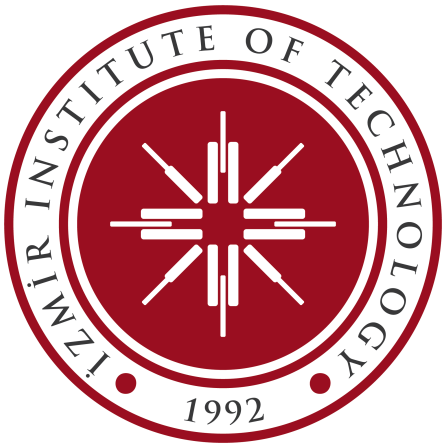

İzmir Institute of Technology

Introduction to Robot Technology

ME460

Project #1

Forward & Inverse Kinematic Analyses of Serial RRP Manipulator

Mert Emrem - 250203015

Mechanical Engineering

Summer 2021

Contents

Introduction

The manipulator architecture assigned is a *spherical arm* (RRP), with dimensional parameter d_1, and joint parameters θ_1, θ_2, s_3 as shown in figure 1.1 below. It is worth noting that the travel direction of the prismatic joint is chosen parallel to the link extending from the second revolute joint, R_2, and this is not necessarily the case for any spherical arm.

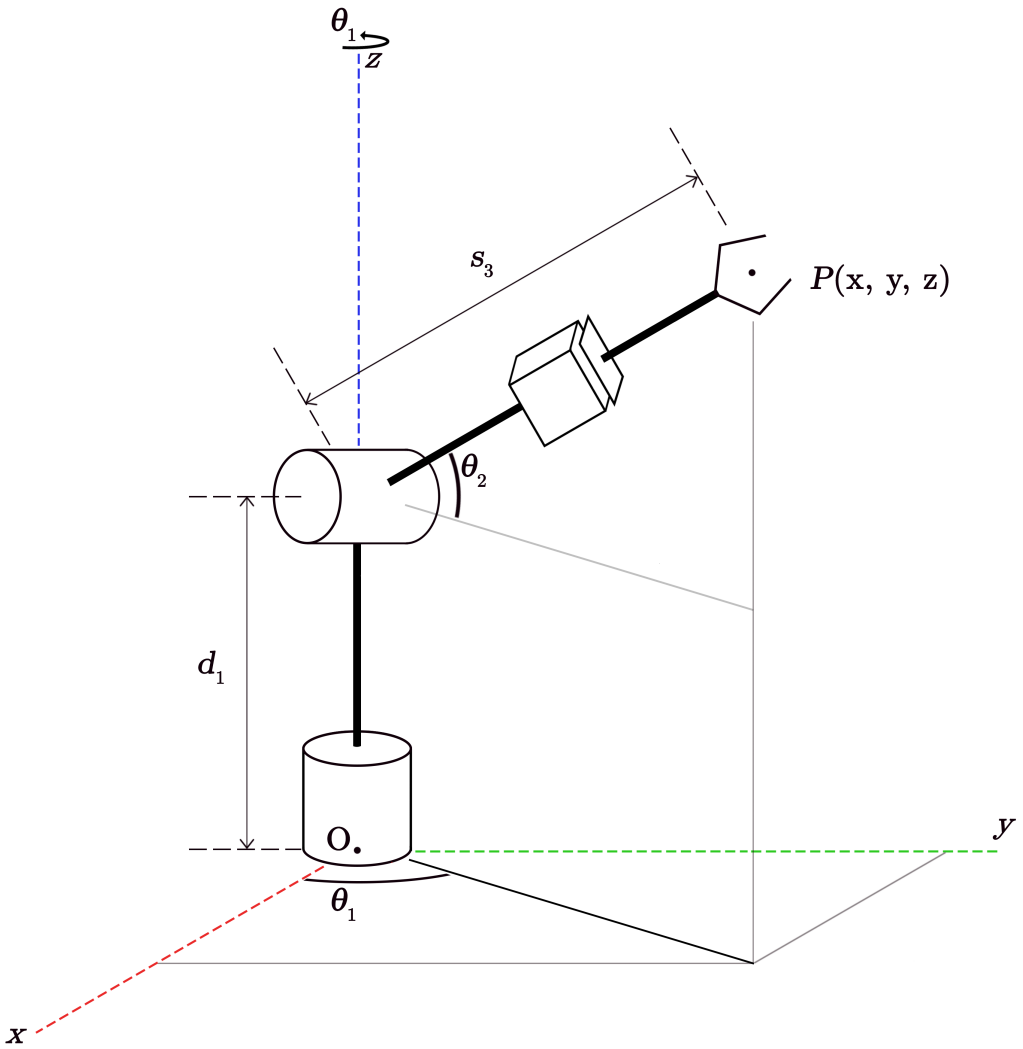

Figure 1.1: Kinematic diagram of the RRP manipulator.

Starting dimensions and constraints are as follows:

Student number: 250203$\underline{01}\dot{5}\dot{5}$

d_1 : $(\dot{1}\dot{5} + 10) \times 5 = 125$ mm,

s_3: $(|\underline{30} - 15|) \times 15 = 125$ mm,

additionally, $s_{3,\,\min} = 0$. It can also be inferred that the workspace of the manipulator is one that spans over a sphere centered at R_2, with a radius of 125 mm ($s_{3,\,\max}$), hence the name *spherical arm*.

Reference frame is at the base point, O (also the first revolute joint, R_1), with z axis coinciding with the rotation axis of R_1, as shown in fig. 1.1.

Initially, the joint space variables are determined and the forward kinematics equations of the manipulator are developed with regards to the end effector point, P, which exists in a three-dimensional task space.

Angle constraints of the angles θ_1 and θ_2 are respectively $0 \rightarrow 2\pi$, and $-\frac{\pi}{2} \rightarrow \frac{\pi}{2}$. As a result, the alternative configuration in which the end effector is upside down is discarded (for any P, $P(\theta_1,\ \theta_2,\ s_3) = P((\theta_1 + \pi),\ (\pi - \theta_2),\ s_3)$). Another way of assigning joint variables is that instead of s_3, d_3 could be used at the prismatic joint along the axis of travel, since serial manipulators are generally described by their subsequent joint frames.

Home position is defined by $\theta_1 = 0$, $\theta_2 = 0$, and $s_3 = 125$ mm.

Forward Kinematics of the Manipulator

In finding joint space parameters, Denavit-Hartenberg convention is often useful, but the manipulator in question is rather straightforward, and therefore the end-effector relations arise from intuition alone.

Forward kinematics equations developed for the manipulator are:

$$P_x = s_3 \cos\theta_2 \cos\theta_1 \tag{2.1}$$

$$P_y = s_3 \cos\theta_2 \sin\theta_1 \tag{2.2}$$

$$P_z = d_1 + s_3 \sin\theta_2 \tag{2.3}$$

Position-level singularities occur at $s_3 = 125$ mm (workspace boundaries). To find the velocity-level singularities, the Jacobian matrix of the manipulator is obtained as follows:

$$\mathbf{J}_e = \begin{bmatrix} \frac{\partial P_x}{\partial \theta_1} & \frac{\partial P_x}{\partial \theta_2} & \frac{\partial P_x}{\partial s_3} \\ \frac{\partial P_y}{\partial \theta_1} & \frac{\partial P_y}{\partial \theta_2} & \frac{\partial P_y}{\partial s_3} \\ \frac{\partial P_z}{\partial \theta_1} & \frac{\partial P_z}{\partial \theta_2} & \frac{\partial P_z}{\partial s_3} \end{bmatrix}$$

$$\mathbf{J}_e = \begin{bmatrix} -s_3 \cos\theta_2 \sin\theta_1 & -s_3 \sin\theta_2 \cos\theta_1 & \cos\theta_2 \cos\theta_1 \\ s_3 \cos\theta_2 \cos\theta_1 & -s_3 \sin\theta_2 \sin\theta_1 & \cos\theta_2 \sin\theta_1 \\ 0 & s_3 \cos\theta_2 & \sin\theta_2 \end{bmatrix}$$

Singularities occur when the determinant of $\mathbf{J}_e = 0$, the full expression is unwieldy, and is therefore left to MATLAB to simplify (see appendix A.1.1). Simplified determinant is

$$\det(\mathbf{J}_e) = s_3^2 \cos\theta_2 = 0 \tag{2.4}$$

Above equation is only satisfied when $s_3 = 0$, which is not possible in practice, and when $\theta_2 = -\frac{\pi}{2}, \frac{\pi}{2}$, which are the fully extended and folded states.

Inverse Kinematics of the Manipulator

Inverse kinematics equations developed for the manipulator are:

$$\theta_1 = \text{atan2}\left(\frac{P_y}{P_x}\right) \tag{3.1}$$

$$\theta_1 = \arctan\left(\frac{l}{r}\right) \tag{3.2}$$

$$s_3 = \sqrt{r^2 + l^2} \tag{3.3}$$

where $l = z - d_1$ and $r = \sqrt{x^2 + y^2}$.

End effector path length is given as $\frac{a_2 + a_3}{4}$, but since this configuration lacks an a_2, and a_3 is substituted by s_3, it is calculated as

$$\frac{s_3}{4} = \frac{125}{4} = 31.25 \text{ mm} \tag{3.4}$$

The path direction is arbitrarily chosen to be orthogonal to the axis $y = -x$, as shown in fig 3.1 below. Destination point and path are mathematically defined in Geogebra.

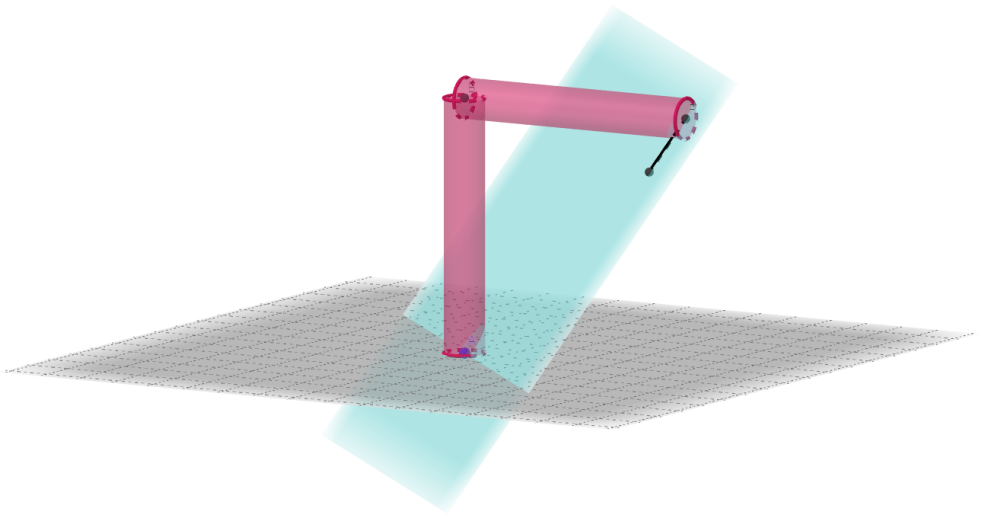

Figure 3.1: Geogebra [1] render of manipulator at home position with the destination point.

Linear interpolation of the trajectory over 100 intervals leads to infinite acceleration at $t = 0$ and $t = T$. This is worked around by applying cubic polynomial with the inputs: travel time T, initial displacement values $< x_0, y_0, z_0 >$, final displacement values $< x_1, y_1, z_1 >$ and current time t.

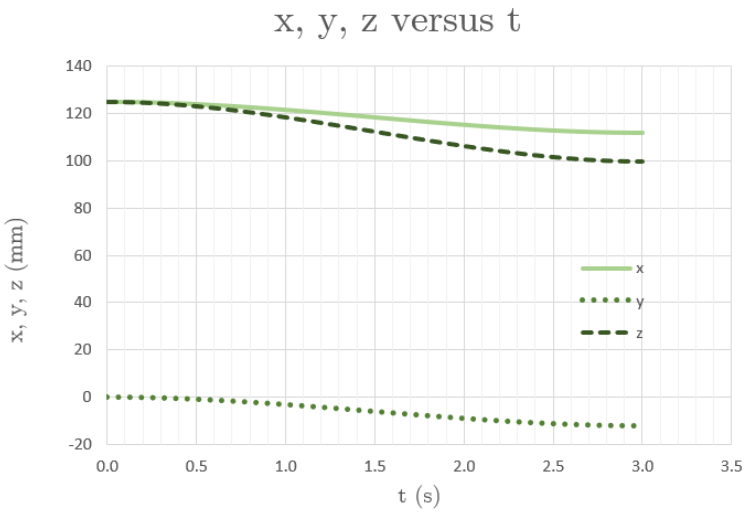

Figure 3.2: Graph of x, y and z values over time.

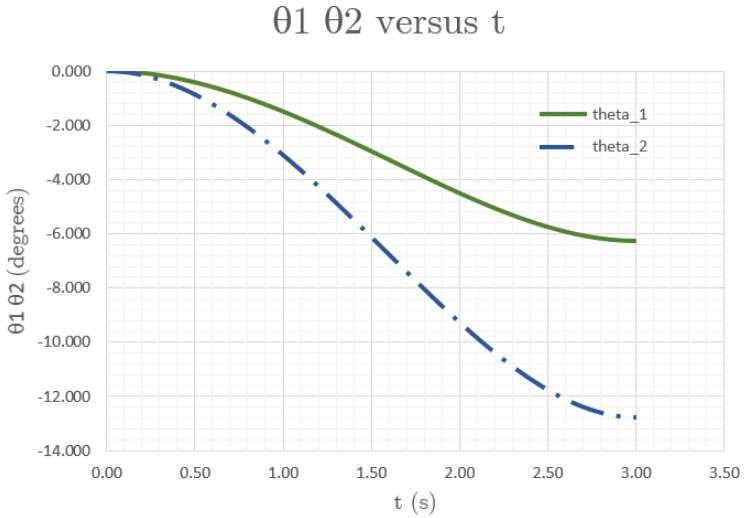

Figure 3.3: Graph of θ_1, and θ_2 values over time.

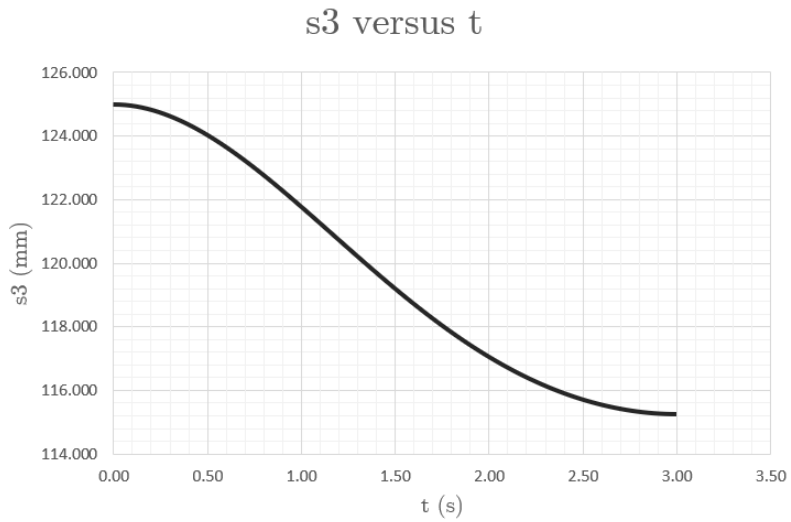

Figure 3.4: Graph of s_3 values over time.

Said cubic polynomial is formulated as follows [2]:

$$\vec{P}(t) = \vec{P}_0 + \frac{3}{T^2}(\vec{P}_1 - \vec{P}_0)\,t^2 - \frac{2}{T^3}(\vec{P}_1 - \vec{P}_0)\,t^3 \tag{3.5}$$

Where P is the displacement vector, T is the total time of travel, subscripts 0 and 1 denote initial and final.

$$\vec{P}_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \tag{3.6}$$

$$\vec{P}_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{3.7}$$

Discussion - Conclusion

Forward and inverse kinematics of the RRP manipulator turns out to be on the simpler side when compared to other manipulators of similar degrees of freedom. Especially so when the travel axis of the prismatic joint aligns with the subsequent link to R_2

The cubic polynomial method employed in the inverse kinematics part of the project is most likely more applicable on actuator-level controls or joint space applications, rather than task space applications such as the one at hand, though it still is an improvement on the constant velocity linear motion.

References

[1] HOHENWARTER, Markus, Michael BORCHERDS , Gabor ANCSIN, Balazs BENCZE and Mathieu BLOSSIER. GeoGebra Classic. computer software. 2021

[2] BIAGIOTTI, Luigi and Claudio MELCHIORRI. Trajectory planning for automatic machines and robots. Berlin: Springer, 2010.

Appendix

A.1 MATLAB Code

A.1.1 Jacobian Matrix

```
%%%% Jacobian matrix and its determinant, symbolically:

syms theta1 theta2 s3 d1

J = jacobian([s3*cos(theta2)*cos(theta1), s3*cos(theta2)*sin(theta1),
    ... d1 + s3*sin(theta2)], [theta1, theta2, s3]);

det(J);

S = simplify(det(J))
```