Robotics

Assignment 02
Report

1_TUE_I

Vasilije Rakcevic
Jingsheng Lyu
Prathamesh More

December 6, 2020

Contents

Chapter 1

Forward Kinematics

1.1 Find DH Parameter for 3DOF Puma

Following are the DH parameters found for the 3-DOF PUMA Robotic Arm

i	α_{i-1}	a$_{i-1}$	d$_i$	θ_i
1	0	0	0	q$_1$
2	0	l$_1$	0	q$_2 - 90°$
3	0	l$_2$	0	q$_3$
4	0	l$_3$	0	0

Table 1.1: **DH Parameters** of 3-DOF RRR Puma

1.2 Transformation Between frames

For convenience let us assume following convention,

$$s_1 = sin(\theta_1) = sin(q_1) \tag{1.1}$$
$$s_{12} = sin(\theta_1 + \theta_2) = sin(q_1 + q_2 - 90°) \tag{1.2}$$
$$s_{123} = sin(\theta_1 + \theta_2 + \theta_3) = sin(q_1 + q_2 + q_3 - 90°) \tag{1.3}$$
$$c_1 = cos(\theta_1) = cos(q_1) \tag{1.4}$$
$$c_{12} = cos(\theta_1 + \theta_2) = cos(q_1 + q_2 - 90) \tag{1.5}$$
$$c_{123} = cos(\theta_1 + \theta_2 + \theta_3) = cos(q_1 + q_2 + q_3 - 90°) \tag{1.6}$$

The forward kinematics ${}^0_4T[q]$ are derived based on the formula for Frame to Frame transformation provided in the tutorials.

$$ {}^0_1T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.7} $$

$$ {}^1_2T = \begin{bmatrix} s_2 & c_2 & 0 & l_1 \\ -c_2 & s_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.8} $$

$$ {}^2_3T = \begin{bmatrix} c_3 & -s_3 & 0 & l_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.9} $$

$$ {}^3_ET = \begin{bmatrix} 1 & 0 & 0 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.10} $$

Finally, we can calculate the final homogeneous transportation matrix with the formulas from 1.7 to 1.10:

$$ {}^{0}_{E}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{E}T = \begin{bmatrix} s_{123} & c_{123} & 0 & c_1 l_1 + s_{12} l_2 + s_{123} l_3 \\ -c_{123} & s_{123} & 0 & s_1 l_1 - c_{12} l_2 - c123 l_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.11} $$

1.3 End Effector position in Operational space

Equation 1.11 is a homogeneous transformation matrix which denotes the transformation of end effector with respect to the base frame. Each matrix element denotes some property of the translation or rotational relations between the frames. Following are the coordinates (Position and orientation angle) of the end effector in the base frame. Following are the equations for the evaluation of this coordinates

$$X = {}^{0}_{E}T[1][4] \tag{1.12}$$
$$Y = {}^{0}_{E}T[2][4] \tag{1.13}$$
$$\alpha = atan2({}^{0}_{E}T[2][1], {}^{0}_{E}T[1][1]) \tag{1.14}$$

Hence,

$$F(\mathbf{q}) = \begin{pmatrix} c_1 l_1 + s_{12} l_2 + s_{123} l_3 \\ s_1 l_1 - c_{12} l_2 - c123 l_3 \\ atan2(-c_{123}, s_{123}) \end{pmatrix} \tag{1.15}$$

1.4 Compute the end effector Jacobian

From the formulas 1.12 to 1.14 we know the corresponding value of X, Y and α. In this section we will take the derivative of these equations w.r.t each joint
Differentiating X from equation 1.6 w.r.t q1,q2 and q3 respectively

$$\frac{dx}{dq1} = -s_1 * l_1 + c_{12} * l_2 + c_{123} * l_3 \tag{1.16}$$
$$\frac{dx}{dq2} = c_{12} * l_2 + c_{123} * l_3 \tag{1.17}$$
$$\frac{dx}{dq3} = c_{123} * l_3 \tag{1.18}$$

Differentiating Y from equation 1.6 $\omega, r, t, q_1, q_2 and q_3$ respectively

$$\frac{dy}{dq1} = c_1 * l_1 - s_{12} * l_2 - s_{123} * l_3 \tag{1.19}$$
$$\frac{dy}{dq2} = -s_{12} - s_{123} * l3 \tag{1.20}$$
$$\frac{dy}{dq3} = -s_{123} * l_3 \tag{1.21}$$

Differentiating α from equation 1.6 w.r.t q1,q2 and q3 respectively

$$\alpha = atan2(-c_{123}, s_{123}) \tag{1.22}$$

$$\alpha = q1 + q2 + q3 - 90 \tag{1.23}$$

$$\frac{d\alpha}{dq1} = 1 \tag{1.24}$$

$$\frac{d\alpha}{dq2} = 1 \tag{1.25}$$

$$\frac{d\alpha}{dq3} = 1 \tag{1.26}$$

Jacobian matrix can be written as,

$$J(q) = \begin{bmatrix} \frac{dx}{dq_1} & \frac{dx}{dq_2} & \frac{dx}{dq_3} \\ \frac{dy}{dq_1} & \frac{dy}{dq_2} & \frac{dy}{dq_3} \\ \frac{d\alpha}{dq_1} & \frac{d\alpha}{dq_2} & \frac{d\alpha}{dq_3} \end{bmatrix} \tag{1.27}$$

Hence Substituting from eq.1.18 to 1.28 in eq.1.29

$$J(q) = \begin{bmatrix} -s_1 * l_1 + c_{12} * l_2 + c_{123} * l_3 & c_{12} * l_2 + c_{123} * l_3 & c_{123} * l_3 \\ c_1 * l_1 - s_{12} * l_2 - s_{123} * l_3 & -s_{12} - s_{123} * l3 & -s_{123} * l_3 \\ 1 & 1 & 1 \end{bmatrix} \tag{1.28}$$

1.5 Understanding the Jacobian Matrix and pose singularities

1.5.1 Robot Pose

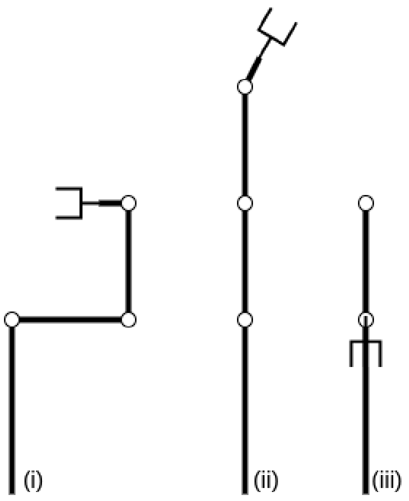

Figure 1.1: Robot Pose

1.5.2 Configuration determination

$$\begin{aligned} \text{i. } \mathbf{q}_1 &= \left(0, 0, -\frac{\pi}{2}\right) \\ \text{ii. } \mathbf{q}_2 &= \left(-\frac{\pi}{2}, \frac{\pi}{2}, 0.1\right) \\ \text{iii. } \mathbf{q}_3 &= \left(-\frac{\pi}{2}, -\frac{\pi}{2}, 0\right) \end{aligned} \tag{1.29}$$

To find the singularity of the robot, we should calculate the determination of the Jacobian matrix.
If the $\mathbf{q}_1 = \left(0, 0, -\frac{\pi}{2}\right)$, we can calculate the Jacobian matrix is following:

$$J(q_1) = \begin{bmatrix} 0.433 & 0.433 & 0.00 \\ 0.467 & 0.055 & 0.055 \\ 1.000 & 1.000 & 1.000 \end{bmatrix} \tag{1.30}$$

$$det(J(q_1)) = -0.1784 \tag{1.31}$$

If the $\mathbf{q}_2 = \left(-\frac{\pi}{2}, \frac{\pi}{2}, 0.1\right)$, we can calculate the Jacobian matrix is following:

$$J(q_2) = \begin{bmatrix} 0.899 & 0.488 & 0.055 \\ -0.005 & -0.005 & -0.005 \\ 1.000 & 1.000 & 1.000 \end{bmatrix} \tag{1.32}$$

$$det(J(q_2)) = 0 \tag{1.33}$$

If the $\mathbf{q}_3 = \left(-\frac{\pi}{2}, -\frac{\pi}{2}, 0\right)$, we can calculate the Jacobian matrix is following:

$$J(q_3) = \begin{bmatrix} -0.076 & -0.488 & -0.055 \\ 0.005 & 0.005 & 0.005 \\ 1.000 & 1.000 & 1.000 \end{bmatrix} \tag{1.34}$$

$$det(J(q_3)) = 0 \tag{1.35}$$

With $\mathbf{q}_1 = \left(0, 0, -\frac{\pi}{2}\right)$, the robot is fully controllable using Jakobian method, because it has no singularity in this case. Intuitively, all linear approximations are such that the next step for reaching some (x,y) goal position in operational space can be uniquely expressed by the joint angles of the robot.

With $\mathbf{q}_2 = \left(-\frac{\pi}{2}, \frac{\pi}{2}, 0.1\right)$, the robot is in singularity, which has been proven by calculated Jacobian, since its determinant is 0. Intuitively, required movement in vertical line will be problematic, since linear approximations for this join configuration are allowing movement strictly horizontal for joints 1 and 2 of the robot.

With $\mathbf{q}_3 = \left(-\frac{\pi}{2}, -\frac{\pi}{2}, 0\right)$, the robot is also in singularity. Intuitively, robot also lost the degree of freedom with respect to vertical movements, since linear approximation in this position only allows strictly horizontal movement from this position for all 3 joints.

To find the singular points of the robot, we should calculate the determinant of its Jacobian. Where the determinant is equal to zero, the Jacobian has lost full rank and the position is singular.

Chapter 2

Trajectory Generation in Joint Space

2.1 Generation of smooth trajectories with polynomial splines

The Robotic Arm needs to go from point A to Point C passing through point B taking 5 seconds i.e. $t_c = 5$ and $t_b = 2.5$.The joint angles for each of the point are mentioned as below

$$\begin{aligned} q_a &= (0,0,0)^T && (2.1) \\ q_b &= (-\frac{\pi}{4}, \frac{\pi}{2}, 0)^T && (2.2) \\ q_c &= (-\frac{\pi}{2}, \frac{\pi}{4}, 0)^T && (2.3) \end{aligned}$$

Assuming the Robotic Arm performs a split spline as following equations

$$\begin{aligned} q_1(t) &= a_1 + a_2 * t^2 + a_3 * t^3 && (2.4) \\ q_2(t) &= b_1 + b_2 * t^2 + b_3 * t^3 && (2.5) \end{aligned}$$

eq(2.4) for motion from point A to B and eq(2.5) for motion from point B to C
let us define some Initial Conditions,
The Velocity at the start and the end will be Zero. hence,

$$\begin{aligned} q_a^{'} &= 0 && (2.6) \\ q_c^{'} &= 0 && (2.7) \end{aligned}$$

2.1.1 Compute cubic spline parameters

From Point A to B

Putting (2.6),(2.7) in (2.4),(2.5)

$$a_1 = 0 \tag{2.8}$$
$$a_2 = 0 \tag{2.9}$$

a_3 can be defined as follows.

$$a_3 = \frac{3}{{t_b}^3} * (q_b - q_a) - \frac{2}{t_b^2} * q_a^{'} - \frac{1}{t_b} * q_b^{'} \tag{2.10}$$

a_4 can be defined as follows.

$$a_4 = -\frac{2}{{t_b}^3} * (q_b - q_a) + \frac{1}{t_b^2} * (q_a^{'} + q_b^{'}) \tag{2.11}$$

From Point B to C

Putting (2.8) in (2.5)

$$b_1 = q_b \tag{2.12}$$
$$b_2 = q_b^{'} \tag{2.13}$$

b_3 can be defined as follows.

$$b_3 = \frac{3}{(t_c - t_b)^3} * (q_b - q_a) - \frac{2}{(t_c - t_b)^2} * q_b^{'} - \frac{1}{(t_c - t_b)} * q_c^{'} \tag{2.14}$$

b_4 can be defined as follows.

$$b_4 = -\frac{2}{(t_c - t_b)^3} * (q_c - q_b) + \frac{1}{(t_c - t_b)^2} * (q_b^{'} + q_c^{'}) \tag{2.15}$$

By Using the above equations we calculated the value of all the parameters for all joints as follows.(Joint 3 has no rotation)

	Joint1(A-B)	joint1(B-C)	Joint2 (A-B)	Joint2 (B-C)
a_0	0	-0.79	0	1.57
a_1	0	-0.31	0	0.63
a_2	-0.25	-0.13	0.5	-1.13
a_3	0.05	0.05	-0.1	0.15

Table 2.1: **DH Parameters** of 3-DOF RRR Puma

2.2 Trajectories in Joint Space

2.2.1 trajectory generation

In this section we can calculate all parameters of the cubic spline with the desired joint angles.
In this we know following terms

$$\begin{aligned} q^{'} &= a_1 + a_2 * t + 3 * a_3 * t && (2.16)\\ q^{''} &= 2 * a_2 + 6 * a_3 * t && (2.17) \end{aligned}$$

Now, according to the conditions, $q^{'}$ will be maximum at $\frac{t_f}{2}$ and $\|q^{''}\|$ will be maximum at 0 and t_f.
Hence first putting t= 0 in eq.(2.17) we get the following

$$2a_2 = q^{''} \tag{2.18}$$

After putting t= t_f in eq.(2.17) we get the following

$$a_3 = \frac{-q^{''}}{3 * t_f} \tag{2.19}$$

After putting these values in 2.16 we get a following result

$$t_f = 4 * \frac{q^{'}}{q^{''}} \tag{2.20}$$

t_o can be calculated by current time function in the code.
After calculation of t_f and t_o using the equations from 2.4 to 2.15 the parameters such as $a_0, a_1, a_2, a_3, t_o, t_f$ for each joint can be computed.

2.2.2 Trajectory to specific point

Following are the plots for the torque acting on each joints while travelling from zero configuration to the desired configuration

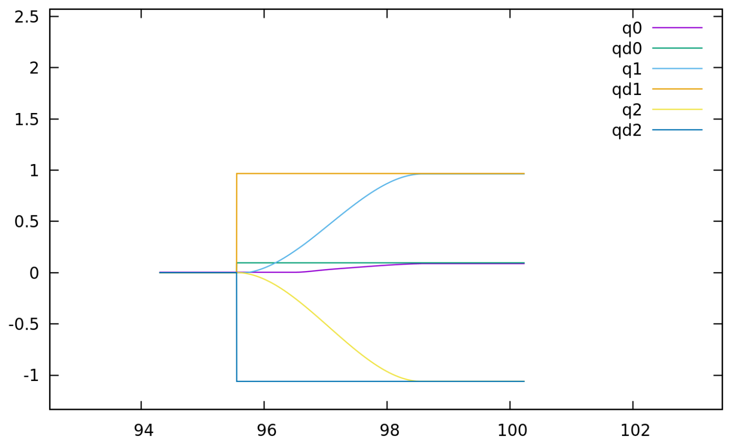

Figure 2.1: q & q_d vs time (t)

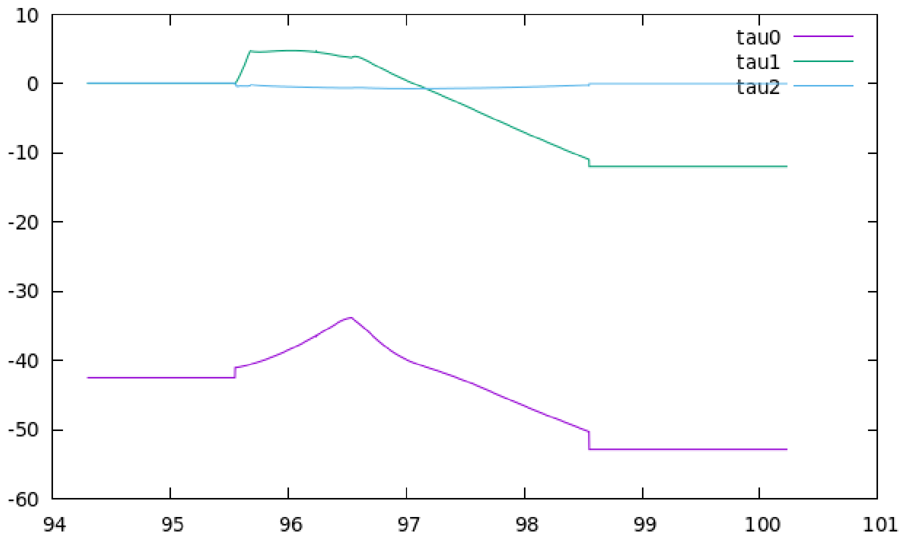

Figure 2.2: Torque vs Time

Chapter 3

Operational Space Control

3.1 Project 2 - Circle

Create a desired trajectory in operational space for the controller. To solve this task, we need to compute the desired positions and velocities of the end effector. $\dot{\beta}$ is the angular velocity. Define the initial time with t_0 and we can calculate the $\delta t = t_{curTime} - t_0$ first let implement a circle generator by the following equations of the circle.

$$x = 0.6 + 0.2cos(\psi) \tag{3.1}$$
$$y = 0.35 + 0.2sin(\psi) \tag{3.2}$$

where $\psi = \beta^{'} * t$ since the speed is constant.
Now, Differentiating x and y we get,

$$x^{'} = -\psi^{'} * 0.2 * sin(\psi) \tag{3.3}$$
$$y^{'} = \psi^{'} * 0.2 * cos(\psi) \tag{3.4}$$

where $\psi' = \beta^{'}$. These are the governing equations for the circular trajectory. The inverse kinematics which converts these desired coordinates to the joint torque can be calculated by the equations below, where 3.6 is taken from the lecture.

$$F(Force) = -k_p(x - x_d) - k_v(x^{'} - x_d^{'}) \tag{3.5}$$
$$T(Torque) = J^T * F \tag{3.6}$$

3.1.1 Parabolic Blend

For performing parabolic Blend we will first calculate t_b

$$\beta^{'} = \int_0^{t_b} \beta^{''}(t)dt \tag{3.7}$$

$$= t_b * \beta^{''} \tag{3.8}$$

putting the value of $\beta^{''}$ as $\frac{2*\pi}{25}$ and value of $\beta^{'}$ as $\frac{2*\pi}{5}$

$$t_b = 5sec \tag{3.9}$$

integrating $\beta^{'}$ we get

$$\psi = \int_0^{t_b} \beta^{'} dt \tag{3.10}$$

$$= \int_0^{t_b} \beta^{''} * t * dt \tag{3.11}$$

$$= \frac{1}{2} * \beta^{''} * t^2 \tag{3.12}$$

This 3.12 will govern the motion of the robotic manipulator in the blend zone. Now, Putting the value of t_b in the 3.12 we get

$$\psi = \pi \tag{3.13}$$

Now for t from 0 to blend time (t_b) 5 secs, the governing equation will be

$$\psi = \frac{1}{2} * \beta^{''} * (t - t_0)^2 \tag{3.14}$$

$$\psi^{'} = \beta^{''} * (t - t_0) \tag{3.15}$$

After the blending time, the angular velocity will be constant - $\frac{2*\pi}{5}$ and will perform 2π every 5 seconds. As calculated in 3.13 the robot will pass exactly π distance, meaning that with constant speed $\frac{2*\pi}{5}$ we need it to pass 4π (two complete circles) distance, and it will take additional 10 sec.
Hence from 5 secs to 15 secs the robotic arm will perform a constant angular velocity trajectory, keeping in mind that the starting phase is equal to value at the end of previous section: $\psi_0 = \pi$. The governing equation for it will be

$$\psi = \pi + \beta^{'} * (t - t_0) \tag{3.16}$$

$$\psi^{'} = \beta^{'} \tag{3.17}$$

After the constant angular velocity trajectory, the robotic arm will again go in to a blend mode and perform following trajectory for the time period from 15secs to 20secs and finally come to rest. The blend time as well as the distance will be the same as in the first blend, however the robot will be slowing down.

$$\psi = \pi + (\pi - \frac{1}{2} * \beta^{''} * (20 - (t - t_0))^2) \tag{3.18}$$

$$\psi^{'} = \beta^{''} * (t - t_0) \tag{3.19}$$

3.1.2 Project 2 - Control

3.1.3 Project 2 - Gain tuning and plots

As we can see, if the K_p increases, the error between x and x_d will decrease.

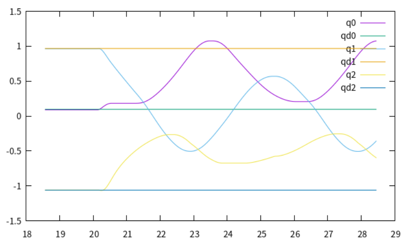

Figure 3.1: joint angles

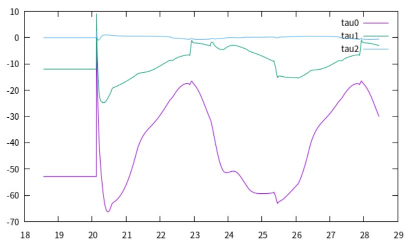

Figure 3.2: torque for proj2

We define the error with dx[] in our controll.cpp file for the plot.

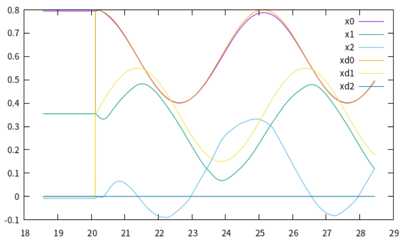

Figure 3.3: Positions and desired position in operational space

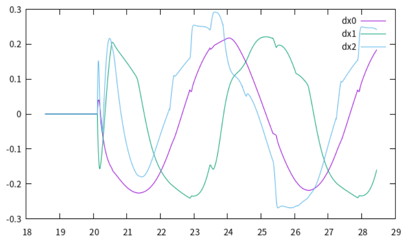

Figure 3.4: Position error between x and x_d

Chapter 4

submission

Student Name	(A1)	A2	A3	A4	(A5)	(B1)	B2	C1	C2	C3	C4	(C5)
Vasilije Rakcevic	x	x	x	x	x	x	x	x	x	x	x	
Jingsheng Lyu	x	x	x	x	x		x	x	x	x		
Prathamesh More	x	x	x	x		x	x	x	x		x	