the desired angle and returns. Additionally, see figure task2-2 overshoot for the explicit scope screenshot.

I also tested overdamping case, we cannot see overshoot as well, meanwhile we cannot see much different from videos, but the scope tells us the rise time is longer than critical case. With $K_v = 10, K_p = 2\sqrt{20}$. In this case $T_r \cong 1.6$ larger than 1.1 .(see figure underdamped)

Task-3 Path trajectory control

In this case we apply what I've done in task-2 and achieve more advanced goal.

The desired chart:

Time (seconds)	Theta-1 (degrees)	Theta-2 (degrees)	Theta-3 (degrees)
0	0	0	90
2	30	-10	70
4	45	130	-85
6	150	10	70
8	180	0	-90

I design the trajectory function by putting desired time and corresponding angel into a matrix. Deal with each column separately and extrapolate the angle with spline function. I set a vector Ts to represent the time with stepsize 0.01. I extract the indices of each matrix elements because I want to output an Nx3 matrix with each angle respectively and the first column as time node, in this case I can simulate a varying input along time.

The simulation can reach the desired angle and it follows the trajectory. See the video clip task3. the same reason as before, we cannot see how close the arm follows the trajectory from the video directly so I only included one video clip.

Set $K_v = 2\sqrt{K_p}$ again and see the scope screenshot when Kp=1,10,100,200 respectively.

See figure: task3Kp=1, task3Kp=10, task3Kp=100, task3Kp=200.

See figure:trajectory compare with the desired angles shown in the scope, they are the same.

The whole simulink design is attached as well.