Cooperative Robotics

Author: Natnael Berhanu Takele
Email: natnael.berhanu.takele@gmail.com
Date: September 1, 2024

General notes

- Exercise 1 is done with the ROBUST matlab main and unity visualization tools. Exercises 2-3 are done with the Franka simulation tools.
- Comment and discuss the simulations, in a concise scientific manner. Further comments, other than the questions, can be added, although there is no need to write 10 pages for each exercise.
- Aid the discussion with screenshots of the simulated environment (compress them to maintain a small overall file size), and graphs of the relevant variables (i.e. activation functions of inequality tasks, task variables, and so on). Graphs should always report the units of measure in both axes, and legends whenever relevant.
- Report the thresholds whenever relevant.
- Report the mathematical formula employed to derive the task jacobians and the control laws when asked, including where they are projected.
- If needed, part of the code can be inserted as a discussion reference.

Use the following template when you need to discuss the hierarchy of tasks of a given action or set of actions:

Table 1: Example of actions/hierarchy table: 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.

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
Task A	I	1		1
Task B	I	2	1	
Task C	E		2	2

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.

The UVMS is a robotic system comprising a 6-degree-of-freedom (6 DOFs) vehicle and a 7-degree-of-freedom (7 DOFs) manipulator. This integrated system excels in underwater environments, combining mobility and dexterity for precise navigation and task execution.

The comprehensive structure of the system can be succinctly represented through a configuration system.

$$c = \begin{bmatrix} q \\ \eta \end{bmatrix} \tag{1}$$

where

$$q = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix} \tag{2}$$

the arm joint vector $\mathbf{q}$ is an element of the set of real numbers $\mathbb{R}^l$ and

$$\eta = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \in \mathbb{R}^6 \tag{3}$$

is the position and orientation of the vehicle. In particular:

$$\eta_1 = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \eta_2 = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{4}$$

where η_1 represents the position of the vehicle frame with respect to the world frame, while η_2 denotes the orientation of the vehicle frame expressed in its roll-pitch-yaw (RPY) representation. This RPY representation facilitates the definition of the rotation matrix ${}^w_v R$.

The vehicle is tasked with reaching a predefined position in the world frame w. Addressing this, we consider two equality control tasks: one for linear position and another for angular attitude. This separation allows independent activation if needed.

Additionally, there are other objectives, such as maintaining the vehicle's horizontal attitude to align with the sea floor. Given the energy-intensive nature of this task, we treat it as an inequality task, activating only when the misalignment surpasses a specified threshold.

This exercise focuses on the implementation of a Task Priority Control system for an Underwater Vehicle Manipulator System (UVMS). The code, developed in MATLAB, is simulated using a Unity-based simulation platform. The experimental framework includes testing on the Robust UVMS variant. Within the Robust UVMS context, we have incorporated three key actions to achieve a sequence of

tasks, namely reaching a specified point, landing, and subsequently manipulating a target object. The implemented actions comprise:

1. **Safe Waypoint Navigation Action**: Ensures secure and efficient navigation through designated waypoints while considering the unique challenges of the underwater environment.

2. **Landing Action**: Facilitates a controlled descent to the desired landing position, maintaining stability and precision.

3. **Fixed-Based Manipulation Action**: Enables the manipulation of a target object once the underwater manipulator has successfully landed, demonstrating a reliable and stable fixed-base operation.

The simulation results provide a comprehensive evaluation of the UVMS's performance, particularly in the Robust configuration, validating the effectiveness of the Task Priority Control approach in achieving complex underwater tasks.

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

The table is read from left to right, with Safe Waypoint Navigation as the first action, Landing as the second action, and Grasping as the third action.

Table 2: The Hierarchy of Tasks

Task	Type	Safe Waypoint Navigation	Landing	Grasping
Zero Velocity Constraint	E, C	-	-	1
Minimum Attitude	I, S	1	-	-
Horizontal Attitude	I, S	2	1	-
Vehicle Alignment to the Target	I, P	-	2	2
Grasping Task	E, AD	-	-	3
Zero Altitude Control	E, AD	-	3	-
Vehicle Position Control	E, AD	3	-	-
Vehicle Orientation Control	E, AD	4	-	-

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

I used "Activation Functions" to demonstrate the robots' behavior as shown in Figure 1. It shows the activation functions related to various control tasks for a robotic system. Here's a summary of the robot's behavior:

1. **Zero Velocity Constraint**: This activation function ensures that the vehicle maintains zero velocity at certain phases, likely during grasping. Its activation remains constant at 1, indicating that this constraint is always enforced.

2. **Minimum Altitude**: This function dictates a minimum altitude constraint, which gradually decreases once a certain threshold is reached, suggesting that the vehicle descends as needed while maintaining a minimum safe altitude.

3. **Horizontal Attitude**: This activation manages the horizontal orientation of the vehicle, ensuring it remains level. It drops to zero sharply after a certain point, indicating that the horizontal attitude control is relaxed once the alignment and other conditions are met.

4. **Vehicle Alignment to the Target**: This function manages the vehicle's alignment with the target. It begins to decrease when the alignment error falls within a specific range (bell-shaped region), showing that precise alignment is crucial until the error is minimized to an acceptable level.

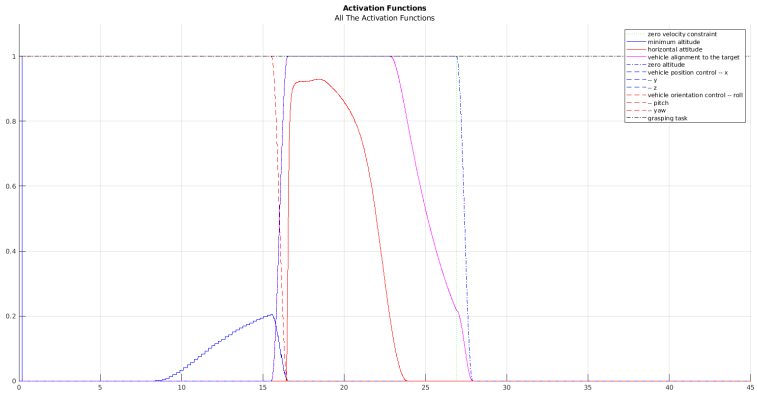

Figure 1: Activation functions related to various control tasks for a UVMS robotic system.

5. **Zero Altitude and Position Control**: These functions control the vehicle's position along the x, y, and z axes. The functions show different phases of activation, with steep inclines and declines indicating when precise positioning is critical, especially during the descent and approach to the target.

6. **Vehicle Orientation Control**: These functions manage the vehicle's orientation. They remain active around the peak regions, indicating precise control during critical phases like alignment and landing.

7. **Grasping Task**: The grasping task becomes active towards the end, suggesting that once all positional and alignment conditions are met, the robot proceeds to execute the grasp.

The overall behavior shows a coordinated sequence of tasks where the robot aligns, descends, maintains safe altitudes, and finally grasps the target, all managed by the respective activation functions that ensure safe and precise execution of each phase.

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

Define O_v and O_t as the origins of the vehicle frame $\langle v \rangle$ and the target frame $\langle t \rangle$, respectively. Let i_v represent the unit vector along the x-axis of the vehicle. The positive angle θ is formed between ${}^w i_v$ and the projected distance vector ${}^w d = {}^w O_t - {}^w O_v$ on the inertial horizontal plane. Refer to Figure 2 for a visual representation of this task.

To achieve alignment between the vehicle and the target, we define the misalignment vector ${}^w \rho = \theta \, {}^w n_{at}$, where ${}^w n_{at}$ is a vector along the axis of rotation, given by ${}^w n_{at} = \frac{1}{\sin\theta} {}^w i_v \times (\frac{{}^w d}{\|w_d\|})$.
In order to align the vehicle, we define our task with the objective of minimizing ρ using its derivative with respect to θ. The partial derivative is:

$$\frac{\partial^w \rho}{\partial \theta} = {}^w n_{at} \dot{\theta} \tag{5}$$

Let's express the previous expression in terms of $\dot{y}$. Initially, for $\dot{\theta}$,

$$\dot{\theta} = {}^w n_{at} \omega_{t/x} \tag{6}$$

$$\dot{\theta} = {}^w n_{at} (\omega_{t/w} - \omega_{x/w}) \tag{7}$$

${}^w i_v$ is a unit vector along the x-axis of the vehicle frame $\langle v \rangle$, thus we can express $\omega_{v/w} = \omega_{x/w}$. Next, for $\omega_{t/w}$:

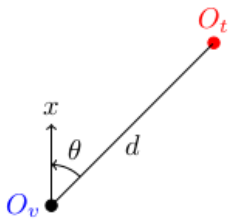

Figure 2: Target Alignment on the Inertial Horizontal Plane

$$\omega_{t/w} = \frac{1}{\|^w d\|^2} \left({}^w d \wedge v_{d/w} \right) \tag{8}$$

where $v_{d/w} = v_{t/w} - v_{v/w}$ by definition. We assume the target is immobile, thus $v_{t/w} = 0$. Finally, the Jacobian relationship is as follows:

$$\dot{\theta} = ({}^w n_{at})^T \begin{bmatrix} 0_{3\times 7} & -\frac{1}{\|^w d\|^2} [{}^w d\wedge] & -I_{3\times 3} \end{bmatrix} \dot{y} = {}^w J_{at} \dot{y} \tag{9}$$

We determine the misalignment vector ρ through the application of the Reduced Versor Lemma. The task reference is calculated with respect to ${}^w\rho_{at}$, aiming for a scenario where, upon alignment, $||{}^w\rho_{at}||$ approaches zero.

$${}^w\dot{\tilde{x}}_{at} = \lambda \left(0 - \|{}^w\rho_{at}\|\right), \quad \lambda \in \mathbb{R}^+ \tag{10}$$

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.

The speed at which the vehicle aligns with the target decreases as the gain decreases. Nevertheless, irrespective of the gain, the vehicle adjusts its position to align with the target, albeit more slowly with smaller gains. Notably, with an extremely small gain (0.001), the vehicle appears to struggle to orient itself effectively.

To achieve behavior independent of the gain, we can introduce a bias term to the task reference.

$${}^w\dot{\tilde{x}}_{at} = \lambda \left(0 - \|{}^w\rho_{at}\|\right) - \mu, \quad \lambda, \mu \in \mathbb{R}^+ \tag{11}$$

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.

If we introduce another action, which involves moving the manipulator after the landing phase, we observe that the entire robot follows the manipulator's movement. To prevent this behavior, it is necessary to add a task that constrains the vehicle's motion during the grasping phase. This would ensure that the action is executed more safely and with greater precision.

As demonstrated in the simulation, the distance between the robot's landing point and the location of the rock is generally sufficient for the end-effector to reach the target position. However, this may not always be the case in real-world scenarios, where the rock's position is unknown. In such cases, an additional task is required to control the distance between the robot and the target once the robot's sensors detect the desired object.

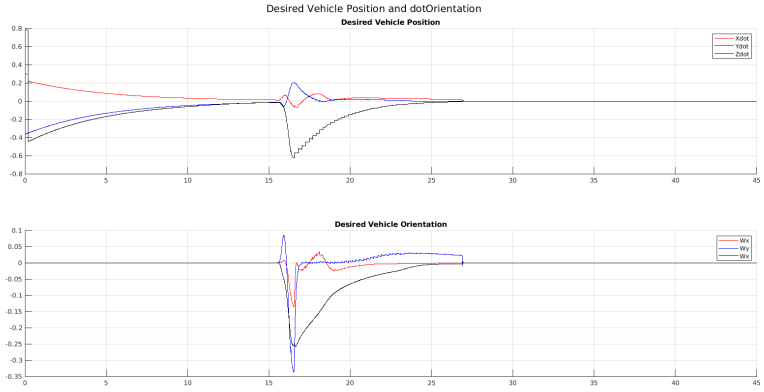

Figure 3: Motion of the vehicle performing the mission

The critical behavior to correct is the robot's motion following the landing phase, as illustrated in the given figure. If the rock is outside the manipulator's workspace after landing, it means that, in our case, it indicates an incorrect landing position assignment. In real scenarios, since the rock's position remains unknown until detected by the robot's sensors, it is essential to implement a control mechanism that adjusts the distance between the robot's current position and the identified target.

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

In the event that the target lies outside the manipulator's workspace after landing, the algorithm lacks a task addressing the distance to the target. To rectify this within the Landing action, we propose the addition of a new inequality task known as "Distance to the Target." This task leverages the distance vector ${}^{w}d$ to achieve a specified target distance. Where ${}^{w}d$ is a projected distance between the position of the nodule and the position of the vehicle.

Table 3: The Hierarchy of Tasks for Improved Landing

Task	Type	Safe Waypoint Navigation	Improved Landing	Grasping
Zero Velocity Constraint	E, C	-	-	1
Minimum Attitude	I, S	1	-	-
Horizontal Attitude	I, S	2	1	-
Vehicle Alignment to the Target	I, P	-	2	2
Distance to the Target	I, P	-	3	-
Grasping Task	E, AD	-	-	3
Zero Altitude Control	E, AD	-	4	-
Vehicle Position Control	E, AD	3	-	-
Vehicle Orientation Control	E, AD	4	-	-

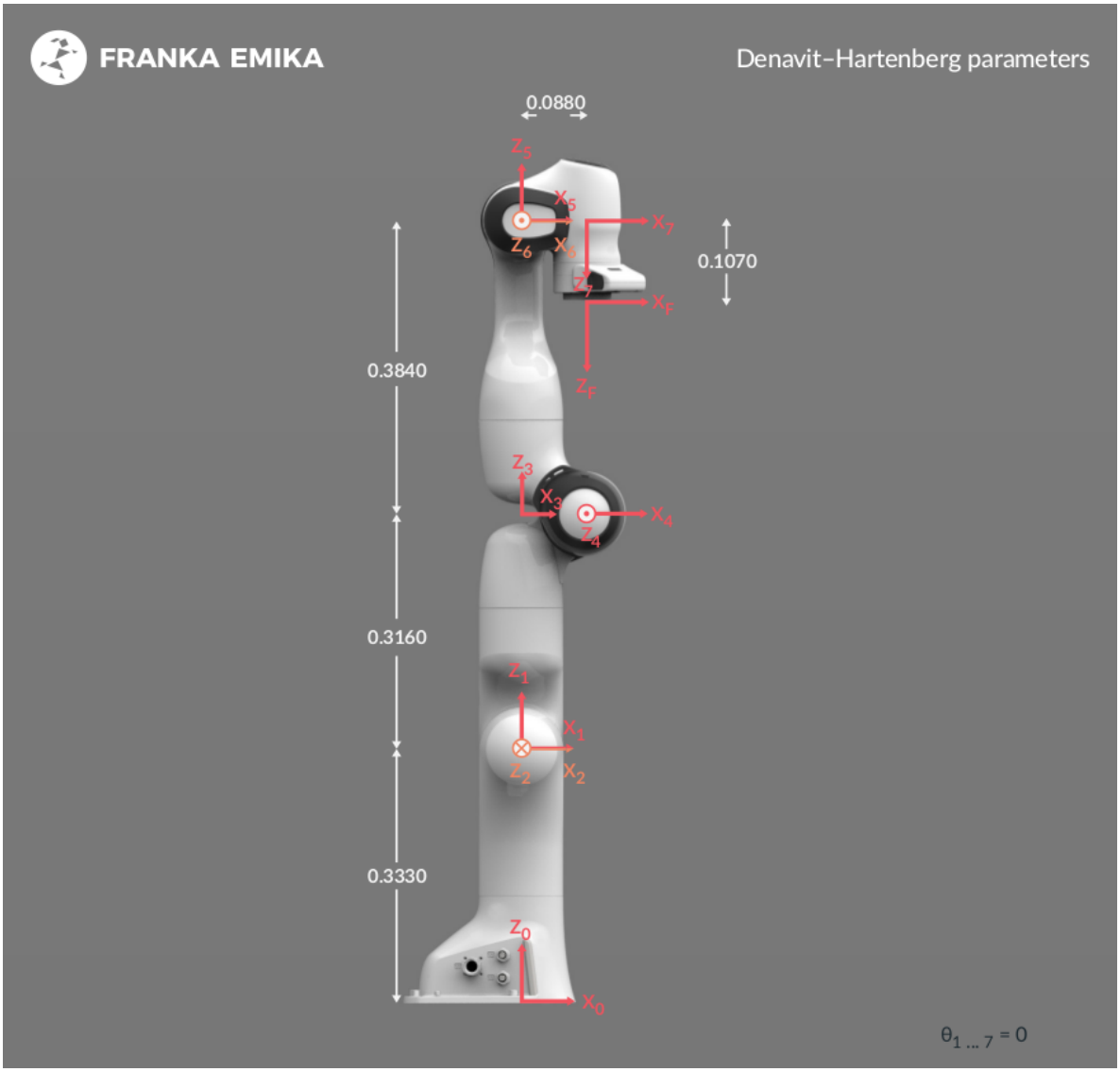

Figure 4: Robot Specifications

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.

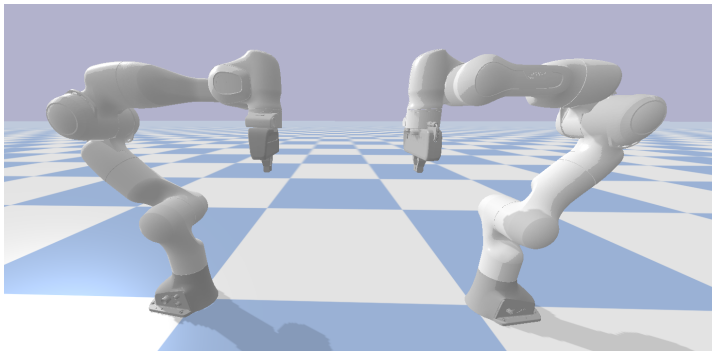

Figure 5: Initial position of the robots

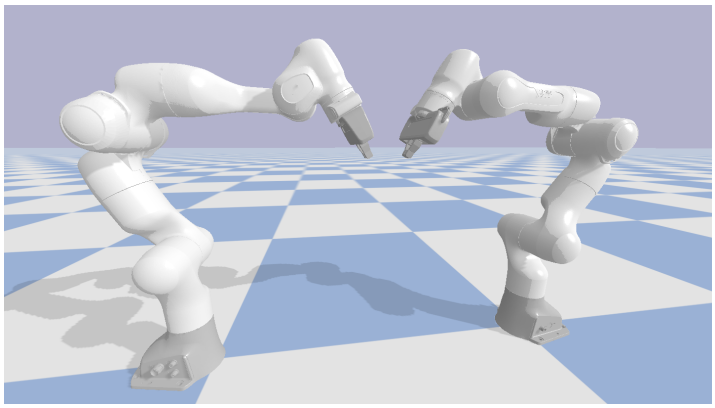

Figure 6: Relative orientation at the grasping position

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.

$$ {}^{w}\mathrm{O}_{o} = [0.5, 0, 0.59]; \tag{12} $$

$$ l_{obj} = 10\,(\mathrm{cm}). \tag{13} $$

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.

The table outlines the task hierarchy for Bimanual Manipulation.

Table 4: The Hierarchy of Tasks for Different Actions

Task	Type	Reach the Object	Move the Object	Stop the Motion
End-Effector Minimum Altitude	I,S	1	2	1
End-Effector Joint Limits	I,S	2	3	-
Bimanual Rigid Constraint	E,C	-	1	-
Tool/End-Effector Pose	E,AD	3	4	-

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

All task Jacobians should have dimensions $m \times 14$, where m corresponds to the row dimension of the task and its reference rate. Since we are controlling the arm joints of both manipulators (each with 7 DOFs), the Jacobian is a 14×14 matrix. The Jacobian, denoted as J_{jl}, is given by:

$$J_{jl} = \begin{bmatrix} I_{7\times7} & 0_{7\times7} \\ 0_{7\times7} & I_{7\times7} \end{bmatrix} \tag{14}$$

To compute the joint limits task reference, always aim to maintain a velocity that moves the joints towards the midpoint between the limits. Specifically, the desired task reference should be positioned centrally between the joint limits, denoted as *pandaArm.jlmin* and *pandaArm.jlmax*. Thus, the reference velocity $\dot{\tilde{x}}_{jl}$ is set to achieve this central position, ensuring the joints are equidistant from the minimum and maximum limits.

$$\dot{\tilde{x}}_{jl} = \lambda \left(\frac{pandaArm.jlmin + pandaArm.jlmax}{2} - q \right) \quad \lambda \in \mathbb{R}^+, \tag{15}$$

where q is the current joint values.

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

For the Bimanual Rigid Grasping Constraint task, the end-effector positions are aligned with the object frame $\langle o \rangle$. To maintain a secure grasp on the object, the end-effectors of both arms must move synchronously, imposing a kinematic constraint expressed as:

$$\dot{\mathbf{x}}_{o,left} = \dot{\mathbf{x}}_{o,right} \tag{16}$$

This constraint can be expressed in terms of the Jacobians and joint velocities as:

$$\mathbf{J}_{o,left}\dot{\mathbf{q}}_{left} = \mathbf{J}_{o,right}\dot{\mathbf{q}}_{right} \tag{17}$$

which simplifies to:

$$\mathbf{J}_{o,left}\dot{\mathbf{q}}_{left} - \mathbf{J}_{o,right}\dot{\mathbf{q}}_{right} = 0 \tag{18}$$

The resulting Jacobian for the rigid grasping constraint is then:

$$pandaArm.Jokc = \begin{bmatrix}\mathbf{J}_{o,left} & -\mathbf{J}_{o,right}\end{bmatrix} \tag{19}$$

Where:

- $\mathbf{J}_{o,left} = pandaArm.ArmL.wJo$, the Jacobian for the left arm extended to the object frame.
- $\mathbf{J}_{o,right} = pandaArm.ArmR.wJo$, the Jacobian for the right arm extended to the object frame.

The combined Jacobian *pandaArm.Jokc* captures the kinematic constraint necessary to synchronize the motion of both arms while grasping the object, ensuring that the end-effectors maintain the required relative positions and velocities to achieve a stable and coordinated grasp.

For the Bimanual Rigid Grasping Constraint task, the task reference is computed to ensure that both end-effectors maintain identical velocities, enforcing a rigid grasp of the object. The common velocity vector for the rigid grasp constraint is set to zero, reflecting the condition that there should be no relative motion between the end-effectors and the object. Mathematically, this is represented as:

$$\dot{\tilde{\mathbf{x}}}_{rc} = \mathbf{0}_{6\times 1} \tag{20}$$

This implies that the desired end-effector velocities in the object frame are constrained to zero, ensuring that the grasp remains rigid and the object does not slip or rotate relative to the manipulators. The task reference calculation aligns the velocities of both arms to this common zero vector, effectively coupling their movements to perform the rigid grasp and subsequent manipulation of the object.

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.

2.4.1 Case 1

In this case, the target position for the robot's end-effector is set to $[0.65, -0.35, 0.28]$. I used "Activation Functions" of both left and right arm manipulators to demonstrate the panda robots' behavior as shown in the figure below.

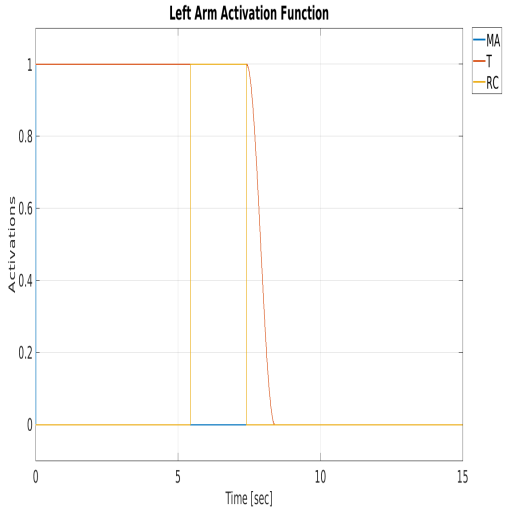

Figure 7: Left Arm Activation Function

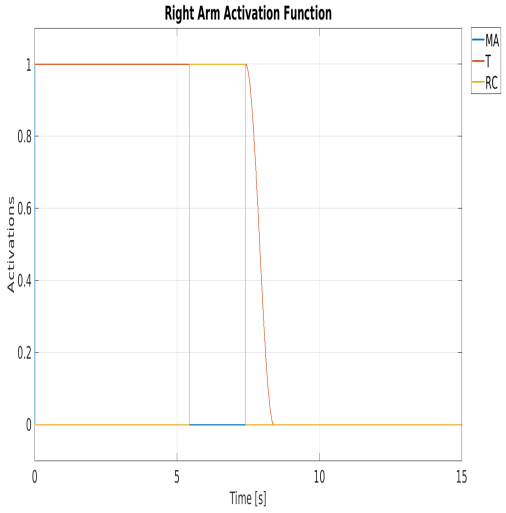

Figure 8: Right Arm Activation Functions

The activation functions of the right and left arms demonstrate synchronized behavior across three tasks: Minimum Altitude (MA), Manipulator Tool (T), and Rigid Constraint (RC). Both arms remain inactive in the Minimum Altitude task throughout the 15-second period, as indicated by the blue curve, which stays at zero. The Manipulator Tool task, represented by the red curve, is fully engaged by both arms from 0 to 9 seconds, after which it deactivates sharply. The Rigid Constraint task, shown by the yellow curve, activates later, starting at around 5 seconds and remains active until 7 seconds, and then declines to zero by 8 seconds. This overlap suggests that during the first 9 seconds, both arms are engaged in manipulating a tool, while simultaneously performing a rigid constraint task from 5 to 9 seconds. After this coordinated action, both arms cease their tasks, implying the completion of the manipulative and constraint operations within this period.

Figures 9, 10, 11, and 12 show the desired and cartesian velocities.

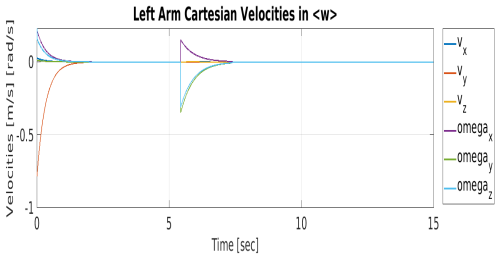

Figure 9: Left Arm Cartesian Velocities

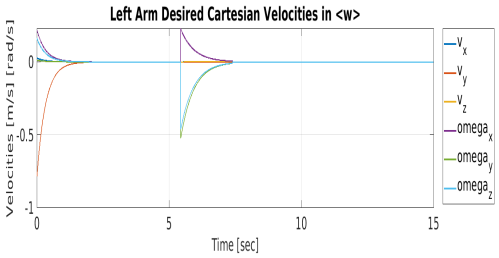

Figure 10: Left Arm Desired Cartesian Velocities

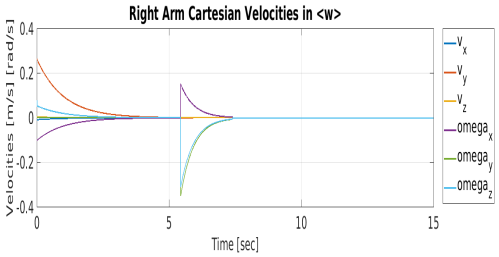

Figure 11: Right Arm Cartesian Velocities

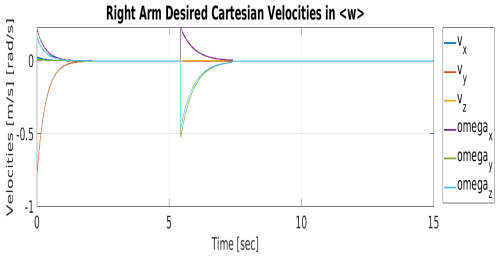

Figure 12: Right Arm Desired Cartesian Velocities

The plots display the Cartesian velocities of the left arm along with the desired velocities. In analyzing the robot's behavior, we observe that the velocities for the left arm in all directions—translational (v_x, v_y, v_z) and rotational $(\omega_x, \omega_y, \omega_z)$—generally align well with the desired values over time. However, discrepancies are noticeable, particularly in the early phases of motion where initial misalignment

occurs, evidenced by the higher deviation in v_y and ω_y. This suggests that the controller may have a lag in achieving the desired velocities promptly, resulting in a transient period before the velocities converge to the desired profiles. Comparing the actual velocities with the desired ones reveals that the control system is largely effective, but may need further tuning to reduce initial discrepancies. These differences are particularly relevant in scenarios requiring high precision, as the early deviations could impact the task performance, especially during delicate operations such as grasping or positioning close to target objects. Further tuning of control parameters or implementing adaptive control strategies might be necessary to enhance the synchronization between the actual and desired velocities, thereby improving the overall precision and stability of the robot's movements.

During the manipulation phase, the rigid constraint task effectively synchronizes the motions of both arms. Due to the higher gain settings, the desired Cartesian velocities for the left arm are higher than those for the right arm. However, the rigid constraint task ensures that the actual Cartesian velocities of both arms are equalized. This is achieved by dynamically adjusting the velocities: the left arm is decelerated to wait for the right arm, while the right arm is accelerated to match the velocity of the left arm. This coordination maintains the rigid connection between the two arms, ensuring synchronized and cohesive movements throughout the manipulation phase.

2.4.2 Case 2

In this case, the target position for the robot's end-effector is set to $[0.5, 0, 0.025]$. Also here, I used "Activation Functions" of both left and right arm manipulators to demonstrate the panda robots' behavior as shown in the figure below.

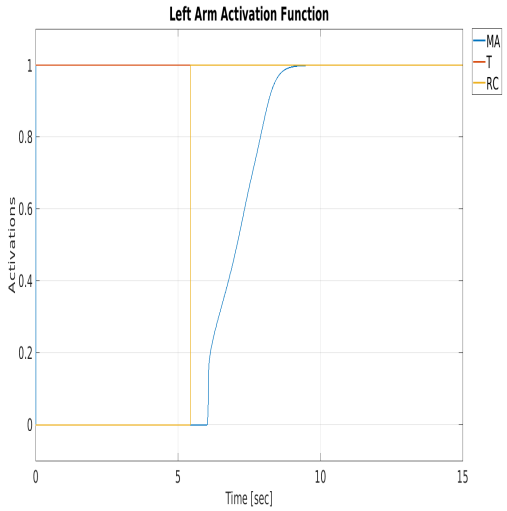

Figure 13: Left Arm Activation Functions

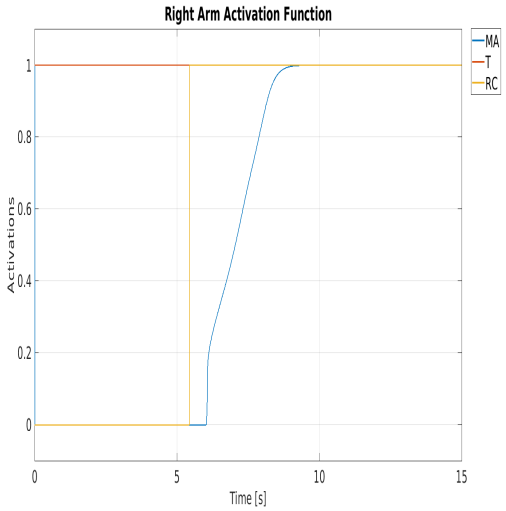

Figure 14: Right Arm Activation Function

The provided plots illustrate the activation functions for the left and right arms of a robot performing a series of tasks over time. In both the left and right arm activation functions, the T (Manipulator Tool) task is immediately activated to its maximum value (1) at the beginning and maintains this activation level until approximately 5 seconds. This indicates that the manipulator tool task is the initial priority for the robot. At around 5 seconds, the T activation drops sharply to zero, and the RC (Rigid Constraint Task) activation takes over, instantly reaching the maximum activation level of 1. This shift signifies a transition from a manipulator task to a rigid constraint task, suggesting a change in the robot's objective or environment that requires adherence to a rigid constraint. Simultaneously, the MA (Minimum Altitude) task remains at zero initially, indicating no concern for altitude during the early stages of the tasks. However, after the rigid constraint task is activated at around 5 seconds, the minimum altitude task (MA) begins to increase gradually. The curve shows a sigmoid-like shape starting from around 5 seconds, slowly ramping up and approaching its maximum activation level. This suggests that as time progresses and the rigid constraint task is stabilized, the robot begins fo-

cusing on maintaining or adjusting its altitude, possibly to optimize positioning or avoid obstacles.

Figures 15, 16, 17, and 18 illustrate the desired and actual Cartesian velocities for the second case. They demonstrate how the rigid constraint task restricts the left arm's movement while propelling the right arm, even with the safety tasks active. This behavior highlights the priority of the rigid constraint in maintaining coordinated motion between the arms, overriding the individual arm velocity settings imposed by safety constraints.

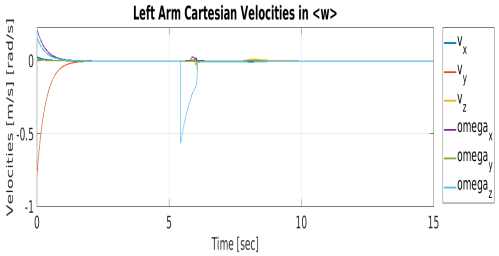

Figure 15: Left Arm Cartesian Velocities

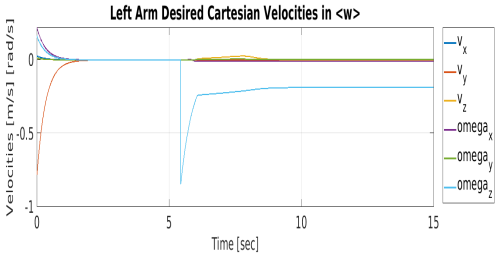

Figure 16: Left Arm Desired Cartesian Velocities

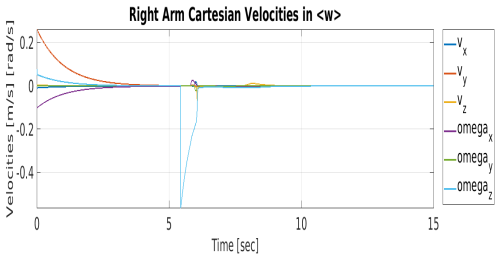

Figure 17: Right Arm Cartesian Velocities

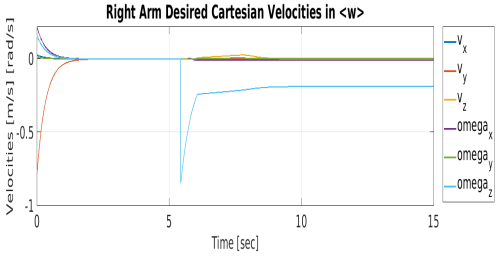

Figure 18: Right Arm Desired Cartesian Velocities

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.

$$ {}^{w}\mathrm{O}_{o} = [0.5, 0, 0.59]^{\top}(m); \tag{21} $$

$$ l_{obj} = 6\,(\mathrm{cm}). \tag{22} $$

   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

$$ {}^{w}\mathrm{O}_{g} = [0.60, 0.40, 0.48]^{\top}(m) \tag{23} $$

   (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.

In cooperative manipulation, joint references are computed in two stages. Initially, each manipulator calculates its own reference independently. Subsequently, a common reference for both manipulators is derived as a weighted average of these individual references.

The following two tables outline the task hierarchy for Cooperative Manipulation.

Table 5: The task Hierarchy for Cooperative Manipulation (Independent)

Task	Type	Reach the Object	Move the Object	Stop the Motion
End-Effector Minimum Altitude	I,S	1	1	1
End-Effector Joint Limits	I,S	2	2	-
Mutual Motion Task	E,C	-	-	-
Tool/End-Effector Pose	E,AD	3	3	-

Table 6: The task Hierarchy for Cooperative Manipulation (Cooperative)

Task	Type	Reach the Object	Move the Object	Stop the Motion
End-Effector Minimum Altitude	I,S	1	2	1
End-Effector Joint Limits	I,S	2	3	-
Mutual Motion Task	E,C	-	1	-
Tool/End-Effector Pose	E,AD	3	4	-

3.2 Q2: Comment the behaviour 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 case, the target position for the robot's end-effector is set to $[0.60, 0.40, 0.48]$. The following figures provide a clear illustration of the robot's behavior as it moves toward this goal.

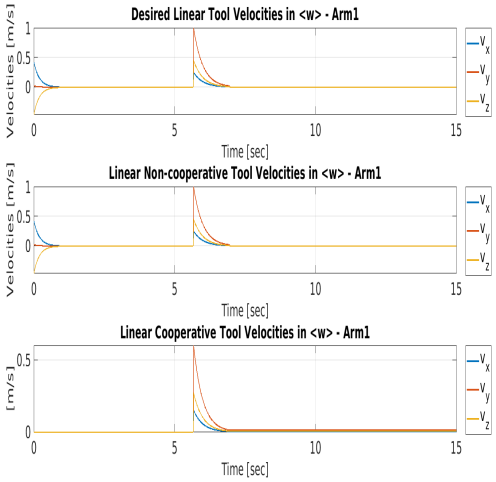

Figure 19: Desired, Non-Cooperative and Cooperative Linear Cartesian Velocity - Arm1

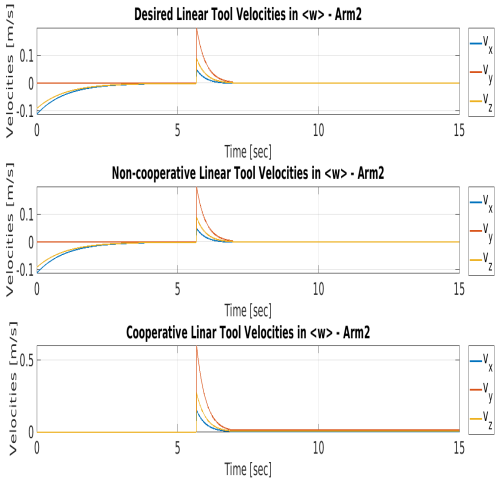

Figure 20: Desired, Non-Cooperative and Cooperative Linear Cartesian Velocity - Arm2

Figures 19, 20, 21, and 22 compare the desired, non-cooperative, and cooperative velocities of the arms. The desired and non-cooperative velocities are identical for both arms. Due to the higher gain used for the position/orientation task of the left arm (arm 1), the non-cooperative velocity of the left arm exhibits a greater magnitude compared to the right manipulator (arm 2). Conversely,

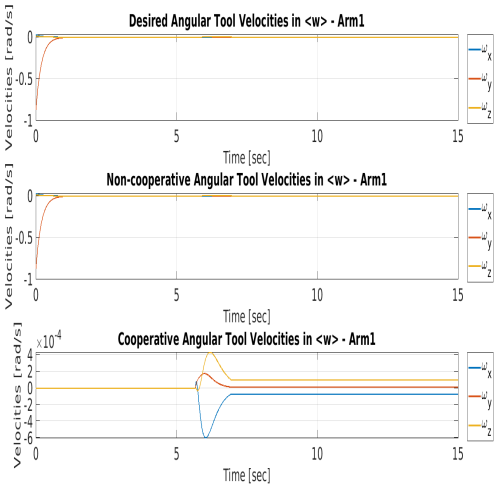

Figure 21: Desired, Non-Cooperative and Cooperative Angular Cartesian Velocity - Arm1

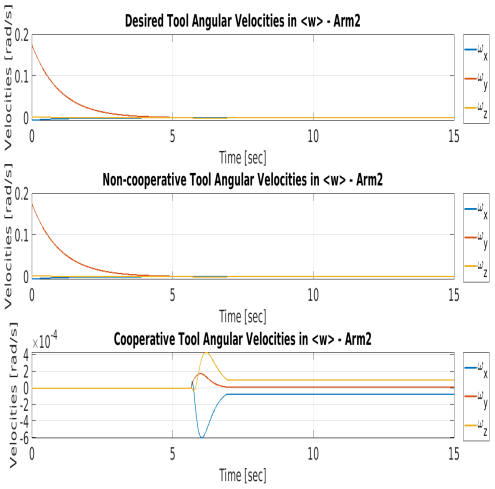

Figure 22: Desired, Non-Cooperative and Cooperative Angular Cartesian Velocity - Arm2

the cooperative velocities of both arms are equal, as they are calculated as a weighted average of the non-cooperative velocities of the left and right arms. This approach ensures that the rigid constraint condition is satisfied, promoting coordinated movement between the arms.