Cooperative Robotics

Authors:	Email:
Dikshant Thakur	s5943225@studenti.unige.it
Ouassim Milous	s5938924@studenti.unige.it
Girum Molla Desalegn	s6020433@studenti.unige.it

Date: 02/02/2025

1 Exercise 1: Implement a Complete Mission

Implement several actions to reach a point, land, and then perform the manipulation of a target object.
Initialize the vehicle at the position:

$$\begin{bmatrix} 8.5 & 38.5 & -36 & 0 & -0.06 & 0.5 \end{bmatrix}^\top$$

Use a "safe waypoint navigation action" to reach the following position:

$$\begin{bmatrix} 10.5 & 37.5 & -38 & 0 & -0.06 & 0.5 \end{bmatrix}^\top$$

Then land, aligning to the nodule. In particular, the x axis of the vehicle should align to the projection, on the inertial horizontal plane, of the unit vector joining the vehicle frame to the nodule frame. Once landed, implement a "fixed-based manipulation action" to reach the target nodule (mimicking the scanning of the nodule). During this manipulation phase, the vehicle should not move for any reason.

1.1 Q1: Report the unified hierarchy of tasks used and their priorities in each action.

Actions list:

- $\mathcal{A}_1$ - Safe Waypoint Navigation Action
- $\mathcal{A}_2$ - Nodule Alignment Action
- $\mathcal{A}_3$ - Safe Landing Action
- $\mathcal{A}_4$ - Fixed Base Manipulatin Action

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$
Horizontal Alignment	[I, S, P]	1	1	1	1
Minimum Altitude	[I, S, P]	2	2		
Vehicle Control	[E, AD]	3			2
Approach Nodule	[E, AD]			2	
Landing	[E, AD]			3	
Tool	[E, AD]				3

Table 1: A number in the action column indicates that the task is activated for that particular action ($\mathcal{A}$).The higher the number, the lower the priority for that action. The "type" column indicates whether the objective is an equality (E) , inequality (I) , action define (AD) safety(S) or prerequisite (P).

Tasks' Defintion

- **Horizontal Alignment**—This task is responsible for maintaining the horizontal position of the vehicle parallel to the sea floor.
- **Minimum Altitude**—This task maintains the mininum altitude between the vehicle and the sea floor.
- **Vehicle Control** - It controls the vehicle to achieve its desired location.
- **Approach Nodule** - This task guides the vehicle to approach the nodule's position.
- **Landing** - This task is responsible for landing the UVMS system on the sea floor.
- **Tool** - This task moves the manipulator arm to the goal position.

1.2 Q2: Comment the behaviour of the robots, supported by relevant plots.

Based on the actions mentioned above, we define our phases and present our observations. In the action of Safe Waypoint Navigation, the vehicle starts from its initial position and moves linearly to its desired position while remaining horizontally stable. The manipulator is not yet active.
After this, while keeping the vehicle aligned with the nodule's x-axis and the sea floor, the entire system moves vertically downward. Once the vehicle lands, the manipulator arm reaches the nodule, which is its desired position.

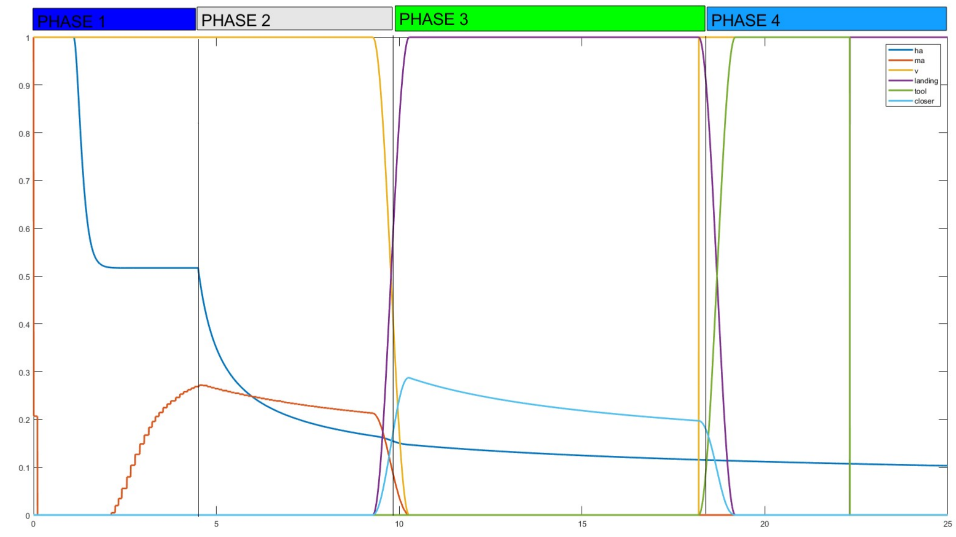

Figure 1: Activation Function

where Y Value is the ouput value of actiovation function, which is in between 0 and 1.

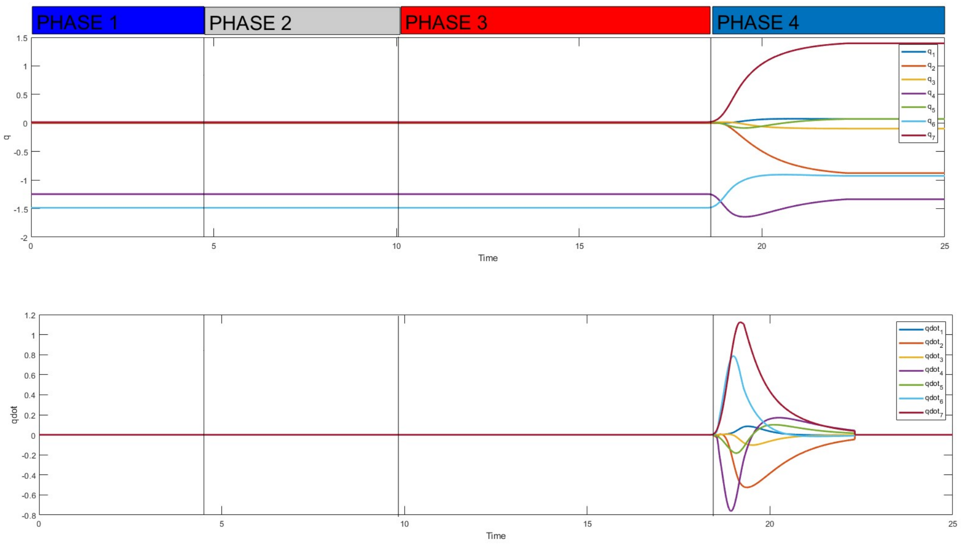

Figure 2: Joint position and Joint velocity

The above two graph represents the joint posiition and its velocity over the time.

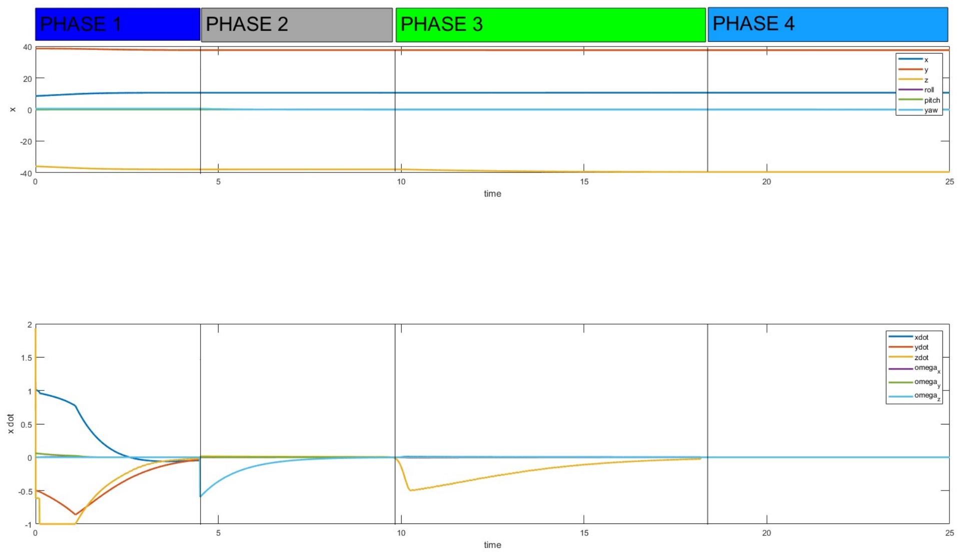

Figure 3: End effector posiition and its velocity

1.3 Q3: What is the Jacobian relationship for the Alignment to Target control task? How was the task reference computed?

To align the vehicle with the nodule, we calculated the yaw angle difference between the vehicle and the nodule. The calculation steps are as follows:
First, we calculated the distance between the current position and the nodule with respect to the world frame.

$$^{w}v_{target} = {^{w}v_{goal}} - {^{w}v_{current}}$$

With this target vector, we find the yaw angle between the current position of the vehicle and the position of the nodule with respect to the world frame.

$$\psi_{target} = tan^{-1}\left[\frac{^{w}v_{target}(y_{component})}{^{w}v_{target}(x_{component})}\right]$$

Once we have the ψ_{target}, we then constrain the yaw angular velocity of the vehicle as follows:

$$\dot{x}_{yaw} = \psi_{target} - \psi_{current}$$

$$\dot{x} = \begin{bmatrix} \mathbf{0}_{5\times 1} \\ \dot{x}_{yaw} \end{bmatrix}$$

The Jacobian we are using is the same one we used in the vehicle position control task. Since we constrained our velocities only for the yaw angular velocity of the vehicle, the only solution we will get is for the yaw angular vehicle velocity.

$$^{w}J_{v} = \begin{bmatrix} \mathbf{0}_{6\times 7} & \mathbf{I}_{6\times 6} \end{bmatrix}$$

1.4 Q4: Try changing the gain of the alignment task. Try three different values: 0.001, 0.1, 1.0. What is the observed behaviour? Implement a solution that is gain-independent guaranteeing that the landing is accomplished aligned to the target.

We observed that with gain 1.0, the time taken by vehicle to reduce the angular error between its body and nodule around z-axis with respect to world frame is faster than if we assign the gain 0.1 and 0.001. To be precise the time take in the first case, when gain is 1, is 10.5 seconds. In the second and third case, which are 0.1 and 0.001 respectively, it takes 52 seconds and in the latter case it moves

infinitinemaly slow.
To get the gain-independent solution we can use the time derivative of point concept. First we need to define the position vector

$$^{a}\Delta\mathbf{r}_{P,Q} = {}^{a}P - {}^{a}Q$$

where P is the desired position and Q is the intial position. $< a >$ is the projection frame.
Then check $^{a}\Delta\mathbf{r}_{P,Q}$ is zero or not. If it is not zero, then find the unit vector ($\widehat{^{a}\Delta\mathbf{r}}_{P,Q}$) of the distance vector ($^{a}\Delta r_{P,Q}$)

$$\widehat{^{a}\Delta\mathbf{r}}_{P,Q} = \frac{^{a}\Delta\mathbf{r}_{P,Q}}{\|^{a}\Delta\mathbf{r}_{P,Q}\|}$$

Now take derivative of the distance vector with respect to frame $< a >$ and calculate the velocity along the unit vector.

$$\frac{d_a}{dt}(^{a}\Delta\mathbf{r}_{P,Q})$$

$$^{a}\dot{P} = \widehat{^{a}\Delta\mathbf{r}}_{P,Q} * v$$

where v is the velocity give to the point.
Update the position of ^{a}P with respect time by using this equation,

$$^{a}P(t + dt) = {}^{a}P(t) + {}^{a}\dot{P}dt + Q(dt)$$

where $^{a}P(t + dt)$ is the updated position of P point and $^{a}P(t)$ is the position of ^{a}P at time t.
Iterate this equation till,

$$^{a}P = {}^{a}Q)$$

. We can implement this solution on both of the case, where ^{a}P is the point at vehicle frame and ^{a}Q is the point where we need to go and align.

1.5 Q5: After the landing is accomplished, what happens if you try to move the end-effector? Is the distance to the nodule sufficient to reach it with the end-effector? Comment the observed behaviour.

After landing task, the manipulator easily achieved its task without vehicle moving.

1.6 Q6: Implement a solution that guarantees that, once landed, the nodule is certainly within the manipulator's workspace.

To ensure that the nodule is always within the workspace of the manipulator arm, we need to control the vehicle in such a way that it moves closer to the nodule while landing. To do this, we need to control the linear x and linear y velocities so that the arm is always in the projection of the nodule. Here are the steps we followed: First we need to find the linear distance between the current position of vehicle and the nodule.

$$e_{linear} = {}^{w}T_g - {}^{w}T_t$$

where e_{linear} is the linear error, which is 3x1 matrix, $< w >$ is world frame, and g and t represent the goal and the tool of the arm, respectively.

$$\dot{x} = \begin{bmatrix} e_{linear_x} \\ e_{linear_y} \\ 0 \end{bmatrix}$$

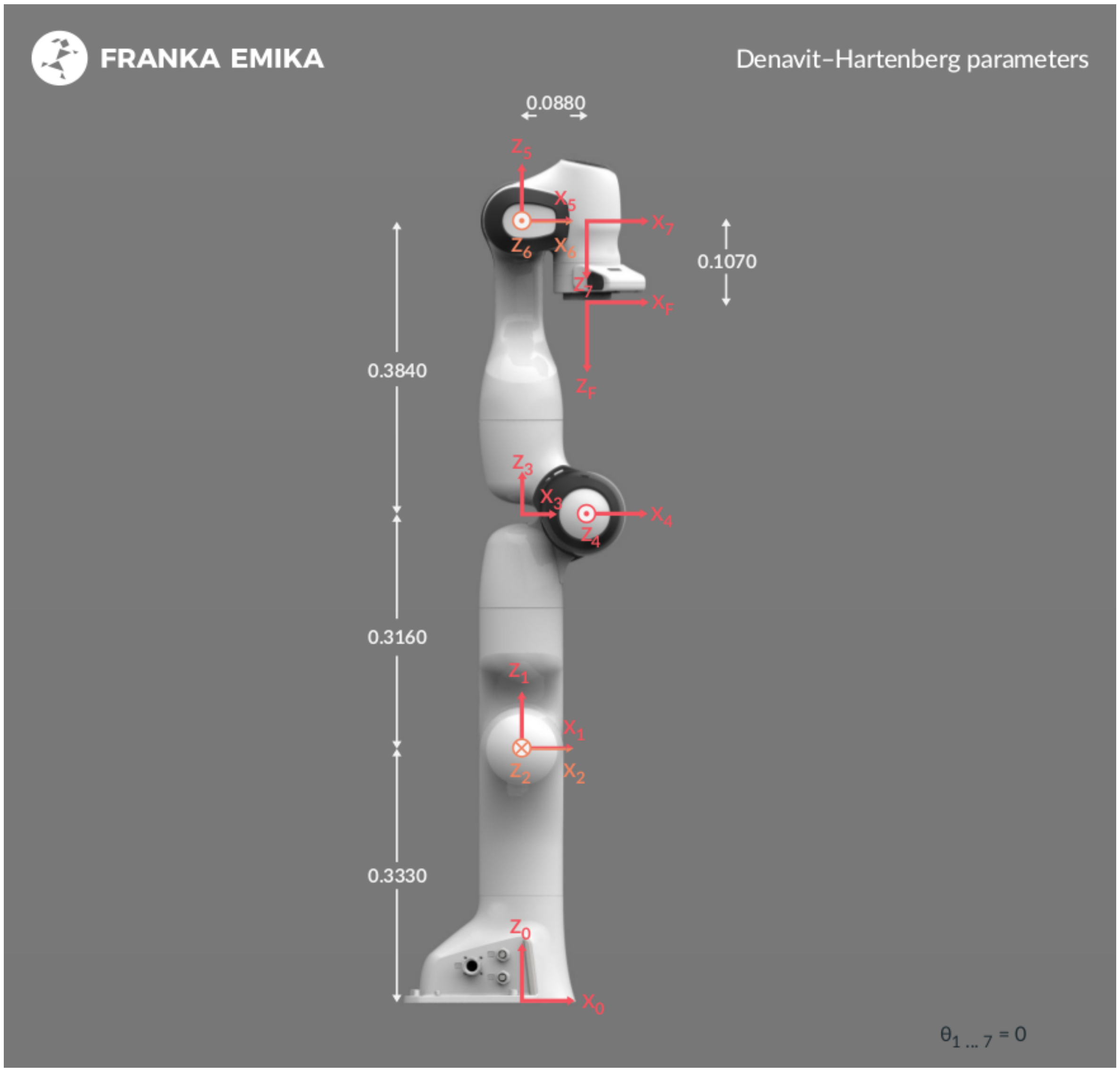

Figure 4: Robot Specifications

2 Exercise 2: Bimanual manipulation

In this exercise it is required to perform a bimanual manipulation by implementing the task priority algorithm considering two manipulators as a single robot. The manipulator adopted for this assignment is the Franka Panda by Emika (see Figure 4)). A simulation in python is provided, in order to visualize the two robots and test your Matlab implementation.

1. The transformations between the world and the base of each robot must be computed knowing that:
   (a) The left arm base coincides with the world frame.
   (b) The right arm base is rotated of π w.r.t. z-axis and is positioned 1.06 m along the x-axis and -0.01 m along the y-axis.
2. The transformation from each base to the respective end-effector is given in the code and is retrieved from the URDF model thanks to the *Robotic System Toolbox* of Matlab.
3. Then, define the tool frame for both manipulators; the tool frames must be placed at 21.04 cm along the z-axis of the end-effector frame and rotated with an angle of -44.9949 deg around the z-axis of the end-effector.

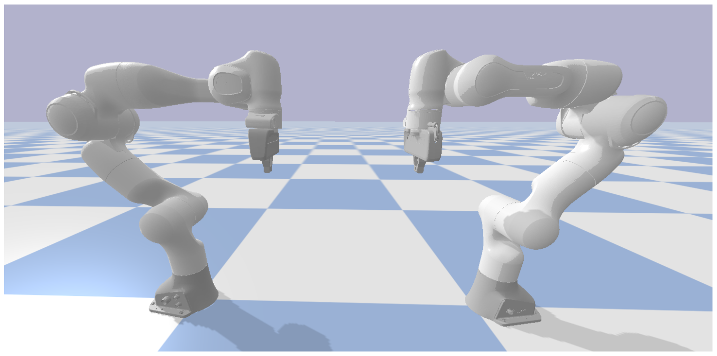
Figure 5: Initial position of the robots

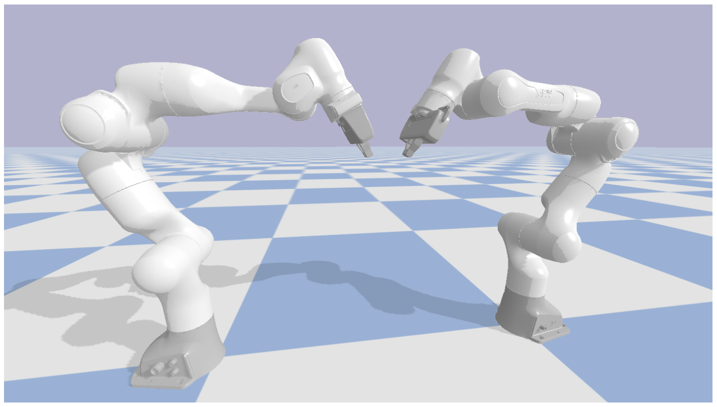
Figure 6: Relative orientation at the grasping position

4. Implement a "move to" action that includes the joint limits task.

5. The first phase foresees to move the tool frame of both manipulators to the grasping points, by implementing a "move to" action and its corresponding tasks. Define the goal frames for each manipulator such that their origin corresponds to the grasping points. **HINT**: the position of the grasping points can be computed by knowing the origin of the object frame ${}^{w}\mathrm{O}_o$ and the object length l_{obj}. The goal orientation of the tool frames is obtained by rotating the tool frames 30 deg around their y-axis; the manipulators should reach the configuration depicted in Figure 6.

$$ {}^{w}\mathrm{O}_o = [0.5, 0, 0.59]; \tag{1} $$

$$ l_{obj} = 10\,(\mathrm{cm}). \tag{2} $$

6. During this phase of the mission, the following safety tasks must be implemented: *end-effector minimum altitude* and *joint limits*. The joint limits specifications can be found in the datasheet of the robot. The minimum altitude from the table should be 15 cm.

Once the manipulators reach the grasping points, the second phase of the mission should start. Now, implement the Bimanual Rigid Grasping Constraint task to carry the object as a rigid body.

1. Define the object frame as a rigid body attached to the tool frame of each manipulator. **HINT:** Compute this quantity after reaching the grasping point.

2. Define the rigid grasp task.

3. Then, move the object to another position while both manipulators hold it firmly, e.g. ${}^{w}\mathrm{O}_g = [0.65, -0.35, 0.28]^\top (m)$

4. Note that the transition for the *Bimanual Rigid Constraint* should be a binary one, i.e., without smoothness. This is the nature of the constraint, i.e., either it exists or not. Modify the *Action-Transition* script seen during the class to take into account the different nature of this task (a constraint one).

Once the object has been moved to the required position you have to implement a final phase of the mission in which the joint velocities are set to zero, and every action is deactivated except for the minimum altitude task.

Repeat the simulation, using a goal position or by changing the grasping in such a way that one of the two manipulators activates multiple safety tasks, to stress the constraint task.

2.1 Q1: Report the unified hierarchy of tasks used and their priorities in each action.

Actions list:

- $\mathcal{A}_1$ - Move-to.
- $\mathcal{A}_2$ - Co-operative move-to
- $\mathcal{A}_3$ - Halt manipulator

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
Joint limits	[I,S]	1	1	1
Minimum Altitude	[I,S	2	2	2
Bimanumial rigid grasping constraint	[E]		3	
Move-To	[E,AD]	3	4	

Table 2: A number in the action column indicates that the task is activated for that particular action ($\mathcal{A}$).The higher the number, the lower the priority for that action. The "type" column indicates whether the objective is an equality (E) , inequality (I) , action define (AD) safety(S) or prerequisite (P).

Tasks' Definition

- **Joint limits:** This is safety task to constraint the each joints of the manipulator , which is responsible to maintain the joint angle in its range.
- **Minimum Altitude:** This is the safety task where the tool of the manipulator keeps the minimum distance from the ground.
- **Bimanual rigid grasping constrain:** In this task we constraint our robot to act together.
- **Move To:** this task the manipulator move to the desired position

2.2 Q2: What is the Jacobian relationship for the Joint Limits task? How was the task reference computed?

The Jacobian for the limits task is the identity matrix. For this task reference, we calculate the velocities in this way, where we constrain the joints to return within their limits. In the formula below, $Jl_{\min_i}$ and $Jl_{\max_i}$ represents the lower and upper joint limits of all joints i_{th} and q_i is the current of position of i_{th} joint.

$$\dot{q}_i = \left(\frac{Jl_{\min_i} + Jl_{\max_i}}{2} \right) - q_i$$

$$^{b}J_{limit} = \mathbf{I}_{14x14}$$

2.3 Q3: What is the Jacobian relationship for the Bimanual Rigid Grasping Constraint task? How was the task reference computed?

We took a reference point, denoted as O (which is the center point of the object in our case), and calculated the Jacobian for both of the arm tool ,left arm tool and right arm tool, with respect to the $< O >$.

$$J_{\text{combined}} = [J_{l,o} \; - \; J_{r,o}]$$

Here, $J_{l,o}$ refers to the Jacobian of the left arm tool with respect to $< O >$ and $J_{r,o}$ refers to the Jacobian of the right arm tool with respect to $< O >$. The negative sign accounts for the opposite direction of motion of the right arm relative to the left.
In the task reference, we aim to make the velocities of both arms equal, meaning that the difference in the end effector velocities should be zero in the context of the Bimanual Rigid Grasping Constraint task. This ensures the synchronization of both arms.

$$J_{o,a}\dot{q}_a = J_{o,b}\dot{q}_b$$

$$[J_{o,a} - J_{o,b}]\dot{y} = 0$$

The task refrence should be zero in our case.

2.4 Q4: Comment the behaviour of the robots, using relevant plots. In particular, show the difference (if any) between the desired object velocity, and the velocities of the two end-effectors in the two cases.

During the initial phase, the left and right manipulators move independently, each following its own trajectory without coordination. This continues until the completion of Phase One. After transitioning to Phase Two, both manipulators begin working cooperatively, utilizing their shared Jacobian matrix to synchronize their movements. This collaborative approach ensures they efficiently reach the desired position. In phase 3, both right and left manipulator half their movement.

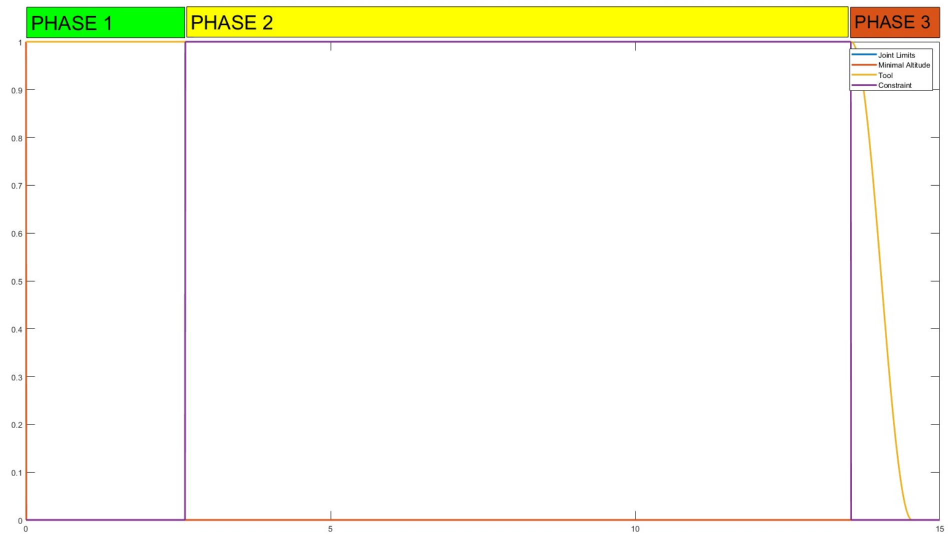

Figure 7: Activation Function for right arm

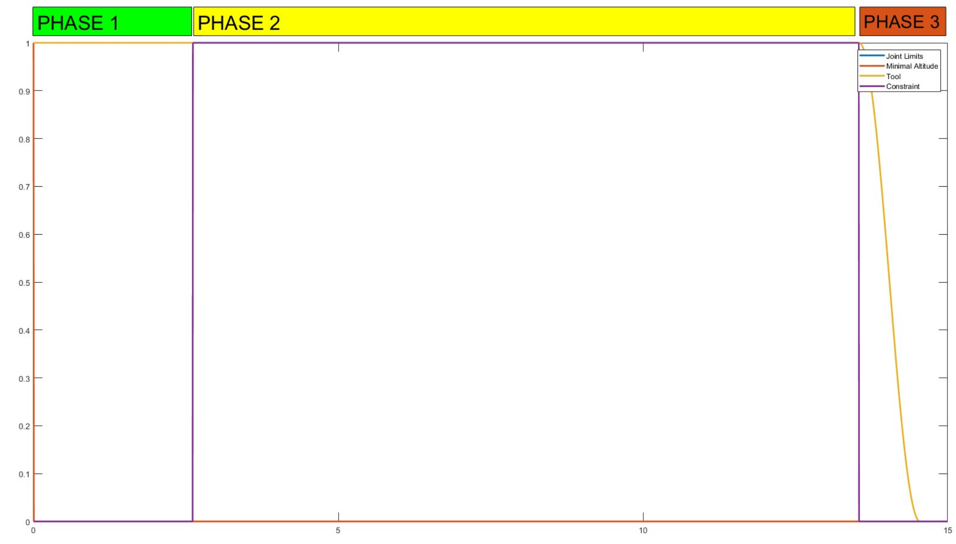

Figure 8: Activation Function for left arm

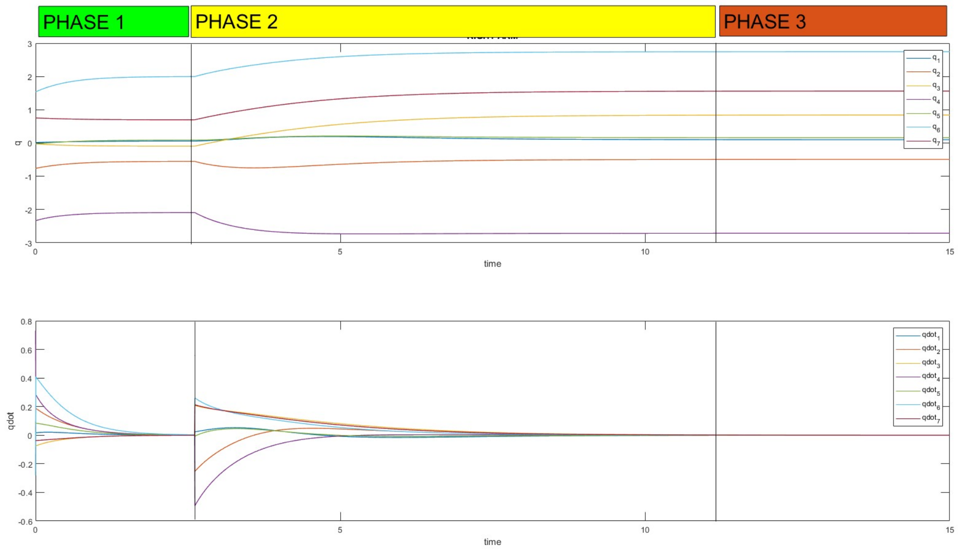

Figure 9: Joint velocity for right arm

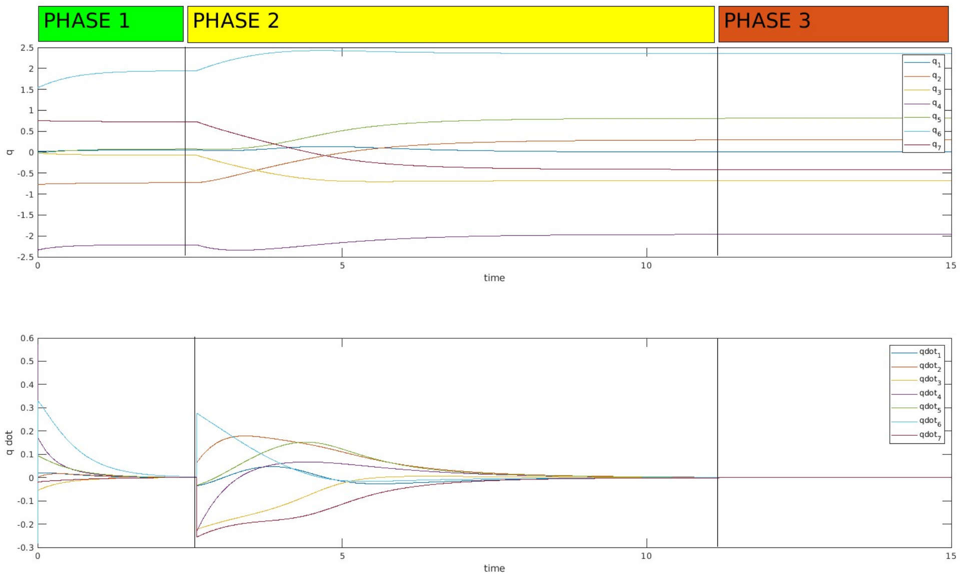

Figure 10: Joint velocity for left arm

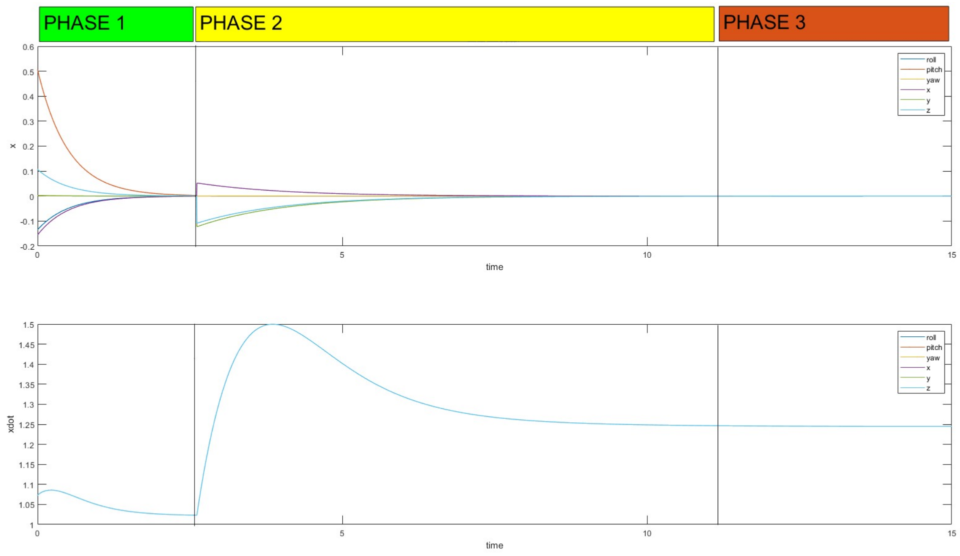

Figure 11: End effector velocity for right arm

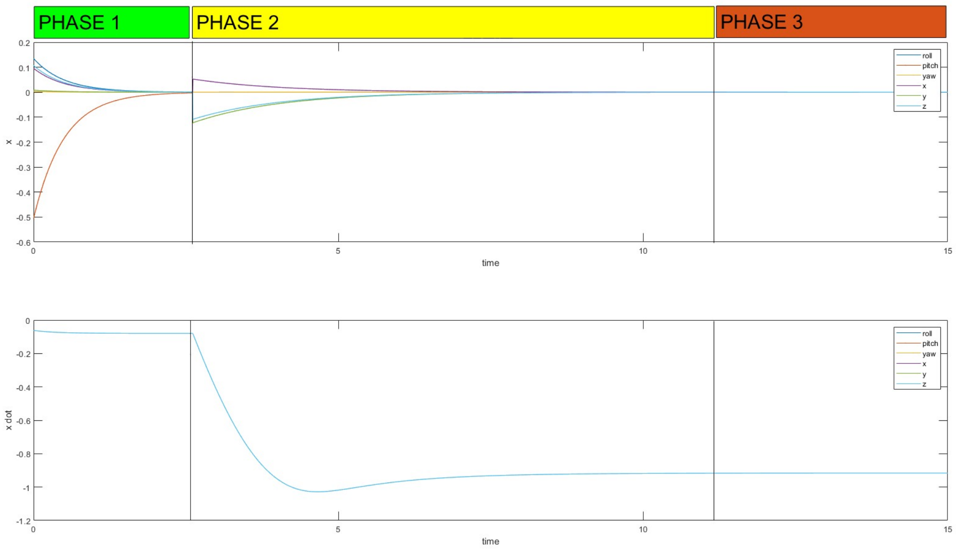

Figure 12: End effector velocity for left arm

3 Exercise 3: Cooperative manipulation

In this exercise, it is required to perform cooperative manipulation by implementing the task priority algorithm considering the two Franka Panda manipulators as two distinct robots.

1. The first phase foresees to move the tool frames to the grasping points, by implementing the "move to" action for both manipulators. **Please note: each manipulator has his own Task Priority Inverse Kinematic Algorithm**. Define the goal frames for each manipulator such that their origin corresponds to the grasping points. **HINT**: the position of the grasping points can be computed by knowing the origin of the object frame ${}^{w}\mathrm{O}_o$ and the object length l_{obj}. The goal orientation of the tool frames is obtained by rotating the tool frames 20 deg around their y-axis.

$$ {}^{w}\mathrm{O}_o = [0.5, 0, 0.59]^\top (m); \tag{3}$$

$$ l_{obj} = 6\,(\mathrm{cm}). \tag{4}$$

   During this phase of the mission, the following safety tasks must be implemented: *end-effector minimum altitude* and *joint limits*. The joint limits specifications can be found in the datasheet of the robot. The minimum altitude from the table should be 15 cm.

2. Once the manipulators reach the grasping points the second phase of the mission should start. Implement the *Cooperative Rigid Constraint* task to carry the object as a rigid body.

   (a) Define the object frame as a rigid body attached to the tool frame of each manipulator.

   (b) Define the rigid grasp task.

   (c) You have to move the object to another position while both manipulators hold it firmly. The desired object goal position is

$$ {}^{w}\mathrm{O}_g = [0.60, 0.40, 0.48]^\top (m) \tag{5}$$

   (d) Compute the *non-cooperative* object frame velocities. **HINT:** Suppose manipulators communicate ideally and can exchange the respective end-effector velocities

   (e) Apply the coordination policy to the *non-cooperative* object frame velocities.

   (f) Compute the *cooperative* object frame velocities.

   Note that the transition for the *Cooperative Rigid Constraint* should be a binary one, i.e., without smoothness. This is the nature of the constraint, i.e., either it exists or not. Modify the *ActionTransition* script seen during the class to take into account the different nature of this task (a constraint one).

3. Once the object has been moved to the required position you have to implement a final phase of the mission in which the joint velocities are set to zero, and every action is deactivated except for the minimum altitude task.

Again, test it twice, once with the provided (reachable) goal, and then with a goal or with a different grasp configuration that triggers the activation of multiple joint limits or a joint limit and minimum altitude by one manipulator. The idea is that this second position should trigger the cooperation policy (i.e., the cooperative velocity should be different than the original desired object velocity).

3.1 Q1: Report the unified hierarchy of tasks used and their priorities in each action, and report clearly the actions used in the two phases of the cooperation algorithm.

We used the hierarchy given below for both arms, maintaining the same order.
Actions list:

- $\mathcal{A}_1$ - Move-to : This action is responsible for moving the robot without cooperation, meaning the robot will only follow joint limits and maintain a minimum altitude
- $\mathcal{A}_2$ - Co-operative move-to : In this action, we constrained both robots to move together while respecting each other's individual constraints
- $\mathcal{A}_3$ - Halt manipulator : In this action, we deactivate all tasks for both robots so that they reach the rest position and remain there

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
Joint limits	[I,S]	1	1	1
Minimum Altitude	[I,S]	2	2	2
Move-To	[AD, E]	3		
Cooperative Rigid Constraint move	[AD,E]		3	

Table 3: A number in a given cell represents the priority of the control task (row) in the hierarchy of the control action (column). The type column indicates whether the objective is an equality (E) or inequality (I) one.

Tasks' definition

- **Joint Limits :** This is a safety task in which we keep the joints within their range.
- **Minimum Altitude:** This is also a safety task in which we have restricted the robot to a specific distance. This means the robot's end effector will not go below that distance with respect to its base.
- **Cooperative Rigid Constraint Move:** In this task, we apply a coordination policy in which we provide velocities to both robots in such a way that they communicate with each other and move together. These velocities are calculated considering all the constraints.

3.2 Q2: Comment on the behavior of the robots, using relevant plots. In particular, make sure there are differences between the desired object velocity and the non-cooperative Cartesian velocities at least in one simulation. Show also how the cooperative velocities of the two end-effectors behave.

In this exercise, we observed that the two independent left and right manipulators first move to the initial waypoint, then they grasp an object and cooperate together to calculate the proper velocities to move to a second goal successfully, without one arm straying away from the other.

In order to achieve this, the two robots exchange several variables: their current velocities, desired velocities, and the admissible space H. We calculate H using the following function:

```
function [H] = ComputeH(pandaArm)
    [U,D,V] = svd(pandaArm.wJt);
    J_pinv = V * pinv(D) * U';
    H = pandaArm.wJt * J_pinv;
end
```

Afterward, the two robots calculate a new velocity in a distributed manner, which will be equal. This creates a constraint that ensures they move together toward the same goal. The new velocity is calculated using this function:

```
function [combinedx] = ComputeCooperativeXdot(desiredx1,desiredx2,x1,x2,h1,h2)
    m1_0 = 1;
    m2_0 = 1;
    m1 = m1_0 + norm([desiredx1 - x1]);
    m2 = m2_0 + norm([desiredx2 - x2]);
    xw = (1/(m1 + m2)) * (m1 * x1 + m2 * x2);
    C = [h1 h2];
    [U,D,V] = svd(C);
    C_pinv = V * pinv(D) * U';
    combinedx = [h1, zeros(6); zeros(6), h2] * [xw; xw] + ...
                [h1, zeros(6); zeros(6), h2] * C_pinv * C * [desiredx1; desiredx2];
end
```

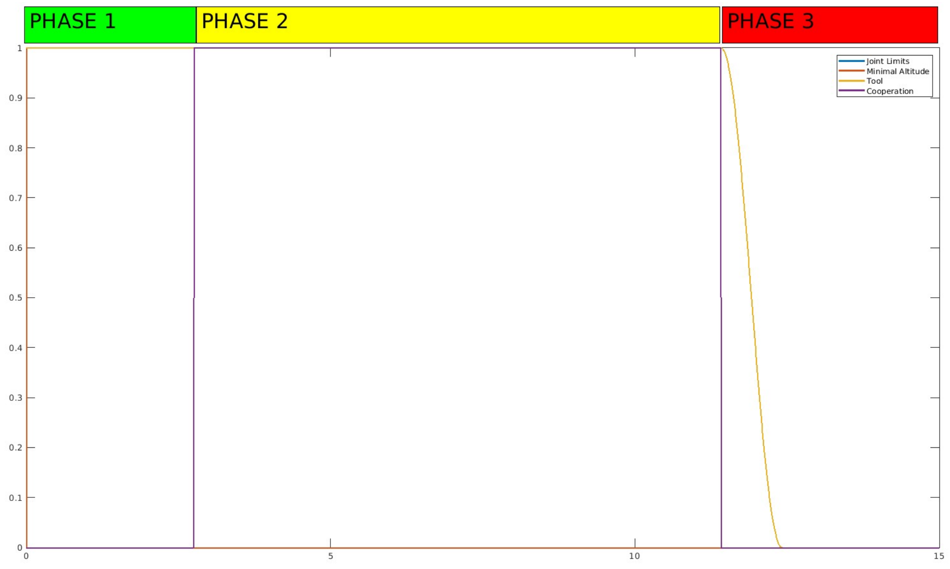

Figure 13: Activation Function for the right arm.
The figure above shows the activation function for the right arm. We notice a behavior similar to that observed in the second exercise. However, in this case, we are using a cooperative task instead of a move-to and constraint task.

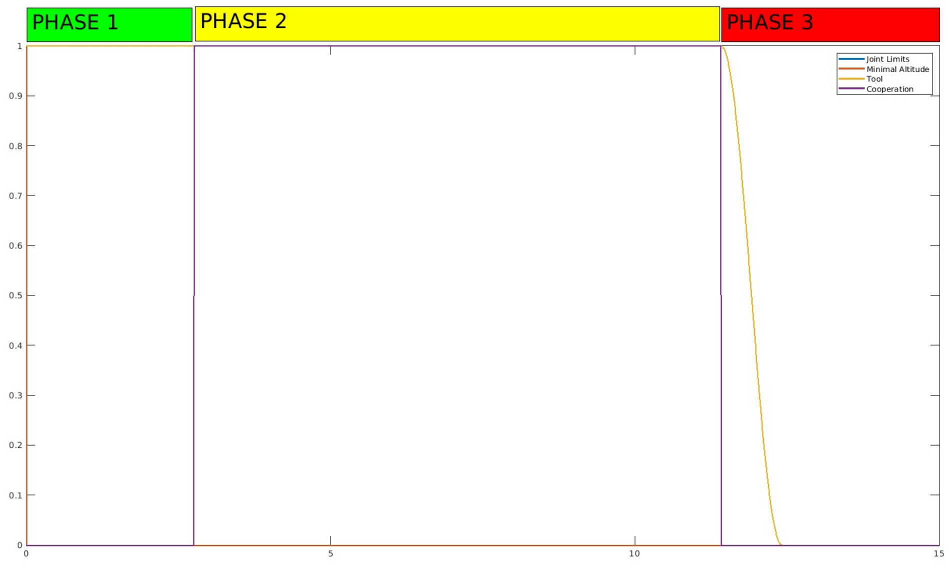

Figure 14: Activation Function for the left arm.
The figure above shows the activation function for the left arm. We observe the same trend as in the right arm, where the cooperation task ensures the movements are coordinated between both arms.

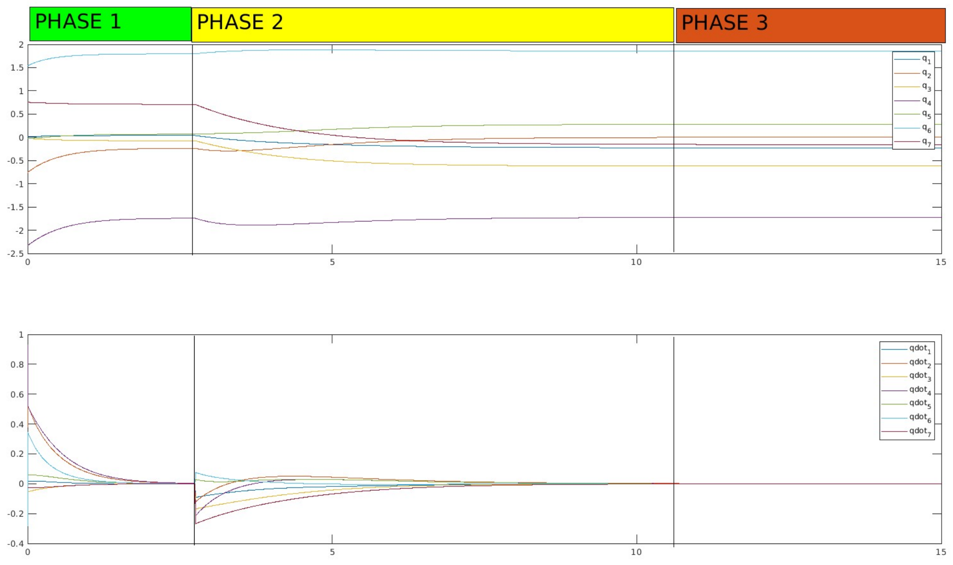

Figure 15: Joint velocity for the right arm.
The plot above illustrates the joint velocities of the right arm. In the beginning of each phase, we assign a new task reference, resulting in high error, which the robot minimizes exponentially over time. In the final phase, the robot halts movement.

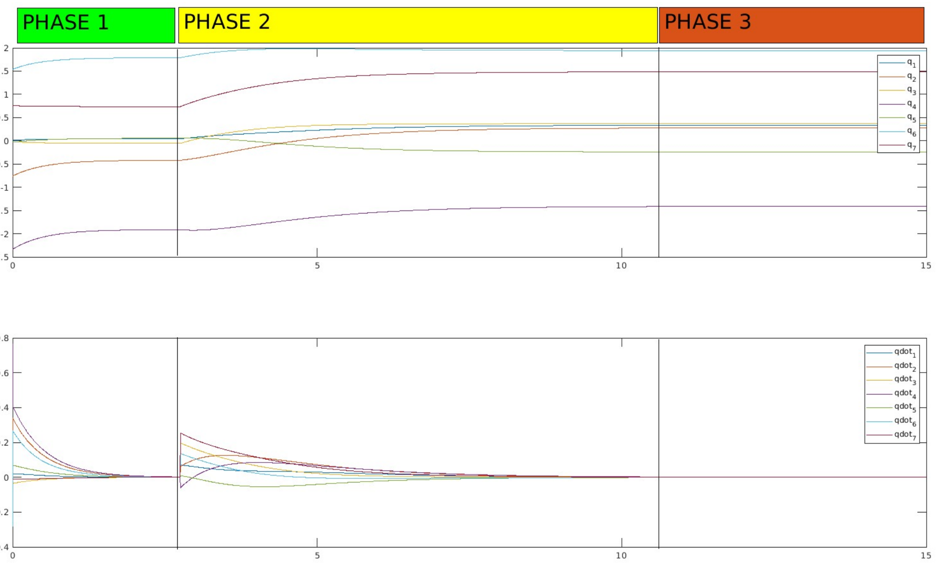

Figure 16: Joint velocity for the left arm.
This plot shows the joint velocities for the left arm. Similar to the right arm, we observe a high error at the beginning of each phase, followed by exponential minimization of the error. In the last phase, the movement stops.

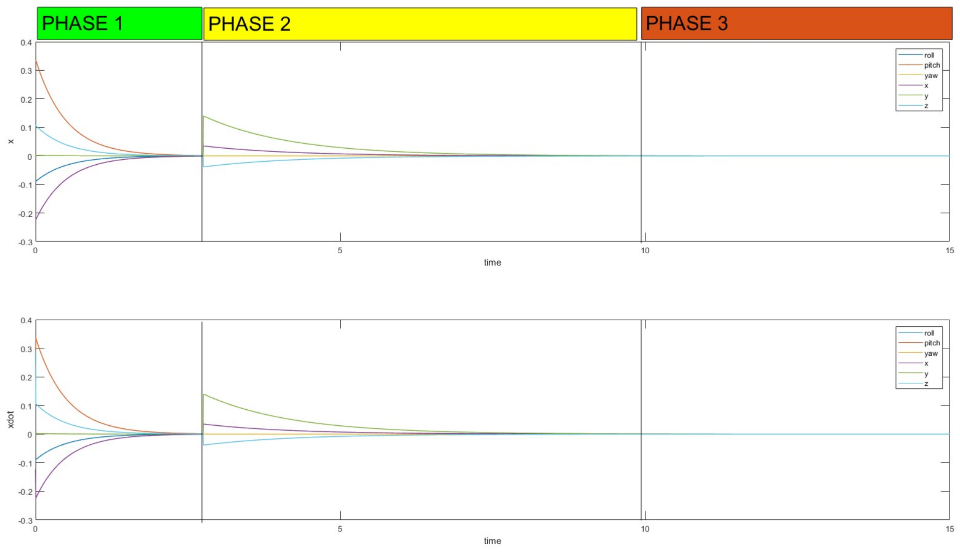

Figure 17: End effector velocity for the right arm.
This figure shows the non-cooperative end-effector velocity for the right arm. The plot compares the desired velocities and the actual velocities over time.

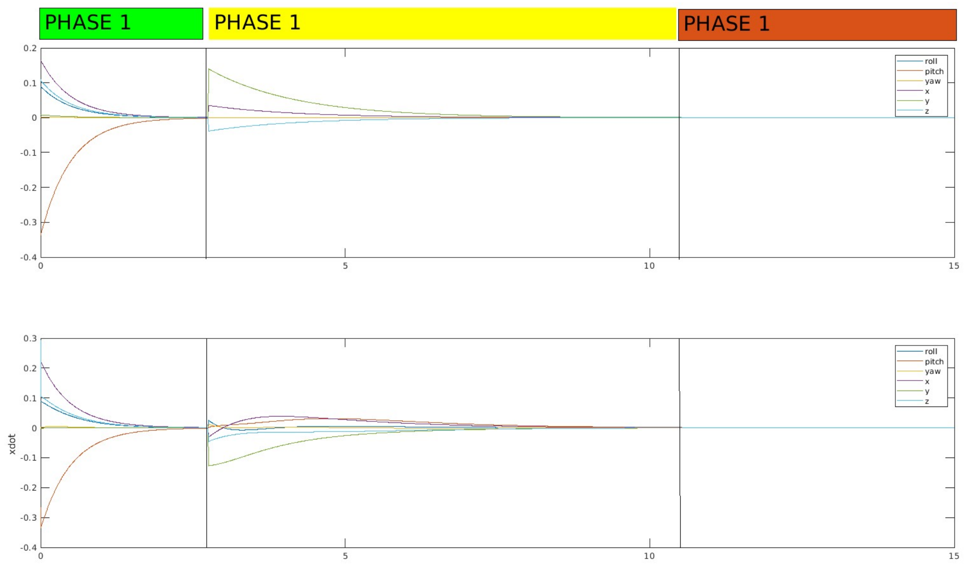

Figure 18: End effector velocity for the left arm.

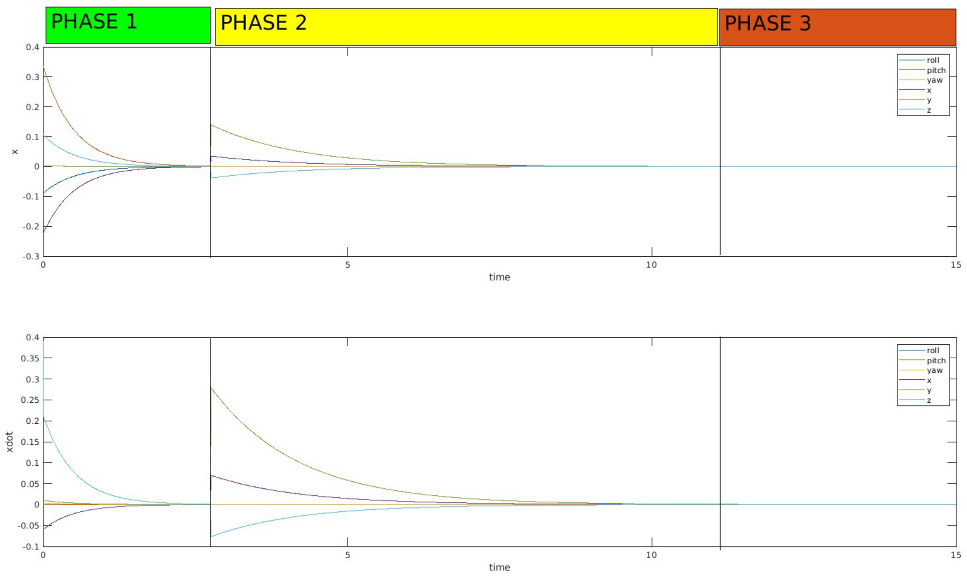

Figure 19: Desired object velocity and the cooperative Cartesian velocity for the right arm.
In this figure, we compare the desired object velocity with the cooperative Cartesian velocity for the right arm. We observe that in the first phase, the results are quite similar. However, in the second phase, when cooperation is used, we notice a higher velocity on the y-axis, indicating how the cooperative task constrains the movement.

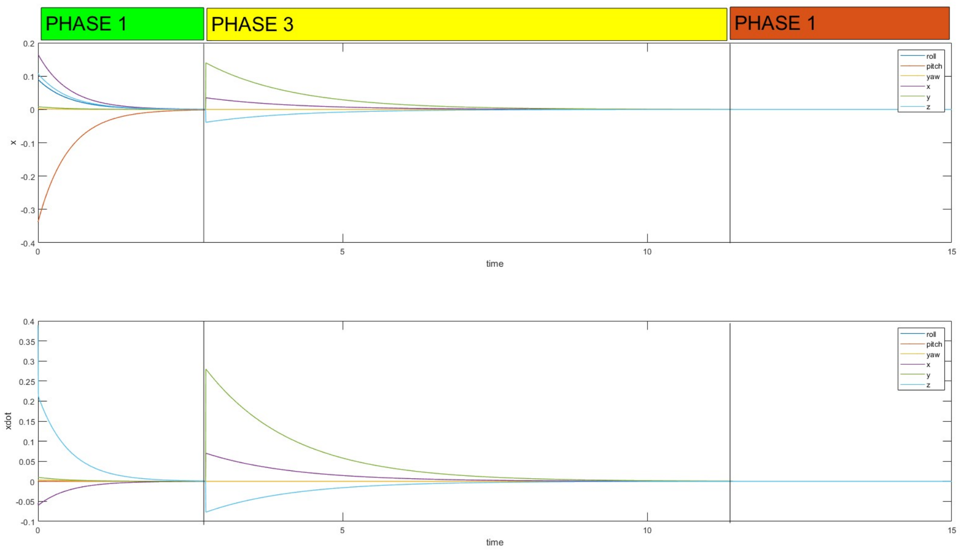

Figure 20: Desired object velocity and the cooperative Cartesian velocity for the left arm.
This figure shows the desired object velocity and the cooperative Cartesian velocity for the left arm. As with the right arm, we see a similar behavior in the second phase, where cooperation results in higher velocities, particularly on the y-axis, illustrating the constraint introduced by the cooperative task.