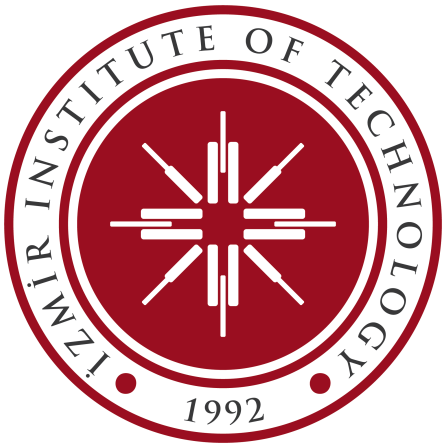

İzmir Institute of Technology

Introduction to Robot Technology

ME460

Project #2

Forward & Inverse Kinematic Analyses of Serial RRP Manipulator

Mert Emrem - 250203015

Mechanical Engineering

Summer 2021

Contents

Introduction

The manipulator architecture is identical to the one used in project #1 [1], namely, *spherical arm* (RRP), with dimensional parameter d_1, and joint parameters θ_1, θ_2, s_3 as shown in figure 1.1 below.

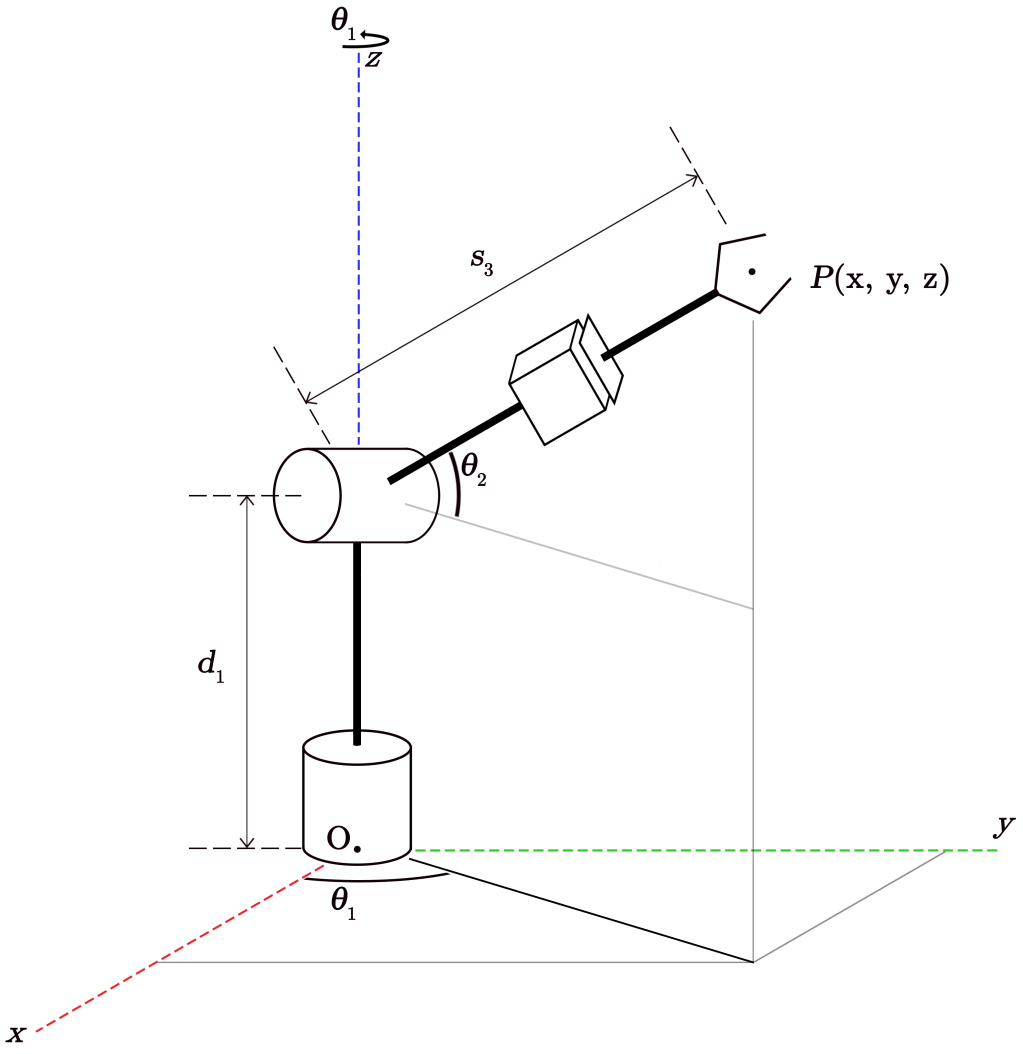

Figure 1.1: Kinematic diagram of the RRP manipulator [1].

Starting dimensions and constraints are as follows:

from project 1:

$d_1 = 125$ mm, $s_{3,\max} = 125$ mm,

Student number: $\underline{25}0203015\dot{}$

$v_{\max} = \dot{5} + 10 = 16$ mm/s ,

$a_{\max} = \dfrac{(\underline{25} + 50)}{2} = 37.5$ mm/s^2 , additionally, $s_{3,\min} = 0$.

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.

In the first project, forward and inverse kinematics relations of the manipulator were derived, and were applied on an $x-y-z$ composite path. In this project, said path is to be the trajectory on which the end-effector travels.

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

This report for the project consists of velocity-level and acceleration-level analyses, in addition to the previously conducted position-level analysis in part 1.

1.1 Workspace of the manipulator

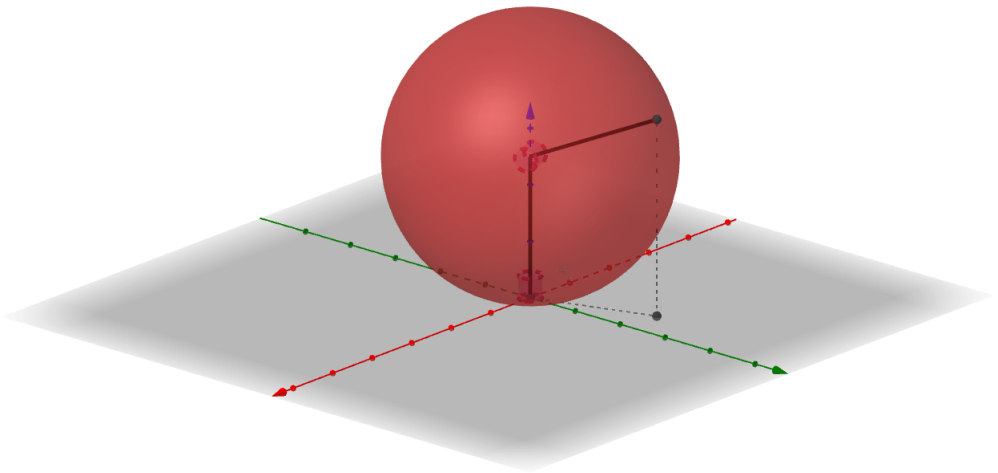

Figure 1.2: Geogebra [2] render of manipulator workspace

As shown in fig. 1.2 above, the workspace of the manipulator, under no angular constraints, is one that is perfectly spherical, with a line of singularity passing through the z axis, which is explained in the later sections.

Position and Velocity-Level Kinematics

Forward kinematics equations developed for the manipulator are:

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

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

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

...Tip point P being a function of θ_1, θ_2, and s_3. The relation of task space velocities to joint space velocities is as follows:

$$\bar{v} = \mathbf{J}_e \dot{\bar{q}} \tag{2.4}$$

$$\bar{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{2.5}$$

$$\dot{\bar{q}} = \begin{bmatrix} \dot{\theta_2} \\ \dot{\theta_2} \\ \dot{s_3} \end{bmatrix} \tag{2.6}$$

Where $\bar{v}$ is the vector of absolute velocities, and $\dot{\bar{q}}$ is the vector of joint space velocities.

The Jacobian matrix $\mathbf{J}_e$ of the manipulator is obtained as follows:

$$\mathbf{J}_e = \begin{bmatrix} \frac{\partial P_x}{\partial \theta_1} & \frac{\partial P_x}{\partial \theta_2} & \frac{\partial P_x}{\partial s_3} \\ \frac{\partial P_y}{\partial \theta_1} & \frac{\partial P_y}{\partial \theta_2} & \frac{\partial P_y}{\partial s_3} \\ \frac{\partial P_z}{\partial \theta_1} & \frac{\partial P_z}{\partial \theta_2} & \frac{\partial P_z}{\partial s_3} \end{bmatrix}_{3\times 3}$$

$$\mathbf{J}_e = \begin{bmatrix} -s_3 \cos\theta_2 \sin\theta_1 & -s_3 \sin\theta_2 \cos\theta_1 & \cos\theta_2 \cos\theta_1 \\ s_3 \cos\theta_2 \cos\theta_1 & -s_3 \sin\theta_2 \sin\theta_1 & \cos\theta_2 \sin\theta_1 \\ 0 & s_3 \cos\theta_2 & \sin\theta_2 \end{bmatrix}_{3\times 3}$$

Subscript e relates to the end-effector. Velocity-level singularities occur when the determinant of $\mathbf{J}_e = 0$, the full expression (see appendix A.1) is unwieldy, and is therefore left to MATLAB to simplify. The simplified determinant is

$$\det(\mathbf{J}_e) = s_3^2 \cos\theta_2 = 0 \tag{2.7}$$

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, wherein θ_1 is indeterminate, as shown in fig 2.1, which represents $\theta_2 = \frac{\pi}{2}$. It can be observed that side-to-side (parallel to the rotation axis of R_2) movement is restricted in said configurations. This position leads to singularity regardless of the s_3 value.

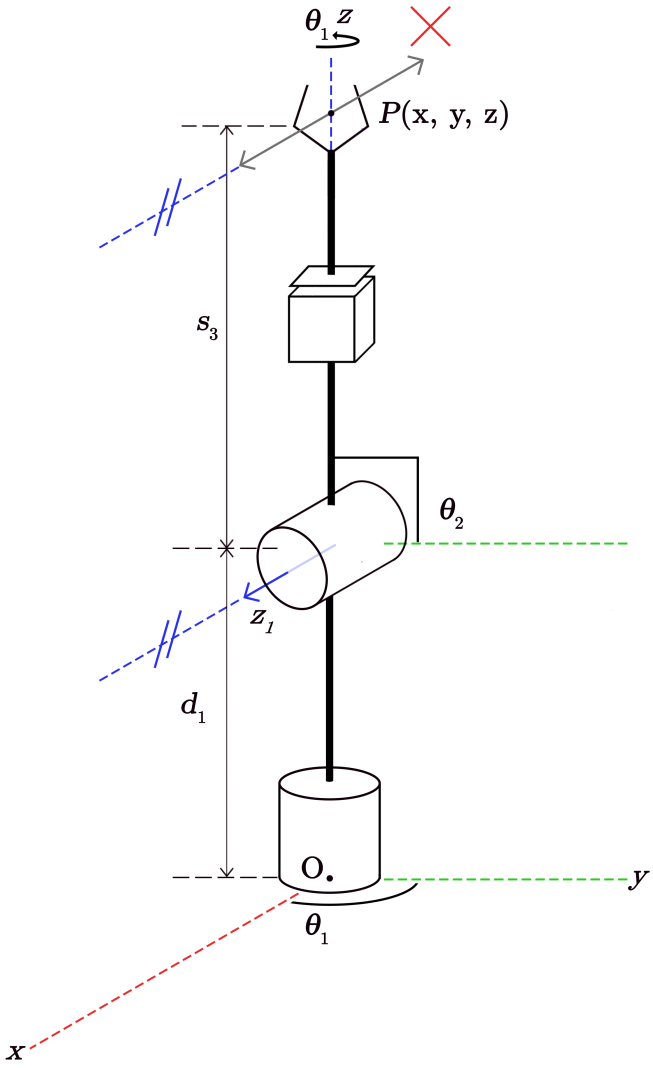

Figure 2.1: Geogebra [2] render of manipulator at home position with the destination point.

Inverse kinematics equations developed for the manipulator are:

$$\theta_1 = \arctan2(P_x, P_y) \tag{2.8}$$

$$\theta_2 = \arctan2(r, l) \tag{2.9}$$

$$s_3 = |(\sigma)\sqrt{r^2 + l^2}| \tag{2.10}$$

where $l = z - d_1$ and $r = |(\sigma)\sqrt{x^2 + y^2}|$, and σ represents sign ambiguity arising from taking the square root of the equation. Values s_3 and r are absolute lengths, and therefore the negative values can be discarded. From 2.10, s_3 has a double solution for $-l$ and $+l$,

From eqs. 2.8 and 2.9, a double solution is apparent, which is for any P,
$P(\theta_1,\ \theta_2,\ s_3) = P((\theta_1 + \pi),\ (\pi - \theta_2),\ s_3)$. A position-level singularity can be inferred from the relation for θ_2, with $r = 0$, for which singularity requires that $x = 0, y = 0$. It is worth noting that the double solution for θ_1 entails a change of θ_2 also.

Going in the opposite direction from 3.1, to obtain joint space velocities from task space velocities, inverse of Jacobian $\mathbf{J}_e$ is computed:

$$\dot{\overline{q}} = \mathbf{J}_e^{-1}\,\overline{v} \tag{2.11}$$

$$\mathbf{J}_e^{-1} = \begin{bmatrix} -\sin(\theta_1)/(s_3\cos(\theta_2)) & \cos(\theta_1)/(s_3\cos(\theta_2)) & 0 \\ -(\cos(\theta_1)\sin(\theta_2))/s_3 & -(\sin(\theta_1)\sin(\theta_2))/s_3 & \cos(\theta_2)/s_3 \\ \cos(\theta_1)\cos(\theta_2) & \cos(\theta_2)\sin(\theta_1) & \sin(\theta_2) \end{bmatrix}_{3\times 3}$$

Equation 2.11 is applied for every timestep of the trajectory, and the values are arranged into an array (see appendix A.2).

Acceleration-Level Kinematics

Adding on to the previous section, one can further differentiate the task-space parameters, $\bar{v}$ into $\dot{\bar{v}}$, as follows

$$\bar{v} = \mathbf{J}_e \dot{\bar{q}} \tag{3.1}$$

$$\dot{\bar{v}} = \bar{a} = \mathbf{J}_e \ddot{\bar{q}} + \dot{\mathbf{J}}_e \bar{q} \tag{3.2}$$

And from eq 2.11, we know that

$$\dot{\bar{q}} = \mathbf{J}_e^{-1} \bar{v} \tag{3.3}$$

therefore multiplying both sides of 3.2, we obtain

$$\ddot{\bar{q}} = \mathbf{J}_e^{-1} \left(\bar{a} - \dot{\mathbf{J}}_e \dot{\bar{q}} \right) \tag{3.4}$$

The derivative of the Jacobian, with simplification steps omitted is shown below, with c and s representing cosine and sine respectively.

$$\dot{\mathbf{J}}_e = \begin{bmatrix} s_3 s(\theta_1) s(\theta_2) - s_3 c(\theta_1) c(\theta_2) - c(\theta_2) s(\theta_1) & s_3 s(\theta_1) s(\theta_2) - s_3 c(\theta_1) c(\theta_2) - c(\theta_1) s(\theta_2) & -s(\theta_1 + \theta_2) \\ c(\theta_1) c(\theta_2) - s_3 c(\theta_1) s(\theta_2) - s_3 c(\theta_2) s(\theta_1) & -s(\theta_1) s(\theta_2) - s_3 c(\theta_1) s(\theta_2) - s_3 c(\theta_2) s(\theta_1) & c(\theta_1 + \theta_2) \\ 0 & (\theta_2) - s_3 s(\theta_2) & c(\theta_2) \end{bmatrix}_{3\times 3}$$

Equation 3.4 is applied for every timestep of the trajectory, and the values are arranged into an array (see appendix).

Trajectory & Motion Profile

From project 1, path length is determined to be 31.25 mm.

The path direction is arbitrarily chosen to be orthogonal to the axis $y = -x$, as shown in fig 4.3 below. Destination point and path are mathematically defined in Geogebra.

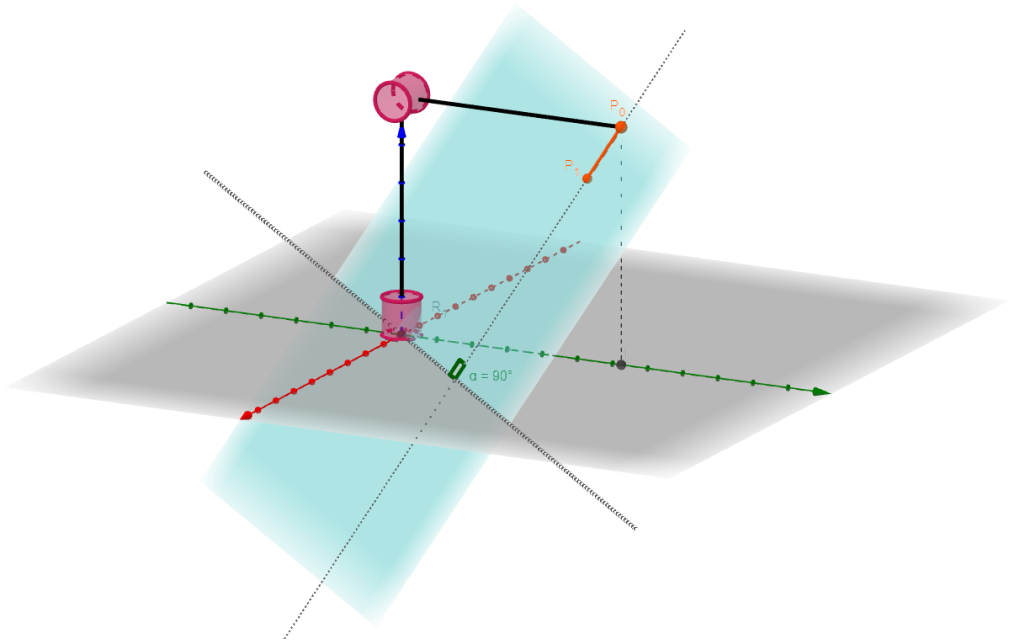

Figure 4.1: Geogebra render of manipulator at home position with the destination point (x axis in green, y axis in red).

The trajectory is sampled at a rate of 10 Hz (per 0.1 seconds) from $t = 0$ and $t = T$. Total travel time T is calculated from the minimum time it takes for the tip point to travel from point P_0 to P_1.

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

Acceleration during trapezoidal motion profile is as follows:

$$\begin{cases} a & 0 \le t \le t_1 \\ 0 & t_1 < t \le t_2 \\ -a & t_2 < t \le T \end{cases}$$

Velocity and subsequently displacement can be derived from the above piecewise function via integration or equations of motion.

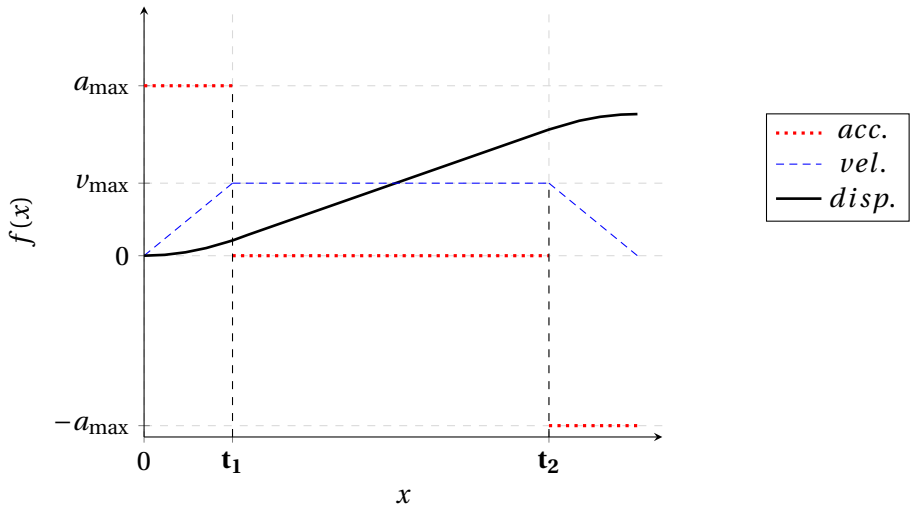

Figure 4.2: Graph of acceleration (mm/s^2), velocity (mm/s) and displacement (mm) versus time (s).

The graph above shows the trapezoidal motion profile, with points of infinite jerk at t_1 and t_2.

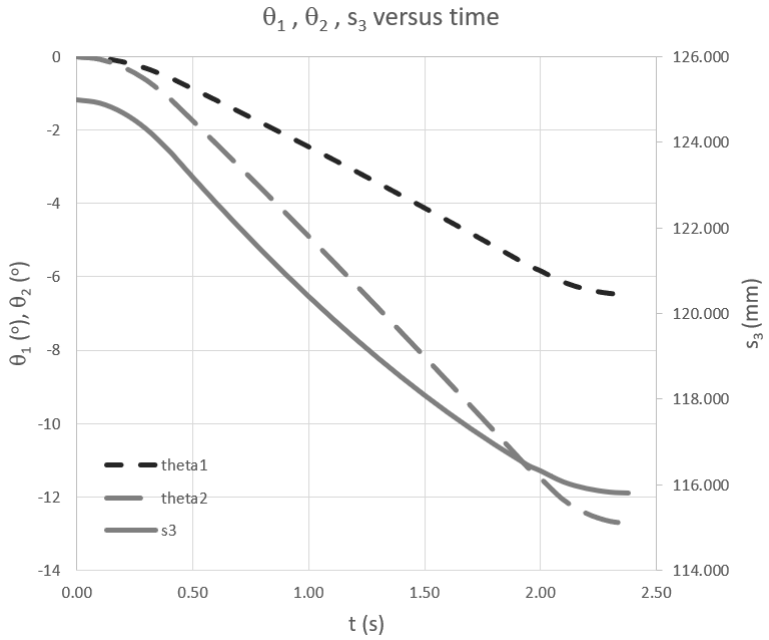

Figure 4.3: Graph of θ_1 (°), θ_2 (°) and s_3 (mm) versus time (s).

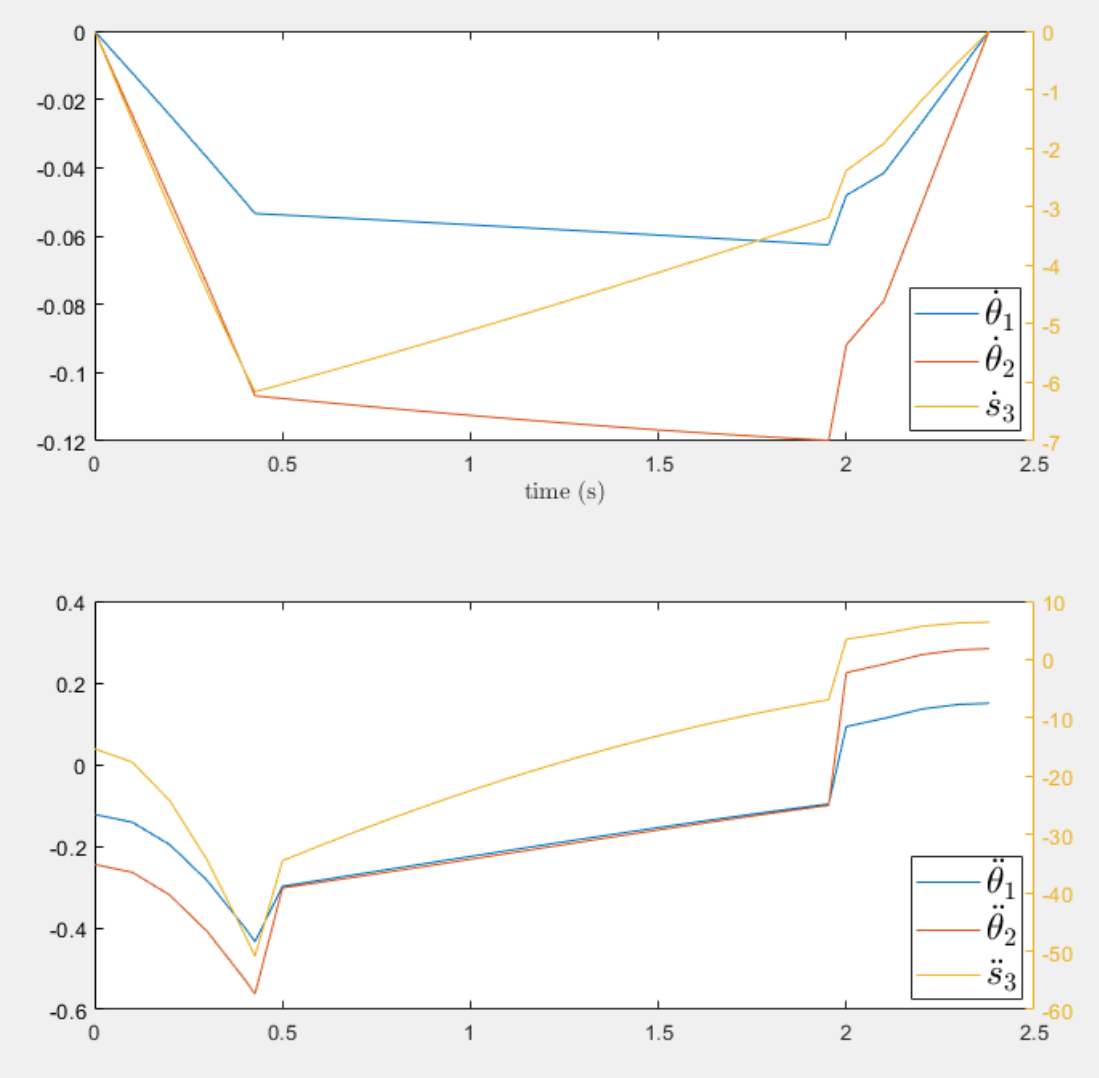

Figure 4.4: MATLAB output graphs of $\overline{q}$, $\dot{\overline{q}}$ versus time (s).

Discussion - Conclusion

Adding to the position-level analyses of project 1, this project included the calculations for velocity-level and acceleration-level analyses, in both forward and inverse kinematics. In addition, the workspace is shown, though it would have been preferable to show it in terms of a grid of points of the end-effector for a more rigorous picture, which is overkill for this specific type of arm.

Trapezoidal motion profile is employed to plan the trajectory of the end-effector, therefore the givens are the task-space variables from which joint-space variables are derived.

References

[1] EMREM, Mert. Forward & Inverse Kinematic Analyses of Serial RRP Manipulator. Project Report, 2021. In: İzmir Institute of Technology

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

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

Appendix

A.1 Equations

→ Unsimplified determinant of Jacobian $\mathbf{J}_e$

$$\begin{aligned}\det(\mathbf{J}_e) = & s_3^2 \cos(\theta_1)^2 \cos(\theta_2)^3 + s_3^2 \cos(\theta_1)^2 \cos(\theta_2) \sin(\theta_2)^2 \\ & + s_3^2 \cos(\theta_2)^3 \sin(\theta_1)^2 + s_3^2 \cos(\theta_2) \sin(\theta_1)^2 \sin(\theta_2)^2\end{aligned}$$

A.2 Data

θ_1 (rad)	θ_2 (rad)	s_3 (mm)	$\dot{\theta_1}$ (rad/s)	$\dot{\theta_2}$ (rad/s)	$\dot{s_3}$ (mm/s)	$\ddot{\theta_1}$ (rad/s^2)	$\ddot{\theta_2}$ (rad/s^2)	$\ddot{s_3}$ (mm/s^2)
0.000	0.000	125.000	0.000	0.000	0.000	-0.122	-0.245	-15.306
-0.001	-0.001	124.924	-0.012	-0.025	-1.526	-0.141	-0.264	-17.583
-0.002	-0.005	124.696	-0.025	-0.049	-3.024	-0.196	-0.320	-24.182
-0.006	-0.011	124.321	-0.037	-0.074	-4.464	-0.285	-0.409	-34.424
-0.010	-0.020	123.806	-0.050	-0.100	-5.818	-0.400	-0.527	-47.219
-0.011	-0.023	123.646	-0.053	-0.107	-6.161	-0.434	-0.562	-50.887
-0.015	-0.030	123.199	-0.054	-0.108	-6.031	-0.298	-0.302	-34.488
-0.021	-0.041	122.604	-0.054	-0.109	-5.851	-0.283	-0.289	-31.893
-0.026	-0.052	122.028	-0.055	-0.110	-5.669	-0.269	-0.275	-29.393
-0.032	-0.063	121.471	-0.056	-0.111	-5.484	-0.254	-0.261	-26.989
-0.037	-0.074	120.931	-0.056	-0.112	-5.297	-0.240	-0.246	-24.685
-0.043	-0.085	120.411	-0.057	-0.113	-5.107	-0.225	-0.232	-22.484
-0.049	-0.097	119.910	-0.057	-0.113	-4.915	-0.211	-0.218	-20.386
-0.054	-0.108	119.428	-0.058	-0.114	-4.721	-0.196	-0.203	-18.394
-0.060	-0.120	118.965	-0.059	-0.115	-4.524	-0.182	-0.189	-16.510
-0.066	-0.131	118.523	-0.059	-0.116	-4.325	-0.168	-0.175	-14.734
-0.072	-0.143	118.100	-0.060	-0.117	-4.124	-0.154	-0.160	-13.068
-0.078	-0.155	117.698	-0.060	-0.117	-3.920	-0.141	-0.146	-11.511
-0.084	-0.166	117.316	-0.061	-0.118	-3.715	-0.128	-0.133	-10.064
-0.090	-0.178	116.955	-0.062	-0.119	-3.508	-0.115	-0.120	-8.726
-0.096	-0.190	116.615	-0.062	-0.120	-3.298	-0.103	-0.107	-7.496
-0.100	-0.196	116.442	-0.063	-0.120	-3.186	-0.097	-0.100	-6.886
-0.102	-0.201	116.332	-0.048	-0.092	-2.383	0.093	0.225	3.473
-0.107	-0.211	116.070	-0.042	-0.079	-1.920	0.113	0.246	4.445
-0.111	-0.218	115.916	-0.027	-0.051	-1.186	0.136	0.269	5.686
-0.113	-0.221	115.831	-0.012	-0.023	-0.514	0.147	0.281	6.272
-0.113	-0.222	115.811	0.000	0.000	0.000	0.150	0.283	6.404

A.3 MATLAB Code

```
%%%%%%%%%%%%%%%%%%%     Mert Emrem - 250203015     %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%     CE301 - Assignment #3      %%%%%%%%%%%%%%%%%%%%%%

clc; clear all

syms f(theta1,theta2,s3) d1

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

DJ(theta1,theta2,s3) = simplify(diff(J,theta1) + diff(J,theta2) + diff(
    J,s3))

Jdet(theta1,theta2,s3) = simplify(det(J))

Jinv(theta1,theta2,s3) = simplify(inv(J))

h = 31.25; amax = 37.5; vmax = 16; t_1 = vmax/amax; sampling = 0.1;
h_1 = (amax*t_1^2)/2;
t_full = (31.25 - h_1*2)/16 + 2*t_1;
t_2 = t_full - t_1;
t_1sub = ceil(t_1 * 10) / 10; t_2sub = ceil(t_2 * 10) / 10;

time = xlsread('250203015_Project2.xlsx','AC5:AC31');
a_p = xlsread('250203015_Project2.xlsx','AD5:AD31');
v_p = xlsread('250203015_Project2.xlsx','AE5:AE31');
theta_1 = xlsread('250203015_Project2.xlsx','AL5:AL31');
theta_2 = xlsread('250203015_Project2.xlsx','AM5:AM31');
s_3 = xlsread('250203015_Project2.xlsx','AP5:AP31');

q = [theta_1 theta_2 s_3]';

x_coeff = 0.5/1.225; y_coeff = 0.5/1.225; z_coeff = 1/1.225;
v_px = x_coeff*v_p; v_py = y_coeff*v_p; v_pz = z_coeff*v_p;
a_px = x_coeff*a_p; a_py = y_coeff*a_p; a_pz = z_coeff*a_p;

v_mat = [-v_px -v_py -v_pz]';
a_mat = [-a_px -a_py -a_pz]';

q_dot = [];

for i = 1:length(v_mat)

```

```
    Jinp = Jinv(q(1,i),q(2,i),q(3,i));
    conc = double(Jinp * v_mat(1:3,i));
    q_dot = [q_dot conc];

end


q_ddot = [];

for i = 1:length(v_mat)

    Jinp = Jinv(q(1,i),q(2,i),q(3,i));
    conc = double(Jinp * (a_mat(1:3,i)-Jdet(q_dot(1,i),q_dot(2,i),q_dot
   (3,i))));
    q_ddot = [q_ddot conc]

end




subplot(2,1,1);
plot(time, q_dot(1, 1:27))

hold on

plot(time, q_dot(2, 1:27))
yyaxis right
plot(time, q_dot(3, 1:27))

xlabel('time (s)', 'interpreter', 'latex')


leg1 = legend('$\dot{\theta}_1$','$\dot{\theta}_2$','$\dot{s}_3$');
set(leg1,'Interpreter','latex');
set(leg1,'FontSize',17);

hold off

subplot(2,1,2);
plot(time, q_ddot(1, 1:27))
hold on
plot(time, q_ddot(2, 1:27))
yyaxis right
plot(time, q_ddot(3, 1:27))


```

```
90 leg1 = legend('$\ddot{\theta}_1$','$\ddot{\theta}_2$','$\ddot{s}_3$');
91 set(leg1,'Interpreter','latex');
92 set(leg1,'FontSize',17);
93 hold off
```