**Robotics Project**

By: Adi Ziv

Lecturer: Ilana nisky

2024

Index

[PROJECT PART 3 3](#_Toc175998552)

[Question 1 3](#_Toc175998553)

[Question 2 5](#_Toc175998554)

[Simulation A, Trajectory 1 5](#_Toc175998555)

[Simulation B, Trajectory 1 13](#_Toc175998556)

[Simulation A, Trajectory 2 20](#_Toc175998557)

[Simulation B, Trajectory 2 28](#_Toc175998558)

[D -First Trajectory 35](#_Toc175998559)

[D -Second Trajectory 35](#_Toc175998560)

[E Bonus 35](#_Toc175998561)

[Matlab Code 36](#_Toc175998562)

# PROJECT PART 3

### Question 1

In this part of the project we saw how external forces affects the robot`s movement. The robot we were given is a PRR robot, with 2 degrees of freedom.

We can write the dynamic equation of the PRR robotic system, as 6 first order equations.

For free movement (without wind), the dynamic equations describing the system are:

Adding a P-D controller

Comparing the above resulting the dynamic equation:

When q is the vector of joints parameters, is the desired joints parameters and are their velocities and acceleration, respectively. In order to solve numerically, we define:

, and   
as is fixed and we get the equations:

### Question 2

For a perfect model for viscosity and lack of wind, meaning that

In the figure below we can see the robot. The workspace is calculated as a circle from the first revolute joint of the robot with a maximum radius possible, the prismatic joint is marked as a black dashed line, the links are in red, the revolute joints are black dots, and the desired trajectory is purple. Both the start and end points are orange and marked as a triangle and asterisk accordingly. All the legends are the same for the rest of the project as necessary. In this question we will see the robot`s movement according to the minimal jerk, in two different trajectories, and for two different alphas, for A alpha is equal to 1 meaning no drag force affects the robot`s movement, and for B alpha is 0.05.

### Simulation A, Trajectory 1

The starting point for this trajectory is (10, 3, 2) and the end point is (13, 1, 2).

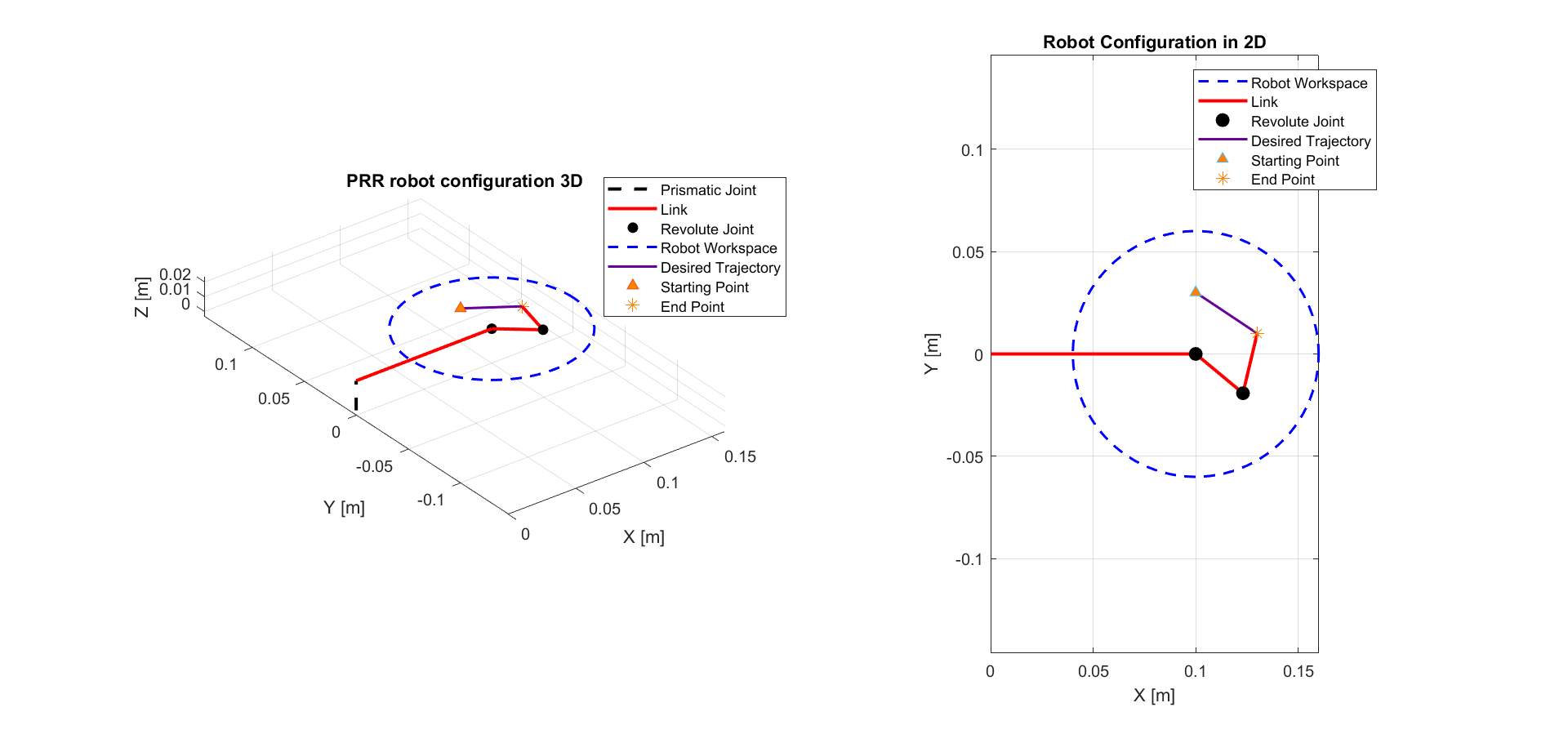
In the figure below we can see the robot`s configuration in 2D and 3D, at the end point of the desired trajectory.

figure - the PRR robot configuration with the desired trajectory in 3D and 2D

The figure shows a PRR robot configuration in both 3D and 2D views. The left subplot illustrates the robot's workspace and desired trajectory in a 3D space, while the right subplot presents the same configuration in a 2D plane. The purple line depicts the desired trajectory according to minimal jerk.

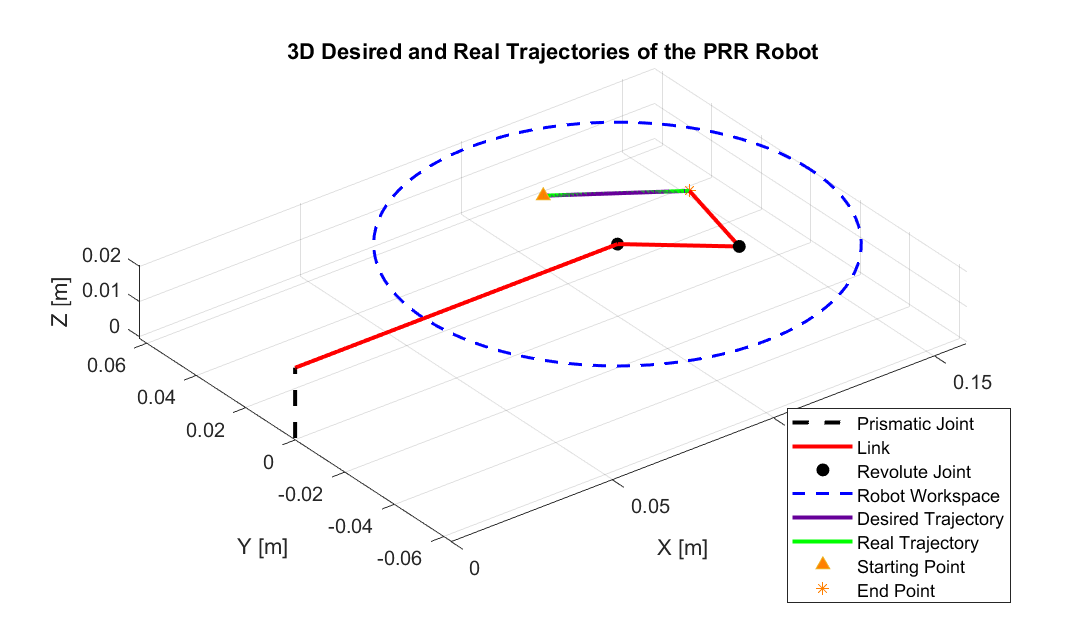


figure - the PRR robot in 3D with the desired and real trajectory for alpha =1

Here is the 3D figure for the robot`s trajectories. In this simulation for we can see that the desired trajectory and the actual trajectory are the same. We assumed that the outcome for the real trajectory will stay the same as the drag force cancels.



figure - The PRR robot desired and real trajectory in 2D for alpha =1

Here for 2D we get a closer look for we can see that the desired trajectory and the actual trajectory are exactly the same. We assumed that the outcome for the real trajectory will stay the same as the drag force cancels.

For both figures for the trajectories of the robot, we can see the real trajectory the robot will follow (without the presence of the drag force), together with the desired. They are identical to each other in this simulation even though they where calculated differently. The actual real trajectory was calculated by ODE solver, while the desired was calculated from the minimal jerk function.

The next figures will show the parameters for both the real and desired trajectories.

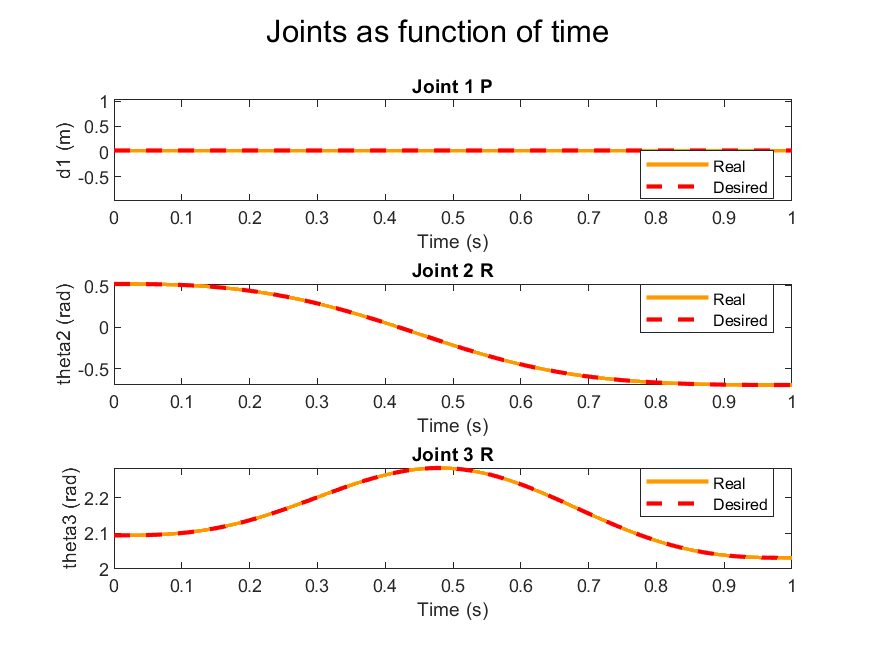
In this figure we can see the joints movement as a function of time. In orange is the real movement (actual) and in red is the desired according to minimal jerk. Because , we can see the desired and real trajectories of the joints coincide. Here for the length of the prismatic joint equals to 0.02, and stay constant during the simulation period. The revolute joint with decreases from positive to negative and the third joint is seemed to get a maximum point. All in all, we can see the robot is following the desired trajectory in an accurate way.

figure - The Joint of the PRR robot as a function of time, with alpha = 1

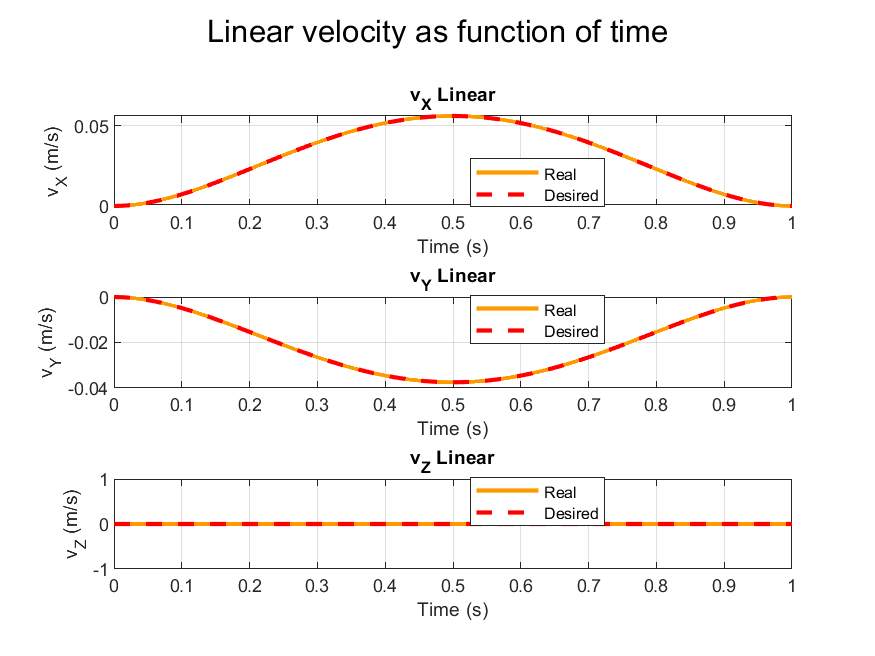


figure - The linear velocities of the PRR robot as function of time, for alpha =1

In this figure we can see the linear velocity as function of time. Because , we can see the desired and real velocities over time graphs coincide. The velocity in the x axis has a maximum point at half of the simulation time, indicating the robot is moving in the positive X direction and increasing its velocity as the time goes. After the robot is passed half the simulation time, the velocity is decreasing until the robot stops and the velocity reaches zero. The linear velocity in the Y axis is having a minimum point half way to the simulation, implying that the robot is moving in the negative direction of the Y axis. The linear velocity in Z does not exist due to the lack of movement in the prismatic joint.

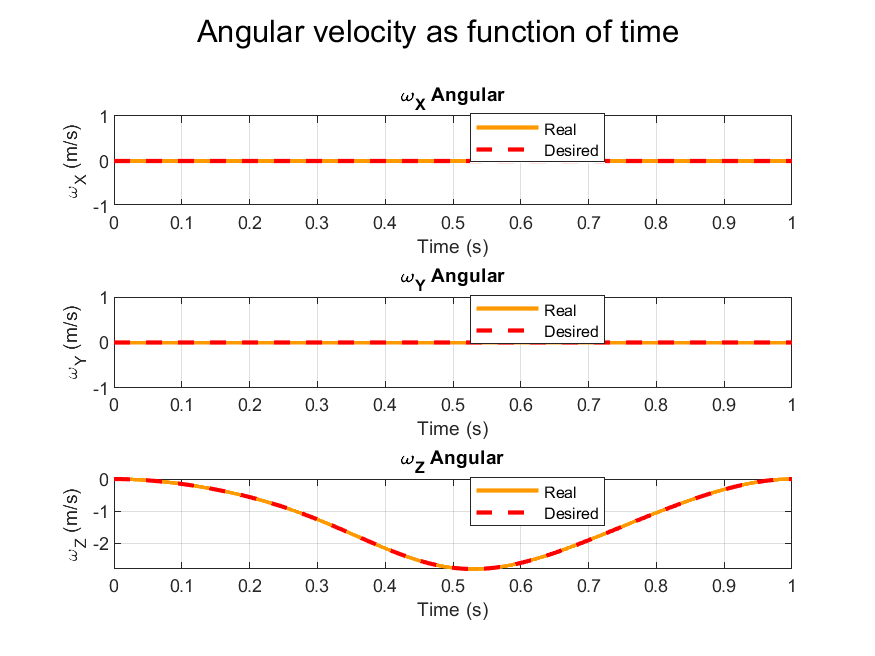


figure -The angular velocities of the PRR robot as function of time, with alpha =1

The figure depicts the angular velocities of the system as functions of time, along the X, Y, and Z axes. The angular velocity in x and y ​ plots, both real and desired angular velocities remain constant at zero, indicating no rotational movement along the X and Y axes. In contrast, the angular velocity of Z plot shows a sinusoidal variation, suggesting a controlled rotational movement around the Z-axis. The close alignment between the real and desired angular velocities in all plots suggests that the robot is accurately following the desired motion trajectory. Because the Z angular velocity is negative, we expect the robot to move with the positive direction of the clock.

תמונה שמכילה טקסט, קו, תרשים, גופן

התיאור נוצר באופן אוטומטי

figure --The Torques of the PRR robot as function of time, with alpha =1

The graph illustrates the torques applied to the robot's joints as functions of time with *=1.* The plot for the first torque​ remains constant at zero throughout the time period. This suggests that no torque is being applied to the first joint, which might be due to the fact that the prismatic joint is locked in a specific position. The second joint`s torque is negative and decreasing, meaning that the force required to this movement is against the positive direction, and indicating that the joint resist the force applied from the robot`s movement. the third joint`s torque is positive and increasing, due to a movement with the positive direction.

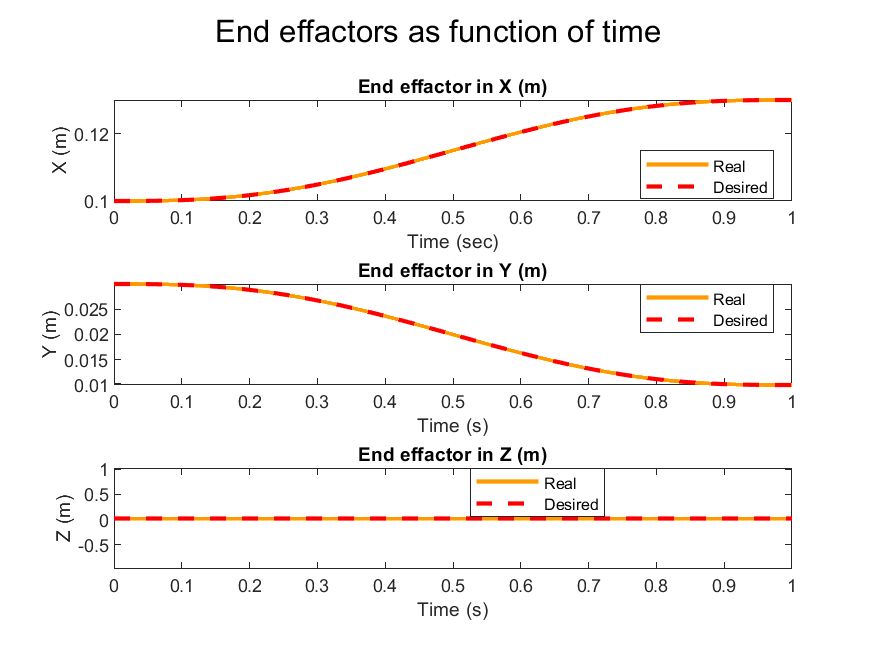


figure 8 -The end effactor positions in X, Y, Z of the PRR robot as function of time, with alpha =1

The figure presents the end effector at the end of the robot's links as functions of time with *=1, in X Y and Z directions.* With alpha equals to 1 we expect that the robot will follow the same route as the desired trajectory, due to no force applied from outside on the robot.

The close match between the real and desired parameters indicate the robot's controlled movement under these conditions.

### Simulation B, Trajectory 1

This part will repeat the simulation with .

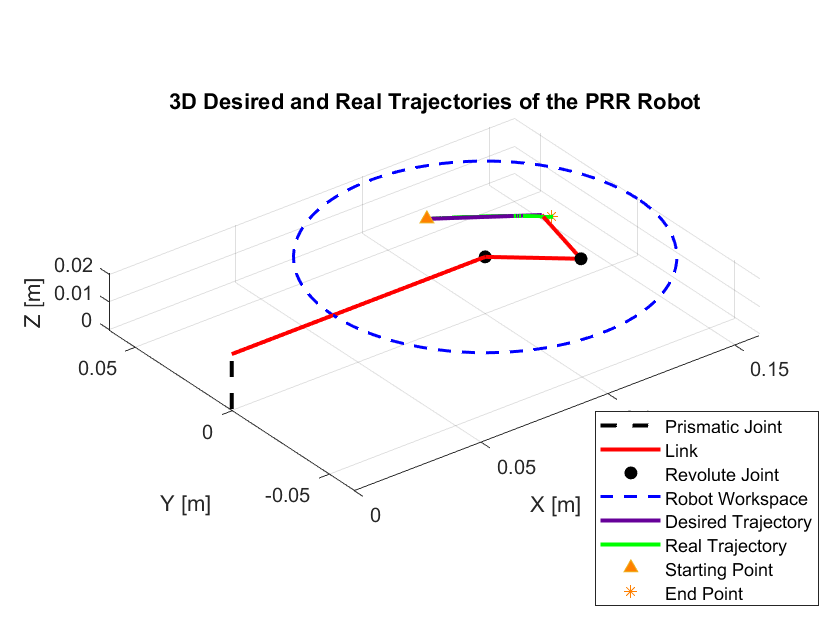


figure - the PRR robot trajectory with alpha = 0.05

In this figure we can observe the robot in 3D, with the desired and real trajectories. In this simulation for 0.05 we can see that the desired trajectory and the actual trajectory are **not** the same. We assume that the outcome for the real trajectory will be different with present of outside forces.



figure - The PRR robot desired and real trajectory in 2D for alpha = 0.05

In this figure we can see the real and desired trajectories in 2D. As we can see, there is a small change in the trajectory when the drag force is affecting the simulation.

There are two sets of graphs below for each parameter, as function of time. Each set represents the time evolution of the joint positions for the robot for the first simulation and together with the current simulation. We will analyze the differences between the parameter for the simulation of .



figure - The Joint of the PRR robot as a function of time, with alpha = 1 and alpha = 0.05

In this figure we can see the values of the joints (meters or radians) dependency on time. For , the real trajectory is in cyan, and desired trajectory is the same as before. We can see the real trajectories of the joints are different between the simulations. As is constant Joint 1 as a function of time is a constant line, because the constant joint does not get affected by the drag force. We can also notice that has a larger margin between the simulations, meaning the force affected the third joint more than others.

  
In this figure we can see the velocity for the simulation of alpha = 0.05 in cyan, and the previous simulation as the legend shows. The real velocity follows the desired path with small deviations, particularly noticeable around the peaks of the curves in the X and Y plots. With a lower , the real velocities slightly deviate from the desired ones. As alpha increases, the robot's motion becomes more controlled, and the real velocities align more closely with the desired velocities. When the velocity is negative this suggests that the robot is moving in the opposite direction to the positive axis. The velocity is going back to zero as the movement ends.

figure - The linear velocities of the PRR robot as function of time , with alpha = 1 and alpha = 0.05



figure - The angular velocities of the PRR robot as function of time , with alpha = 1 and alpha = 0.05

This figure shows the angular velocities of the two simulations in X, Y, and Z axis. In the simulation where (in cyan) the real and desired do not coincide and there is a gap in the Z angular velocity.

In these graphs we can see only in Z axis we have angular velocity, as expected by the robot`s work space we sat to this simulation. The angular velocity is shown to start from zero, than getting negative until reaching a peak point and than going back up to zero. This is matching what we would expect by looking at the simulation video, because the robot is moving with the clock`s positive direction.



figure -The torques of the PRR robot as function of time , with alpha = 1 and alpha = 0.05

This image presents the torques as function of time for both simulations, simulation b with is depicted in cyan on top of the previous orange. As we can see the torque for the first joint is constant, which is as we expect due to the constant prismatic joint in a robot with only 2 degrees of freedom. The second torque is negative and the third is positive, due to a movement against and with the positive direction of the movement accordingly.

For these two graphs the torques are similar. the first torque is constant zero, the second torque is negative and the third is positive This behavior indicates that initially, a negative torque is applied to the second revolute joint to counteract any angular momentum or inertia. As time progresses, the requirement for this torque reduces as the joint reaches the desired configuration or as external forces bring the joint closer to equilibrium. The third torque, starts at 0 [Nm] and increases up to about 3 [Nm], showing an opposite to . This increase suggests that a positive torque is required at the third revolute joint, likely to accelerate the joint from its initial state to the desired position. As the joint reaches its desired state, the torque required stabilizes, which is why the plot levels off toward the end.

This figure shows the end effector as functions of time with *=1*, and in cyan in X Y and Z directions. The outcome from simulation b real trajectory differs from simulation a and the desired trajectory mostly in X and Y plots, in the area of the minimum and maximum points in the graph, which I can assume are less stable movements from the start of the movement. As expected, no change is seen in Z because it is constant.

figure - The torques of the PRR robot as function of time , with alpha = 1 and alpha = 0.05

The graphs collectively demonstrate how changing the control parameter affects the robot's ability to follow desired trajectories. When is set to 1, the robot exhibits perfect tracking performance with the real and desired trajectories overlapping. However, when is reduced to 0.05, the control precision decreases, leading to deviations between the actual and intended paths of both individual joints and the overall end-effector trajectory in 2D space.

### Simulation A, Trajectory 2

The starting point for this trajectory is (10, 3, 2) and the end point is (8, 4, 2).

In the figure below we can see the robot`s configuration in 2D and 3D, at the end point of the desired trajectory

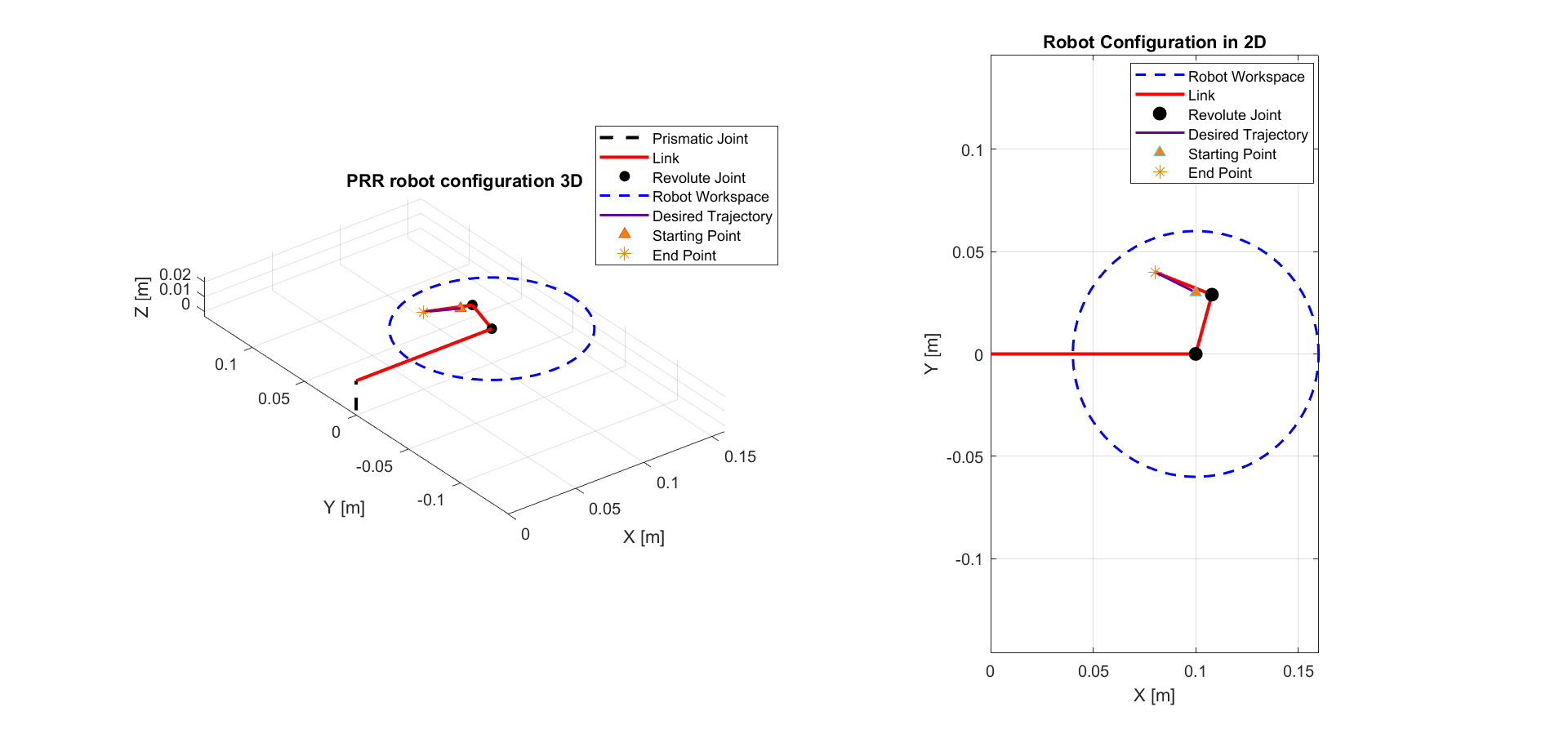


figure - the PRR robot configuration with the desired trajectory in 3D and 2D

The figure shows a PRR robot configuration in both 3D and 2D views. The left subplot illustrates the robot's workspace and desired trajectory in a 3D space, while the right subplot presents the same configuration in a 2D plane. The purple line depicts the desired trajectory according to minimal jerk. In this simulation . We assumed that the outcome for the real trajectory will stay the same as the drag force cancels.

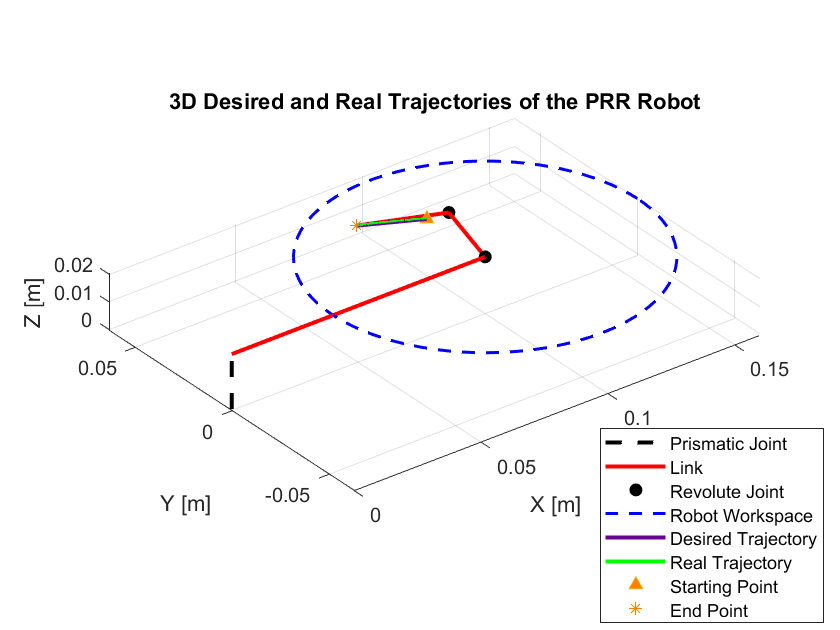


figure - The PRR robot in 3D with the desired and real trajectory for alpha =1

Here is the 3D figure for the robot`s trajectories. In this simulation for we can see that the desired trajectory and the actual trajectory are the same. We assumed that the outcome for the real trajectory will stay the same as the drag force cancels.



figure - The PRR robot desired and real trajectory in 2D for alpha =1

Here for 2D we get a closer look for we can see that the desired trajectory and the actual trajectory are exactly the same. We assumed that the outcome for the real trajectory will stay the same as the drag force cancels.

For both figures for the trajectories of the robot, we can see the real trajectory the robot will follow (without the presence of the drag force), together with the desired. They are identical to each other in this simulation even though they where calculated differently. The actual real trajectory was calculated by ODE solver, while the desired was calculated from the minimal jerk function.

The next figures will show the parameters for both the real and desired trajectories.

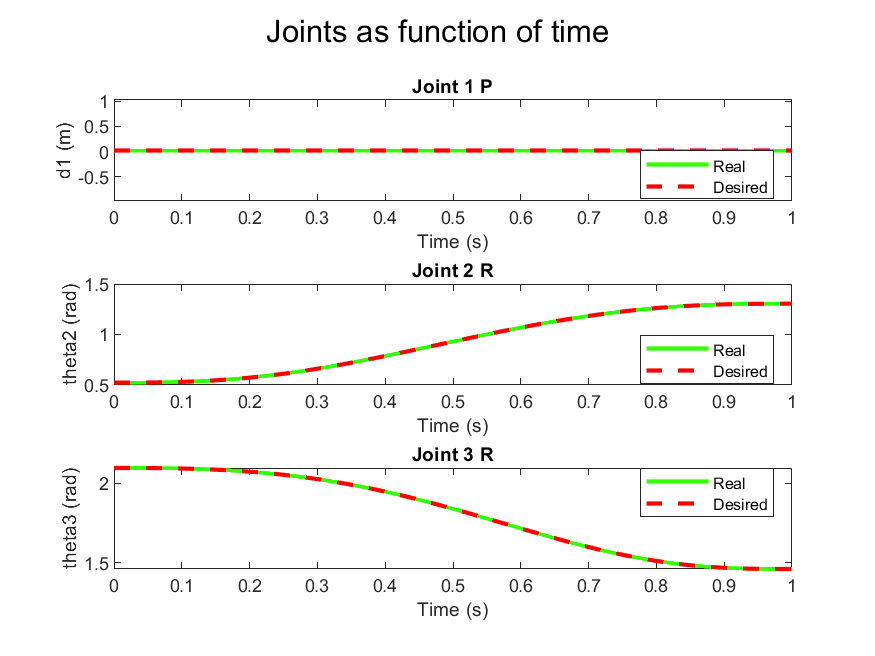


figure 19 - The Joint of the PRR robot as a function of time, with alpha = 1

In this figure we can see the joints parameters. Because , we can see the desired and real trajectories of the joints coincide. Here for the length of the prismatic joint equals to 0.02, and stay constant during the simulation period. The revolute joint with increases gradually and the third joint is seemed to decrease. All in all, we can see the robot is following the desired trajectory in an accurate way.

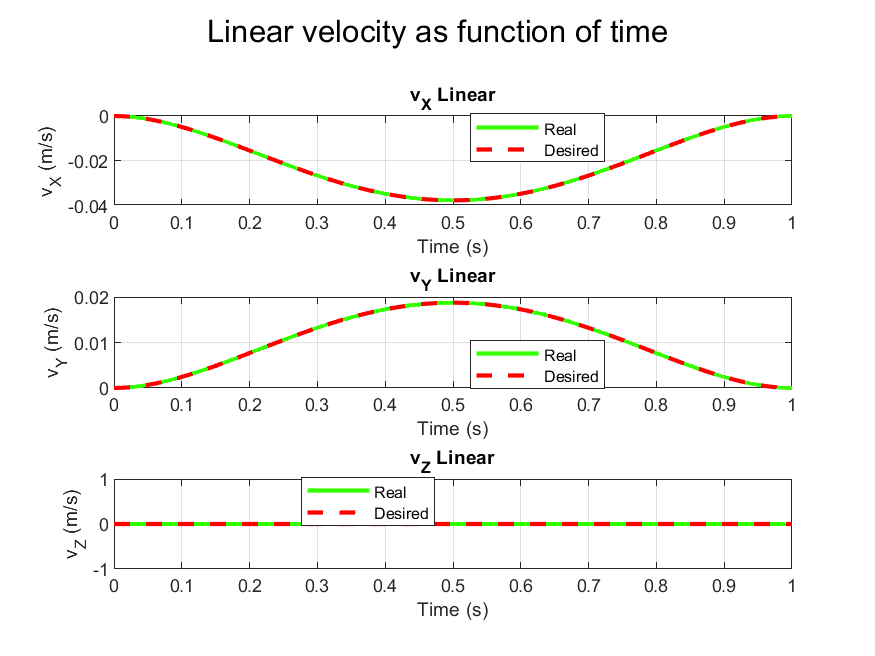


figure - The linear velocities of the PRR robot as function of time, for alpha =1

Because , we can see the desired and real velocities over time graphs coincide. The velocity in the x axis has a minimum point at half of the simulation time, indicating the robot is moving in the negative X direction and increasing its velocity in that direction as the time goes. After the robot is passed half the simulation time, the velocity is decreasing until the robot stops and the velocity reaches zero. The linear velocity in the Y axis is having a maximum point half way to the simulation, implying that the robot is moving in the positive direction of the Y axis. The linear velocity in Z does not exist due to the lack of movement in the prismatic joint.

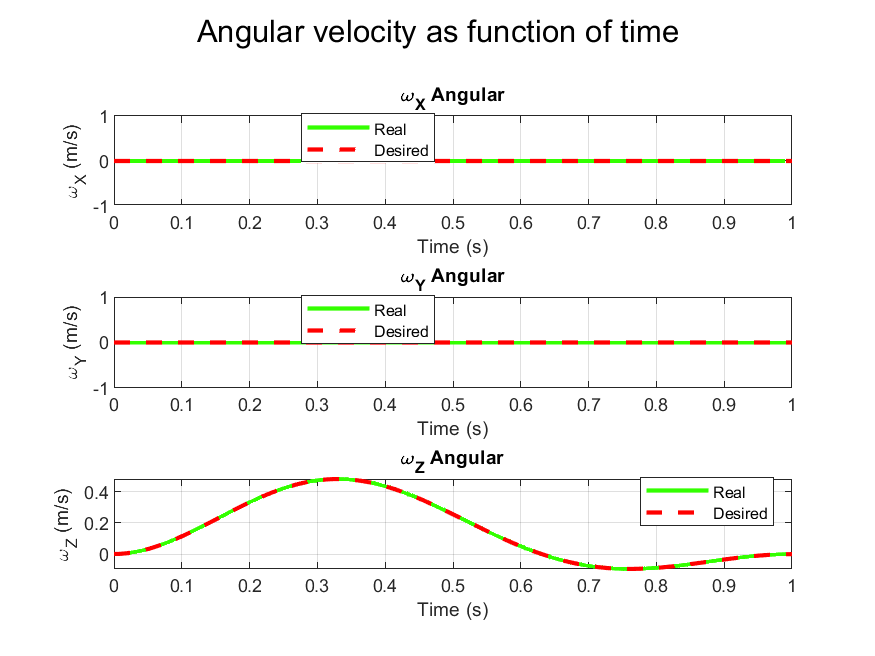


figure - The angular velocities of the PRR robot as function of time, with alpha =1

The figure depicts the angular velocities of the system as functions of time, along the X, Y, and Z axes. The angular velocity in x and y ​ plots, both real and desired angular velocities remain constant at zero, indicating no rotational movement along the X and Y axes. In contrast, the angular velocity of Z plot shows a sinusoidal variation, suggesting a controlled rotational movement around the Z-axis. The close alignment between the real and desired angular velocities in all plots suggests that the robot is accurately following the desired motion trajectory. Because the Z angular velocity is positive, we expect the robot to move with the negative direction of the clock.

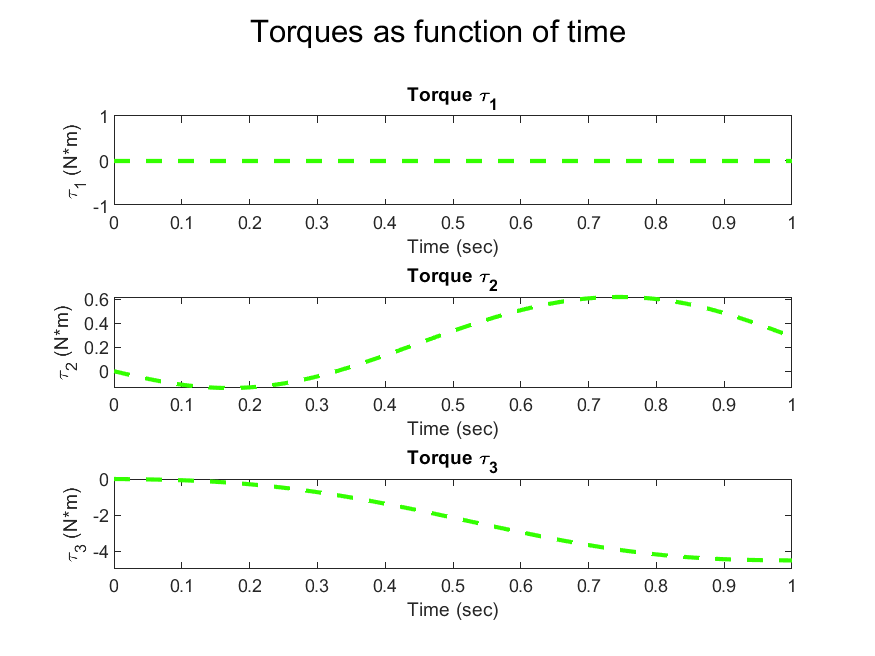


figure - The Torques of the PRR robot as function of time, with alpha =1

The graph illustrates the torques applied to the robot's joints as functions of time with *=1.*

The plot for the first torque​ remains constant at zero throughout the time period. This suggests that no torque is being applied to the first joint, which might be due to the fact that the prismatic joint is locked in a specific position. The second joint`s torque is positive and increasing until the maximum point, meaning that the force required to this movement is with the positive direction, and indicating that the joint resist the force applied from the robot`s movement. the third joint`s torque is negative and decreasing gradually, indicates that the movement in this joint is stable.

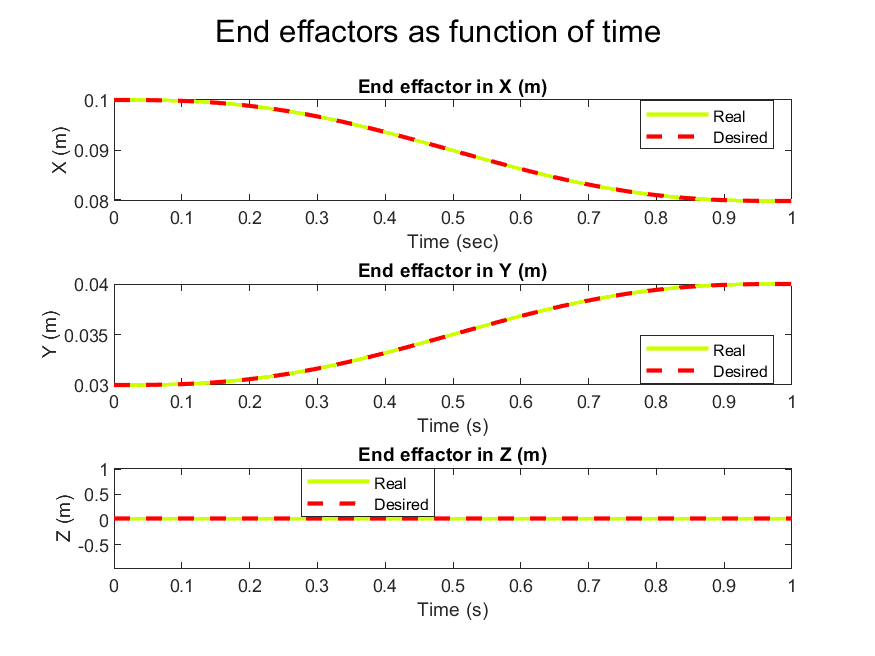


figure - The end effactor positions in X, Y, Z of the PRR robot as function of time, with alpha =1

The figure presents the end effector at the end of the robot's links as functions of time with *=1, in X Y and Z directions.* With alpha equals to 1 we expect that the robot will follow the same route as the desired trajectory, due to no force applied from outside on the robot. The End effector is moving with the positive direction of Y and negative of X. no movement in Z because the prismatic joint is locked.

### Simulation B, Trajectory 2

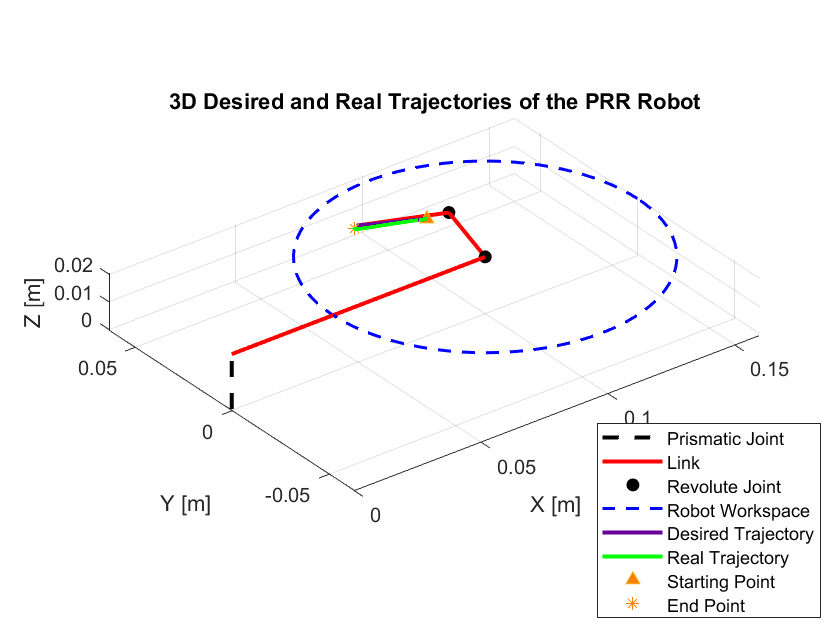
This part will repeat the simulation with 

figure - the PRR robot trajectory with alpha = 0.05

In this figure we can observe the robot in 3D, with the desired and real trajectories. In this simulation for 0.05 we can see that the desired trajectory and the actual trajectory are **not** the same. We assume that the outcome for the real trajectory will be different with present of outside forces.



figure - The PRR robot desired and real trajectory in 2D for alpha = 0.05

In this figure we can see the real and desired trajectories in 2D. As we can see, there is a small change in the trajectory when the drag force is affecting the simulation.

There are two sets of graphs below for each parameter, as function of time. Each set represents the time evolution of the joint positions for the robot for the first simulation and together with the current simulation. We will analyze the differences between the parameter for the simulation of .



figure - The Joint of the PRR robot as a function of time, with alpha = 1 and alpha = 0.05

In this figure we can see the values of the joints (meters or radians) dependency on time. For , the real trajectory is in blue, and desired trajectory is the same as before. We can see the real trajectories of the joints are different between the simulations. As is constant Joint 1 as a function of time is a constant line, because the constant joint does not get affected by the drag force. We can also notice that has a larger margin between the simulations, meaning the force affected the second joint more than others.



figure - The linear velocities of the PRR robot as function of time , with alpha = 1 and alpha = 0.05

In this figure we can see the velocity for the simulation of alpha = 0.05 in blue, and the previous simulation as the legend shows. The real velocity follows the desired path with small deviations, particularly noticeable around the peaks of the curves in the X and Y plots. With a lower , the real velocities slightly deviate from the desired ones. As alpha increases, the robot's motion becomes more controlled, and the real velocities align more closely with the desired velocities. When the velocity is negative this suggests that the robot is moving in the opposite direction to the positive axis. The velocity is going back to zero as the movement ends, which points on the robot`s stability.



figure - The angular velocities of the PRR robot as function of time , with alpha = 1 and alpha = 0.05

This figure shows the angular velocities of the two simulations in X, Y, and Z axis. In the simulation where (in blue) the real and desired do not coincide and there is a gap in the Z angular velocity.

In these graphs we can see only in Z axis we have angular velocity, as expected by the robot`s work space we sat to this simulation. The angular velocity is shown to start from zero, then getting positive until reaching a peak point and than going back to zero. This is matching what we would expect by looking at the simulation video, because the robot is moving with the clock`s negative direction.



figure - The torques of the PRR robot as function of time , with alpha = 1 and alpha = 0.05

This image presents the torques as function of time with *=1*, in yellow and in cyan*.* As we can see the torque for the first joint is constant, which is as we expect due to the constant prismatic joint in a robot with only 2 degrees of freedom. The second torque is positive and the third is negative, due to a movement with and against the positive direction of the movement accordingly. As the joints reaches their desired state, the torque required stabilizes.



figure - The torques of the PRR robot as function of time , with alpha = 1 and alpha = 0.05

This figure shows the end effector as functions of time with *=1*, and in blue in X Y and Z directions. The outcome from simulation b real trajectory differs from simulation a and the desired trajectory mostly in X and Y plots, in the area of the minimum and maximum points in the graph, which I can assume are less stable movements from the start of the movement. As expected, no change is seen in Z because it is constant.

### D -First Trajectory

For this simulation from (10, 3, 2) to (13, 1, 2) we can see the videos of the animations to understand the robot`s movement best.

For the video is saved under the name "robot\_animation\_a1"

For the video is saved under the name "robot\_animation\_b1"

### D -Second Trajectory

For this simulation from (10, 3, 2) to (4, 8, 2) we can see the videos of the animations to understand the robot`s movement best.

For the video is saved under the name "robot\_animation\_a2"

For the video is saved under the name "robot\_animation\_b2"

### E Bonus

The next image describes the App for injecting values for the simulation of the robot.

Steps:

1. Enter starting point.
2. Enter end point.
3. Enter alpha between 0 to 1.
4. Press "Simulate"



### Matlab Code

function PRR\_simulation(P\_s,P\_e,alpha) % this is the main function

% P\_s [cm] is the initial position of the tool in Cartesian coordinates and P\_e [cm]

% is the end position of the tool at the end of the movement

%% Definitions of the robotic system form

l1=10\*0.01;%[m]

l2=3\*0.01;%[m]

l3=3\*0.01;%[m]

Kp=5;% Controller gains

Kd=3;

% links size, mass,

m2=3; %[kg]

m3=5; %[kg]

h=1\*0.01; %[m]

w=1\*0.01;

tau = 1;

customPurple = [0.4, 0, 0.6]; % Deep violet

customOrange = [1, 0.5, 0];

%% find the joints angales for the desired trajectory

xs=P\_s(1)\*0.01; ys=P\_s(2)\*0.01; zs=P\_s(3)\*0.01; % starting point in [m]

xe=P\_e(1)\*0.01; ye=P\_e(2)\*0.01; ze=P\_e(3)\*0.01;% ending point in [m]

[D,xdydzd]=minimal\_jerk(xs,ys,zs,xe,ye,ze,tau,l1,l2,l3);

disp(D.a2)%for me

disp(D.a3)

function [desiredTrajectory,xdydzd]=minimal\_jerk(xs,ys,zs,xe,ye,ze,tau,l1,l2,l3)

% calculate the desierd trajectory according to minimal jerk :

t = linspace(0, tau, 100); % Time vector with 100 time steps from 0 to tau

% Preallocate space for trajectory, velocity, and acceleration

q = zeros(length(t), 3);

velocity = zeros(length(t), 3); % Joint velocities

acceleration = zeros(length(t), 3); % Joint accelerations

xdydzd = zeros(length(t), 3); % Joint locations matrix

%calculate the joints angles and velocties and accelrations for

% this desired trajectory:

for i = 1:length(t)

% Minimal jerk trajectory parameter

s = (t(i)/tau)^3 \* (-10 + 15\*(t(i)/tau) - 6\*(t(i)/tau)^2);

% Position of the end-effector

xdydzd(i, :) = [xs + s\*(xs-xe), ys + s\*(ys-ye), zs + s\*(zs-ze)];

% Compute inverse kinematics for the current position

[d1, t2, t3] = inverse\_kinematics(xdydzd(i, 1), xdydzd(i, 2), xdydzd(i, 3), l1, l2, l3);

% Store the joint positions

q(i, :) = [d1, t2, t3];

end

% Calculate derivatives to obtain velocity and acceleration

velocity(:, 1) = gradient(q(:, 1), t);

velocity(:, 2) = gradient(q(:, 2), t);

velocity(:, 3) = gradient(q(:, 3), t);

acceleration(:, 1) = gradient(velocity(:, 1), t);

acceleration(:, 2) = gradient(velocity(:, 2), t);

acceleration(:, 3) = gradient(velocity(:, 3), t);

% find the joints angales for the desired trajectory

desiredTrajectory.time = t;

desiredTrajectory.d1 = q(:, 1);

desiredTrajectory.t2 = q(:, 2);

desiredTrajectory.t3 = q(:, 3);

desiredTrajectory.v1 = velocity(:, 1);

desiredTrajectory.v2 = velocity(:, 2);

desiredTrajectory.v3 = velocity(:, 3);

desiredTrajectory.a1 = acceleration(:, 1);

desiredTrajectory.a2 = acceleration(:, 2);

desiredTrajectory.a3 = acceleration(:, 3);

end

% nestes inverse kinematics function to get joint angles

function [d1, t2, t3] = inverse\_kinematics(xe, ye, ze,l1,l2,l3)

%distance from center

r\_squared = (l1-xe)^2 + ye^2;

% check if the position is within the workspace of the robot

if sqrt(r\_squared) > l3+l2

error('The position is out of reach.');

end

%find d1

d1= ze;

%the cosine sentence for theta 3

%r^2=l2^2+l3^2-2l1\*l2\*cos(thetha 3)

cos\_theta3 = (r\_squared - (l2^2) - (l3^2)) / (2 \* l2 \* l3);

%check if theta is real

if cos\_theta3< -1 || cos\_theta3> 1

error('The position is out of reach.');

end

%extract theta3

t3 = acos(cos\_theta3);

%sin^2+cos^2=1

sin\_theta3 = sqrt(1 - cos\_theta3^2);

%find the second solution for theta 2 [cos theta3 = cos -theta3]

t2 = atan2(ye, xe-l1) - atan2(l3 \* sin\_theta3, l2 + l3 \* cos\_theta3);

end

%% plot the robot world- workspace and links

% Define the workspace limits

[d1, t2, t3] = inverse\_kinematics(xe, ye, ze, l1, l2, l3);

% figure()

% plot\_workspace(d1,t2,t3); %just plot robot 3D

%

function [x2,y2,z2,x3,y3,z3]=plot\_workspace(d1,t2,t3)

% Plot the workspace of the robot, including links and joint

legend\_labels = {};

x2 = l1+l2\*cos(t2);

y2 = l2\*sin(t2);

z2 = d1;

x3 = x2 + l3\*cos(t2 + t3);

y3 = y2 + l3\*sin(t2 + t3);

z3 = d1;

% drawing the links of the robot in its current configuration

%figure

hold on;

plot3([0 ,0], [0,0] ,[0,d1], 'k--' ,'LineWidth', 2); legend\_labels{end+1} = 'Prismatic Joint';

plot3([0, l1], [0, 0], [d1, d1], 'r-', 'LineWidth', 2); legend\_labels{end+1} = 'Link';

plot3([l1, x2], [0, y2], [d1, z2], 'r-', 'LineWidth', 2); legend\_labels{end+1} = '';

plot3([x2, x3], [y2, y3], [d1, z3], 'r-', 'LineWidth', 2); legend\_labels{end+1} = '';

hold on;

% drawing the joints of the robot in its current configuration

plot3(l1, 0, d1, 'ko', 'MarkerFaceColor', 'k');legend\_labels{end+1} = 'Revolute Joint';

plot3(x2, y2, z2, 'ko', 'MarkerFaceColor', 'k'); legend\_labels{end+1} = '';

[x\_circle, y\_circle]=xy\_circle();

plot3(x\_circle, y\_circle, d1 \* ones(size(x\_circle)), 'b--', 'LineWidth', 1.5);

legend\_labels{end+1} = 'Robot Workspace';

% Label the axes

xlabel('X [m]');

ylabel('Y [m]');

zlabel('Z [m]');

title('PRR robot configuration 3D');

grid on;

axis equal;

view(3); % 3D view

hold on

end

function [x\_circle, y\_circle]=xy\_circle()

% Define the workspace limits

% Plot the circular workspace boundary

theta = linspace(0, 2\*pi, 100); % Create a circle in XY plane

x\_circle = l1 + (l2 + l3) \* cos(theta); %l2+l3 is the max radius for the workspace

y\_circle = (l2 + l3) \* sin(theta);

end

function plot\_desired()

%this function plots the desired trajectory using the equasion

xd = xdydzd(:,1);

yd = xdydzd(:,2);

zd = xdydzd(:,3);

plot3(xd, yd, zd, 'Color', customPurple, 'LineWidth', 1.5);legend\_labels{end+1} = 'Desired Trajectory';

plot3(xd(1), yd(1),zd(1), '^', 'MarkerSize', 6, 'MarkerFaceColor',customOrange); legend\_labels{end+1} = 'Starting Point';

plot3(xd(end), yd(end),zd(end), '\*', 'MarkerSize', 8, 'MarkerEdgeColor', customOrange); legend\_labels{end+1} = 'End Point';

end

function plot\_workspace2D()

legend\_labels = {};

[x\_circle, y\_circle]=xy\_circle();

plot(x\_circle, y\_circle, 'b--', 'LineWidth', 1.5);legend\_labels{end+1} = 'Robot Workspace';

xy\_circle();hold on

plot([0, l1, x2, x3], [0, 0, y2, y3], '-r', 'LineWidth', 2);legend\_labels{end+1} = 'Link';

plot(l1, 0, 'ok', 'MarkerSize', 8, 'MarkerFaceColor', 'k');legend\_labels{end+1} = 'Revolute Joint';

plot(x2, y2, 'ok', 'MarkerSize', 8, 'MarkerFaceColor', 'k');legend\_labels{end+1} = '';

title('Robot Configuration in 2D');

xlabel('X [m]');

ylabel('Y [m]');

grid on;

axis equal;

hold on;

end

% The workspace is based on the reach of the robot in the XY plane

%% plots for the desired trajectory

t = linspace(0, tau, 100);

figure()

subplot(1,2,1)

[x2,y2,~,x3,y3,~]=plot\_workspace(d1,t2,t3);%plot the robot

hold on

plot\_desired()%add desired trajectory

legend(legend\_labels,Location='best')

hold on

subplot(1,2,2)

plot\_workspace2D(); %plot work space with the desired trajectory

plot\_desired()

legend(legend\_labels,Location='best')

% Desired trajectory 3D

hold on;

%% ode solver :

initial\_positions = [D.d1(1);D.t2(1);D.t3(1)];

initial\_velocities = [0;0;0];

initial\_torques = [0;0;0];

P0=[initial\_positions, initial\_velocities, initial\_torques]; %initial condition

t\_s =0 ; % Initialization of the solved vector (x)

t\_e=1;

%run on all time points using ode45

%options = odeset('RelTol', 1e-3, 'AbsTol', [1e-3, 1e-3, 1e-3, 1e-3, 1e-3, 1e-3, 1e-3, 1e-3, 1e-3]);

[T,X] = ode45(@(t,x) springfunc(t,x,D,alpha, Kp, Kd), [t\_s t\_e], P0);%,options

function dx = springfunc(t,x,D,alpha, Kp, Kd)

%this is the nested function that defines the state-space model.

% x(1) is position, x(2) is velocity and t is the time for the simulation solver

% in this function you should create the dynamic equations of the robotic device

%set inputs:

l2=3\*0.01;%[m]

l3=3\*0.01;%[m]

g=9.81;%[N/m]

d1 =x(1);

t2=x(2); %positions theta2 theta3

t3=x(3);

%Define the sin cos angles

s2=sin(t2);

s3=sin(t3);

c2=cos(t2);

c3=cos(t3);

s23=sin(t2+t3);

c23=cos(t2+t3);

%inertia tenzor

Izz2=m2\*(l2^2+w^2)/12;

Izz3=m3\*(l3^2+w^2)/12;

%M,V,G

M = [Izz2 + Izz3 + l2^2\*(m2/4 +m3) + m3\*l3\*(l3/4 +l2\*c3), Izz3+ m3\*((l2^2)/4 + l2\*l3\*c3/2) ;

Izz3+ m3\*((l3^2)/4 + l3\*l2\*c3/2), Izz3+(m3\*(l3^2))/4];

V = [-l2\*l3\*m3 \*s3\*x(5)\*x(6) - (m3 \*l2\*l3\*s3\*(x(6))^2)/2;

(m3\*l2\*l3\*s3\*(x(5))^2)/2];

G = [-1.5\*0.01\*g\*(m3\*s23 + (m2 + 2\*m3)\*s2);

-l3\*g\*m3\*s23/2];

% then dx is the first order equations for the solver (as you showed in q.1a in part 3)

% variables that are defined in the main function are available here as well.

% You can add other inputs for this function if needed

d1\_d = interp1(D.time, D.d1, t);

t2\_d = interp1(D.time, D.t2, t);

t3\_d = interp1(D.time, D.t3, t);

d1\_d\_dot = interp1(D.time, D.v1, t);

t2\_d\_dot = interp1(D.time, D.v2, t);%daccelaration

t3\_d\_dot = interp1(D.time, D.v3, t);

d1\_d\_ddot = interp1(D.time, D.a1, t);

t2\_d\_ddot = interp1(D.time, D.a2, t);

t3\_d\_ddot = interp1(D.time, D.a3, t);

dx = zeros(9,1);

%drag force

c = 500; %the proportional constant

Fdrag = c\*3\*0.01\*[x(5)\*(s2 + s23)+x(6)\*s23 ; -x(5)\*(c2 + c23)-x(6)\*c23 ; 0];

% jacobian

J\_L=[0 ,-3\*0.01\*(s2 + s23), -3\*0.01\*s23 ; 0, 3\*0.01\*(c2 + c23), 3\*0.01\*c23 ; 1, 0, 0];

JL\_Fdrag = J\_L'\*Fdrag;

% torques of the joints with an addition of a proportional-derivative (PD)

% controller with kp kd

q\_accelaration = M \( M \* [t2\_d\_ddot; t3\_d\_ddot] - (1-alpha)\*JL\_Fdrag(2:3) - Kp \*([t2; t3] - [t2\_d; t3\_d]) - Kd \* ([x(5); x(6)] - [t2\_d\_dot; t3\_d\_dot]));

%differential equations

dx(1) =0;% velocity =0, we have 2 DF

dx(2) = x(5);

dx(3) =x(6);

dx(4) =0;% acceleration (from dynamics) =0

dx(5)=q\_accelaration(1);

dx(6)=q\_accelaration(2);

% torques calculations:

taus = M \*[t2\_d\_ddot; t3\_d\_ddot] + V + G + alpha\*JL\_Fdrag(2:3)- Kp\*([t2; t3] - [t2\_d; t3\_d]) ...

- Kd \* ([x(4); x(5)] - [t2\_d\_dot; t3\_d\_dot]);

dx(7)=0; %doesnt exist for a constant prismatic

dx(8)=taus(1); %tau2

dx(9)=taus(2); %tau3

end

%% forward kinemtics

% nestes here you forward kinematics function

function [T\_mat]= DH2mat(DH\_table)

%this function gets the DH table (nX4) and returnes the T forward kinematix matrix

[n,~] = size(DH\_table);

%declarations

T\_mat = eye(4);%will be inserted with T values

%T\_temp = zeros(4);

%creat temp t

for i = 1:n

%take variables out of the DH

a\_i1 =DH\_table(i,1);

alpha\_i1=DH\_table(i,2);

d\_i=DH\_table(i,3);

theta\_i=DH\_table(i,4);

%for row i create matrix T from i-1 to i

%row 1

T\_temp(1,1)=cos(theta\_i);

T\_temp(1,2)=-sin(theta\_i);

T\_temp(1,4)=a\_i1;

%row 2

T\_temp(2,1)=sin(theta\_i)\*cos(alpha\_i1);

T\_temp(2,2)=cos(theta\_i)\*cos(alpha\_i1);

T\_temp(2,3)=-sin(alpha\_i1);

T\_temp(2,4)=-sin(alpha\_i1)\*d\_i;

%row 3

T\_temp(3,1)=sin(theta\_i)\*sin(alpha\_i1);

T\_temp(3,2)=cos(theta\_i)\*sin(alpha\_i1);

T\_temp(3,3)=cos(alpha\_i1);

T\_temp(3,4)=cos(alpha\_i1)\*d\_i;

%row 4 all zeros except 4,4

T\_temp(4,4)=1;

%multiply the matrices to update T\_mat

T\_mat=T\_mat\*T\_temp;

end

end

%% Plotting the Desired and Real Trajectories

% Initialize figure for plotting trajectories

% plotting in 3D

legend\_labels = {};

% Plot the Desired Trajectory

desired\_positions = zeros(3, length(D.time)); % Preallocate

for v = 1:length(D.time)

% Using DH2MAT to calculate end-effector position for each time step

DH\_table\_desired = [0, 0, D.d1(v), 0;

l1, 0, 0, D.t2(v);

l2, 0, 0, D.t3(v);

l3, 0, 0, 0];

T\_desired = DH2mat(DH\_table\_desired);

desired\_positions(:, v) = T\_desired(1:3, 4); % Extract end-effector position

end

figure;

plot\_workspace(D.d1(end),D.t2(end),D.t3(end));

plot3(desired\_positions(1, :), desired\_positions(2, :), desired\_positions(3, :), 'Color',customPurple, 'LineWidth', 2);

legend\_labels{end+1} = 'Desired Trajectory';

% Plot the Real Trajectory

real\_positions = zeros(3, length(T)); % Preallocate

for b = 1:length(T)

% Using DH2MAT to calculate end-effector position for each time step

DH\_table\_real = [0, 0, X(b, 1), 0; %real = X ( notation)

l1, 0, 0, X(b, 2);

l2, 0, 0, X(b, 3);

l3, 0, 0, 0];

T\_real = DH2mat(DH\_table\_real);

real\_positions(:, b) = T\_real(1:3, 4); % Extract end-effector position

end

plot3(real\_positions(1, :), real\_positions(2, :), real\_positions(3, :), 'g-', 'LineWidth', 2);legend\_labels{end+1} = 'Real Trajectory';

plot3(real\_positions(1, 1), real\_positions(2, 1), real\_positions(3, 1), '^', 'MarkerSize', 6, 'MarkerFaceColor',customOrange); legend\_labels{end+1} = 'Starting Point';

plot3(real\_positions(1,end), real\_positions(2,end), real\_positions(3,end), '\*', 'MarkerSize', 6, 'MarkerEdgeColor', customOrange); legend\_labels{end+1} = 'End Point';

grid on; xlabel('X [m]'); ylabel('Y [m]'); zlabel('Z [m]');

title('3D Desired and Real Trajectories of the PRR Robot');

hold on;

% Add legend

legend(legend\_labels,Location='best');

view(3); % Set view to 3D

hold off;

% plotting in 2D

figure() %plot2D

plot\_workspace2D(); %plot work space

plot(desired\_positions(1, :), desired\_positions(2, :), 'Color',customPurple, 'LineWidth', 2);legend\_labels{end+1} = 'Desired Trajectory';

plot(real\_positions(1, :), real\_positions(2, :), 'g-', 'LineWidth', 2);legend\_labels{end+1} = 'Real Trajectory';

plot(real\_positions(1, 1), real\_positions(2, 1), '^', 'MarkerSize', 6, 'MarkerFaceColor',customOrange); legend\_labels{end+1} = 'Starting Point';

plot(real\_positions(1,end), real\_positions(2,end), '\*', 'MarkerSize', 6, 'MarkerEdgeColor', customOrange); legend\_labels{end+1} = 'End Point';

grid on; xlabel('X [m]'); ylabel('Y [m]');

title('2D Desired and Real Trajectories of the PRR Robot');

hold on;

legend(legend\_labels,Location='best')

%% plotting angles, velocities and torques as function of time

% Define hue values excluding red (0 and 1 are red, so we use other values)

hues = [0.1, 0.2, 0.3, 0.5, 0.6, 0.7, 0.9]; % 0.1 = yellow, 0.3 = green, 0.5 = cyan, 0.7 = blue, 0.9 = purple

% Create a palette using full saturation and brightness

palette = hsv2rgb([hues' ones(size(hues')) ones(size(hues'))]);

randColor = palette(randi(length(palette)), :);

%% Plot joint parameters as a function of time

figure(4);

hold on;

legend\_labels = {};%allocate place for legend

subplot(3, 1, 1);

plot(T, X(:, 1),'color',randColor,'LineWidth',2);

legend\_labels{end+1} = 'Real';

hold on

plot(D.time,D.d1,'r--','LineWidth',2)

legend\_labels{end+1} = 'Desired';

xlabel('Time (s)');

ylabel('d1 (m)');

title('Joint 1 P');

legend(legend\_labels,Location="best");

subplot(3, 1, 2);

plot(T, X(:, 2),'color',randColor,'LineWidth',2);

hold on

plot(D.time,D.t2,'r--','LineWidth',2)

xlabel('Time (s)');

ylabel('theta2 (rad)');

title('Joint 2 R');

legend(legend\_labels,Location="best");

subplot(3, 1, 3);

plot(T, X(:, 3),'color',randColor,'LineWidth',2);

hold on

plot(D.time,D.t3,'r--','LineWidth',2);

xlabel('Time (s)');

ylabel('theta3 (rad)');

title('Joint 3 R');

sgtitle('Joints as function of time')

legend(legend\_labels,Location='best');

%% Plot joint linear and angular velocities as a function of time

linear\_v\_desired = zeros(length(D.time),3);%allocate place for desired velocities trajectory

angular\_v\_desired = zeros(length(D.time),3);%allocate place for desired velocities trajectory

linear\_v\_real = zeros(length(D.time),3);%allocate place for real velocities trajectory

angular\_v\_real = zeros(length(D.time),3);%allocate place for real velocities trajectory

J\_A = [ %save the linear part of the Jacobian

0, 0, 0;

0, 0, 0;

0, 1, 1

];

%find d1 t2 t3 for each move in the real & desired trajectory

for o = 1:length(D.time) %desired

s2t = sin(D.t2(o));c2t = cos(D.t2(o)); s23t = sin(D.t2(o) + D.t3(o));c23t = cos(D.t2(o) + D.t3(o)); %desired trajectory angles

J\_L\_d=[ %desired Jacobian linear

0 ,-3\*0.01\*(s2t + s23t), -3\*0.01\*s23t ;

0, 3\*0.01\*(c2t + c23t), 3\*0.01\*c23t ;

1, 0, 0];

J\_desired = [J\_L\_d;J\_A];

derivatives\_desired\_point = [0; D.v2(o); D.v3(o)]; % running on o

v\_w\_desired = J\_desired\*derivatives\_desired\_point; % computation of the linear and angular velocities

linear\_v\_desired(o,:) = v\_w\_desired(1:3)'; %save to vector

angular\_v\_desired(o,:) = v\_w\_desired(4:6)';

end

for o = 1:length(T) %real

s2t = sin(X(o,2));c2t = cos(X(o,2)); s23t = sin(X(o,2) + X(o,3));c23t = cos(X(o,2) + X(o,3)); %real trajectory angles

J\_L\_r=[ %real Jacobian linear

0 ,-3\*0.01\*(s2t + s23t), -3\*0.01\*s23t ;

0, 3\*0.01\*(c2t + c23t), 3\*0.01\*c23t ;

1, 0, 0];

J\_real = [J\_L\_r;J\_A];

derivatives\_real\_point = [0; X(o,5); X(o,6)]; %theta 2 dot, theta 3 dot

v\_w\_real= J\_real\*derivatives\_real\_point; % get the linear and angular velocities

linear\_v\_real(o,:) = v\_w\_real(1:3)'; %save to vector

angular\_v\_real(o,:) = v\_w\_real(4:6)';

end

%Plotting linear velocities with desired and real

figure(5);

hold on;

subplot(3, 1, 1); %X linear

plot(T,linear\_v\_real(:,1),'color',randColor,'LineWidth',2);%linear real

hold on

plot(D.time,linear\_v\_desired(:,1),'r--','LineWidth',2);%linear desired

xlabel('Time (s)');

ylabel('v\_X (m/s)');

grid on;

title('v\_X Linear');

legend(legend\_labels,Location='best');

subplot(3, 1, 2);%Y linear

plot(T,linear\_v\_real(:,2),'color',randColor,'LineWidth',2);%linear real

hold on

plot(D.time,linear\_v\_desired(:,2),'r--','LineWidth',2);%linear desired

xlabel('Time (s)');

ylabel('v\_Y (m/s)');

grid on;

title('v\_Y Linear');

legend(legend\_labels,Location='best');

subplot(3, 1, 3);%Z linear

plot(T,linear\_v\_real(:,3),'color',randColor,'LineWidth',2);%linear real

hold on

plot(D.time,linear\_v\_desired(:,3),'r--','LineWidth',2);%linear desired

xlabel('Time (s)');

ylabel('v\_Z (m/s)');

grid on;

title('v\_Z Linear');

legend(legend\_labels,Location='best');

sgtitle('Linear velocity as function of time')

%Plotting angular velocities with desired and real

figure(6)

hold on;

subplot(3, 1, 1); %X Angular

plot(T,angular\_v\_real(:,1),'color',randColor,'LineWidth',2);%angular real

hold on

plot(D.time,angular\_v\_desired(:,1),'r--','LineWidth',2);%angular desired

xlabel('Time (s)');

ylabel('\omega\_X (m/s)');

grid on;

title('\omega\_X Angular');

legend(legend\_labels,Location='best');

subplot(3, 1, 2); %Y Angular

plot(T,angular\_v\_real(:,2),'color',randColor,'LineWidth',2);%angular real

hold on

plot(D.time,angular\_v\_desired(:,2),'r--','LineWidth',2);%angular desired

xlabel('Time (s)');

ylabel('\omega\_Y (m/s)');

grid on;

title('\omega\_Y Angular');

legend(legend\_labels,Location='best');

subplot(3, 1, 3); %Z Angular

plot(T,angular\_v\_real(:,3),'color',randColor,'LineWidth',2);%angular real

hold on

plot(D.time,angular\_v\_desired(:,3),'r--','LineWidth',2);%angular desired

xlabel('Time (s)');

ylabel('\omega\_Z (m/s)');

grid on;

title('\omega\_Z Angular');

legend(legend\_labels,Location='best');

sgtitle('Angular velocity as function of time')

%% Plot joint torques as a function of time

figure(7);

hold on

subplot(3, 1, 1);

plot(T, X(:, 7),'color',randColor,'LineWidth',2,'LineStyle','--');

xlabel('Time (sec)');

ylabel('\tau\_1 (N\*m)');

title('Torque \tau\_1');

subplot(3, 1, 2);

plot(T, X(:, 8),'color',randColor,'LineWidth',2,'LineStyle','--');

xlabel('Time (sec)');

ylabel('\tau\_2 (N\*m)');

title('Torque \tau\_2');

subplot(3, 1, 3);

plot(T, X(:, 9),'color',randColor,'LineWidth',2,'LineStyle','--');

xlabel('Time (sec)');

ylabel('\tau\_3 (N\*m)');

title('Torque \tau\_3');

sgtitle('Torques as function of time')

%% Plot end effector as a function of time

figure(8);

hold on

subplot(3, 1, 1);

plot(T, real\_positions(1,:)','color',randColor,'LineWidth',2);

hold on

plot(D.time,desired\_positions(1,:)','r--','LineWidth',2);

xlabel('Time (sec)');

ylabel('X (m)');

title('End effactor in X (m)');

legend(legend\_labels,Location='best');

subplot(3, 1, 2);

plot(T, real\_positions(2,:)','color',randColor,'LineWidth',2);

hold on

plot(D.time,desired\_positions(2,:)','r--','LineWidth',2);

xlabel('Time (s)');

ylabel('Y (m)');

title('End effactor in Y (m)');

legend(legend\_labels,Location='best');

subplot(3, 1, 3);

plot(T, real\_positions(3,:)','color',randColor,'LineWidth',2);

hold on

plot(D.time,desired\_positions(3,:)','r--','LineWidth',2);

xlabel('Time (s)');

ylabel('Z (m)');

title('End effactor in Z (m)');

legend(legend\_labels,Location='best');

sgtitle('End effactors as function of time')

%% Simulation video

% Define video writer object

videoFilename = 'robot\_animation.avi'; % Name of the output video file

videoObj = VideoWriter(videoFilename); % Create a VideoWriter object

videoObj.FrameRate = 3; % Set the frame rate (adjust to control the speed)

open(videoObj); % Open the video file for writing

% Calculate the total number of frames needed for a 2-second video

total\_frames = 30; % 30 fps

% Calculate the indices for downsampling your data

indices = round(linspace(1, length(D.time), total\_frames));

% make X same size as D.time

X\_mod = interp1(linspace(T(1), T(end), size(X, 1)), X, D.time);

%simulation

% Plot the workspace grid

d1=D.d1(1);

[xc,yc]=xy\_circle();

figure;

hold on;

%legend\_labels={};

plot3(xc, yc, ones(1, 100) \* d1, 'b--', 'LineWidth', 1);

plot\_desired();

legend\_labels = {'Desired Trajectory', 'Real Trajectory', 'Workspace Boundary'};

legend(legend\_labels, 'Location', 'best'); % Set legend initially

for j = 1:length(indices)

% Extract the joint angles at the current time step from X - real

idx = indices(j);

d1 = X\_mod(idx,1);

t2 = X\_mod(idx,2);

t3 = X\_mod(idx,3);

DH\_table\_real = [0, 0, d1, 0; %real = X ( notation)

l1, 0, 0, t2;

l2, 0, 0, t3;

l3, 0, 0, 0];

T\_real = DH2mat(DH\_table\_real);

real\_positions(:, j) = T\_real(1:3, 4); % Extract end-effector position

% Clear the previous plot

cla;

plot3(xc, yc, ones(1, 100) \* d1, 'b--', 'LineWidth', 1); % Replot static elements

plot\_desired(); % Replot desired trajectory

plot3(real\_positions(1, 1:j), real\_positions(2, 1:j), real\_positions(3, 1:j), 'g-', 'LineWidth', 2);

% Update the real trajectory plot data

plot\_workspace(d1, t2, t3);

grid on; xlabel('X [m]'); ylabel('Y [m]'); zlabel('Z [m]');

% Delete the existing legend

legend\_handle = findobj(gcf, 'Type', 'Legend');

if ~isempty(legend\_handle)

delete(legend\_handle);

end

% Set legend after plotting all elements

legend({'Workspace Boundary','Desired Trajectory','Start Point','Desired End Point','Real Trajectory', 'Prismatic Joint','Link','','', 'Revolute Joint',''}, 'Location', 'northeastoutside');

% plot\_desired()

title(sprintf('Robot Configuration at t = %.2f s', D.time(idx)));

drawnow;

axis equal;

% Capture the current figure as a frame

frame = getframe(gcf); % Get current frame of the figure

writeVideo(videoObj, frame); % Write the frame to the video file

pause(0.001); % Pause to visualize the movement (adjust time as needed)

end

% Close the video file

close(videoObj); % Close the video file after all frames have been written

end