Cooperative Robotics

Authors:
Luna Gava, lunagava@me.com
Marta Lagomarsino, marta.lago@hotmail.it
Date:
February 10, 2020

General notes

- Exercises 1-4 are done with the ROBUST matlab main and unity visualization tools. Exercises 5-6 are done with the DexROV matlab main and unity visualization 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.

1 Exercise 1: Implement a "Safe Waypoint Navigation" Action.

1.1 Adding a vehicle position control objective

Initialize the vehicle far away from the seafloor. An example position could be

$$\begin{bmatrix}10.5 & 35.5 & -36 & 0 & 0 & \pi/2\end{bmatrix}^\top$$

Give a target position that is also sufficiently away from the seafloor, e.g.,

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

Goal: Implement a vehicle position control task, and test that the vehicle reaches the required position and orientation.

1.1.1 Q1: Report the hierarchy of task used and their priorities.

The requirement of the task is having the UVMS reaching a target position expressed in global coordinates. Within this framework, in addition to have the vehicle in the required position, we can add other objectives to perform the task with better performances and in a safer way. In exercise 1.1, we decided to include the task of maintaining a horizontal attitude during the motion, in order to avoid vehicle overturning. Hence, the hierarchy of tasks used in this exercise is (from the highest to the lowest priority objective):

1. Vehicle horizontal attitude (inequality, safety);
2. Vehicle position control task (inequality, action-defining), which controls both position x,y,z and orientation.

The priority of objectives follows the natural order, without over-constraining the system: safety objectives first (horizontal attitude), then actual action-defining objectives (vehicle position).

The horizontal attitude is an *inequality* objective, which requires the norm of the misalignment vector $\boldsymbol{\rho}$ between the absolute world frame z axis and the vehicle's one to be lower than a given value:

$$\|\boldsymbol{\rho}\| \leq \rho_M$$

The vehicle position task requires the vehicle frame $< v >$ roughly aligned with the goal frame $< g >$. In order to achieve this purpose, the following *inequality* conditions should be satisfied:

$$\|\boldsymbol{r}_v\| \leq r_{v,M}$$

$$\|\boldsymbol{\theta}_v\| \leq \theta_{v,M}$$

where $\boldsymbol{r}_v$ is the position error and $\boldsymbol{\theta}_v$ the orientation error.

What is the Jacobian relationship for the Vehicle Position control task?

The Jacobian relationship is defined as: $\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{y}}$, where the system velocity vector $\dot{\boldsymbol{y}} \in \mathbb{R}^{13}$ of the UVMS is:

$$\dot{\boldsymbol{y}} = \begin{bmatrix}\dot{\boldsymbol{q}} \\ \boldsymbol{\nu}\end{bmatrix}$$

and $\dot{\boldsymbol{q}} \in \mathbb{R}^7$ is the joint velocity vector and $\boldsymbol{\nu} \in \mathbb{R}^6$ is the stacked vector of the vehicle linear and angular velocity vectors, with components on the vehicle frame $< v >$:

$$\boldsymbol{\nu} = \begin{bmatrix}{}^v\mathbf{v} \\ {}^v\boldsymbol{\omega}\end{bmatrix}$$

The Jacobian matrix $\boldsymbol{J}(\boldsymbol{c})$ depends on the system configuration vector $\boldsymbol{c} \in \mathbb{R}^{13}$ of the UVMS as:

$$\boldsymbol{c} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{\eta} \end{bmatrix}$$

where $\boldsymbol{q} \in \mathbb{R}^7$ is the arm configuration vector and $\boldsymbol{\eta} \in \mathbb{R}^6$ is the vehicle generalized coordinate position vector, which is the stacked vector of the vehicle frame position $\boldsymbol{\eta_1}$ with respect to the inertial frame $< w >$, and the orientation vector $\boldsymbol{\eta_2}$, expressed in terms of the three angles yaw, pitch and roll.

In the specific case of this exercise, $\boldsymbol{J} \in \mathbb{R}^{6x13}$ is implemented in such a way to control only the vehicle position and orientation without taking into account the joint velocity vector of the arm. Therefore the Jacobian is simply a zero matrix of dimension 6x7 flanked by an identity matrix of dimension 6:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{0}_{6x7} & \boldsymbol{I}_{6x6} \end{bmatrix}$$

This Jacobian implementation preserves only the linear and angular velocities of the vehicle in output, but not the joint velocity vector of the arm. In fact, multiplying $\dot{\boldsymbol{y}}$ by a zero matrix produces a column vector of zeros, while multiplying by the identity matrix results in the column vector $\boldsymbol{\nu}$ itself.

Activation functions

We computed the activation functions of the objectives involved in this exercise:

- A_{ha}: activation function (1x1) of the vehicle horizontal attitude objective;
- $\boldsymbol{A}_{target}$: activation matrix (6x6) of the vehicle position control task.

Since both the objectives are bounded above inequalities, the activation functions are implemented as *increasing bell-shaped* functions. More specifically, the entries $a^i_{(j)}$ organized in the diagonal matrix $\boldsymbol{A}_j$ are defined as follows:

$$a^i_{(j)} = \begin{cases} 1, & \text{if } x_{(j)} > x_{(j),M} \\ s_j(x), & \text{if } x_{(j),M} - \Delta_{(j)} \leq x_{(j)} \leq x_{(j),M} \\ 0, & x_{(j)} < x_{(j),M} - \Delta_{(j)} \end{cases}$$

where $s_j(x)$ is any sigmoid function exhibiting a continuous behavior from 0 to 1. The $\Delta_{(j)}$ value allows to create a buffer zone, where the inequality is already satisfied, but the activation value is still greater than zero, to prevent any chattering problem around the inequality control objective threshold.

In particular, the values of the activation function A_{ha} is a function of the norm of the misalignment vector $\boldsymbol{\rho}$ (control objective variable of this task), while the structure of $\boldsymbol{A}_{target} \in \mathbb{R}^{6x6}$ is a little more complex:

$$\boldsymbol{A}_{target} = \begin{bmatrix} \boldsymbol{I}_{3x3} * s_{target}(\boldsymbol{r}_v) & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{3x3} & \boldsymbol{I}_{3x3} * s_{target}(\boldsymbol{\theta}_v) \end{bmatrix}$$

where $s_{target}(\boldsymbol{r}_v)$ and $s_{target}(\boldsymbol{\theta}_v)$ are increasing bell shaped functions, which depend on the position and orientation error respectively. To better understand the trend of the $\boldsymbol{A}_{target}$, the linear and angular part are plotted separately in the following (Figure 1a).

How was the task reference computed?

For equality or inequality control objectives, a suitable feedback reference rate $\dot{\bar{x}}$ that drives our x toward any arbitrary desired point $\bar{x}$, where the objective is satisfied, is *basic proportional law*:

$$\dot{\bar{x}} \triangleq -\lambda(x - \bar{x}), \quad \lambda > 0$$

where λ is a positive gain to control the convergence speed.

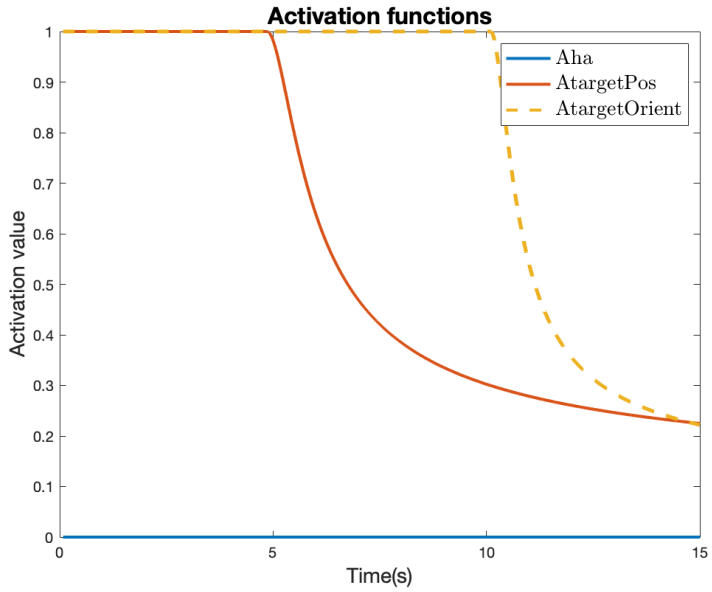
For this exercise, we call the function `CartError` by passing as inputs the transformation matrix of the target position frame with respect to vehicle frame ${}^{v}_{target}\boldsymbol{T}$ and an identity matrix of the same dimension $\boldsymbol{I}_{4x4}$. It provides as output the rotation (or misalignment) vector $\rho_{target,v}$ between the two

frames (computed using the Versor Lemma) and the basic vector $r_{target,v}$.
Then, we apply the basic proportional law to compute the linear and angular reference velocities:

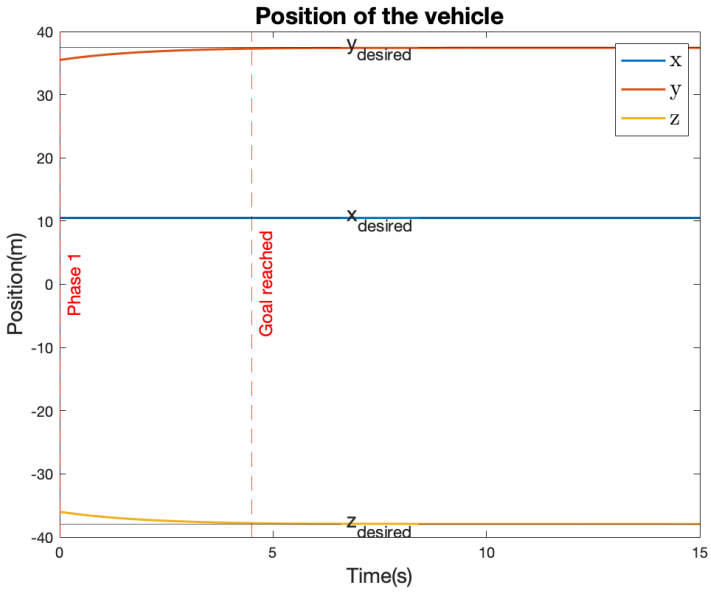
$$\dot{\hat{\boldsymbol{x}}}_{target} = \begin{bmatrix} {}^{v}\boldsymbol{v}_{target,v} \\ {}^{v}\boldsymbol{\omega}_{target,v} \end{bmatrix} = -\lambda \begin{bmatrix} \boldsymbol{r}_{target,v} \\ \boldsymbol{\rho}_{target,v} \end{bmatrix}$$

Eventually, we limit the requested velocities calling the function `Saturate.m`.

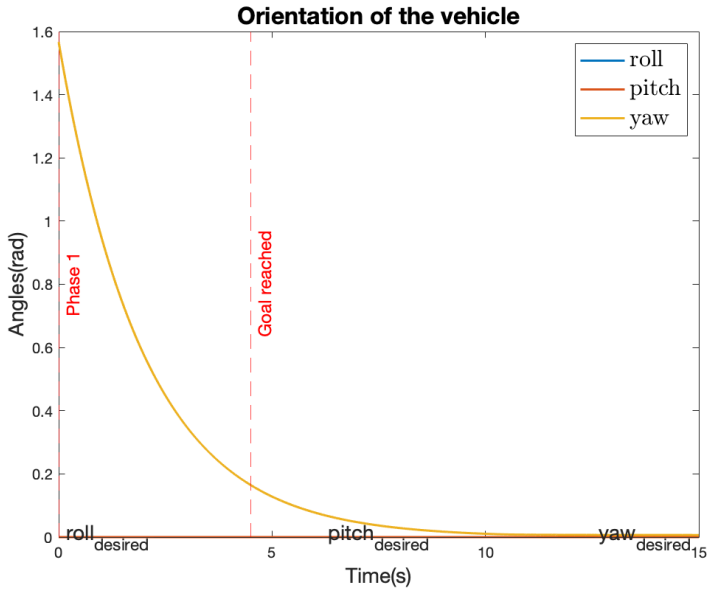
Figure 1a shows that the vehicle position task is active as the simulation starts, then the activation value $\boldsymbol{A}_{targetPos}$ and $\boldsymbol{A}_{targetOrient}$ decrease approaching the target position and orientation respectively. The horizontal attitude task is active, however in this specific combination of initial and desired configurations A_{ha} is always zero because there is no misalignment between the absolute world frame z axis and the vehicle's one. The goal is considered to be reached when the position and orientation error are below a given threshold, as illustrated in Figures 1b and 1c.



(a) Activation functions active in the first exercise.



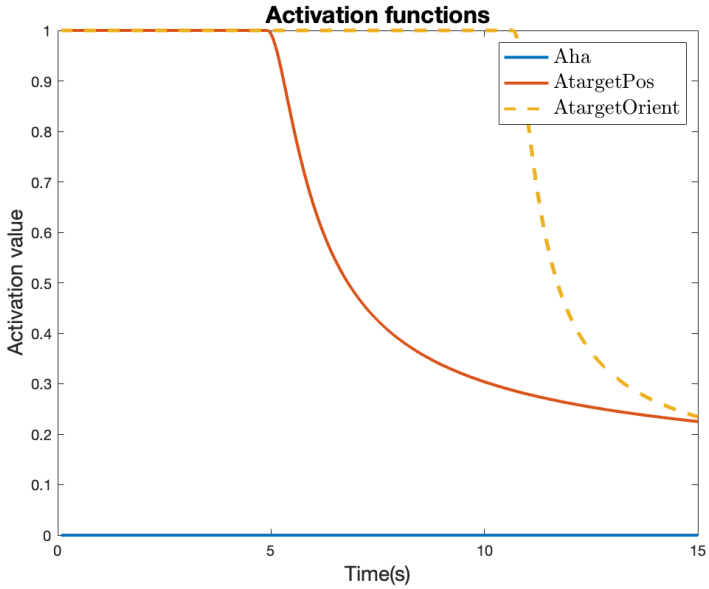
(b) Position of vehicle reaching the new desired position $[x_{desired}, y_{desired}, z_{desired}]$



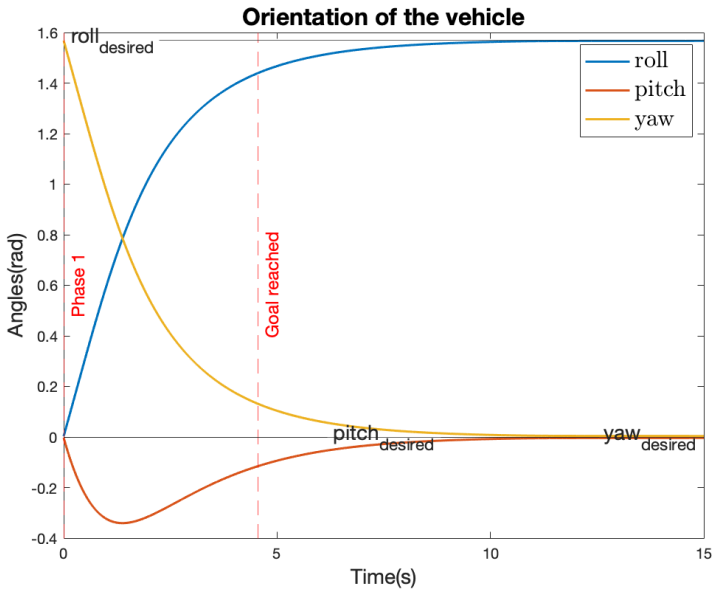
(c) Orientation of vehicle reaching the new desired orientation $[roll_{desired}, pitch_{desired}, yaw_{desired}]$

1.1.2 Q2: What is the behaviour if the Horizontal Attitude is enabled or not? Try changing the initial or target orientation in terms of roll and pitch angles. Discuss the behaviour.

To analyze the behaviour of the vehicle with or without horizontal attitude objective, we set the initial orientation of the vehicle to [0, 0, $\pi/2$] and we would like to reach a desired orientation with roll angle different from zero (i.e., $\phi = \pi/2$). If the horizontal attitude is *disabled*, the vehicle is in charge of changing its orientation. As we can see in Figure 2, if the activation function A_{ha} is 0, the roll angle reaches the desired angle $\pi/2$.



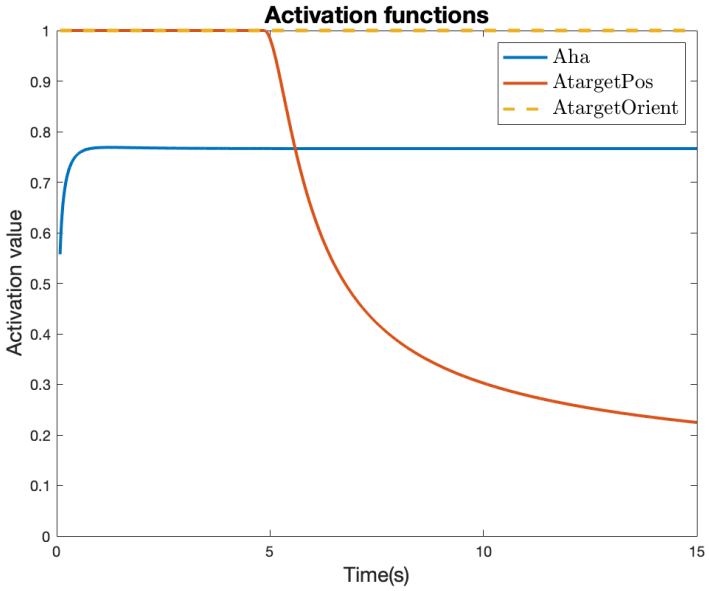
(a) Activation functions of horizontal attitude objective and vehicle position objective.



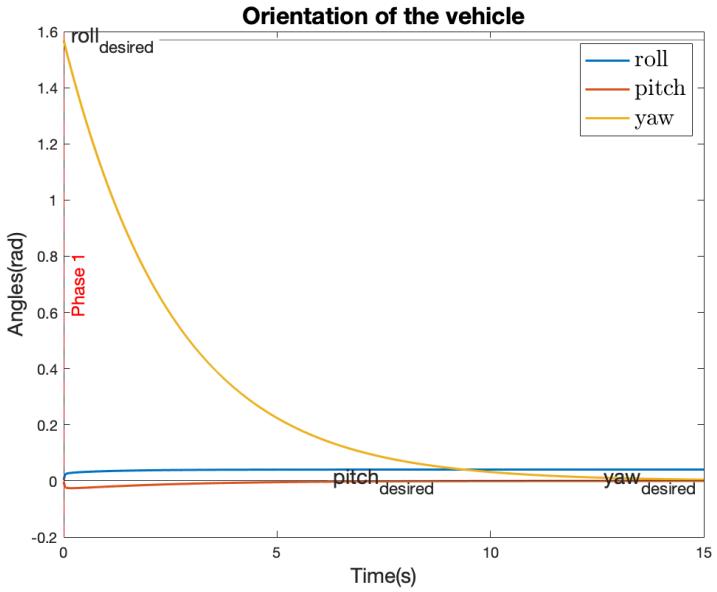
(b) Orientation of the vehicle in terms of roll, pitch and yaw.

Figure 2: Vehicle moving from initial orientation $[0, 0, \pi/2]$ to desired orientation $[\pi/2, 0, 0]$ with Horizontal attitude objective *disabled*.

Contrarily, if the horizontal attitude objective is activated (namely $A_{ha} \neq 0$) and it has higher priority with respect to the vehicle position control task, the horizontal attitude task prohibits to fulfill the action (see Figure 3). The two objectives are **in conflict**: the kernel of horizontal attitude task does not allow the rotation of the vehicle around its x-axis.



(a) Activation function of horizontal attitude objective and vehicle position objective.

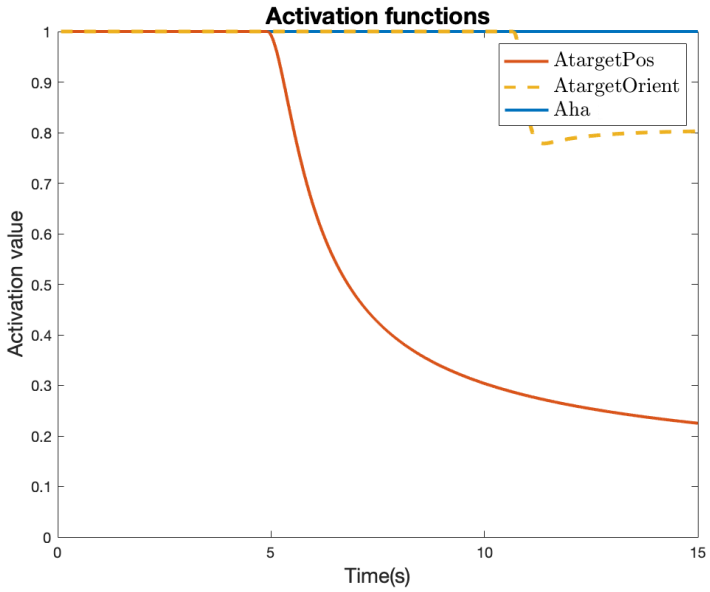


(b) Orientation of the vehicle in terms of Roll, Pitch and Yaw.

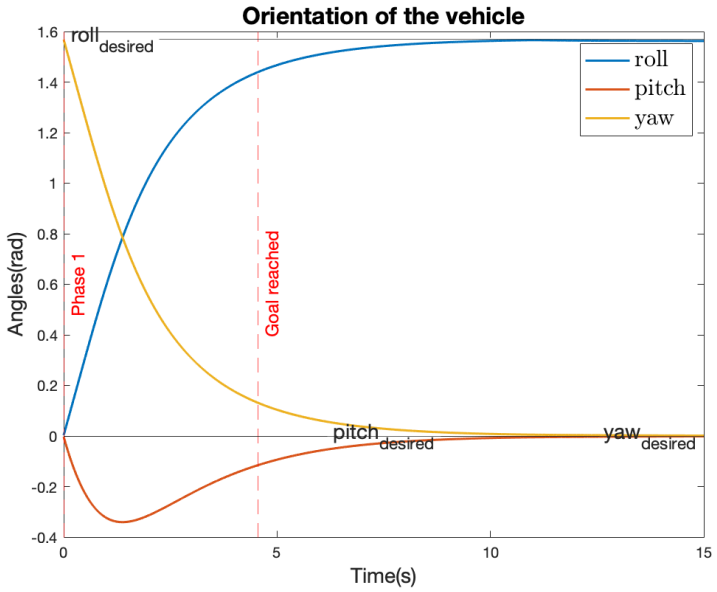
Figure 3: Vehicle moving from initial orientation $[0, 0, \pi/2]$ to desired orientation $[\pi/2, 0, 0]$ with Horizontal attitude objective *enabled*.

1.1.3 Q3: Swap the priorities between Horizontal Attitude and the Vehicle Position control task. Discuss the behaviour.

If we swap the priorities, the behaviour changes since the objectives are in conflict. The vehicle can perform a rotation around x or y axes, because the horizontal attitude task has lower priority with respect to the vehicle position control task. Figure 11 proves that the roll and pitch angles change in time and reach the desired value.



(a) Activation functions of horizontal attitude and target position



(b) Orientation of the vehicle in terms of Roll, Pitch and Yaw

Figure 4: Horizontal attitude enabled with a lower priority with respect to vehicle position control task

1.1.4 Q4: What is the behaviour if the Tool Position control task is active and what if it is disabled? Which of the settings should be used for a Safe Waypoint Navigation action?

The goal of this exercise is to guide the vehicle in a certain desired position. Therefore, the tool position control task is not required and can simply be *disabled*. The velocity of the arm is not changed by the algorithm and the body maintains the same initial joint configuration.

If the tool position control task is *active* with a higher priority than the vehicle position control task, the tool objective can modify the desired position of the vehicle (the desired vehicle position becomes a sort of optimization for performing the action of reaching something with the tool).

If instead it has a lower priority, the desired vehicle position is always reached and only if possible, the tool reaches its goal location.

1.2 Adding a safety minimum altitude control objective

Initialize the vehicle at the position:

$$\begin{bmatrix}48.5 & 11.5 & -33 & 0 & 0 & -\pi/2\end{bmatrix}^\top$$

Choose as target point for the vehicle position the following one:

$$\begin{bmatrix}50 & -12.5 & -33 & 0 & 0 & -\pi/2\end{bmatrix}^\top$$

Goal: Implement a task to control the altitude from the seafloor. Check that at all times the minimum distance from the seafloor is guaranteed.

1.2.1 Q1: Report the hierarchy of task used and their priorities. Comment how you choose the priority level for the minimum altitude.

For this exercise, the vehicle must maintain a certain distance from the seafloor to avoid collision. Hence, we add a control objective to the hierarchy of the previous exercise section. Specifically, the hierarchy of tasks ordered with descending priorities is the following:

1. Vehicle minimum altitude (inequality, safety);
2. Vehicle horizontal attitude (inequality, safety);
3. Vehicle position control task (inequality, action-defining), which both controls over position and orientation.

The priority of objectives follows the natural order: safety objectives first (minimum altitude and horizontal attitude), then actual action-defining objectives (vehicle position). The minimum altitude must have the highest priority, otherwise if the desired configuration of the vehicle is "dangerous" for the UVMS itself, the latter could crash the seafloor.

The minimum altitude objective can simply be stated as maintaining the altitude h above a certain threshold satisfying the following *inequality* condition:

$$h \geq h_{minAltitude}$$

Since this objective is a bounded below inequality, the activation function is a *decreasing bell-shaped* function. More specifically, the entries $a^i_{(j)}$ organized in the diagonal matrix A_j are defined as follows:

$$a^i_{(j)} = \begin{cases} 1, & \text{if } x_{(j)} < x_{(j),m} \\ s_j(x), & \text{if } x_{(j),m} \leq x_{(j)} \leq x_{(j),m} + \Delta_{(j)} \\ 0, & x_{(j)} > x_{(j),m} + \Delta_{(j)} \end{cases}$$

where $s_j(x)$ is any smooth, strictly decreasing function joining the two extrema and Δ is a value that creates a buffer zone, where the inequality is already satisfied, but the activation value is still greater than zero, preventing chattering problems around the inequality control objective threshold $x_{(j),m}$. Section 1.2.3 describes the implementation of the minimum altitude activation function $A_{minAltitude}$ more in detail.

1.2.2 Q2: What is the Jacobian relationship for the Minimum Altitude control task? How was the task reference computed?

For the minimum altitude control task, the Jacobian matrix takes into account only the z component (depth) of the position of the vehicle. We extract the third row of the rotation matrix of the vehicle with respect to the world, in order to be sure to compute the correct distance (Euclidean distance) also if the horizontal attitude task is not active. Hence, the Jacobian matrix is implemented as follow:

$$\boldsymbol{J} = \begin{bmatrix}0 & 0 & 1\end{bmatrix} * \begin{bmatrix}\mathbf{0}_{3x7} & {}^w_v\boldsymbol{R} & \mathbf{0}_{3x3}\end{bmatrix}$$

As regards the task reference, we apply the *basic proportional law* to compute the linear and angular reference velocities:

$$\dot{\tilde{x}}_{minAlt} = -\lambda[h - (h_{minAltitude} + 0.5)]$$

where h is the current altitude of the vehicle from the seafoor and $h_{minAltitude}$ is the value of the minimum altitude permissible to ensure safety. We add the value 0.5 to the $h_{minAltitude}$ to provide a robust control to the vehicle (more detailed description in section 1.2.3).

The current altitude h from the seafloor is computed using the measurements acquired by the sensor embedded on the vehicle, by taking into account that the vehicle can be oriented in a different way with respect to the world frame:

$$h = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} * {}^{w}_{v}\boldsymbol{R} * \begin{bmatrix} 0 & 0 & sensorDistance \end{bmatrix} \quad (1)$$

1.2.3 Q3: Try imposing a minimum altitude of 1, 5, 10 m respectively. What is the behaviour?

The activation function $A_{minAltitude}$ is implemented as a *decreasing bell shaped* function, namely:

1. if the current altitude h of the vehicle from the seafloor is below a fixed minimum altitude $h_{minAltitude}$, the activation function is equal to 1.
2. if h is above $h_{max} = h_{minAltitude} + \Delta$, $A_{minAltitude}$ is equal to 0 (we use $\Delta = 0.5$);
3. in the middle (see Figure 6), $A_{minAltitude}$ has a smooth trend.

The following drawings illustrates two possible situations where the UVMS may be. The red arrow represents the upwards push of the vehicle given by the thrusters in order to comply with the minimum altitude objective.

In situation 1, the difference between $h_{minAltitude}$ and h is positive, therefore the task tends to increase the distance from the seafloor. The $\Delta = 0.5$ allows to pass through the security limit and place the vehicle in the region $[minAltitude,\ minAltitude + 0.5]$ (Drawing 5 and Figure 9).

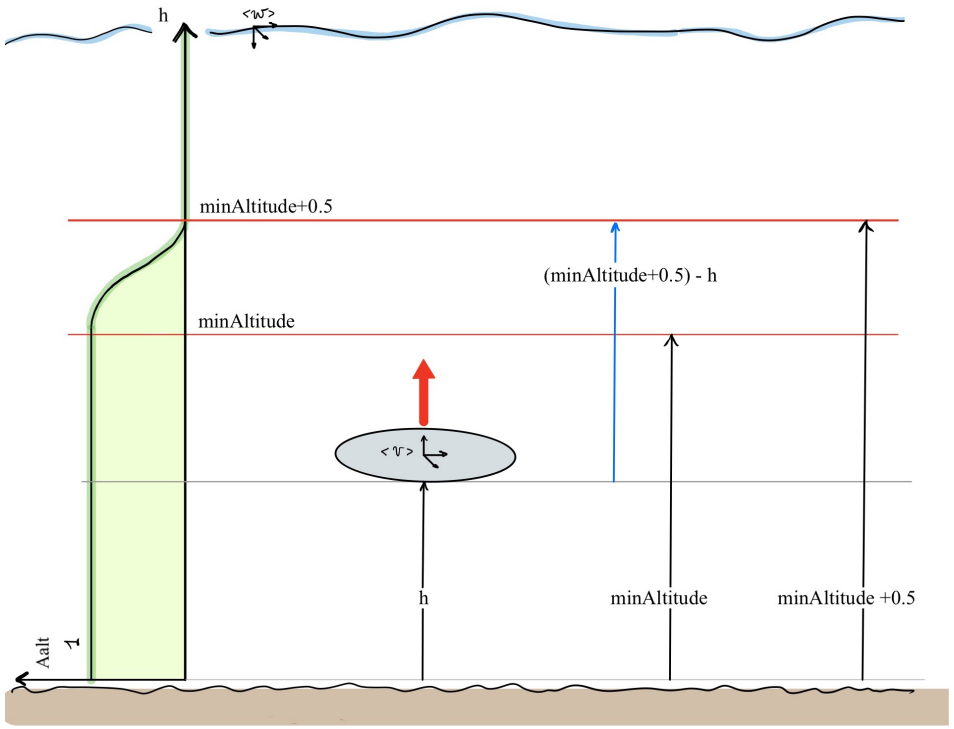


Figure 5: Scheme of UVMS below the minimum altitude

In situation 3, the vehicle has an altitude h that is in the middle between $h_{minAltitude}$ and $h_{minAltitude} + 0.5$ (Drawing 6). The difference is positive, so the UVMS is slightly driven upwards. However, the control is minimal, since the difference is very little and the activation function is decreasing.

The parameter $\Delta = 0.5$ has a really important role in this case because if we had set the reference $\bar{h} = h_{minAltitude}$, the difference would be negative and the task would tend to push the vehicle downwards.

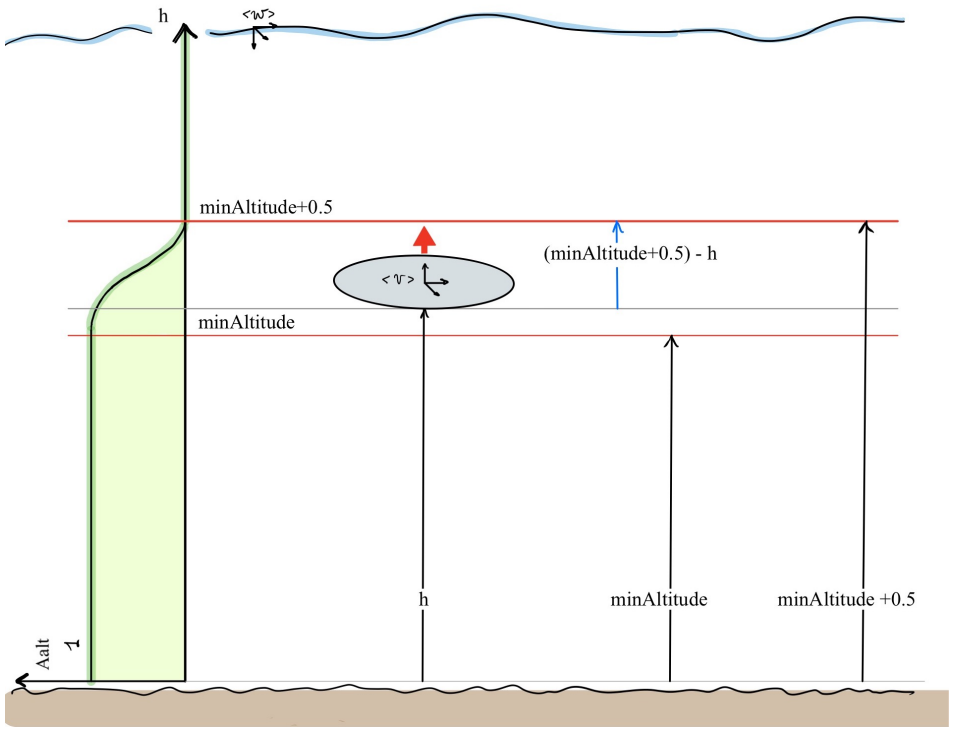


Figure 6: Scheme of UVMS between minimum altitude and minimum altitude+Δ

We start setting the minimum altitude to 1 ($h_{minAltitude} = 1$) and the initial and the final positions as the ones reported above. In this case, the vehicle maintains an altitude of about 1 m and it is able to reach the target position (Figure 7).

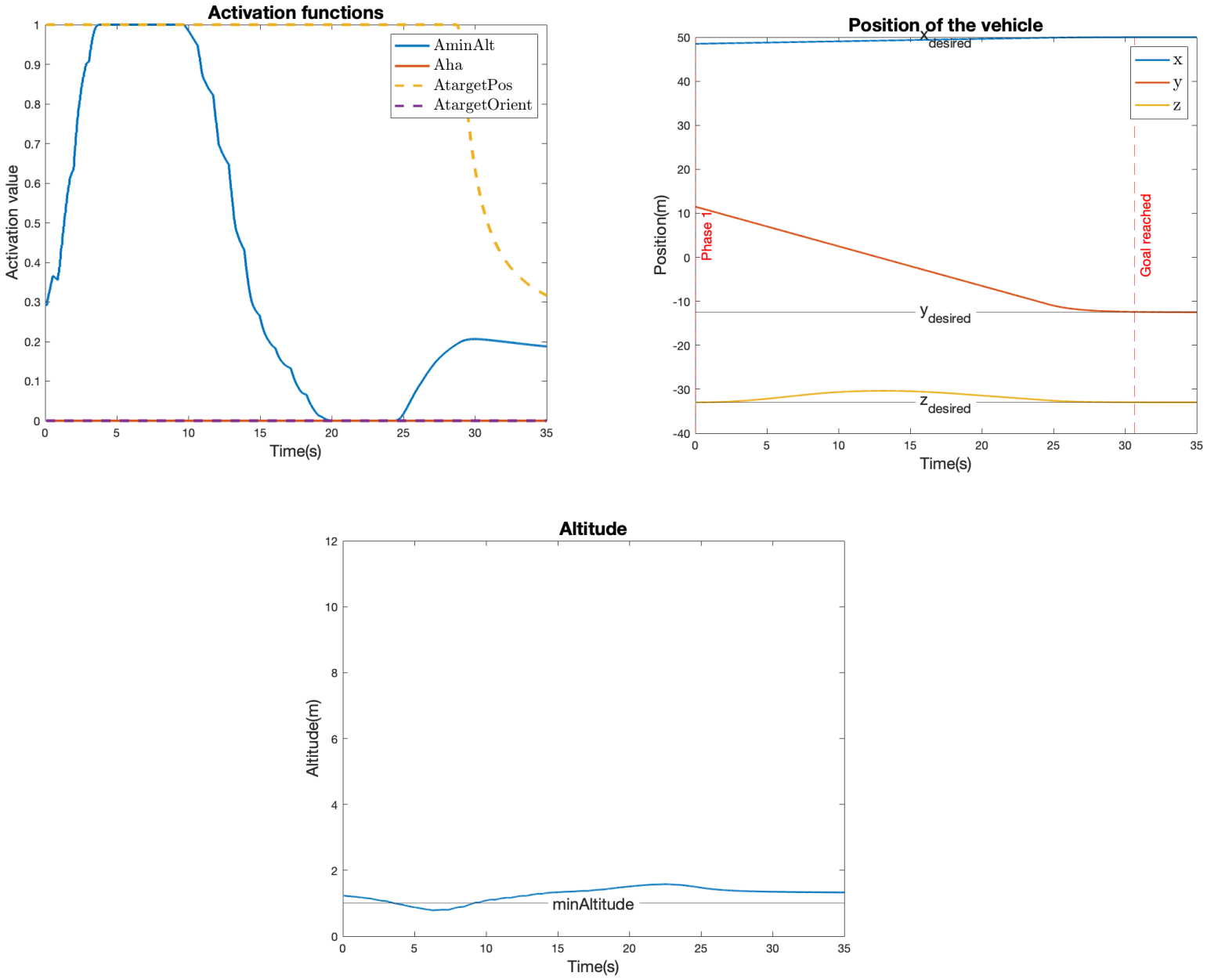
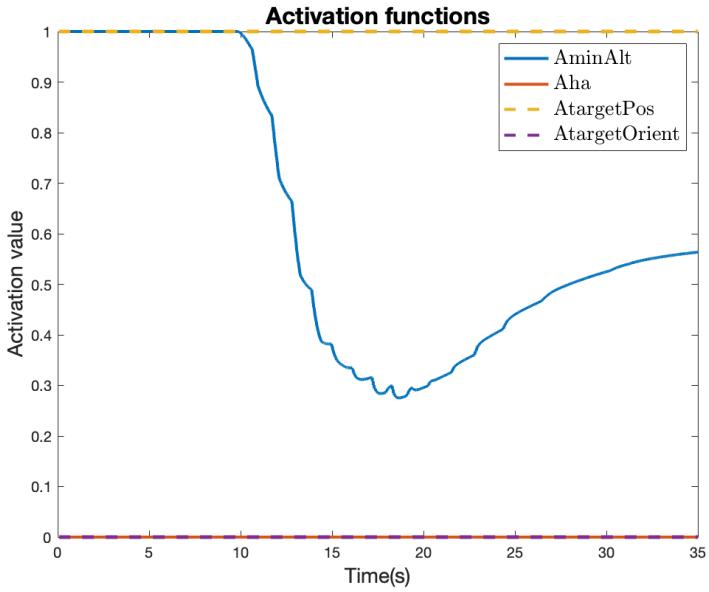
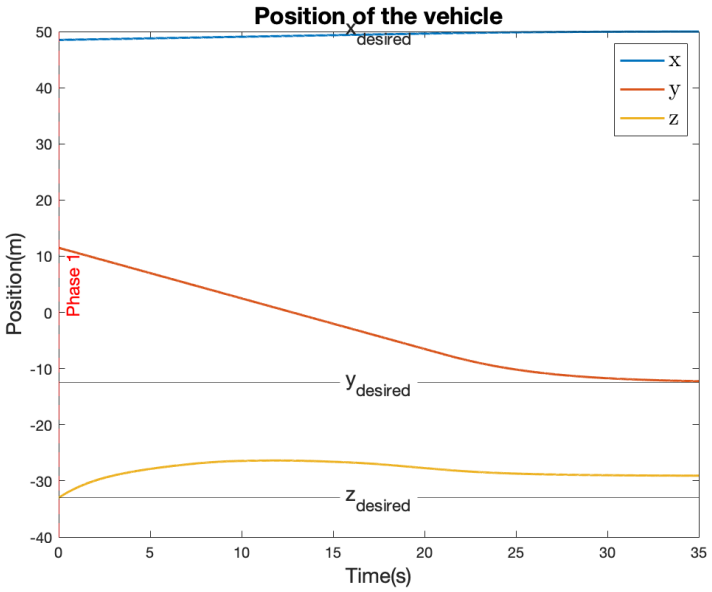


Figure 7: Results obtained setting $h_{minAltitude} = 1$

Subsequently, we set the minimum altitude equal to 5 or 10. In these cases (Figures 8 and 9), the activation function associated to the minimum altitude is always different from 0 ($h < h_{minAltitude}$) and the vehicle cannot achieve the desired position (since the desired altitude of the vehicle is less than the required minimum altitude).





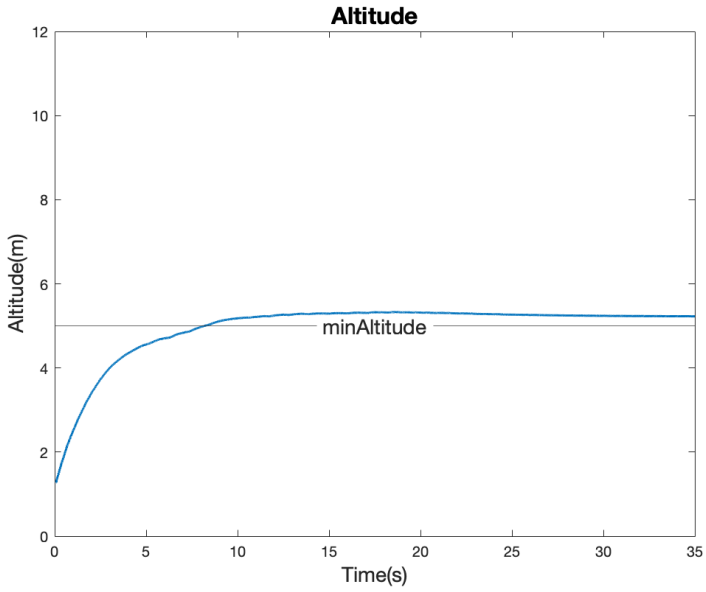


Figure 8: Results obtained setting $h_{minAltitude} = 5$

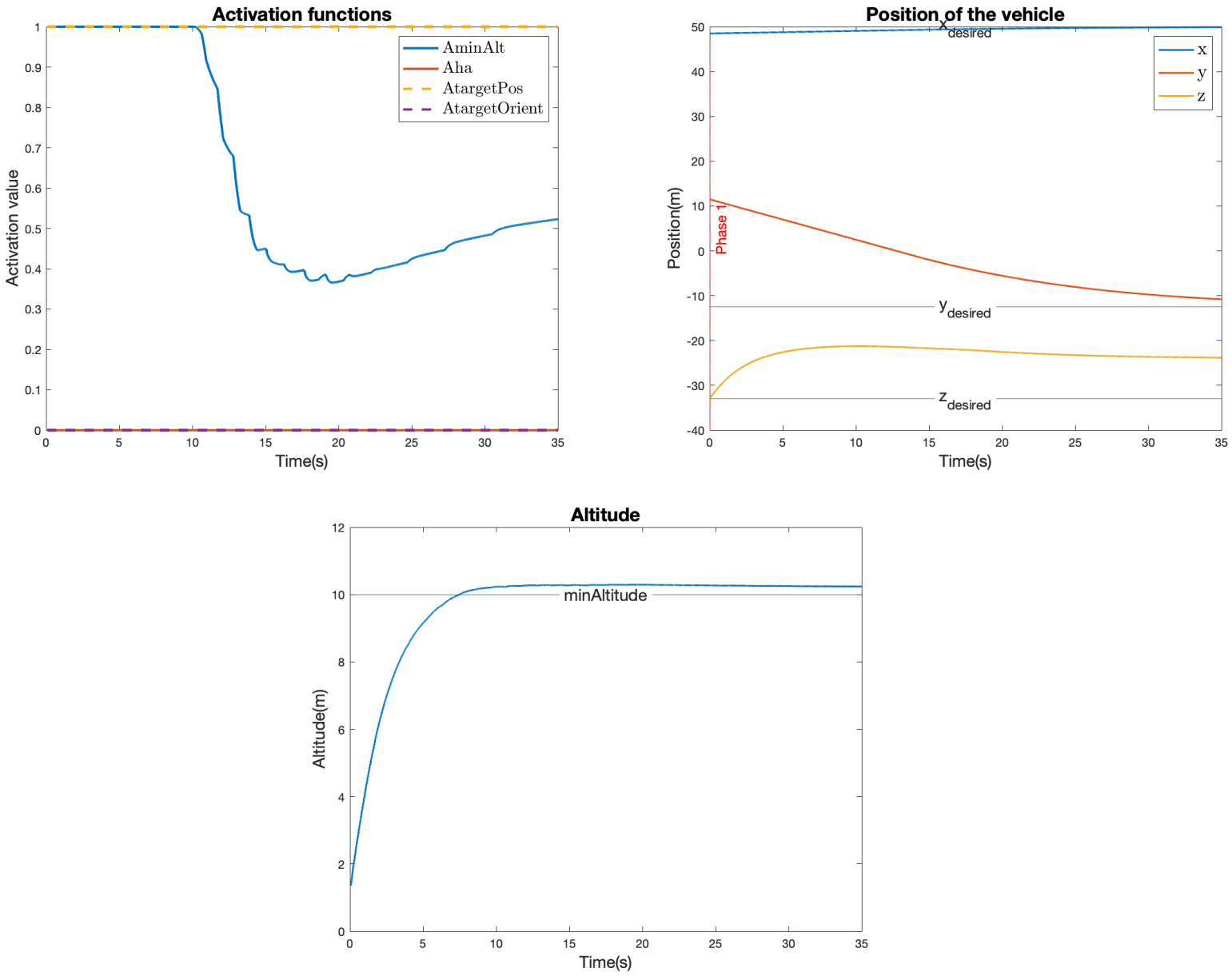


Figure 9: Results obtained setting $h_{minAltitude} = 10$

1.2.4 Q4: How was the sensor distance processed?

The sensor distance *sensorDistance* is processed to compute the current altitude of the vehicle from the seafloor by taking into account the orientation of the vehicle frame with respect to the world frame (see equation 1).

2 Exercise 2: Implement a Basic "Landing" Action.

2.1 Adding an altitude control objective

Initialize the vehicle at the position:

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

Goal: add a control task to regulate the altitude to zero.

2.1.1 Q1: Report the hierarchy of task used and their priorities. Comment how you choose the priority level for the altitude control task.

The landing is achieved using the following hierarchy of objectives (ordered with descending priorities):

1. Vehicle horizontal attitude (inequality, safety);
2. Vehicle altitude (equality, action-defining).

The vehicle altitude control objective has a lower priority than the vehicle horizontal one because we want to land maintaining the vehicle parallel to the seafloor (under the hypothesis of horizontal seafloor) for safety reasons.

The vehicle altitude is implemented as an *equality* control objective:

$$h = h_0$$

The vehicle altitude task must be always maintained active, i.e. $A_{altitude} \equiv 1$

2.1.2 Q2: What is the Jacobian relationship for the Altitude control task? How was the task reference computed?

As in the previous exercise 1.2.2, the Jacobian matrix takes into account only the z component of the vehicle and it is implemented as follows:

$$\boldsymbol{J} = \begin{bmatrix}0 & 0 & 1\end{bmatrix} * \begin{bmatrix}\mathbf{0}_{3x7} & {}^{w}_{v}\boldsymbol{R} & \mathbf{0}_{3x3}\end{bmatrix}$$

The task reference is implemented through the *basic proportional law*:

$$\dot{\bar{x}}_{Altitude} = -\lambda(h_{Altitude} - \bar{h})$$

where h is the current altitude measured by the sensor and $\bar{h}$ is desired final altitude, namely zero. Actually, the sensor does not correspond to the lowest part of the vehicle; therefore, we set the $\bar{h} = h_0 = 0.1$m.

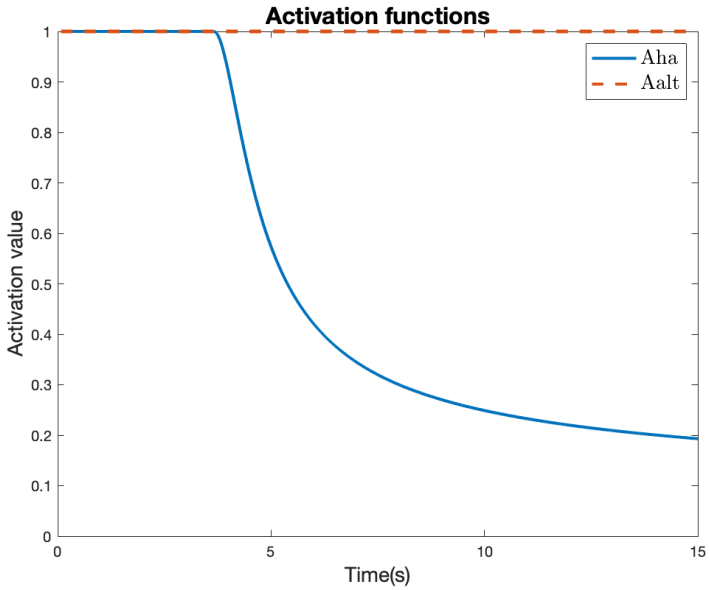
Figures 10b and 10c show respectively the time trend of position of the vehicle and altitude during the landing action.

2.1.3 Q3: how does this task differ from a minimum altitude control task?

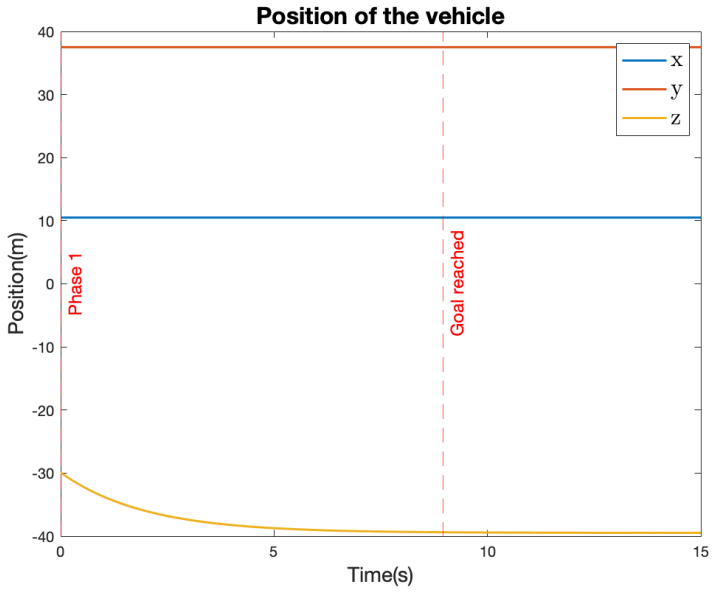
The current task is an equality objective, so it has to be always active (the corresponding activation function is always equal to 1, as we can see in Figure 10a).

Contrarily, the minimum altitude control task is an inequality objective that should be activated only when necessary. The corresponding activation function is implemented as a decreasing bell shaped activation function (see Figures 7, 8, 9), namely it decreases and the task is completely deactivated when the current altitude is above a given threshold plus Δ.

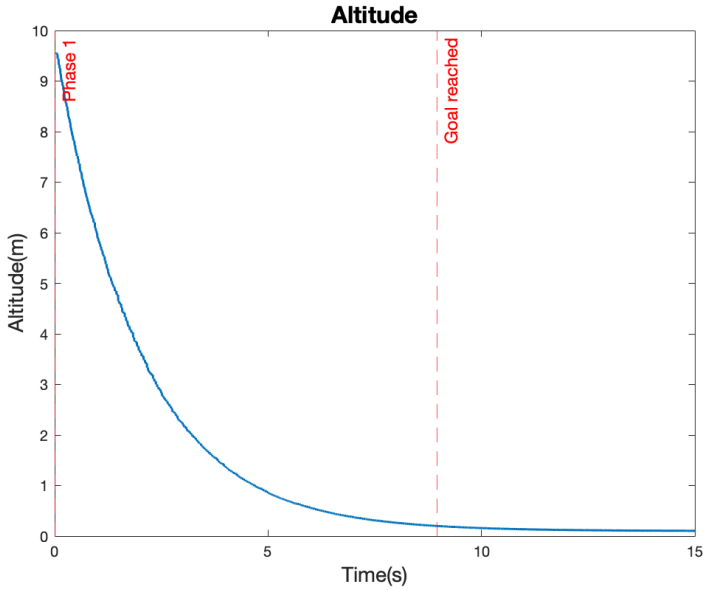
Summarizing, the minimum altitude objective is activated only when really necessary, while the altitude objective is always active.



(a) Activation functions



(b) Position x,y,z of the vehicle



(c) Altitude of the vehicle with respect to the seafloor

Figure 10: Altitude control task (equality objective)

2.2 Adding mission phases and change of action

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:

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

When the position has been reached, land on the seafloor using the basic "landing" action.

2.2.1 Q1: Report the unified hierarchy of tasks used and their priorities.

The exercise requires the implementation of two phases: the first phase corresponds to safe waypoint navigation, while the second phase regards the landing action.

The hierarchy of tasks of the *first phase*, ordered with descending priorities, is the following:

1. Vehicle horizontal attitude objective (inequality, safety);
2. Vehicle position control task (inequality, action-defining), which both controls the position and orientation of the vehicle;

The hierarchy of objectives of the *second phase* is:

1. Vehicle horizontal attitude (inequality, safety);
2. Vehicle altitude (equality, action-defining).

The vehicle horizontal attitude objective has the highest priority in both phases because we want to oblige the vehicle to move remaining always horizontal for safety reasons. Vehicle position and vehicle altitude are mission-oriented objectives, namely what the system really needs to execute to accomplish the user defined mission.

2.2.2 Q2: How did you implement the transition from one action to the other?

We propose different approaches to solve this problem.

First solution

The transition from the vehicle control position task to the landing action is performed by checking the norm of the basic vector between the target and the vehicle frames. If the latter is below a given threshold, the function `UpdateMissionPhase.m` updates the mission phase value to 2. Therefore, the vehicle starts landing. This is done by setting a specific activation function that has a different value in the two phases. In particular, we set:

$$\boldsymbol{A}_{target} = \begin{cases} a^i_{target}(x) & \text{if mission\_phase} = 1 \\ 0 & \text{if mission\_phase} = 2 \end{cases}$$

$$A_{altitude} = \begin{cases} 0 & \text{if mission\_phase} = 1 \\ 1 & \text{if mission\_phase} = 2 \end{cases}$$

This solution presents a discontinuity in the solution. The issue is less dramatic than the ones which could be involved in inequality objectives because chattering may not happen in this case (once we reach the second phase, we cannot go back to the previous phase).

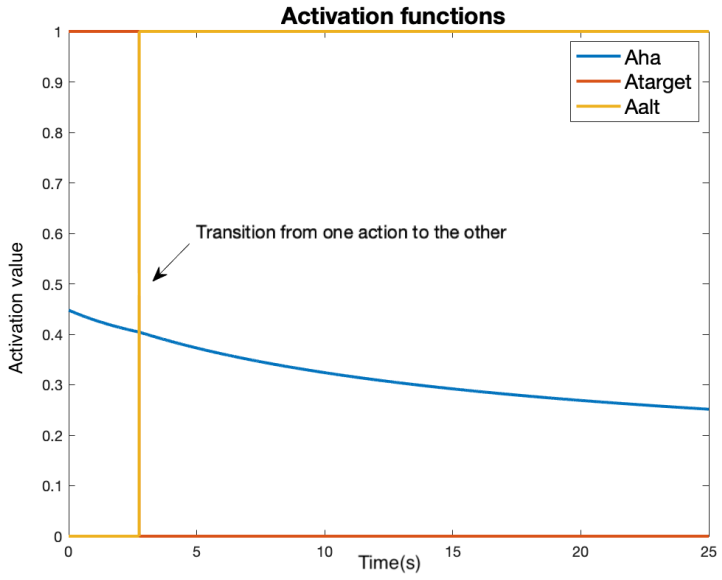


Figure 11: Activation functions during the transition from the first phase to the second one

Second solution

To avoid discontinuities, transitions from an action to another one should be smoothly activated. We implemented the action sequencing by modifying the overall activation function A_j as the product of two functions:

$$A_j(x,p) = a_j^i(x)\ a_j^p(p)$$

where

- $a_j^i(x)$ is a function of the control objective variable x_j, and its purpose is to deactivate the task whenever the inequality objective is satisfied, to avoid overconstraining the system. For equality control objectives it clearly holds that $a_j^i(x) \equiv 1$.
- $a_j^p(p)$ is a continuous sigmoidal function of a new parameter p external to the control task itself (**external activation**).

In particular, the new involved parameter p is the norm of the basic vector between the target and the vehicle and the activation function for the vehicle position control task is computed as:

$$\boldsymbol{A}_{target} = \boldsymbol{A}_{target} * \ Aexternal_{target}$$

where $Aexternal_{target}$ is a increasing bell shaped function that is equal to 0 if the norm is below 0.5, gradually increases in the interval $[0.5, 0.8]$ and it is stable to 1 for a norm higher than 0.8. On the other hand, the activation function associated to the lading is:

$$A_{alt} = 1 * \ Aexternal_{alt}$$

where $Aexternal_{alt}$ is a decreasing bell shaped function that is equal to 1 if the norm is below 0.5, gradually decreases in the interval $[0.5, 0.8]$ and it is stable to 0 for a norm higher that 0.8.

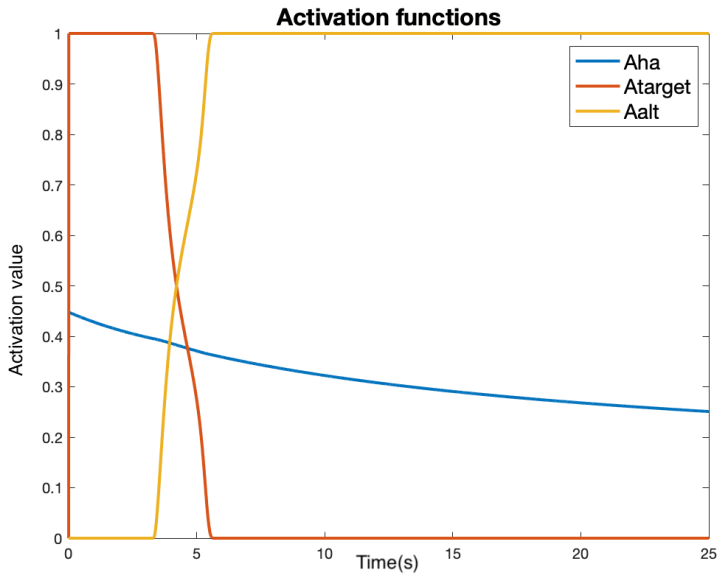


Figure 12: Activation functions during the transition from the first phase to the second one

Since the actual code implements the third solution, we report a screenshot of the code in Figure 13 implementing this second solution. The final result is shown in Figure 12.

```
function [uvms, mission] = UpdateMissionPhase(uvms, mission)
    switch mission.phase
        % add policy for changing phase
        case 1
        % Reach the vehicle desired position using Safe Waypoint Navition Action

            % External activation functions whose parameter is the norm of
            %   the basic vector between the target and the vehicle frame
            %   to obtain a smooth transition between actions
            % external activation function of vehicle position control task != 0 for norm > 0.5
            uvms.Aexternal.target = IncreasingBellShapedFunction(0.5, 0.8, 0, 1, norm(uvms.vTtarget(1:3,4)));
            % external act.f. of landing = 0 for norm > 0.5
            uvms.Aexternal.alt = DecreasingBellShapedFunction(0.5, 0.8, 0, 1, norm(uvms.vTtarget(1:3,4)));

            if norm(uvms.vTtarget(1:3,4)) < 0.5
                mission.phase = 2; % Landing
            end
        case 2
        % Landing
            uvms.Aexternal.target = 0;
            uvms.Aexternal.alt = 1;
    end
end
```

Figure 13: Implementation of external activation function in UpdateMissionPhase.m

Third solution

The function of the mission phases $a_j^p(p)$ is implemented as a function of the *time elapsed* within the current phase, allowing the proper activation of new tasks during a phase transition, and deactivation of objectives that are not anymore relevant.

We have implemented the function `InitMissionPhase` which defines the number of phases, the active tasks in each phase, the exit condition of each action and the duration of the transition. Moreover, it initializes time variables which change during the execution.

The function `taskSequence` reports the list of all tasks ordered with descending priority. For each of this task, it calls the functions `UpdateActivation` and `iCAT_task`. First of all, it initializes the projection matrix $\boldsymbol{Q}_0 = \boldsymbol{I}$ and the vector $\boldsymbol{\rho}_0 = \boldsymbol{0}$. The control vector $\boldsymbol{\rho}$ contains the joint velocity vectors of the arm and the linear and angular velocity of the vehicle:

$$\boldsymbol{\rho}_p = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{p}} \end{bmatrix}$$

The function `iCAT_task` implements the *task priority framework* methodology called iCAT (inequality Control objectives, Activations and Transitions). This technique exploits novel regularization methods to activate and deactivate any row of a given task in a prioritized hierarchy without incurring in practical discontinuities (that actually arise in presence of activation functions), while maintaining as much as possible the invariance properties of the other active tasks.

The problem of tracking with priorities the given reference velocities of each task can be found as the solution of a sequence of minimization problems introduced within the *task oriented regularization*. The function `iCAT_task` computes the *new* $\boldsymbol{\rho}$ value based on the Jacobian and reference value of this task, its activation function and the previous value of the projection matrix. It returns also the new value of the projection matrix.

The `iCAT_task` calls the function `iCAT_pseudoInverse`, which performs the general pseudoinverse minimizing problem and provides as output the pseudoinverse $(\boldsymbol{JQ})^{\#,\boldsymbol{A},\boldsymbol{Q}}$. In particular, thanks to the function `RegPseudoInverse` function, the solution of regularized minimization problem is computed by exploiting the *unifying formula* for the pseudo inverse.

The computation of the pseudoinverse is useful for computing other quantities needed for the calculation of the *new* $\boldsymbol{\rho}$:

$$\boldsymbol{W}_k = \boldsymbol{J}_k\boldsymbol{Q}_{k-1}(\boldsymbol{J}_k\boldsymbol{Q}_{k-1})^{\#,\boldsymbol{A}_k,\boldsymbol{Q}_{k-1}}$$

and the new value of the projection matrix:

$$\boldsymbol{Q}_k = \boldsymbol{Q}_{k-1}(\boldsymbol{I} - (\boldsymbol{J}_k\boldsymbol{Q}_{k-1})^{\#,\boldsymbol{A}_k,\boldsymbol{I}}\boldsymbol{J}_k\boldsymbol{Q}_{k-1}$$

and finally the *new* $\boldsymbol{\rho}$ is equal to:

$$\boldsymbol{\rho}_k = \boldsymbol{\rho}_{k-1} + \boldsymbol{Q}_{k-1}(\boldsymbol{J}_k\boldsymbol{Q}_{k-1})^{\#,\boldsymbol{A}_k,\boldsymbol{I}}\boldsymbol{W}_k(\dot{\hat{\boldsymbol{x}}}_k - \boldsymbol{J}_k\boldsymbol{\rho}_{k-1})$$

where $k = 1, ..., p$ with p equal to the total number of priority levels.

Before calling the `iCAT_task` for each objective, `UpdateMissionPhase` checks if the end condition of a phase is met by calling the corresponding callback. If the requirements are satisfied, the transition starts.

The function `UpdateActivation` activates or deactivates a task during a phase transition by post multiplying the activation function of a specific objective by a sigmoidal function. The possible transitions are:

1. inactive → active
2. active → inactive

For instance, if the j-task was inactive in the previous action $\mathcal{A}_1$ and it is active at the current action, $\mathcal{A}_2$, $a_j^p(p)$ is an *increasing bell shaped* function whose parameter is the current time and the buffer zone starts when the transition is detected and has the duration specified in `InitMissionPhase`.

On the contrary, the deactivation of the j-task (active in the previous action $\mathcal{A}_1$ and inactive at the current action $\mathcal{A}_2$) is implemented with a *decreasing bell shaped* function.

Figure 14 shows the result using the described architecture. As we can see, until x_j has not reached the desired value $\bar{x}_j$, the task is active $a_j^i = 1$ and the feedback reference rate will drive x_j toward the region where its value is acceptable.

In the transition zone, the feedback reference rate will still try to drive x_j towards the point $\bar{x}_j$. However, the priority of this task is not any longer fully enforced. Indeed, as a_j^i decreases from 1 to 0, lower priority tasks will have the opportunity to influence the current task.

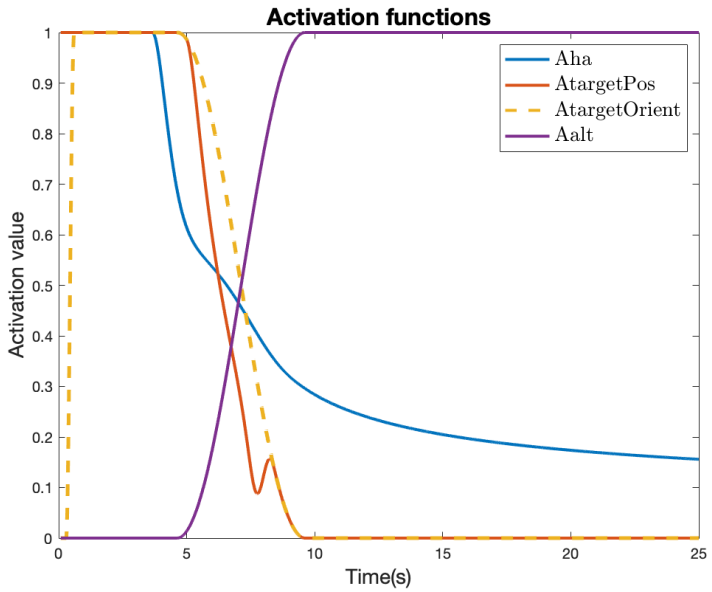
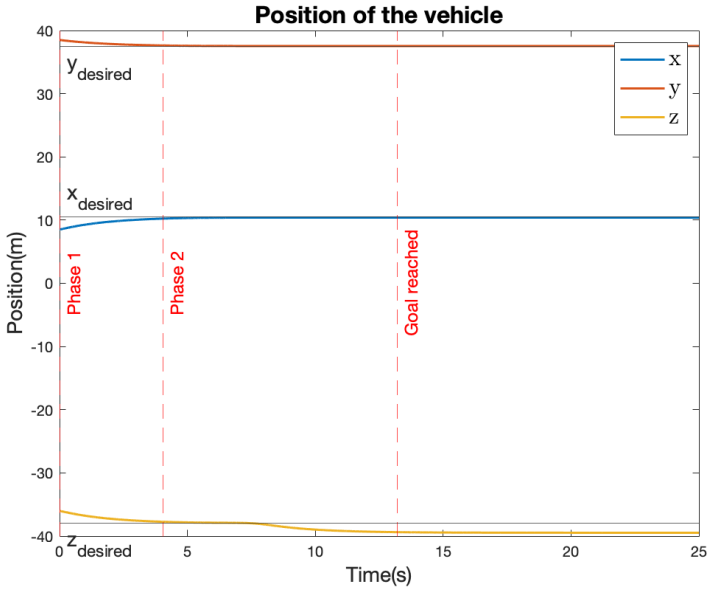
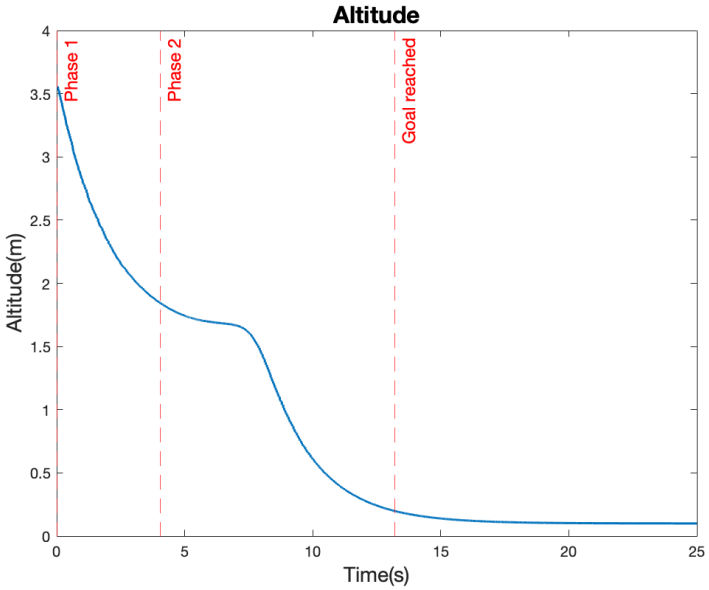


Figure 14: Activation functions during the transition from the first phase to the second one

Figure 15 shows the position of the vehicle with respect to the inertial frame and its distance from the seafloor. It's possible to easily distinguish the two mission phases. In particular, the vehicle reaches the desired position at the end of the first phase. Then, the vehicle performs the landing: its altitude decreases down to zero.



(a) Position x,y,z of the vehicle



(b) Altitude of the vehicle with respect to the seafloor

Figure 15: Safe waypoint navigation and landing actions

3 Exercise 3: Improve the "Landing" Action

3.1 Adding an alignment to target control objective

If we use the landing action, there is no guarantee that we land in front of the nodule/rock. We need to add additional constraints to make the vehicle face the nodule. The position of the rock is contained in the variable `rock_center`. 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.

Goal: Add an alignment task between the longitudinal axis of the vehicle (x axis) and the nodule target. 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.

3.1.1 Q1: Report the unified hierarchy of tasks used and their priorities. Comment the behaviour.

The goal of this exercise is achieved through two phases: the UVMS performs safe waypoint navigation and then the landing action.

The first phase corresponds to *safe waypoint navigation* in order to reach the target position and the corresponding hierarchy of objectives is:

1. Vehicle horizontal attitude (inequality, safety);
2. Vehicle position control task (inequality, action-defining), which controls both control on position and orientation.

In the second phase, the *landing* is achieved using the following hierarchy of objectives (highest to lowest priority objective):

1. Vehicle horizontal attitude (inequality, safety);
2. Vehicle longitudinal alignment to the nodule (inequality, operational prerequisite);
3. Vehicle altitude (equality, action-defining).

The order is motivated by the fact that we want to land maintaining the vehicle parallel to the floor (considering the floor horizontal). The order of the other two objectives is not relevant because the two tasks are not in conflict. Indeed, the vehicle altitude objective controls the z axis, while the longitudinal alignment works on the horizontal plane.

The longitudinal alignment objective consists in aligning the vehicle's longitudinal axis (x-axis) to a particular direction. An alignment with the object to be manipulated (i.e. the rock) could be very convenient to avoid having the arm assumed weird configurations, and also to avoid moving too much the center of mass. This objective is a prerequisite for the execution of the landing action. It requires to satisfy the following *inequality* objective:

$$\|\boldsymbol{\zeta}\| \leq \zeta_M$$

where ζ is the misalignment vector between the vehicle's longitudinal axis and the desired direction (vehicle-nodule).

3.1.2 Q2: What is the Jacobian relationship for the Alignment to Target control task? How was the task reference computed?

The goal of this exercise is to add the alignment task between the longitudinal axis of the vehicle and the nodule target. To fulfill this purpose, we implemented rigorous methodological steps.

The two vectors involved are:

- the x axis of the vehicle, that for simplifying the notations we will call:

$$\boldsymbol{a} := {}^{v}\boldsymbol{i_v}$$

- the unit vector obtained by projecting on the inertial horizontal plane the vector joining the vehicle frame to the nodule frame. To simplify the notation, we will call it $\boldsymbol{b}$ in the following. $\boldsymbol{b}$ is computed starting from the distance vector joining the vehicle to the nodule $\boldsymbol{r}_{v,nodule}$ and projecting it on the inertial x-y plane (by the matrix P_{xy}):

$$^{v}projection\_\boldsymbol{r}_{v,nodule} = \boldsymbol{P}_{xy} *^{v} \boldsymbol{r}_{v,nodule}, \quad \boldsymbol{P}_{xy} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

  hence the corresponding unit vector is:

$$\boldsymbol{b} := \frac{^{v}projection\_\boldsymbol{r}_{v,nodule}}{\|^{v}projection\_\boldsymbol{r}_{v,nodule}\|}$$

Then, we used the `ReducedVersorLemma` function in order to obtain the misalignment vector between the two vectors:

$$\boldsymbol{\rho} = \boldsymbol{n}\theta$$

The vectors defined so far are better illustrated in Figure 16.

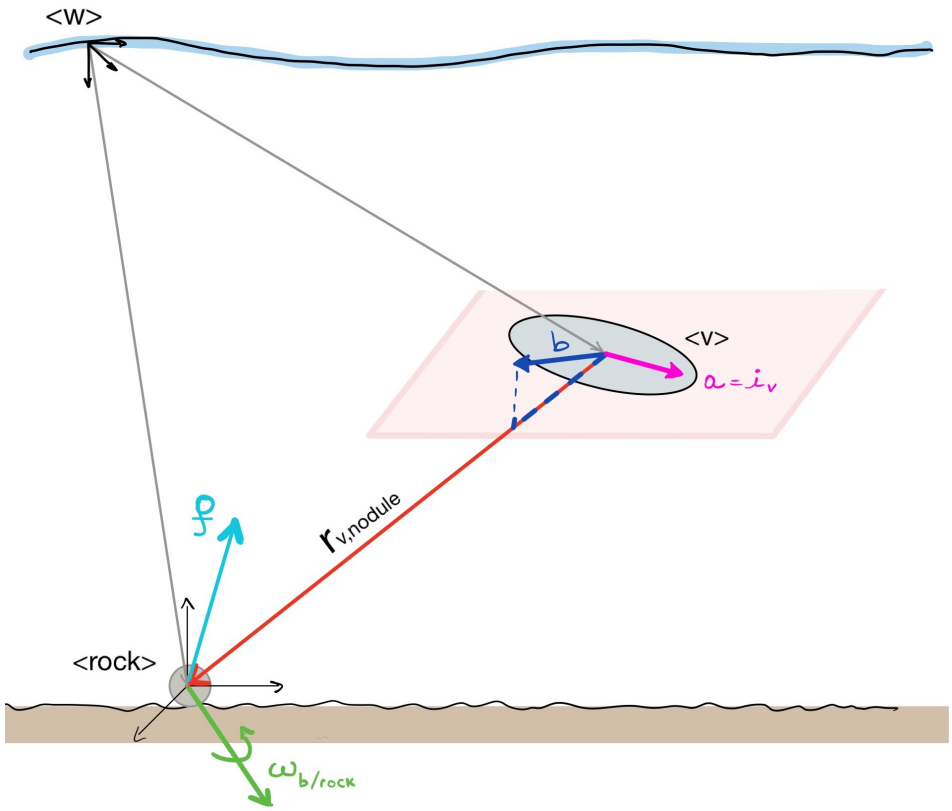


Figure 16: The UVMS and the vectors involved in the alignment to target control task

The **task reference** follows the *basic proportional law*:

$$\dot{\tilde{\boldsymbol{x}}} = \lambda \boldsymbol{\rho} = \lambda \boldsymbol{n} \theta \tag{2}$$

such that the *rotation vector* $\boldsymbol{\rho}$ between the two considered vectors will converge to 0.

To evaluate the Jacobian matrix, we have to compute the **time derivative** of $\boldsymbol{\rho}$ with respect to an observer α:

$$D_\alpha \boldsymbol{\rho} = \boldsymbol{n}\dot{\theta} + \theta D_\alpha \boldsymbol{n} = \tag{3}$$
$$= (\boldsymbol{n}(\boldsymbol{n}\cdot))\boldsymbol{\omega_{b/a}} + N_\alpha(\theta)(\boldsymbol{\omega_{b/\alpha}}, \boldsymbol{\omega_{a/\alpha}})$$

where $\boldsymbol{N}_\alpha(\theta)$ depends on two separate terms $\boldsymbol{\omega_{b/\alpha}}$ and $\boldsymbol{\omega_{a/\alpha}}$.

If we choose either the observer α sit on $\boldsymbol{a}$ or on $\boldsymbol{b}$, the equation 3 can be rewritten as:

$$D_a \boldsymbol{\rho} = (\boldsymbol{n}(\boldsymbol{n}\cdot))\boldsymbol{\omega_{b/a}} + \boldsymbol{N}_a(\theta)\boldsymbol{\omega_{b/a}} \tag{4}$$

where

$$\boldsymbol{\omega_{b/a}} = \boldsymbol{\omega_{b/\alpha}} - \boldsymbol{\omega_{a/\alpha}}$$

As already mentioned, our goal is to bring the misalignment vector to zero.
This goal can be achieved in two different ways according to the desired behaviour of the misalignment vector during the convergence to zero:

1. control on $\|\boldsymbol{\rho}\|$ using only one degree of freedom;
2. control on $\boldsymbol{\rho}$ imposing three degrees of freedom.

If we are not interested in the behaviour of the misalignment vector while it is going to zero, the goal can be easily achieved imposing only **one degree of freedom**. It is enough to control the **norm of the misalignment vector** imposing the basic proportional law:

$$\dot{\tilde{x}} = -\lambda \|\boldsymbol{\rho}\| = -\lambda\theta$$

If we want that the misalignment vector goes to zero, while the vehicle is going straight with respect to an observer α sit on a particular frame, the goal can be achieved imposing three degree of freedom.

Control on norm of misalignment vector

In this case, we can omit the reference to an observer in the equation 3:

$$D\boldsymbol{\rho} = \boldsymbol{n}\dot{\theta} + \theta \boldsymbol{N}(\theta) \tag{5}$$

By scalar pre-multiplying both sides of equation 5 for $\boldsymbol{n}$, we obtain:

$$\boldsymbol{n} \cdot D\boldsymbol{\rho} = \boldsymbol{n} \cdot \boldsymbol{n}\dot{\theta} + \boldsymbol{n} \cdot \boldsymbol{N}(\theta) \tag{6}$$

We know a priori that the the product $\boldsymbol{n} \cdot \boldsymbol{N}(\theta)$ is zero and the above equation can be simplified:

$$\boldsymbol{n} \cdot D\boldsymbol{\rho} = \boldsymbol{n} \cdot \Big((\boldsymbol{n}(\boldsymbol{n}\cdot))\boldsymbol{\omega_{b/a}}\Big) \tag{7}$$

where $\boldsymbol{n}(\boldsymbol{n}\cdot)$ performs the projection of the vector along the direction of $\boldsymbol{n}$.
This operation can be implemented using the projection matrix P_n:

$$\boldsymbol{P}_n = \boldsymbol{n}\boldsymbol{n}^{\boldsymbol{T}} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \begin{bmatrix} n_x & n_y & n_z \end{bmatrix} = \begin{bmatrix} n_x^2 & n_x n_y & n_x n_z \\ n_y n_x & ny^2 & n_y n_z \\ n_z n_x & n_z n_y & n_z^2 \end{bmatrix} \tag{8}$$

The equation 7 becomes:

$$\boldsymbol{n} \cdot D\boldsymbol{\rho} = \boldsymbol{n} \cdot \boldsymbol{P}_n \ \boldsymbol{\omega_{b/a}} \tag{9}$$

We can rewrite the scalar product between $\boldsymbol{n}$ and the projection matrix, obtaining:

$$\boldsymbol{n} \cdot D\boldsymbol{\rho} = \boldsymbol{n}^{\boldsymbol{T}} \boldsymbol{P}_n \ \boldsymbol{\omega_{b/a}} \tag{10}$$

where

$$\boldsymbol{\omega_{b/a}} = \boldsymbol{\omega_{b/w}} - \boldsymbol{\omega_{a/w}}$$

In particular, the angular velocity of the longitudinal axis of the vehicle with respect to the world is the velocity of the vehicle itself:

$$\boldsymbol{\omega_{a/w}} = \boldsymbol{\omega_{v/w}} \tag{11}$$

Instead, the angular velocity of the projection of the vector joining the vehicle frame to the nodule frame can be computed as follows:

$$\boldsymbol{\omega_{b/w}} = \frac{\left\|\boldsymbol{v_{b/w}}\right\|}{\left\|{}^{v}\boldsymbol{r}_{v,nodule}\right\|} \left(\frac{{}^{v}\boldsymbol{r}_{v,nodule}}{\left\|{}^{v}\boldsymbol{r}_{v,nodule}\right\|} \wedge \frac{\boldsymbol{v_{b/w}}}{\left\|\boldsymbol{v}_{b/w}\right\|} \right) \tag{12}$$

However, we know that $\boldsymbol{v}_{b/w} = -\boldsymbol{v}_{v/w}$. We can immediately notice that the equation 12 can be simplified as:

$$\boldsymbol{\omega_{b/w}} = \frac{-1}{\left\|\boldsymbol{r}_{v,nodule}\right\|^2} \left({}^{v}\boldsymbol{r}_{v,nodule} \wedge \boldsymbol{v_{v/w}} \right)$$

The wedge product can be substituted by the skew symmetric matrix and the distance vector can be projected on the horizontal plane performing a matrix multiplication ($\boldsymbol{P}_{xy}$):

$$\boldsymbol{\omega_{b/w}} = \frac{-1}{\left\|\boldsymbol{r}_{v,nodule}\right\|^2} \left[\boldsymbol{r}_{v,nodule}\right]_{\mathrm{x}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{v_{v/w}}$$

Substituting this calculation in the equation 10, we obtain:

$$\boldsymbol{n} \cdot D\boldsymbol{\rho} = \boldsymbol{n}^{\boldsymbol{T}} P_n \left(\frac{-1}{\left\|\boldsymbol{r}_{v,nodule}\right\|^2} \left[\boldsymbol{r}_{v,nodule}\right]_{\mathrm{x}} \boldsymbol{P}_{xy} \ \boldsymbol{v_{v/w}} - \boldsymbol{\omega_{v/w}} \right) \tag{13}$$

Eventually, we compute the Jacobian for the longitudinal alignment to target control task as:

$$\boldsymbol{J}_{la} = \boldsymbol{n}^{\boldsymbol{T}} P_n \begin{bmatrix} \boldsymbol{0}_{3x7} & \frac{-1}{\left\|\boldsymbol{r}_{v,nodule}\right\|^2} \left[\boldsymbol{r}_{v,nodule}\right]_{\mathrm{x}} \boldsymbol{P}_{xy} & -\boldsymbol{I}_{3x3} \end{bmatrix} \tag{14}$$

Three degrees of freedom control

The desired behaviour (equation 2) can be substituted in the equation 4:

$$\boldsymbol{n}\lambda\theta = (\boldsymbol{n}(\boldsymbol{n}\cdot))\boldsymbol{\omega_{b/a}} + \boldsymbol{N}_a(\theta)\boldsymbol{\omega_{b/a}}$$

The solution that solves at best the first problem is **unique** and it is the one aligned with $\boldsymbol{n}$:

$$\boldsymbol{\omega_{b/a}} = \boldsymbol{n}\lambda\theta$$

This justify the fact that we can neglect $\boldsymbol{N}_a(\theta)$ if we sit on $\boldsymbol{a}$ or $\boldsymbol{b}$.
The term $\boldsymbol{\omega_{b/a}}$ is already known from the previous computation:

$$\frac{-1}{\left\|\boldsymbol{r}_{v,nodule}\right\|^2} \left[\boldsymbol{r}_{v,nodule}\right]_{\mathrm{x}} \boldsymbol{P}_{xy} \ \boldsymbol{v_{v/w}} - \boldsymbol{\omega_{v/w}} = \boldsymbol{n}\lambda\theta \tag{15}$$

From the previous equation we can individuate the Jacobian for the alignment to target control task as:

$$\boldsymbol{J}_{la} = \begin{bmatrix} \boldsymbol{0}_{3x7} & \frac{-1}{\left\|\boldsymbol{r}_{v,nodule}\right\|^2} \left[\boldsymbol{r}_{v,nodule}\right]_{\mathrm{x}} \boldsymbol{P}_{xy} & -\boldsymbol{I}_{3x3} \end{bmatrix} \tag{16}$$

Task reference

The task reference is implemented through the *basic proportional law*:

$$\dot{\bar{x}}_{longAlign} = -\lambda(\zeta - \bar{\zeta})$$

where ζ is the misalignment vector between the unit vector $\boldsymbol{b}$ joining the vehicle frame to the nodule frame and vector x-axis of the vehicle frame (vector $\boldsymbol{a}$). The misalignment vector is computed calling the function `ReducedVersonLemma` and passing as inputs the vectors $\boldsymbol{b}$ and $\boldsymbol{a}$.

Activation functions

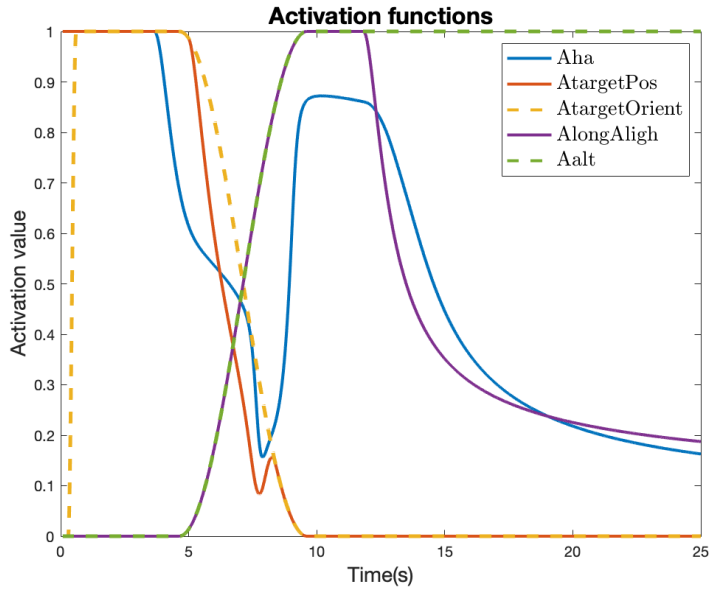
Since the longitudinal alignment objective is bounded above, the corresponding activation function $A_{longAlign}$ will be an *increasing bell-shaped* function (see section 1.1.1 for the definition).
In particular, the structure of $\boldsymbol{A}_{longAlign} \in \mathbb{R}^{3x3}$ is:

$$\boldsymbol{A}_{longAlign} = \boldsymbol{I}_{3x3} * s_{longAlign}(\boldsymbol{\zeta})$$

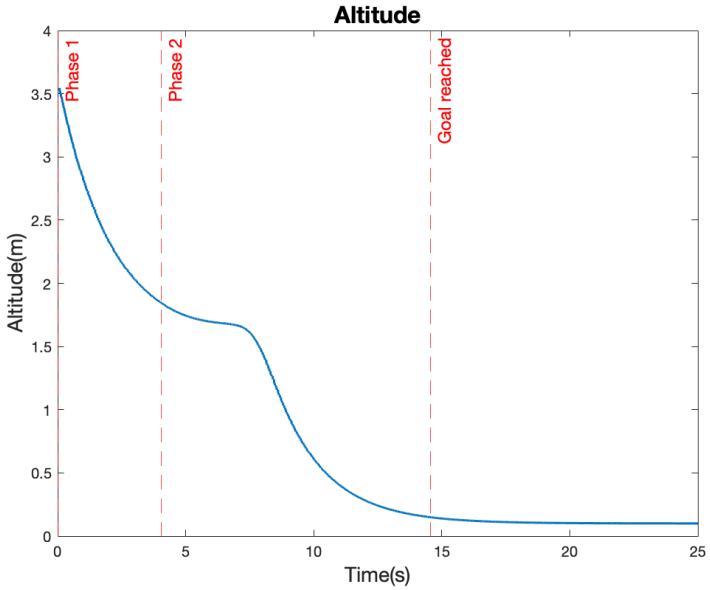
where $s_{longAlign}(\boldsymbol{\zeta})$ is the smooth function, which depends on the misalignment vector $\boldsymbol{\zeta}$.

The next figures show the most significant plots of the implemented sequence of tasks, highlighting the different phases of the mission.

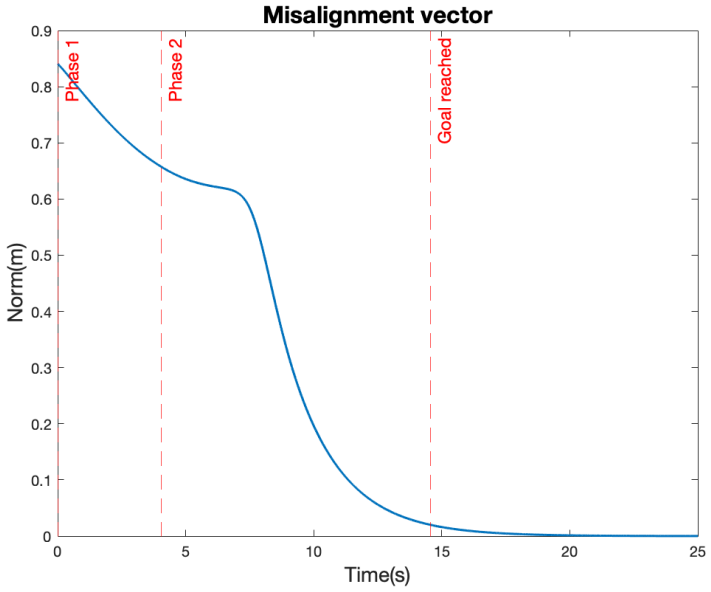
Regarding the *second phase*, we decided to report the trend of the altitude from the seafloor and **norm of misalignment vector** between the longitudinal axis of the vehicle and the projection on the inertial horizontal plane of the unit vector joining the vehicle frame to the nodule frame. The plots prove that both are reduced down to zero.



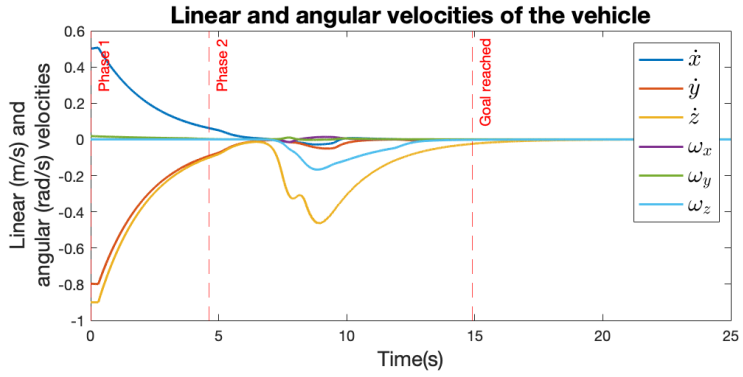
(a) Activation functions



(b) Altitude of the vehicle from the seafloor



(c) Norm of the misalignment vector between the longitudinal axis of the vehicle and the goal direction.



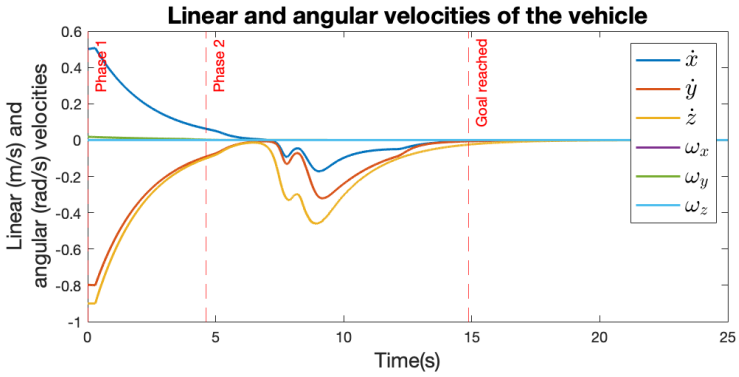
(d) Linear and angular velocities of the vehicle

Figure 17: "Landing" Action with alignment toward the nodule ($\lambda_{la} = 0.5$)

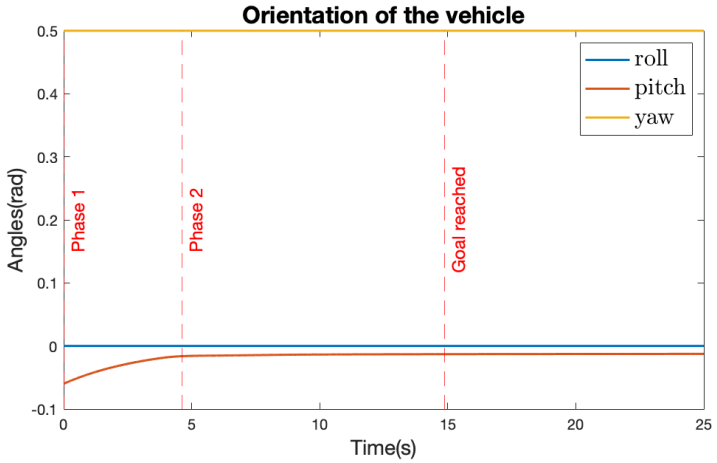
If we enable the control of the linear part only, namely the corresponding Jacobian is:

$$\boldsymbol{J}_{la} = \begin{bmatrix} \mathbf{0}_{3x7} & \frac{-1}{\|\boldsymbol{r}_{v,nodule}\|^2} \left[\boldsymbol{r}_{v,nodule}\right]_{\mathrm{x}} \boldsymbol{P}_{xy} & \mathbf{0}_{3x3} \end{bmatrix}$$

the vehicle translates until it reaches the alignment to the nodule without changing its orientation (Figure 19b). Figure 19a shows that in the second phase the vehicle acquires a linear velocity ($\dot{x}, \dot{y}, \dot{z} \neq 0$), while the angular part is null ($\omega_x, \omega_y, \omega_z = 0$).



(a) Linear and angular velocities of the vehicle



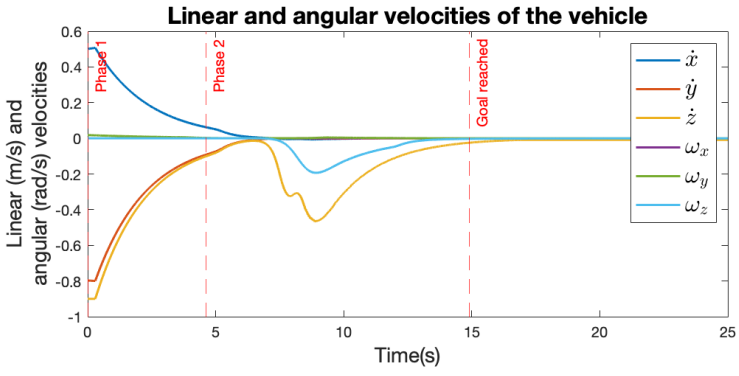
(b) Orientation of the vehicle w.r.t $< w >$

Figure 18: "Landing" Action with alignment toward the nodule keeping the vehicle orientation fixed

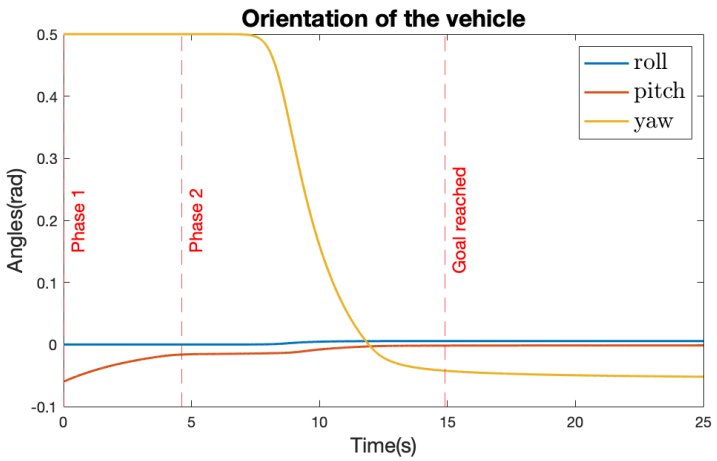
On the other hand, if we control only the orientation of the vehicle, namely the corresponding Jacobian is

$$\boldsymbol{J}_{la} = \begin{bmatrix} \mathbf{0}_{3x7} & \mathbf{0}_{3x7} & -\boldsymbol{I}_{3x3} \end{bmatrix} \tag{17}$$

the vehicle keeps its position fixed and rotates around its axes for pointing toward the nodule. In the case of the exercise, the kernel space of the horizontal attitude task (higher priority objective) does not allow the variation of roll and pitch angles. The vehicle performs a rotation around the z-axis (see in Figure 18b), namely it acquires an angular velocity $\omega_z \neq 0$ (Figure 18a, note that $\dot{z} \neq 0$ is caused by the landing action).



(a) Linear and angular velocities of the vehicle

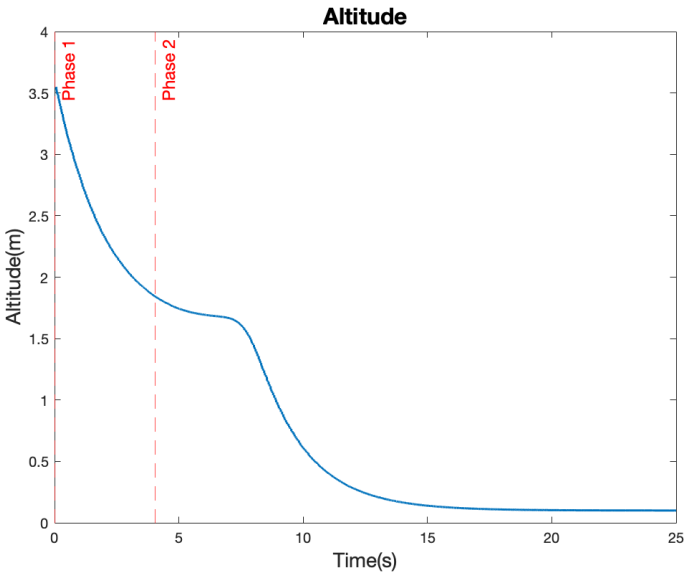


(b) Orientation of the vehicle w.r.t $< w >$

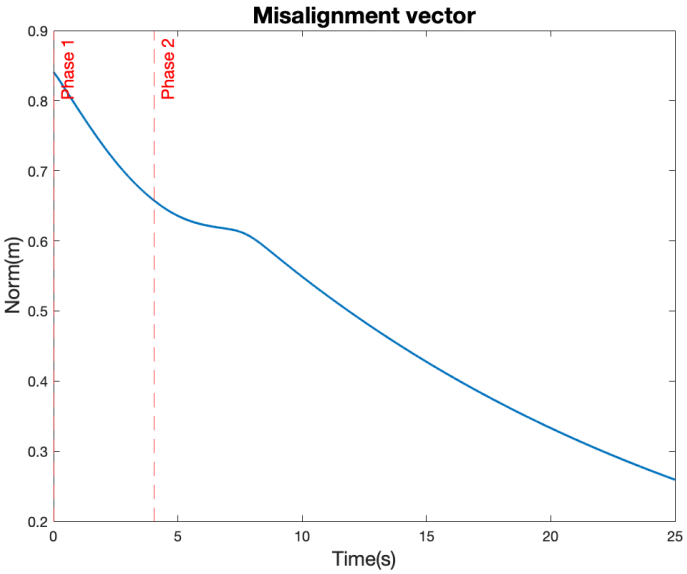
Figure 19: "Landing" Action with alignment toward the nodule keeping the vehicle position fixed

3.1.3 Q3: Try changing the gain of the alignment task. Try at least three different values, where one is very small. What is the observed behaviour?

The gain determines how fast the misalignment vector converges to zero. We consider the goal reached when both the two tasks are completed. If the gain is set to a very small value (i.e., $\lambda = 0.05$), the vehicle land to the sea floor (vehicle altitude objective completed) without having reached the desired alignment yet. So, the vehicle aligns to the nodule crawling across the seafloor. In fact, in Figure 20 the altitude of the vehicle is zero, but the misalignment vector has not reached the zero yet.
Instead, with higher values of the gain (i.e., $\lambda = 0.5$ or $\lambda = 1$), the two phases are accomplished as illustrated in Figures 17 and 21.

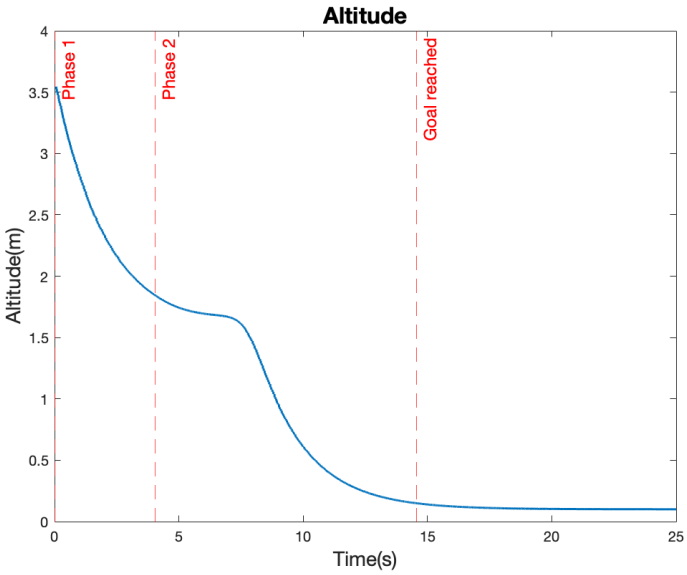


(a) Altitude of the vehicle from the seafloor

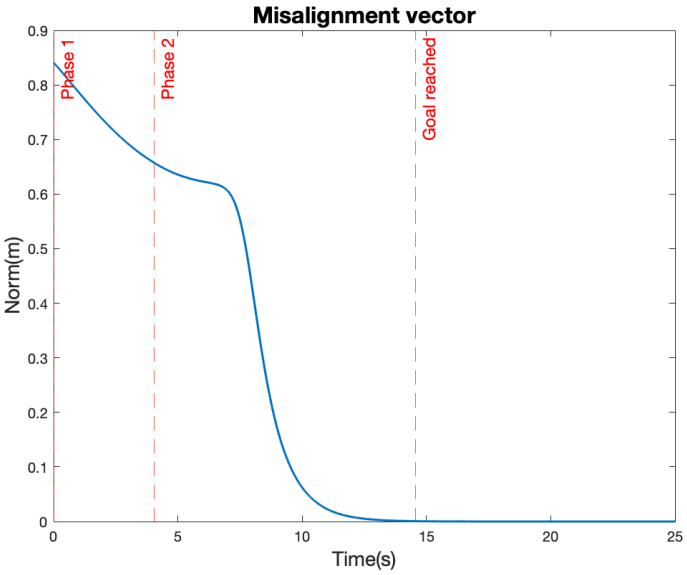


(b) Misalignment vector between the longitudinal axis of the vehicle and the goal direction

Figure 20: Gain of longitudinal alignment task $\lambda_{la} = 0.05$



(a) Altitude of the vehicle from the seafloor

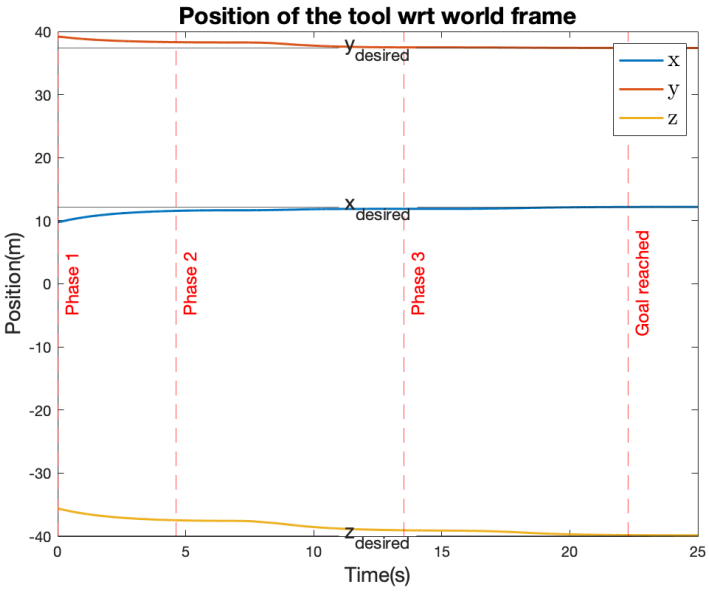


(b) Misalignment vector between the longitudinal axis of the vehicle and the goal direction

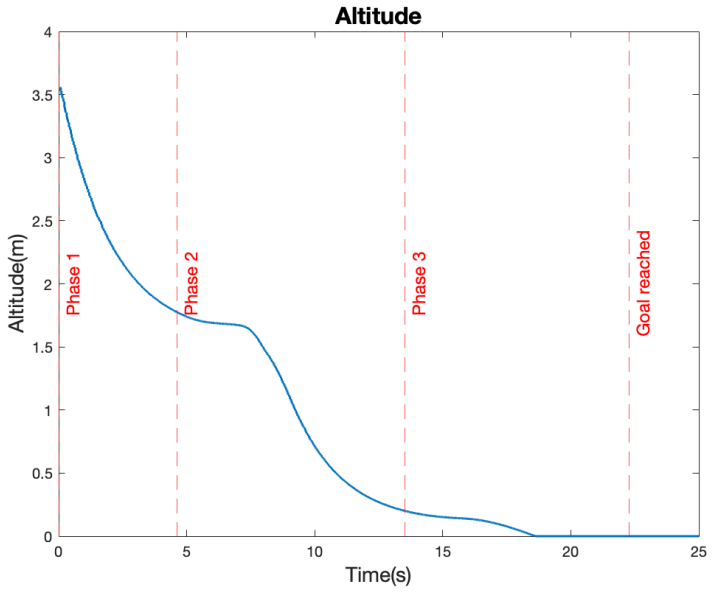
Figure 21: Gain of longitudinal alignment task $\lambda_{la} = 1$

3.1.4 Q4: 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.

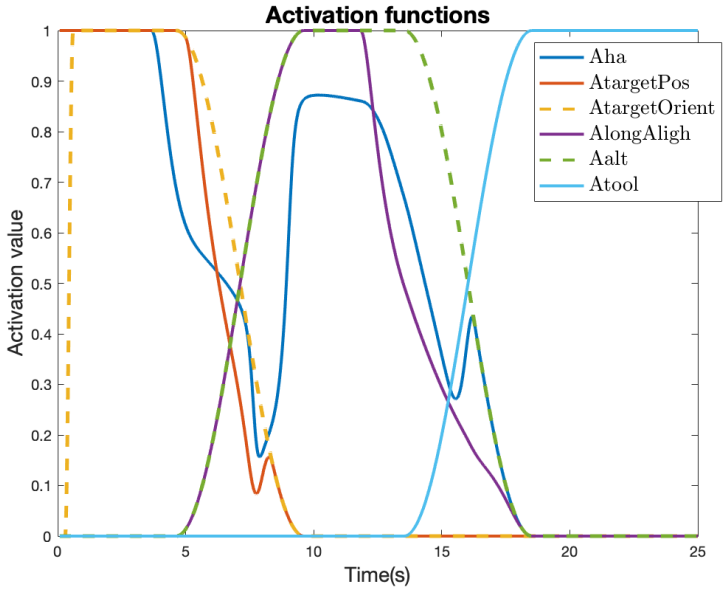
We add a *third phase* considering active only the Tool-frame position control task. After the landing is accomplished (second phase), the end-effector tries to reach the nodule. However, if we set only this objective in the third phase, the vehicle is free to move, so it is not constrained to keep the reached position. We notice that the vehicle tends to tilt. In fact, the roll and pitch angles starts increasing and the activation function of the tool is active to maintain the desired position of the tool (see Figure 22).



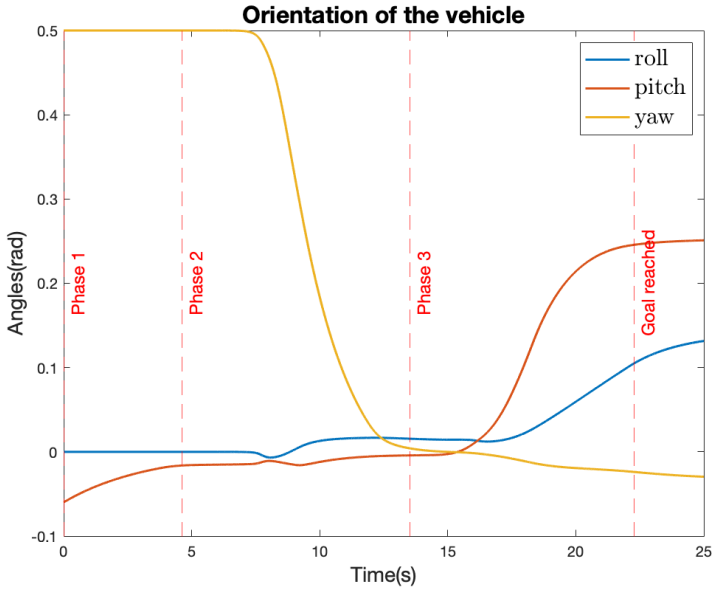
(a) Position of the tool



(b) Distance from the seafloor



(c) Activation functions

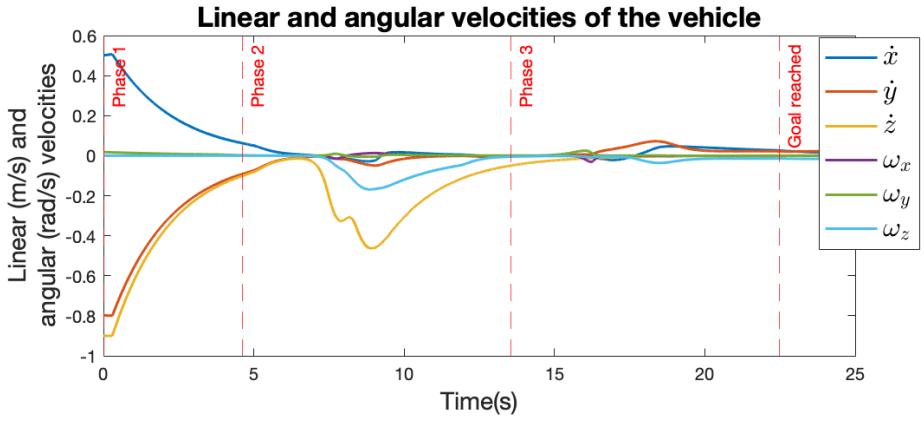


(d) Orientation of the vehicle

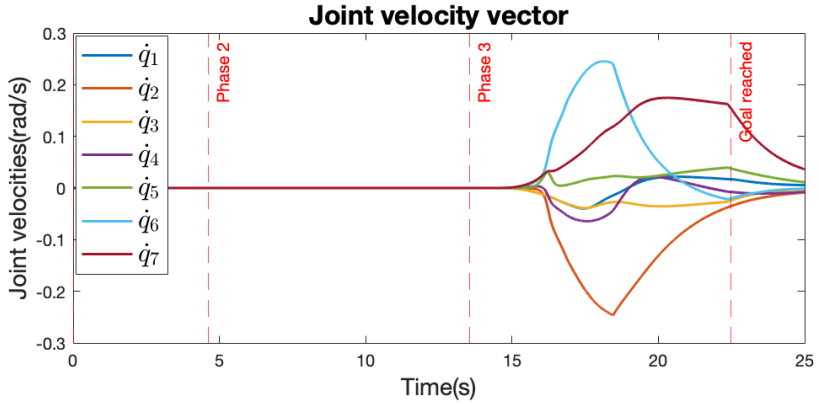
Figure 22: Results obtained when in the third phase only the tool-frame position control task is present

If instead we keep active all objectives of the second phase during the third phase, the vehicle

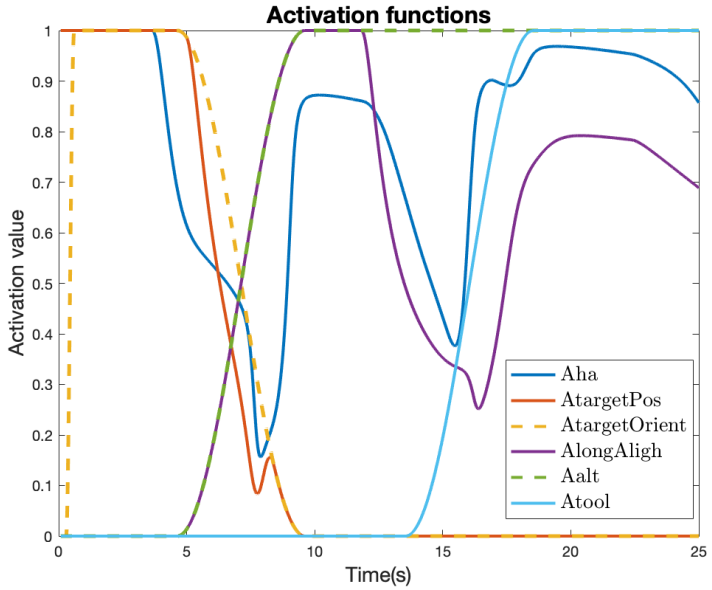
translates on the floor until the tool reaches the desired position. From figure 23a we can notice that, in the third phase, the vehicle acquires a linear velocity $\dot{x}$ (blue line) and $\dot{y}$ (red line) greater than zero.



(a) Velocities of the vehicle



(b) Joint velocities



(c) Activation functions

Figure 23: End effector action

4 Exercise 4: Implementing a Fixed-base Manipulation Action

4.1 Adding non-reactive tasks

To manipulate as a fixed based manipulator, we need to constraint the vehicle to not move, otherwise the tool frame position task will make the vehicle move.

Goal: Add a constraint task that fixes the vehicle velocity to zero. Land on the seafloor. Try reaching the rock position with the end-effector, and observe that the vehicle does not move.

4.1.1 Q1: Report the unified hierarchy of tasks used and their priorities. At which priority level did you add the constraint task?

To implement this exercise, we left unchanged the first and the second phase of the previous exercise. We concentrate only on the third phase to which we add the vehicle motion minimization objective in order to overcome the problem we have just encountered in section 3.1.4.
The hierarchy of objectives in this third phase (highest to lowest priority objective) is the following:

1. End-effector/Tool-frame linear position control (equality, action-defining);
2. End-effector/Tool-frame angular position control (equality, action-defining);
3. Vehicle motion minimization (equality, optimization).

The third phase is accomplished when the tool reaches the desired configuration and the tool reached the rock (within a threshold). The hierarchy follows a specific order: first a mission-oriented task and then an optimization objective.

The Tool-frame position objective requires that the tool frame $< t >$, rigidly attached to the end-effector space, converges to a given goal frame $< g >$. In other words, the following two *equality* objectives must be eventually satisfied:

$$\boldsymbol{r} = 0,\ \boldsymbol{\vartheta} = 0$$

where $\boldsymbol{r}$ is the position error and $\boldsymbol{\vartheta}$ the orientation error between the tool and goal frames.

The purpose of the vehicle motion minimization objective is to make the vehicle roughly stationary during the grasping and the manipulation, unless its movement is strictly needed. This is a consequence of the fact that, usually, the control performances of the vehicle are fairly worse than those of the arms. The vehicle motion minimization is an *equality* objective, which has to satisfy the following condition:

$$\boldsymbol{v}_v = \boldsymbol{v}_0$$

Activation functions

Exercise 4 adds two new objectives whose corresponding activation functions are:

- $\boldsymbol{A}_t$: activation matrix (6x6) of the tool-frame position and orientation objective;
- $\boldsymbol{A}_{fixvehicle}$: activation matrix (6x6) of the vehicle motion minimization objective.

Since both the objectives are *equalities*, the activation functions must be always activated, namely $\boldsymbol{A}_t \equiv \boldsymbol{I}_{6x6}$ and $\boldsymbol{A}_{fixvehicle} \equiv \boldsymbol{I}_{6x6}$.

4.1.2 Q2: What is the Jacobian relationship for the Vehicle Null Velocity task? How was the task reference computed?

The Jacobian matrix takes into account only the vehicle; in particular it contains an identity matrix in order to maintain the same position and orientation:

$$\boldsymbol{J}_{fixVehicle} = \begin{bmatrix} \mathbf{0}_{6x7} & \boldsymbol{I}_{6x6} \end{bmatrix}$$

Since we would like to keep the same position, the reference is set zero $\dot{\tilde{\boldsymbol{x}}} = [\mathbf{0}_{6x1}]$.

Figure 24 reports the activation functions of the active tasks in the appropriate phases.

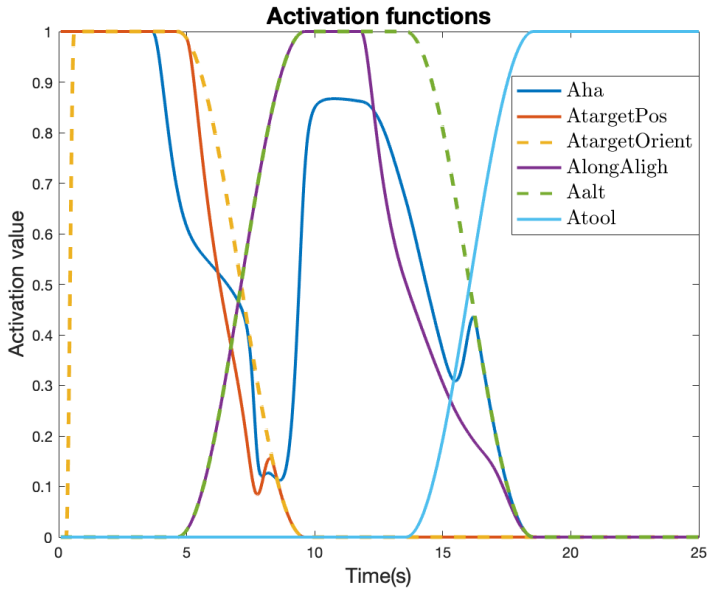


Figure 24: Activation functions

To underline the behaviour of the UVMS during the mission, we decided to report the configuration and the velocities of both the arm and the vehicle. During the first and second phase, the vehicle changes its position keeping the arm fixed. Then, in the third phase, the arm starts changing its configuration while the vehicle is constrained to not move.

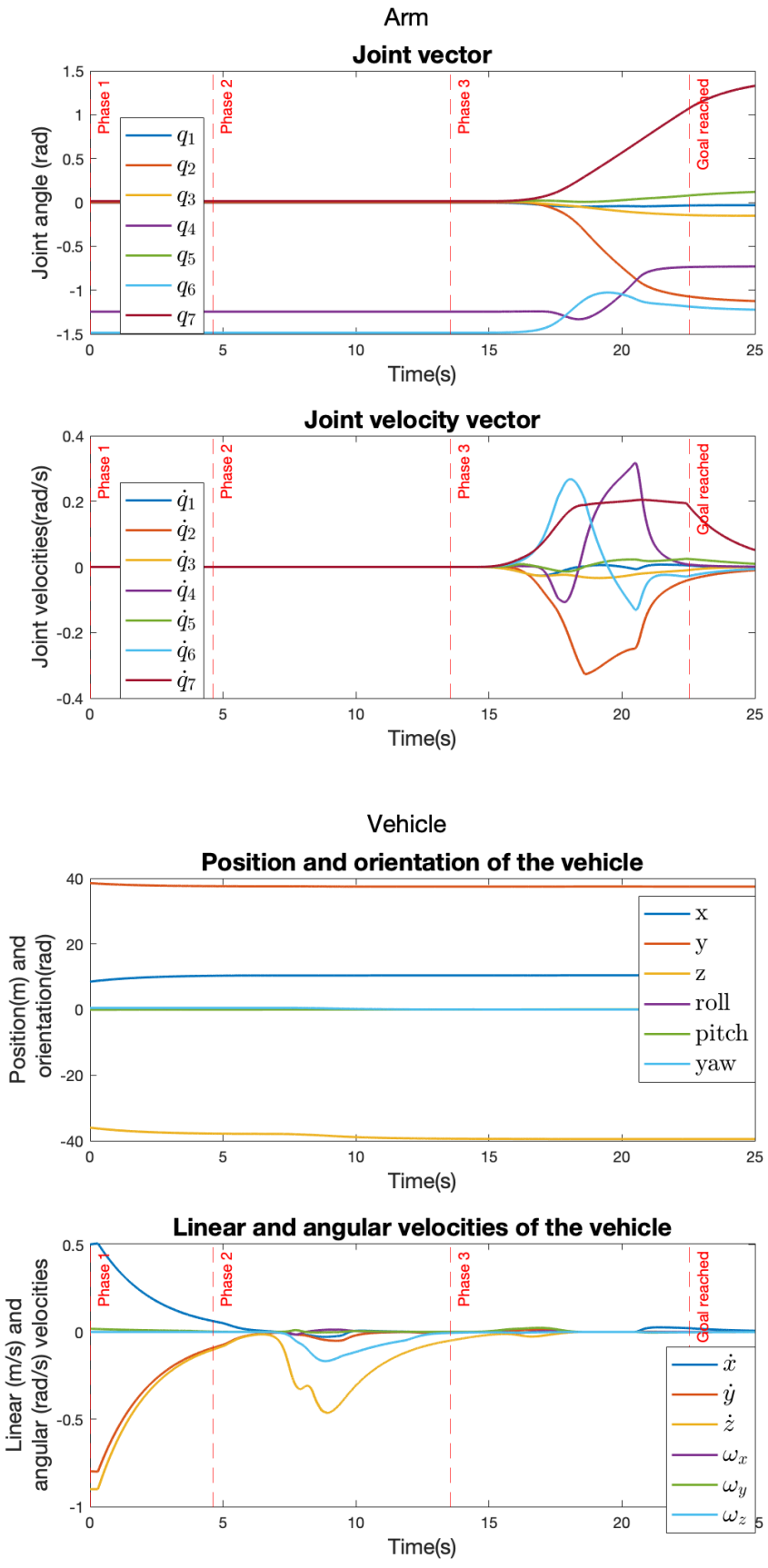
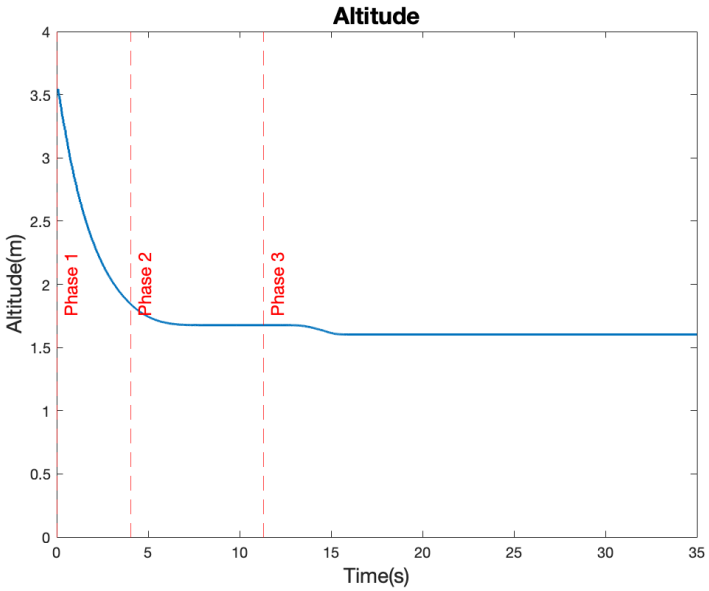


Figure 25: Configuration and velocities of the arm and the vehicle

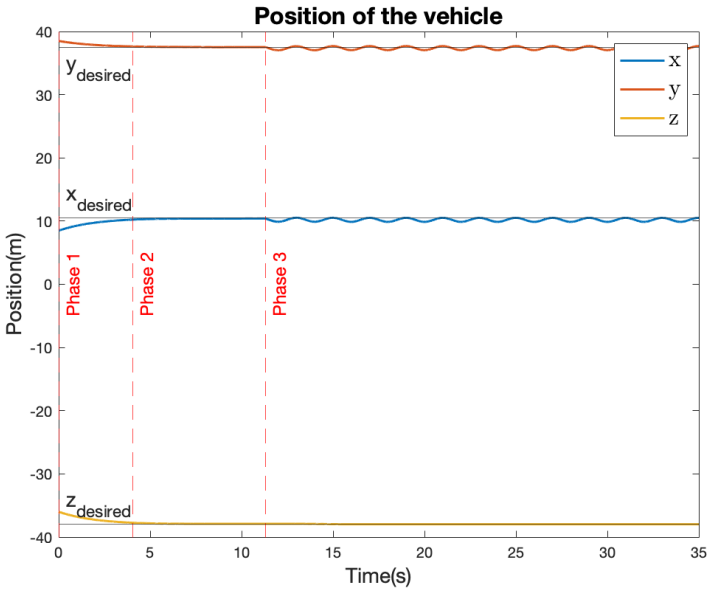
4.1.3 Q3: Suppose the vehicle floating, i.e. not landed on the seafloor. What would happen, if due to currents, the vehicle moves?

The implemented task acts in the loop as optimization task: once entered in the third phase, between all possible solutions the algorithm selects the one which keeps the vehicle still ($\boldsymbol{v}_v = 0$) and just moves the arm. The current is an external factor, therefore it can affect the vehicle velocity. The result is a displacement of the vehicle. At each iteration, the task constrains the position to the value caused by the external disturbance.

To prove the statement, we remove the landing task in the second phase in such a way to allow floating manipulation and we add a sinusoidal velocity disturbance (before integration) acting on the vehicle. Figure 26 shows that in the third phase the vehicle has a velocity different from zero and changes its x and y position even if the objective to fix the UVMS is active.



(a) Altitude of the vehicle from the seafloor.



(b) Activation functions

Figure 26: Floating manipulation in presence of current.

4.2 Adding a joint limit task

Let us now constrain the arm with the actual joint limits. The vector variables `uvms.jlmin` and `uvms.jlmax` contain the maximum and minimum values respectively.

Goal: Add a joint limits avoidance task. Land on the seafloor. Try reaching the rock position with the end-effector, and observe that the vehicle does not move and that all the joints are within their limits.

4.2.1 Q1: Report the unified hierarchy of tasks used and their priorities. At which priority level did you add the constraint task?

The hierarchy of objectives in the *third phase* (highest to lowest priority objective) is the following:

1. Arm joint limits (inequality, safety);
2. Arm manipulability (inequality, safety);
3. Tool-frame linear position control task (equality, action-defining);
4. Tool-frame angular position control task (equality, action-defining);
5. Vehicle motion minimization (equality, optimization).

The hierarchy of objectives follows a natural objectives: first safety objective, then action-defining and optimization objectives. The manipulator must operate within its joint limits, which means having the following inequality control objectives fulfilled:

$$q_{i,m} \leq q_i \leq q_{i,M}, \quad \text{with} \quad i = 1, ..., 7$$

where q_i is the i-th joint variable and $q_{i,m}$, $q_{i,M}$ are the lower and higher joint bounds.

Not all the arm configurations are the same from the dexterity point of view. In certain configurations, the control effort to achieve a desired end-effector velocity is greater than in others. For these reasons, the arm must maintain a certain minimum level of dexterity, to be able to perform the manipulation tasks. If the manipulability measure $\mu \triangleq \sqrt{\boldsymbol{J}\boldsymbol{J}^T}$ is small, we are close to a kinematic singularity, i.e. a configuration where the determinant is zero. In its vicinities, the control action $\dot{\boldsymbol{q}}$ tends to infinity for certain $\dot{\boldsymbol{x}}$. The objective thus can be stated in the following terms:

$$\mu > \mu_m$$

Hence, since the objective is a below bounded inequality, the activation function A_{mu} is a *decreasing bell shaped* function (see section 1.2.1 for the definition) of the manipulability measure μ.

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

The Jacobian matrix takes into account only the arm:

$$\boldsymbol{J}_{jl} = \begin{bmatrix} \boldsymbol{I}_{7x7} & \boldsymbol{0}_{7x6} \end{bmatrix}$$

The task reference follows the *basic proportional law*:

$$\dot{\tilde{x}}_{jl_i} = -\lambda(q_i - \bar{q}_i)$$

where the desired configuration $\bar{q}_i$ is taken as the center between the lower limit `uvms.jlmin`(i) and the upper limit `uvms.jlmax`(i):

$$\bar{q}_i = \frac{\texttt{uvms.jlmax}(i) + \texttt{uvms.jlmin}(i)}{2}$$

The task is an *inequality* objective, therefore the activation function $\boldsymbol{A}_{jl}$ has not to be always active.

We implemented $\boldsymbol{A}_{jl}$ as a 7x7 *diagonal matrix* containing on the diagonal the sum of a decreasing $s_{dec}(q_i)$ and an increasing bell shaped function $s_{inc}(q_i)$.

In particular, each element of the diagonal $A_{jl}(i,i)$ depends on q_i (i-th joint of the arm with $i = 1, ..., 7$) and it is computed as follows:

$$A_{jl}(i,i) = \begin{cases} 0, & \text{if} \quad \texttt{uvms.jlmin(i)} + \Delta < q_i < \texttt{uvms.jlmax(i)} - \Delta \\ s_{dec}(q_i), & \text{if} \quad \texttt{uvms.jlmin(i)} < q_i < \texttt{uvms.jlmin(i)} + \Delta \\ s_{inc}(q_i), & \text{if} \quad \texttt{uvms.jlmax(i)} - \Delta < q_i < \texttt{uvms.jlmax(i)} \\ 1, & \text{if} \quad q_i < \texttt{uvms.jlmin(i)} \quad \text{or} \quad q_i > \texttt{uvms.jlmax(i)} \end{cases}$$

Figure 27 displays the activation function for the 6-th joint, namely $A_{jl}(6,6)$.

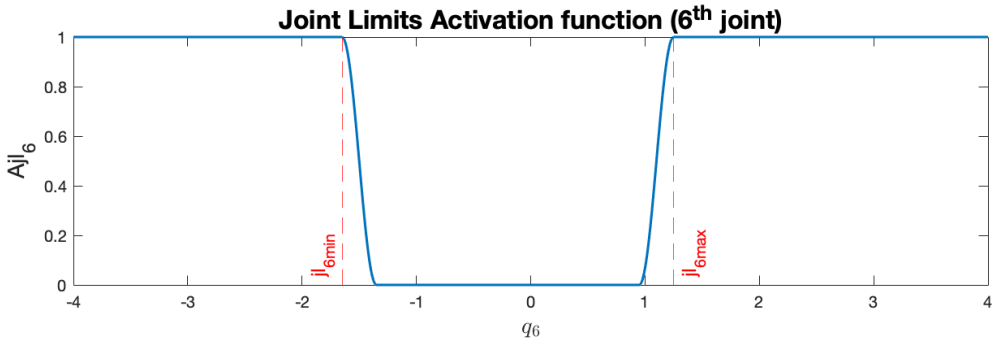
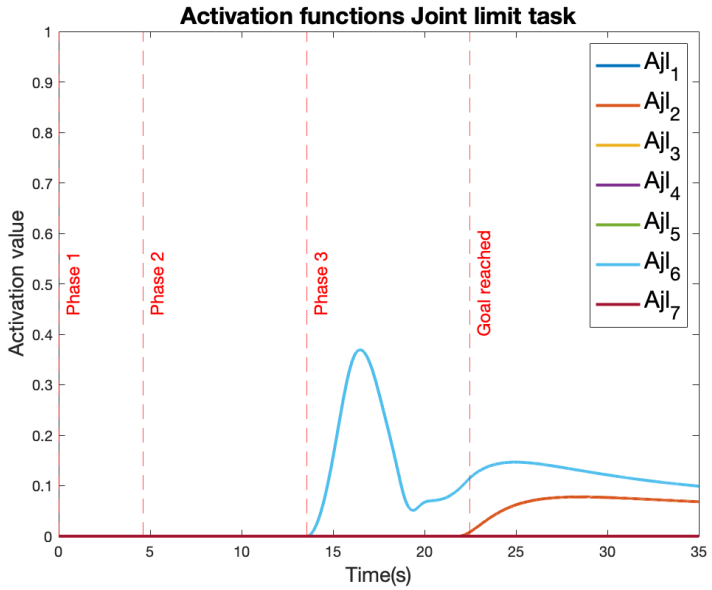


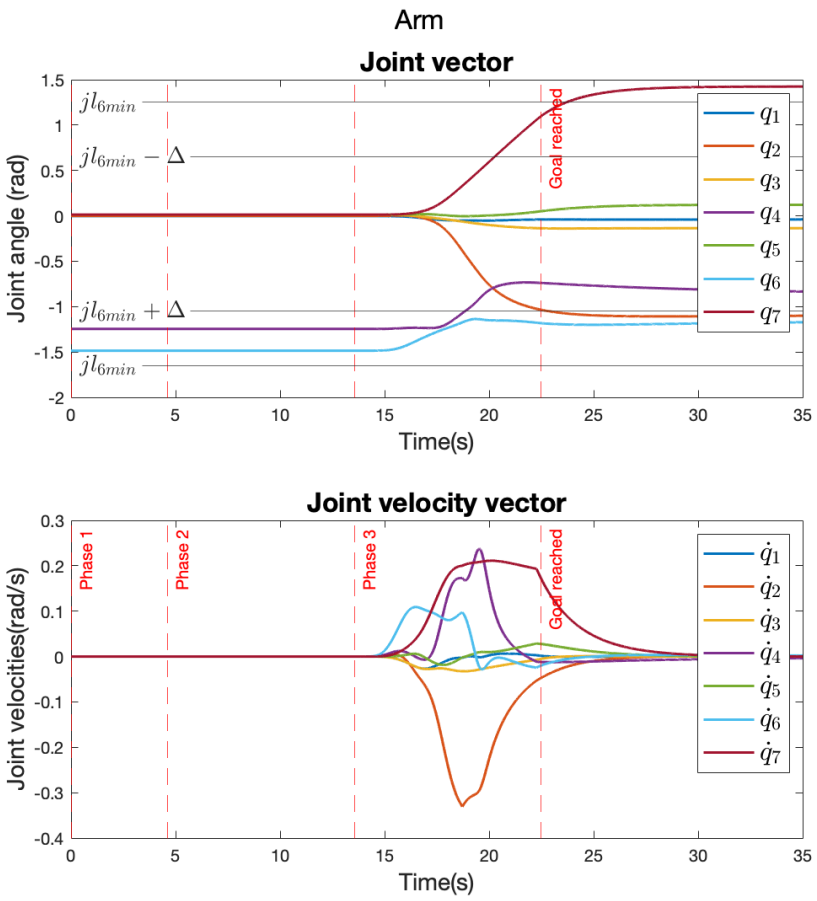
Figure 27: Joint limits activation function for 6^{th} joint

Figure 28 shows the activation functions of all joints of the arm. We can notice that only the activation functions of joint 2 and joint 6 are different from zero starting from the third phase from which the joint limits objective is active.

In particular, the activation of 6-th joint is demonstrated by the fact that the corresponding joint vector (light blue line) is within the interval $\texttt{uvms.jlmin(6)} < q_6 < \texttt{uvms.jlmin(6)} + \Delta$ (Figure 28b).



(a) Activation functions



(b) Joint vector and velocity vector of the arm

Figure 28: Joint limits task

5 Exercise 5: Floating Manipulation

5.1 Adding an optimization control objective

Use the DexROV simulation for this exercise.

The goal is to try to optimize the joint positions, if possible, to keep the first four joints in a "preferred shape", represented by the following vector

$$\begin{bmatrix} -0.0031 & 1.2586 & 0.0128 & -1.2460 \end{bmatrix}^\top$$

Goal: Add an optimization objective to keep the first four joints of the manipulator in the preferred shape. Observe the behaviour with and without the task.

5.1.1 Q1: Report the unified hierarchy of tasks used and their priorities. At which priority level did you add the optimization task?

The hierarchy of objectives in the *third phase* (highest to lowest priority objective) is the following:

1. Joint Limits (inequality, safety);
2. Arm manipulability (inequality, prerequisite);
3. Vehicle horizontal attitude (inequality, safety);
4. Tool-frame linear position control (equality, action-defining);
5. Tool-frame angular position control (equality, action-defining);
6. Arm preferred shape (inequality, optimization).

This exercise adds the objective aimed to have the arm maintaining a predefined desired shape. The purpose of this objective is to minimize arm movements during the vehicle movement, in such a way the vehicle can exhibit better control performances. Maintaining the arm in a preferred shape allows to perform repetitive tasks minimizing the internal motions and can also be used as an alternative method to ensure good dexterity. As optimization objective, this task has a lower priority than system safety objectives, hence it does not influence the mission. The arm preferred shape requires the following objective to be satisfied:

$$\|\boldsymbol{e}_q\| \leq e_{qM}$$

where $\boldsymbol{e}_q \triangleq \begin{bmatrix} \bar{q}_i - q_i \\ . \\ . \end{bmatrix}$ is the difference between the preferred and the actual joint vector.

Since this difference must be lower than the maximum value e_{qM}, the activation function of the arm preferred shape $\boldsymbol{A}_{opt}$ is a 7x7 diagonal matrix where i-th element on the diagonal is:

$$\boldsymbol{A}_{opt}(i,i) = \begin{cases} s_{inc}(\|e_q(i)\|), & \text{if } i = 1,2,3 \text{ or } 4 \\ 0 & \text{otherwise} \end{cases}$$

where $s_{inc}(\|\boldsymbol{e}_q\|)$ is the *increasing bell shaped* function, since the activation must be 1 when the norm of the difference between the preferred and the current joint vector is higher than the maximum value e_{qM}.

5.1.2 Q2: What is the Jacobian relationship for the Joint Preferred Shape task? How was the task reference computed?

The Jacobian matrix takes into account only the arm; in particular it is a 4x13 matrix because we want to fix the first 4 joints of the arm:

$$\boldsymbol{J}_{opt} = \begin{bmatrix} \boldsymbol{I}_{4x4} & \boldsymbol{0}_{4x3} & \boldsymbol{0}_{4x6} \end{bmatrix}$$

As far as the task reference is concerned, the latter is computed as follow:

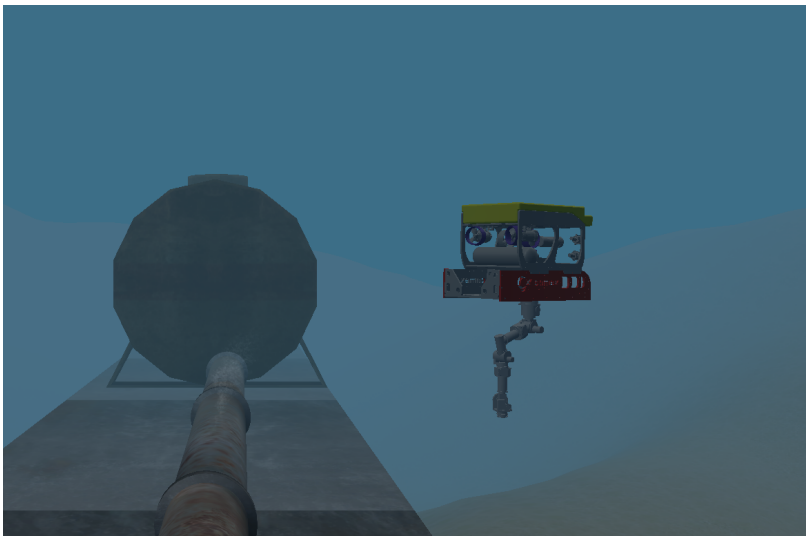
$$\dot{\bar{\boldsymbol{x}}}_{opt} = -\lambda(\boldsymbol{q}_{(1:4)} - \bar{\boldsymbol{q}}) \tag{18}$$

namely, we compute the difference between the current configuration of the first four joints minus the "preferred shape" $\bar{\boldsymbol{q}} = \begin{bmatrix} -0.0031 & 1.2586 & 0.0128 & -1.2460 \end{bmatrix}^\top$

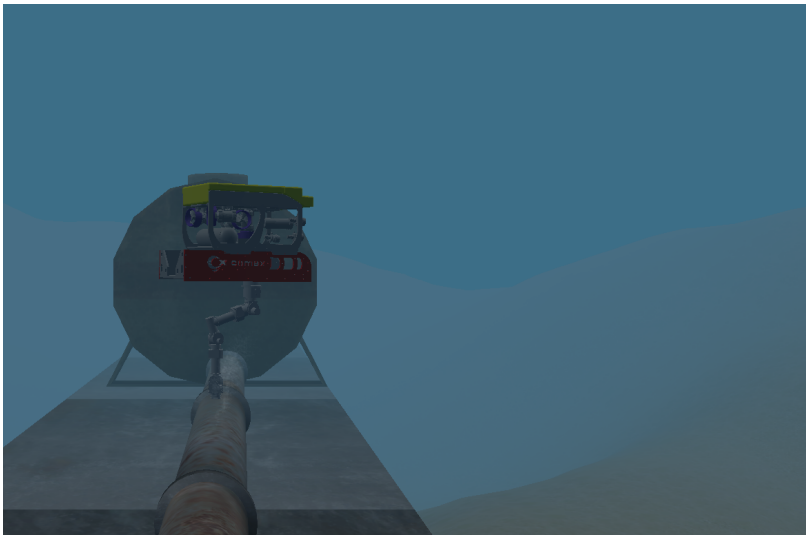
5.1.3 Q3: What is the difference between having or not having this objective?

If the arm preferred shape objective is active, then q_1, q_2, q_3 and q_4 remain constant in time. As shown in Figure 30a, only the remaining three joints can assume a velocity different from zeros.

The vehicle moves keeping the first four joints fixed; therefore the arm remains more or less in the same configuration, as you see in the screenshot of the simulated environment (Figure 29a). The final vehicle positions is just above the pipe (Figure 29b).



(a) Vehicle moving towards the target position



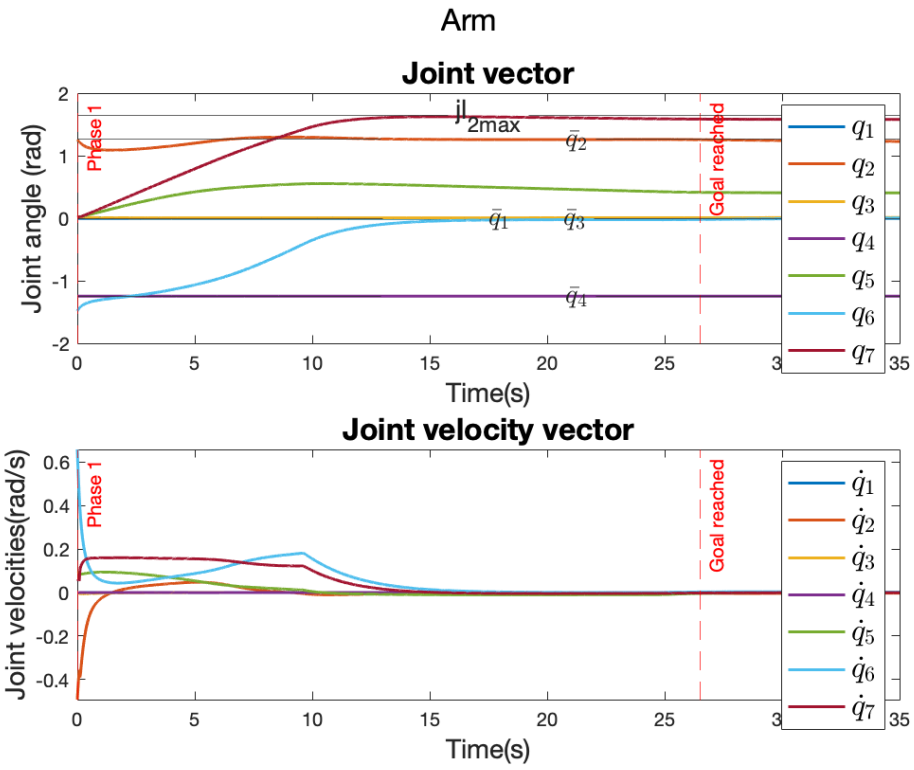
(b) Final vehicle position

Figure 29: Screenshots of UVMS with Arm preferred shape objective **active**

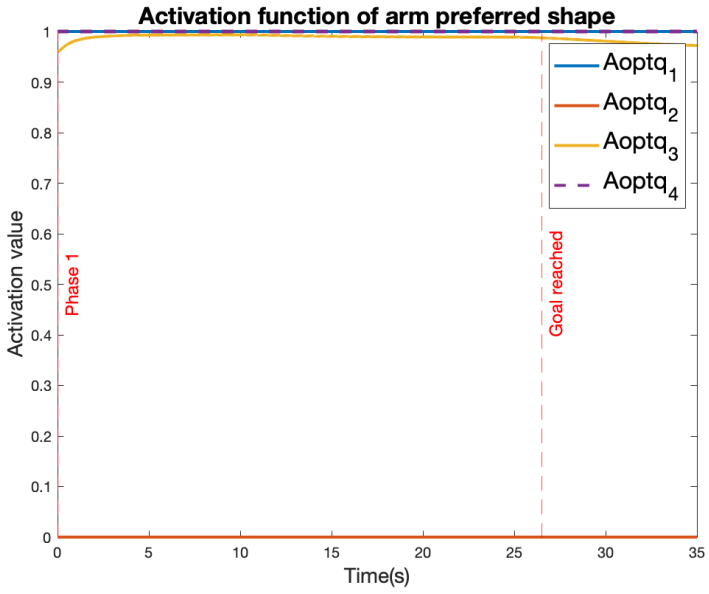
As we have already said, the arm preferred shape is an optimization objective, which does not influence the mission, but allow to choose between multiple solutions, if they exists. To better explain what is going on, we decided to plot also the case in which the joint limit task is not active (Figure 31).

We can notice that at the beginning it seams that there is no a solution with $q_1 = \bar{q}_1$, $q_2 = \bar{q}_2$, $q_3 = \bar{q}_3$ and $q_4 = \bar{q}_4$. Hence, q_2 moves away from $\bar{q}_2$. This fact is particularly evident in Figure 31a, where the joint limit task is inactive.

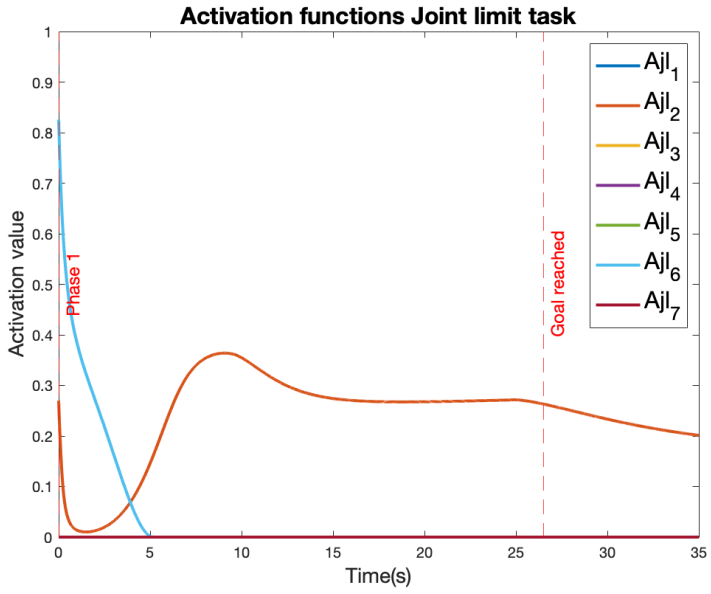
In Figure 30a, the joint limit objective is active and q_2 is close to the upper joint limit. As a consequence, Ajl_2 is active ($\neq 0$) and tends to distance q_2 from its limit.



(a) Joint vector and joint velocity vector of the arm

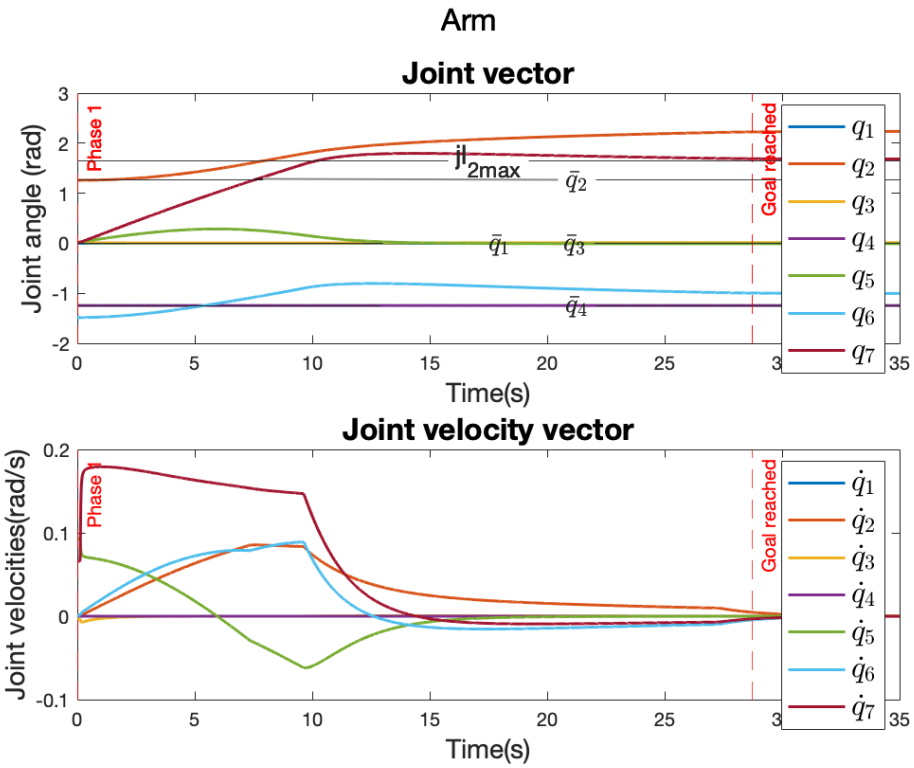


(b) Activation function of arm preferred shape

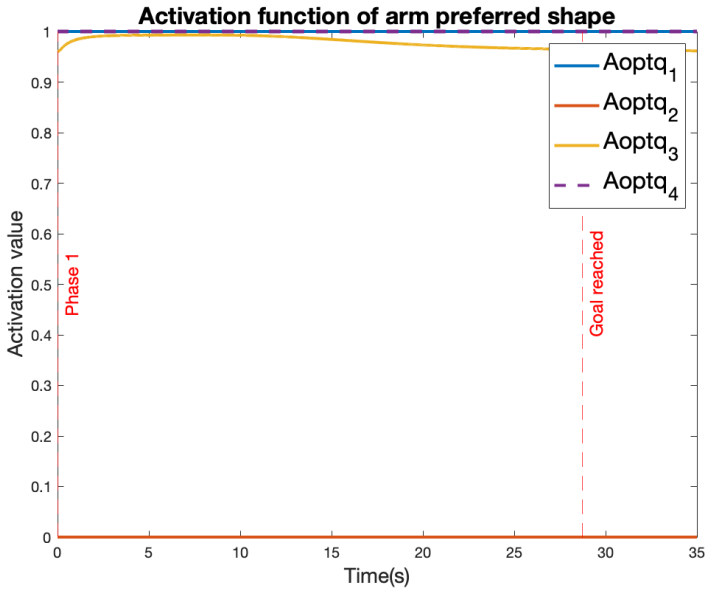


(c) Activation function of joint limit task

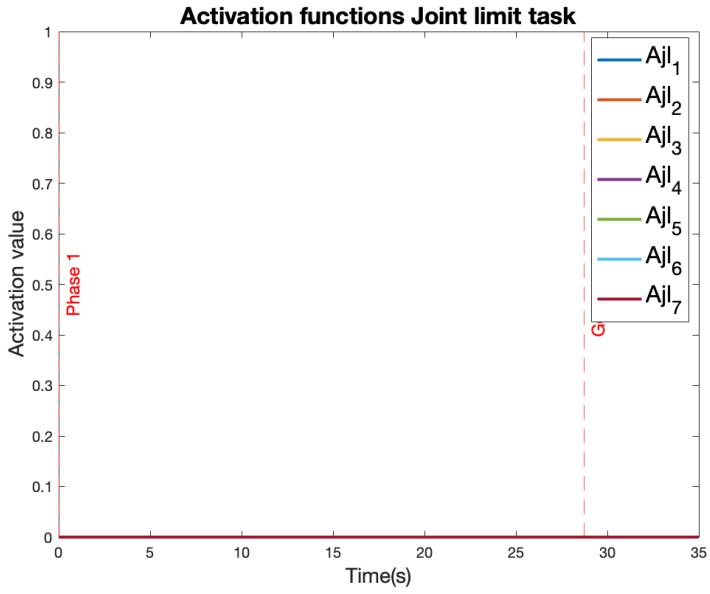
Figure 30: Floating manipulation with Arm preferred shape objective **active** (*joint limit task active*)



(a) Joint vector and joint velocity vector of the arm



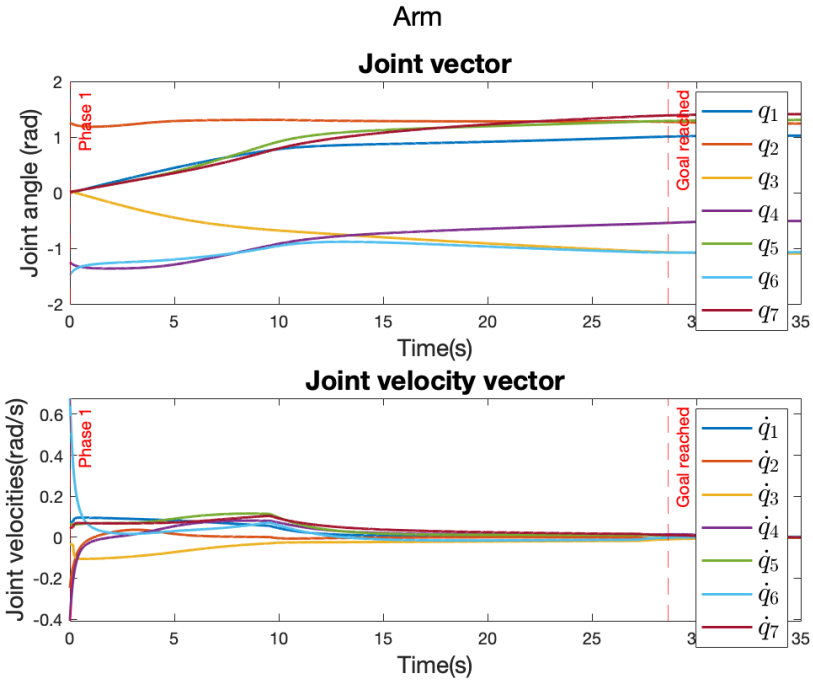
(b) Activation functions



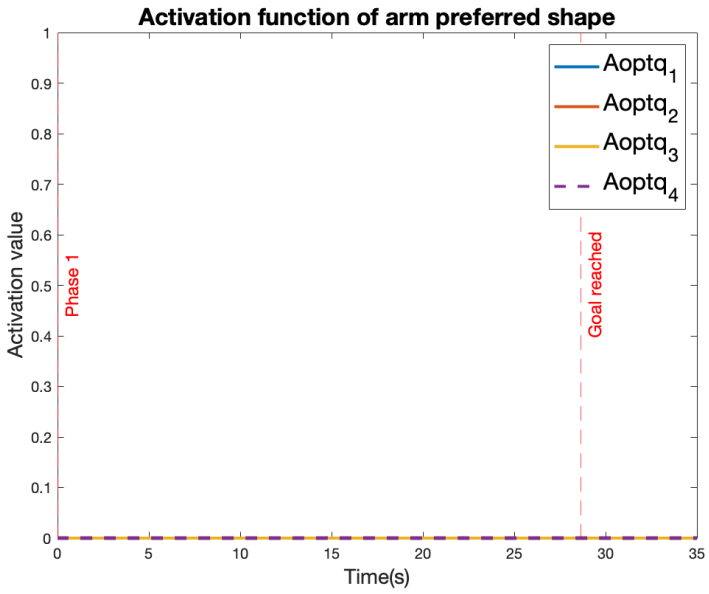
(c) Activation function of joint limit task

Figure 31: Floating manipulation with Arm preferred shape objective **active** (*joint limit task inactive*)

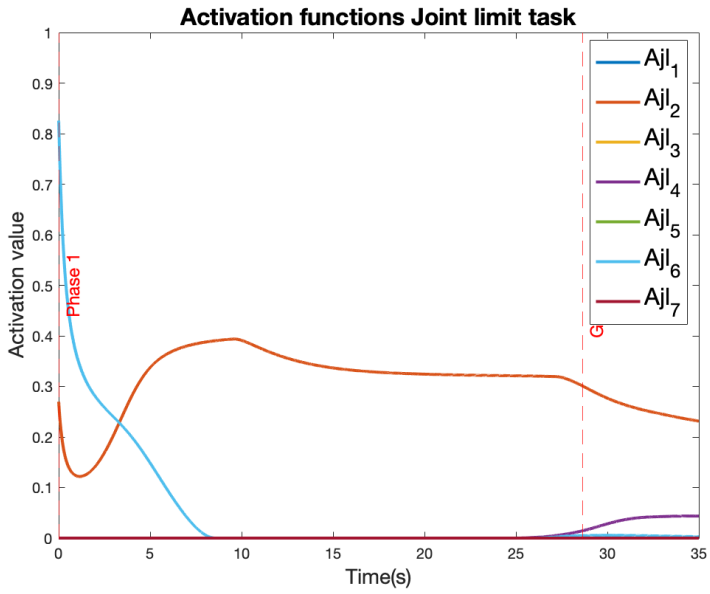
On the other hand, if the optimization is deactivated (Figure 32a), the arm changes its configuration during time. In particular, the arm points its end-effector toward the tool goal position (Figure 33a). In the final configuration, the arm will be extended (Figure 33b).



(a) Joint vector and joint velocity vector



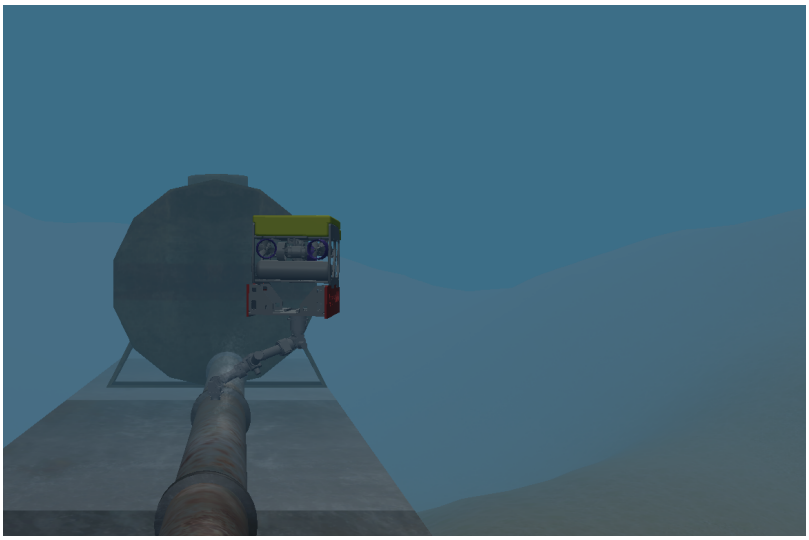
(b) Activation functions



(c) Activation function of joint limit task

Figure 32: Floating manipulation with Arm preferred shape objective **inactive** (*joint limit task active*)

(a) Vehicle moving towards the target position



(b) Final vehicle position

Figure 33: Screenshots of UVMS with Arm preferred shape objective **inactive**

5.2 Adding mission phases

Let us now structure the mission in more than one phase. In the first phase, exploit the previous exercises, and implement a safe waypoint navigation. Move the vehicle to a location close to the current defined end-effector goal position, just slightly above it. Then, trigger a change of action and perform floating manipulation.

Goal: introduce mission phases in the floating manipulation scenario. Observe the difference.

5.2.1 Q1: Report the unified hierarchy of tasks used and their priorities. Which task is active in which phase/action?

This exercise is organized in two phases or actions.
The first action corresponds to *safe waypoint-navigation* and the hierarchy of objectives is the following:

1. Vehicle horizontal altitude (inequality, safety);
2. Vehicle position control task (inequality, action-defining).

The second action regards *floating manipulation* and the hierarchy of objectives is the following:

1. Joint limits (inequality, safety);
2. Arm manipulability (inequality, prerequisite);
3. Vehicle horizontal attitude (inequality, safety);
4. Tool-frame linear position control (equality, action-defining);
5. Tool-frame angular position control (equality, action-defining);
6. Arm preferred shape (inequality, optimization);
7. Vehicle motion minimization (equality, optimization).

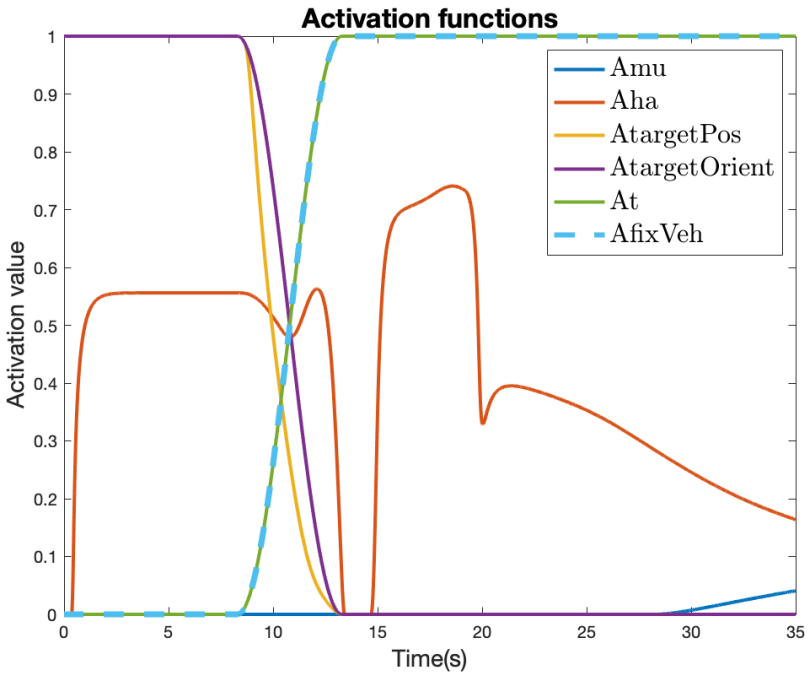


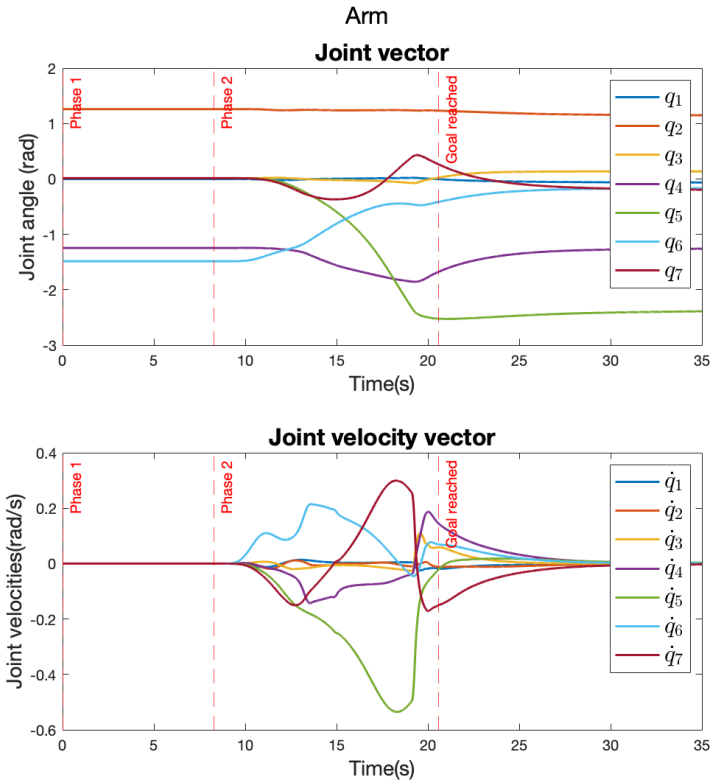
Figure 34: Activation functions to show which tasks are active in the two phases.

5.2.2 Q2: What is the difference with the previous simulation (still in exercise 5), where only one action was used?

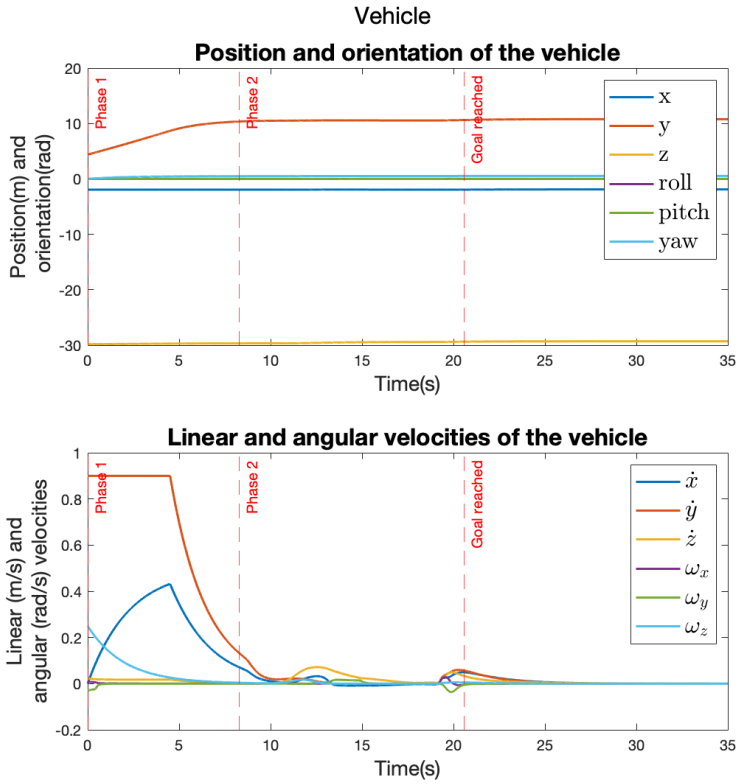
The hierarchy of tasks in Exercise 5.1 does not involve the vehicle position control. The UVMS is "driven" by the end-effector position tracking.

In Exercise 5.2, we structure the mission in two phases. In the first phase, the vehicle position control objective makes the base changing its position and orientation keeping fixed the arm. Once the desired vehicle configuration has been reached (second phase), the arm starts moving, as shown in Figures 35a. The vehicle motion minimization objective fixes the vehicle position and orientation, when possible, as depicted in Figure 35b.

A relevant issue of the proposed architecture, which emerges performing floating manipulative operations, is that the effects of the vehicle position tracking inaccuracy are taken into account. The manipulator and vehicle motions are not kinematically decoupled.



(a) Joint vector and joint velocity vector of the arm



(b) Position, orientation and velocity of the vehicle

(c) Screenshot at the end of the simulation.

Figure 35: Safe waypoint navigation and floating manipulation phases

6 Exercise 6: Floating Manipulation with Arm-Vehicle Coordination Scheme

6.1 Adding the parallel arm-vehicle coordination scheme

Let us now see how the two different subsystems (arm and vehicle) can be properly coordinate. Introduce in the simulation a sinusoidal velocity disturbance acting on the vehicle, and assume the actual vehicle velocity measurable. To do so, add a constant (in the inertial frame) velocity vector to the reference vehicle velocity before integrating it in the simulator.

Goal: modify the control part to implement the parallel arm-vehicle coordination scheme. Observe that, even with a disturbance acting on the vehicle, the end-effector can stay in the required constant position.

6.1.1 Q1: Which tasks did you introduce to implement the parallel coordination scheme?

The idea is to decouple manipulator and vehicle motions, otherwise the disturbances of the floating base would propagate through the coupled kinematics immediately to the end effector of the manipulator. To fulfil this mission, we have to introduce two optimizations TPIK 1 and TPIK 2 (Task Priority Inverse Kinematics) running in parallel (Figure 36).

Given the action $\mathcal{A}$ to be executed, the parallel scheme which coordinates the vehicle and arm movements works as follows:

- The first optimization TPIK 1 considers the vehicle with the manipulator and it provides $\dot{\boldsymbol{y}}_1 = \begin{bmatrix} \bar{\boldsymbol{v}}_1 \\ \dot{\boldsymbol{q}}_1 \end{bmatrix}$ where only the vehicle reference velocity $\bar{\boldsymbol{v}}_1$ is used, while the manipulator part $\dot{\boldsymbol{q}}_1$ is discarded.
- The second optimization TPIK 2 considers the vehicle as totally non-controllable: it imposes the current measured velocity of the vehicle base and computes the joint velocity vector of the arm that optimizes the execution of the task. The output of this subsystem is: $\dot{\boldsymbol{y}}_2 = \begin{bmatrix} \boldsymbol{v} \\ \dot{\boldsymbol{q}}_2 \end{bmatrix}$.

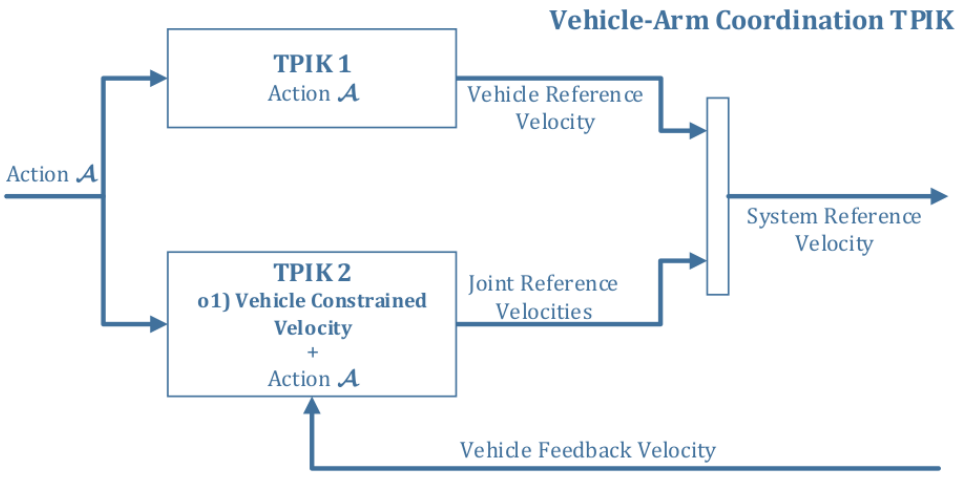


Figure 36: A detailed view of the Kinematic Control Layer, with the two Task Priority Inverse Kinematics blocks highlighted, performing the vehicle velocity tracking compensation scheme.

Implementation

The procedure is implemented by calling the function `taskSequence` twice.

At each iteration, the current vehicle velocity (both linear and angular components) is saved in a temporary variable, called $\boldsymbol{current_v}$. Then, the velocity $\bar{v}_1$ that has to be achieved by the vehicle is computed by calling the function `taskSequence` (TPIK 1). The latter can also acquire $\boldsymbol{current_v}$ as

an additional input parameter. If the "measured" vehicle velocity is provided, the *vehicle constrained velocity task* is placed on top of the task hierarchy of action $\mathcal{A}$. We described this objective in Section 4, however this task is placed at highest priority therefore it is no more an optimization, but it becomes a prerequisite.

The following equality condition has to be satisfied for the floating base:

$$\boldsymbol{v}_v = \boldsymbol{current_v}$$

The task has to be maintained always active, therefore the activation function is:

$$\boldsymbol{A}_{fixVehicle} = \boldsymbol{I}_{6x6}$$

The Jacobian is simply implemented in such a way to consider only the vehicle part and not the arm:

$$\boldsymbol{J}_{fixVehicle} = \begin{bmatrix} \boldsymbol{0}_{6x7} & \boldsymbol{I}_{6x6} \end{bmatrix}$$

The function `taskSequence` is called a second time with the additional parameter and it provides as output the reference joint velocity $\dot{\tilde{\boldsymbol{q}}}_2$.

The outputs of the two subsystems should be forwarded to the corresponding parts of the DCL to be tracked. We add a sinusoidal velocity disturbance (in the world frame) to the computed the reference vehicle velocity before integrating it in the simulator:

$$\bar{\boldsymbol{v}} = \bar{\boldsymbol{v}}_1 + d(t)$$

where

$$d(t) = \begin{bmatrix} {}^{v}_{w}\boldsymbol{R} & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{3x3} & {}^{v}_{w}\boldsymbol{R} \end{bmatrix} * sin(\boldsymbol{w}t) * \begin{bmatrix} a_x & a_y & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

is *sinusoidal disturbance* in the inertial frame and then projected in the vehicle frame to be coherent with the velocity that we would track. The parameters a_x and a_y specifies the amplitudes of the sinusoidal disturbance along the x- and y-axis respectively.

Thanks to the TPIK 2 optimization, the arm joint velocities are always optimal ones based on the current vehicle velocities, independently of any vehicle inaccuracy in tracking the desired ones generated by TPIK 1. Figure 38a shows the vehicle base position given the sinusoidal disturbance. Despite the disturbance signal, the tool keeps the target position as shown in Figure 37.

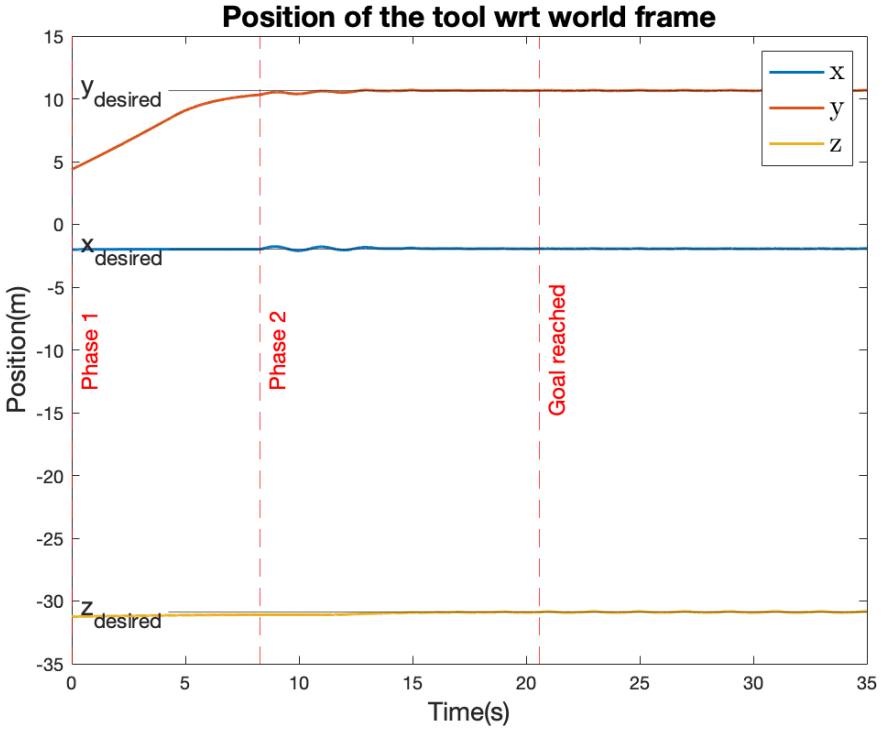
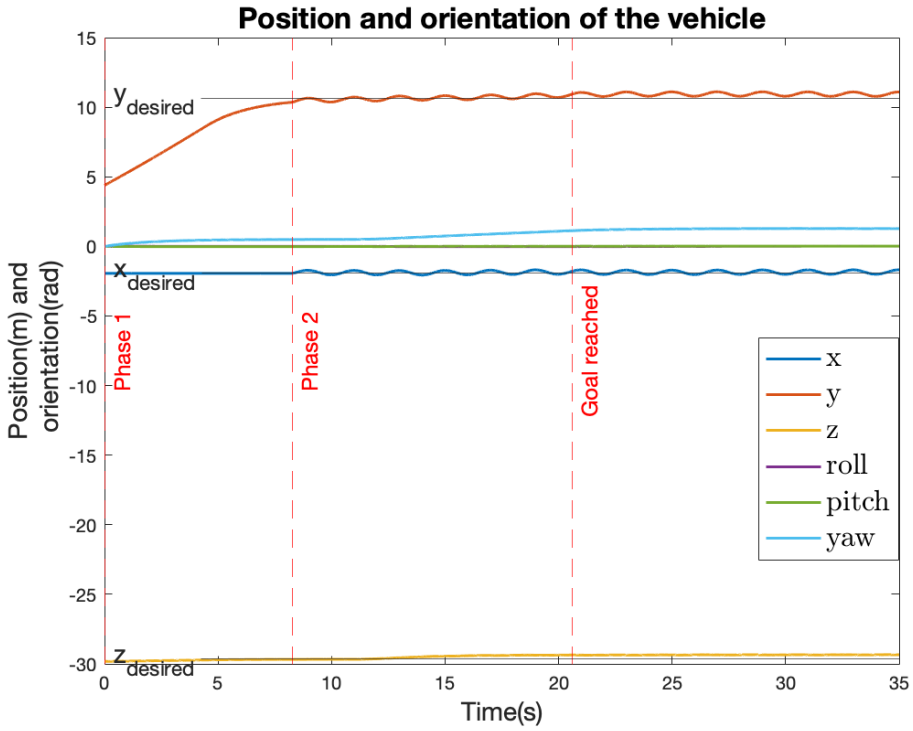
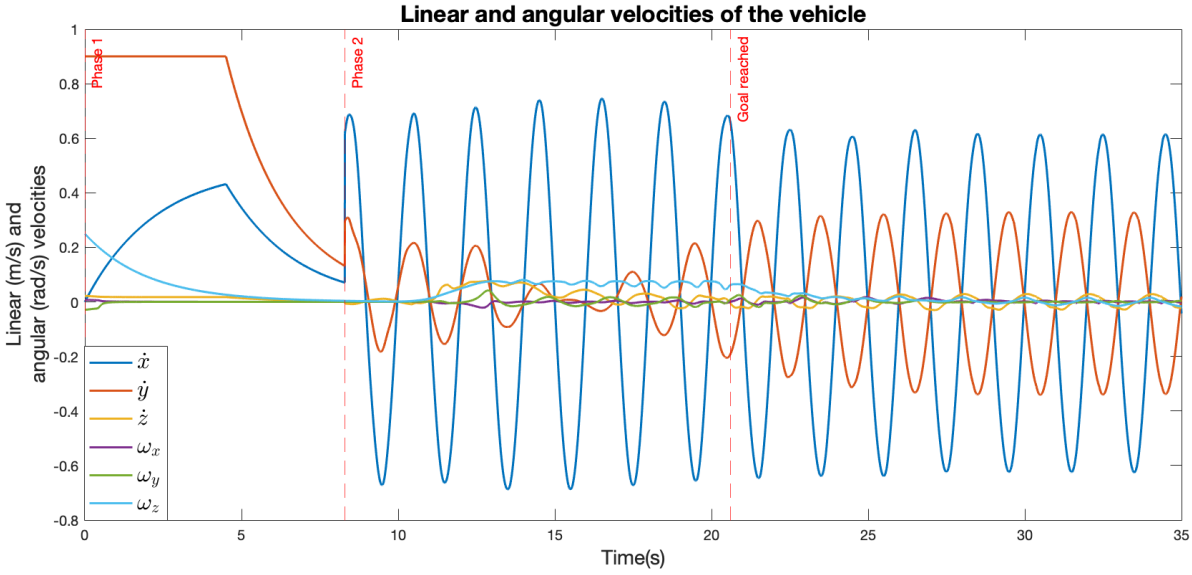


Figure 37: Position of the *arm's tool* with respect to the world frame when the a sinusoidal velocity disturbance (amplitude 0.5 along the x axis and 0.5 along the y axis) is acting on the vehicle.



(a) Position and orientation of the vehicle with respect to the inertial frame



(b) Linear and angular velocities of the vehicle with respect to the inertial frame

Figure 38: *Vehicle* subjected to a sinusoidal velocity disturbance of amplitude 0.5 along the x axis and 0.5 along the y axis

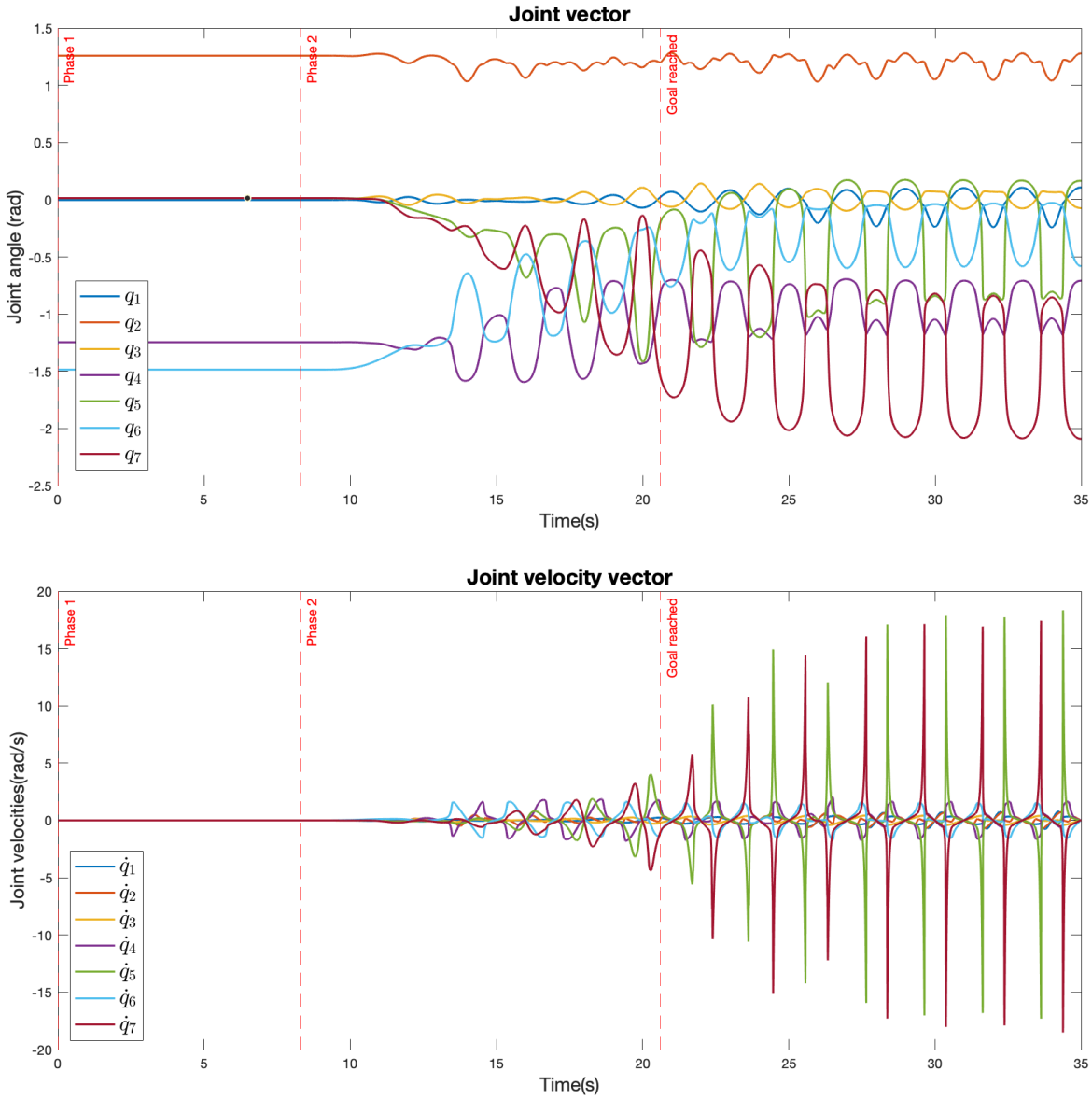
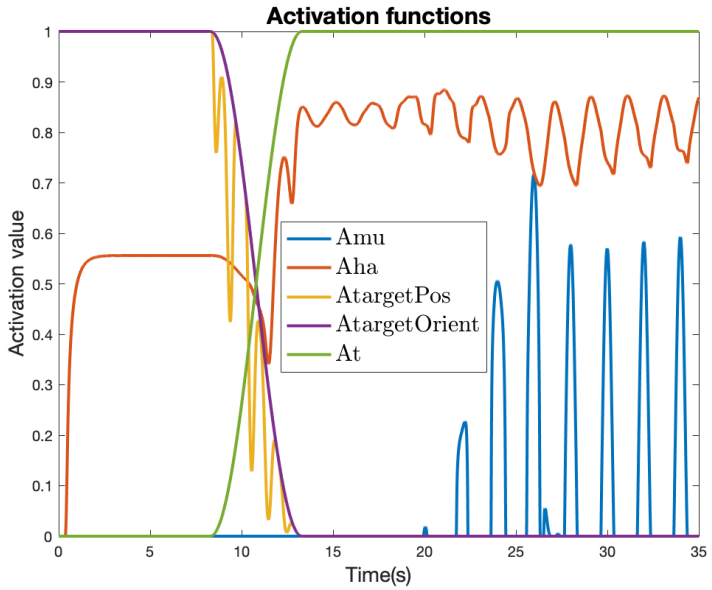
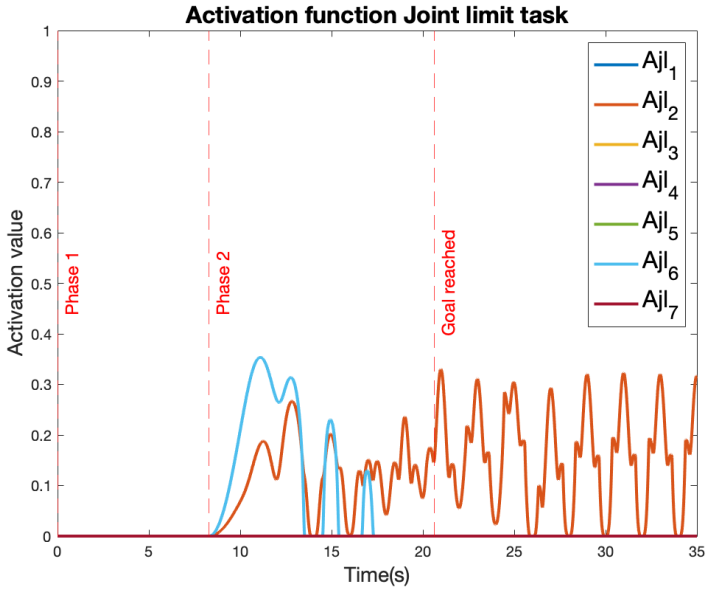


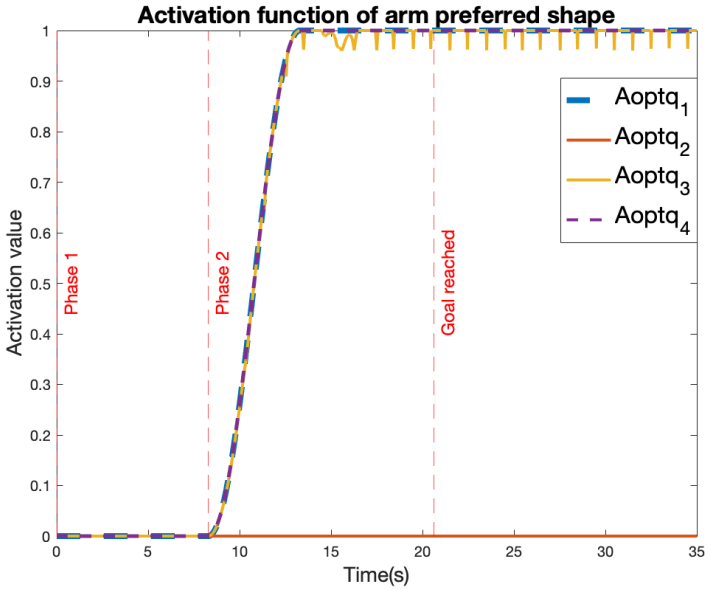
Figure 39: *Vehicle* subjected to a sinusoidal velocity disturbance of amplitude 0.5 along the x axis and 0.5 along the y axis



(a) Tasks active in action $\mathcal{A}$



(b) Activation function Joint limit task



(c) Activation function of arm preferred shape

Figure 40: Activation functions

6.1.2 Q2: What happens if the sinusoidal disturbance becomes too big?

If the sinusoidal disturbance becomes too big (i.e., $a_x = 2.5$ and $a_y = 2.5$), the system is not in charge of compensate and the disturbance of the floating base would propagate through the coupled kinematics to the end effector of the arm: the tool cannot keep precisely the desired position as demonstrated in Figure 41.

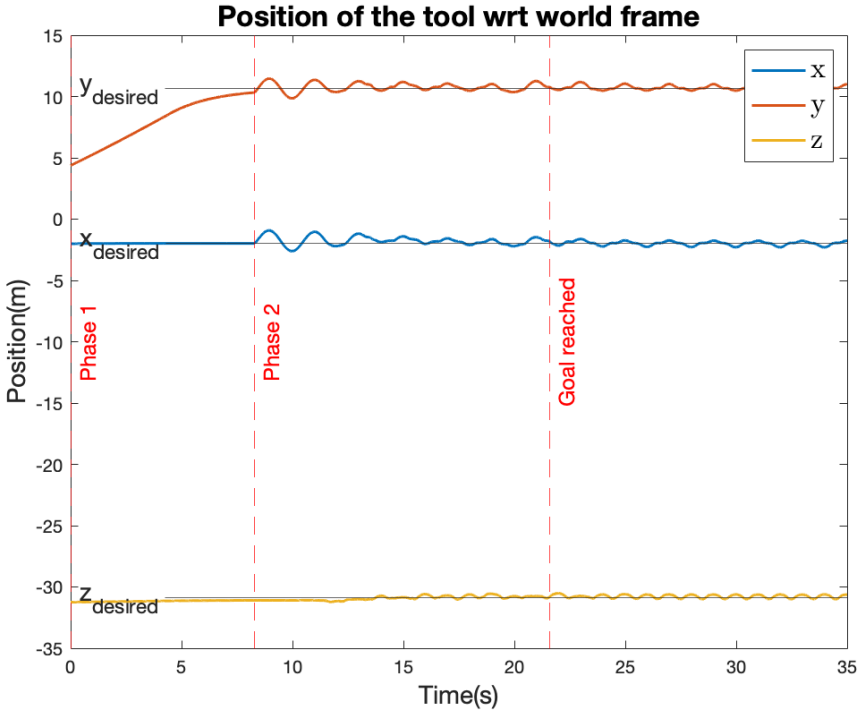


Figure 41: Position of the *tool* with sinusoidal velocity disturbance ($a_x = 2.5$, $a_y = 2.5$) on the vehicle.

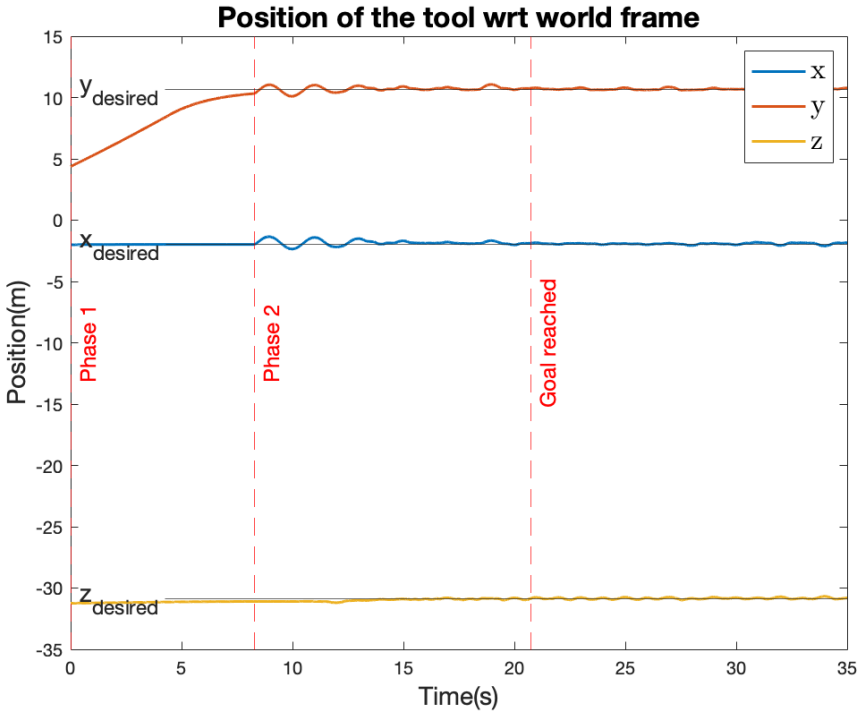
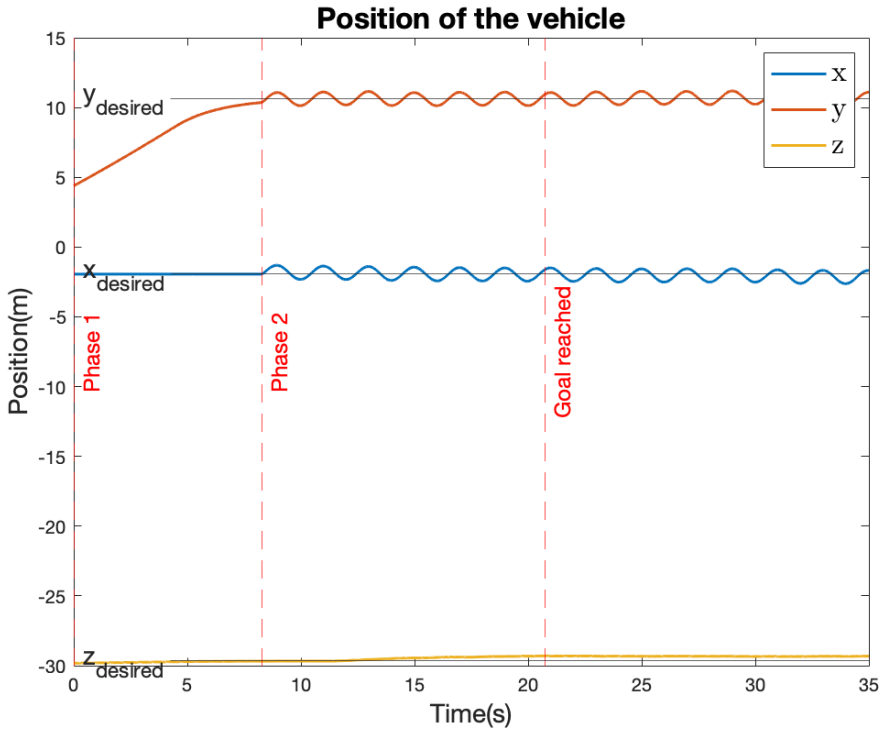
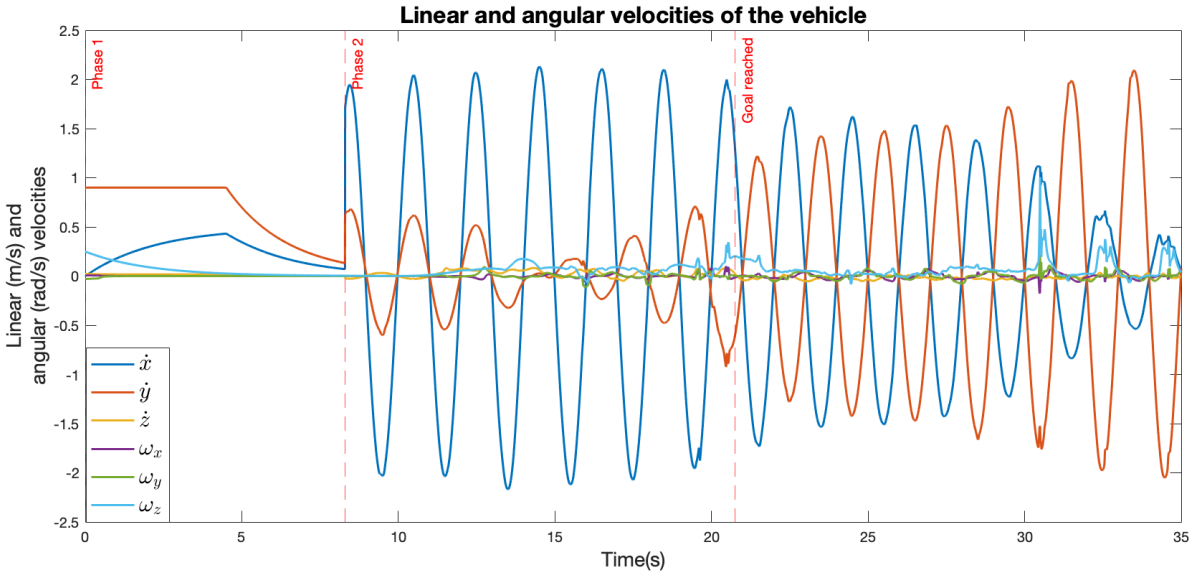


Figure 42: Position of the *tool* with sinusoidal velocity disturbance ($a_x = 1.5$, $a_y = 1.5$) on the vehicle.



(a) Position and orientation of the vehicle with respect to the inertial frame



(b) Linear and angular velocities of the vehicle with respect to the inertial frame

Figure 43: *Vehicle* subjected to a sinusoidal velocity disturbance of amplitude 1.5 along the x axis and 1.5 along the y axis

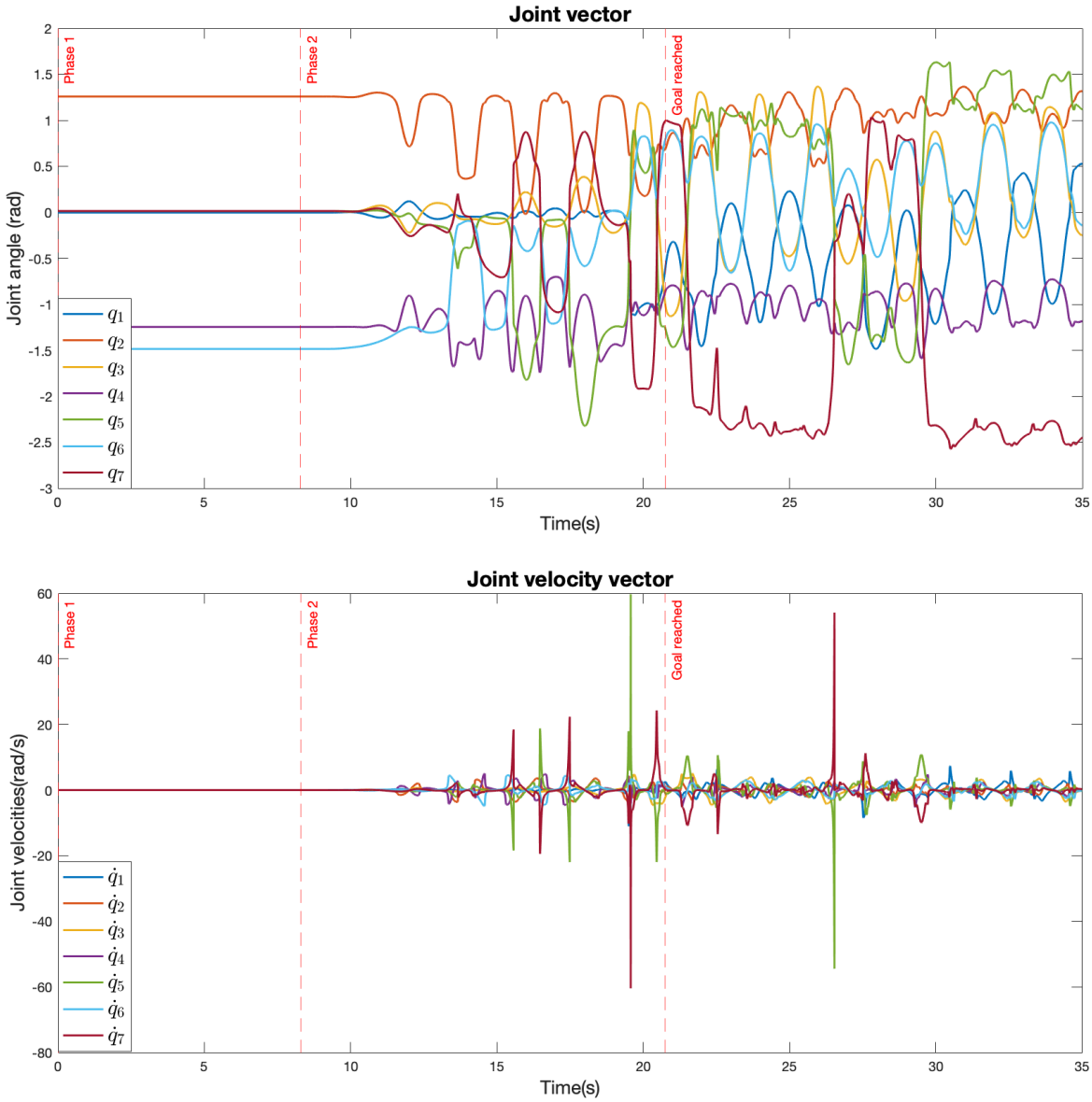
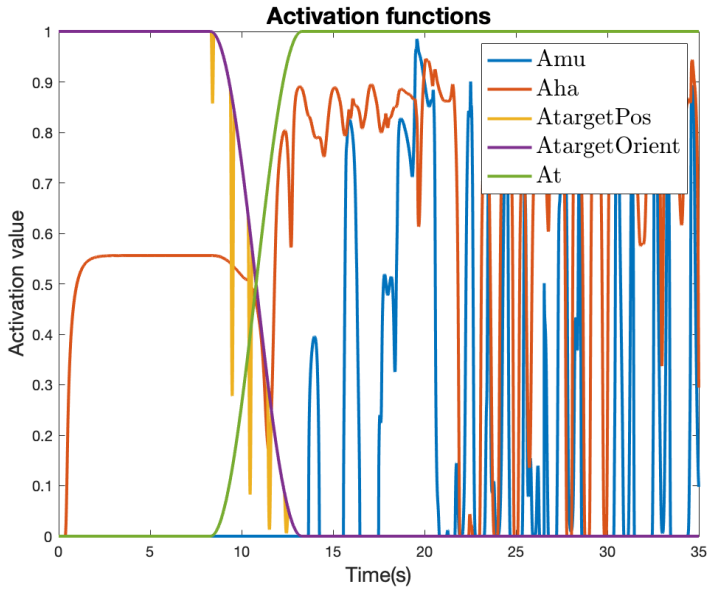
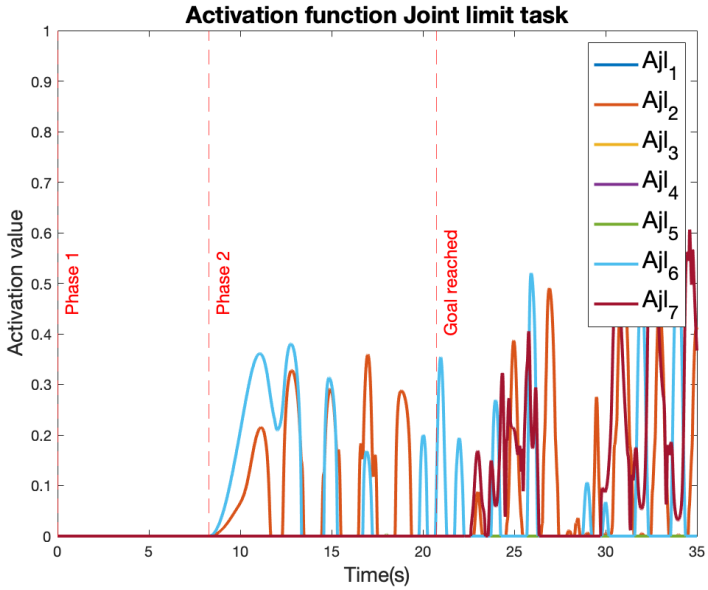


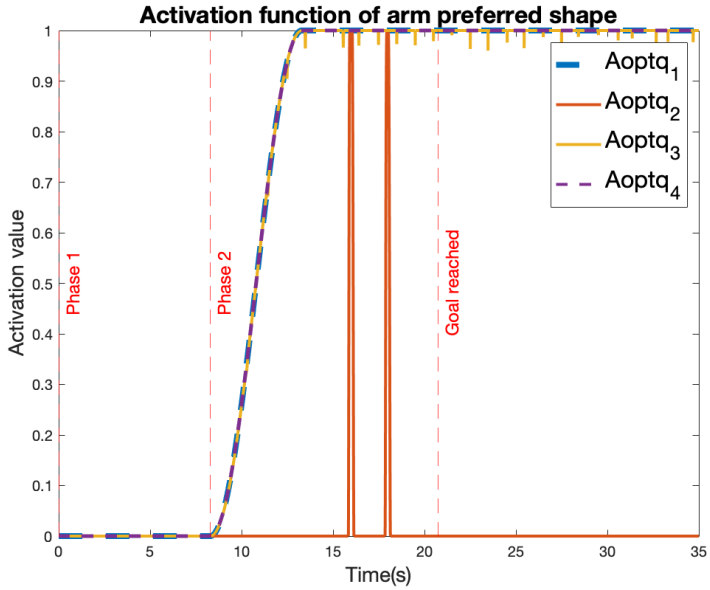
Figure 44: *Vehicle* subjected to a sinusoidal velocity disturbance of amplitude 1.5 along the x axis and 1.5 along the y axis



(a) Tasks active in action $\mathcal{A}$



(b) Activation function Joint limit task



(c) Activation function of arm preferred shape

Figure 45: Activation functions