Example - RRR Robot

Harsh Maithani

Problem statement

Given the DH parameters of a RRR robot, as well as cartesian trajectory points stored in an external file -

1. Determine the Forward Kinematics
2. Determine the Inverse Kinematics
3. Fetch cartesian trajectory points from an external file and formulate a trajectory planner to send intermediate points to the robot.

Solution

The DH parameters of the robot, along with the joint limits are -

Link number	Twist(α)	Link length(a)	Link offset (d)	Joint angle (θ)	Min. limit	Max. limit
0	0	0	-	-	-	-
1	pi/2	10	0	θ_1	-pi	pi
2	0	5	0	θ_2	-pi/2	pi/2
3	0	5	0	θ_2	-pi	pi

Table 1: DH Parameters

For prototyping we used the Peter Corke RTB Toolbox in MATLAB that has in-built Forward Kinematics and Inverse Kinematics functionality. The toolbox is provided in the submission folder for convenience but it is recommended to download it directly from the website of Peter Corke. The Inverse Kinematics functionality can be erroneous though.

```
robot:: 3 axis, RRR, stdDH, fastRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|         10|     1.5708|          0|
|  2|         q2|          0|          5|          0|          0|
|  3|         q3|          0|          5|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

Figure 1: Robot DH parameters with Peter Corke RTB Toolbox

The DH parameters as defined by Craig are :

- a_i = the distance from Z_i to Z_{i+1} measured along X_i
- α_i = the angle from Z_i to Z_{i+1} measured about X_i
- d_i = the distance from X_{i-1} to X_i measured along Z_i
- θ_i = the angle from X_{i-1} to X_i measured about Z_i

Using these definitions, the robot can be visualized as in the figure as -

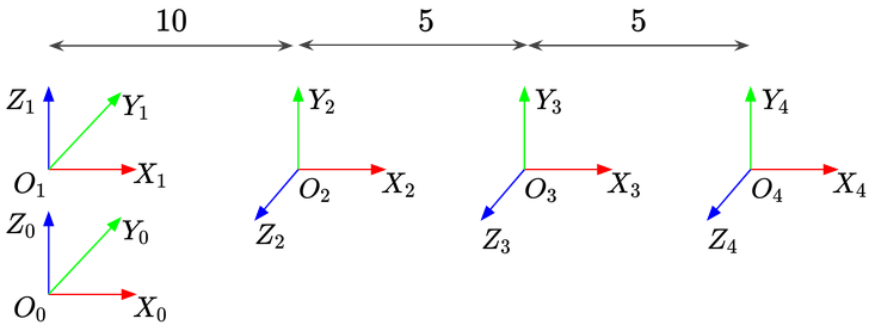

Figure 2: Robot visualized. Frames O_0 and O_1 are identical.

Forward Kinematics

The Forward Kinematics equations can be derived as

$$\boldsymbol{x} = \Big[L_1 + L_2\ cos(\theta_2) + L_3\ cos(\theta_2 + \theta_3)\Big]\ cos(\theta_1) \tag{1}$$

$$\boldsymbol{y} = \Big[L_1 + L_2\ cos(\theta_2) + L_3\ cos(\theta_2 + \theta_3)\Big]\ sin(\theta_1) \tag{2}$$

$$\boldsymbol{z} = \Big[L_2\ sin(\theta_2) + L_3\ sin(\theta_2 + \theta_3)\Big] \tag{3}$$

To view the results in MATLAB run `main.m`

```
In C++

x = (L1 + L2 * cos(q2) + L3 * cos(q2 + q3)) * cos(q1);
y = (L1 + L2 * cos(q2) + L3 * cos(q2 + q3)) * sin(q1);
z = (     L2 * sin(q2) + L3 * sin(q2 + q3));
```

Inverse Kinematics

The Inverse Kinematics equations can be derived as

$$\boldsymbol{q_1} = tan^{-1}\left(\frac{y}{x}\right) \tag{4}$$

$$\boldsymbol{q_3} = cos^{-1}\left[\frac{\left(\frac{x}{cos(q_1)} - L_1\right)^2 + z^2 - L_2^2 - L_3^2}{2\ L_2\ L_3}\right] \tag{5}$$

$$\boldsymbol{q_2} = tan^{-1}\left(\frac{m}{n}\right) - tan^{-1}\left(\frac{\sqrt{n^2 + m^2 - z^2}}{z}\right) \tag{6}$$

where

$m = L_2 + L_3\ cos(q_3)$ and $n = L_3\ sin(q_3)$

For q_2 we have referred to the book by Craig - Introduction to Robotics, Mechanics and Control (Page 377).

To view the results in MATLAB run `main.m`

```
In C++

q1 = atan(y/x);    // Angles in radians
q3 = acos((pow((x/cos(q1) - L1),2) + pow(z,2) - pow(L2,2) - pow(L3,2))/(2*L2*L3));

m = L2 + L3 * cos(q3);
n = L3 * sin(q3);
q2 = (atan2(m,n)-atan2(sqrt(pow(n,2)+pow(m,2)-pow(z,2)),z));
```

Joint Angles	RTB Toolbox	Analytical Equations	C++
$q_1 : 0.15$	$x : 17.85 \quad q_1 : 0.15$	$x : 17.85 \quad q_1 : 0.15$	$x : 17.85 \quad q_1 : 0.15$
$q_2 : 0.29$	$y : 2.698 \quad q_2 : 0.29$	$y : 2.69 \quad q_2 : 0.29$	$y : 2.69 \quad q_2 : 0.29$
$q_3 : 0.57$	$z : 5.219 \quad q_3 : 0.57$	$z : 5.21 \quad q_3 : 0.57$	$z : 5.21 \quad q_3 : 0.57$

Table 2: Verification of Forward and Inverse Kinematics. Inverse Kinematics from the RTB Toolbox can be erroneous.

Trajectory planner

We use the trajectory planning equations for minimizing the jerk in the paper by Huaizhong Li (provided in the submission folder). The trajectory consists of two phases - an acceleration phase and an immediate deceleration phase. There are more trajectory planning algorithms in robotics textbooks. The one we have chosen for this problem ensures that the jerk is minimized.

- Acceleration phase ($t \in [0, t_1]$)

$$d = \frac{A}{2}\, n^2 \left[\frac{1}{2} \left(m\ t \right)^2 - \left(1 - cos(m\ t) \right) \right] \tag{7}$$

- Deceleration phase ($t \in [t_1, t_2]$)

$$d = \frac{1}{4} A T_1^2 + \frac{1}{2} A T_1 T_2 + \frac{A}{2} n^2 \left[\frac{(2\pi)^2 (t - t_2)}{T_1} - \frac{1}{2} \left(\frac{2\pi (t - t_2)}{T_1} \right)^2 + \left(1 - cos \left(\frac{2\pi (t - t_2)}{T_1} \right) \right) \right] \tag{8}$$

  where

$$A = 8 \frac{q_f - q_i}{T^2},\ T_1 = \left(\frac{2(q_f - q_i)}{A} \right)^{1/2},\ T_2 = T_1\ ,\ m = \frac{2\pi}{T_1}\ ,\ n = \frac{T_1}{2\pi}$$

To view the results in MATLAB run the file `complete_planner.m` or run `main.m` with the last line uncommented.

```
In C++

// Acceleration phase
displacement = (A/2)*(pow(n,2))*(0.5*(pow((m*t),2))-(1-cos(m*t)));  // Displacement

// Deceleration phase
displacement = 0.25*A*(pow(T1,2))+0.5*A*T1*T2+0.5*A*(pow(n,2))
      *( (pow((2*PI),2)*(t-t_start)/T1)-0.5*pow((m*(t-t_start)),2)
      +(1-cos(m*(t-t_start))) ) ;
```