Cooperative Robotics

Authors : Francesco Ganci, Serena Paneri
EMAILs : francesc.ganci@gmail.com, serenapaneri2605988@gmail.com
Date : May 2022 - June 2022
GitHub Repository : https://github.com/programmatoroSeduto/CooperativeRoboticsProject

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.

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

Table 1: Example of actions/hierarchy table: a number in a given cell represents the priority of the control task (row) in the hierarchy of the control action (column). The type column indicates whether the objective is an equality (E) or inequality (I) one.

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

1 Exercise 1: Implement a "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: What is the Jacobian relationship for the Vehicle Position control task? How was the task reference computed?

The simulation implements the Vehicle Position Control Task in two separated tasks:

- **Position Control Task** : the task enables the robot to reach a target point in the space
- **Orientation Control Task** : the task performs an alignment of the robot to a given target frame

Concerning the Position Control Task, given a *fixed* point in the space, the objective is formulated as *making the origin of the vehicle frame coinciding with a given point in the space*. Here's the objective function (in the simulation, the goal P has been called *vehicleGoalPosition* with transformation matrix *wTgt*):

$$\underline{x}(\underline{c}) = O_v$$

Let's assume that the point P is fixed with respect to the world frame. The derivative of the objective function can be written as the simple vehicle velocity:

$$\dot{\underline{x}} = \underline{v}_{v/w}$$

So, after projected the above-mentioned relation in the world frame, and recalling that the simulation provides the projected configuration vector $\underline{y}$ in the vehicle frame, the Jacobian for the Position Control Task is:

$${}^w J_{\text{pos}} = \begin{bmatrix}O_{3\times 17} & {}^w R_v & O_{3\times 3}\end{bmatrix}$$

Concerning the Orientation Control Task, the reasoning is pretty much the same as before. The objective function is the misalignment vector obtained from the versor lemma, $\underline{\rho}_{v/Tg}$.

$$\underline{x}(\underline{c}) = \underline{\rho}_{v/Tg}$$

We could derive the entire vector $\underline{\rho}_{v/Tg}$ and project it, but it's simpler to think to directly control the angular velocity of the vehicle $\underline{\omega}$, hence the derivative of the objective function may be written simply as:

$$\dot{\underline{x}} = \underline{\omega}_{v/w}$$

Please take into account that, the frame we're trying to reach, is a fixed frame with respect to the world one. After projecting the equation with respect to the world frame, the Jacobian is:

$${}^w J_{\text{orient}} = \begin{bmatrix}O_{3\times 17} & O_{3\times 3} & {}^w R_v\end{bmatrix}$$

The task reference for the Position Control Task is simply the difference between the current coordinates of the vehicle (i.e. the coordinates of the origin of the vehicle frame) and the point the vehicle is trying to reach. A gain value $\lambda = 0.5$ has been chosen for the task.

$$\bar{\dot{x}} = 0.5 \cdot (P - O_v)$$

The one for the Orientation Control Task, is the misalignment vector between the vehicle frame and the target one. Also here $\lambda = 0.5$ has been used.

$$\bar{\dot{x}} = 0.5 \cdot \underline{\rho}_{v/Tg}$$

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.

The Horizontal Attitude is an objective of inequality type and it belongs to the category of the safety tasks. For that reason, it needs to have an higher priority with respect to the action defining tasks, such as those on position and orientation control above mentioned.
The hierarchy should become, after including this safety task, something like that:

Task	Type	$\mathcal{A}_1$
Horizontal Attitude	I,S	1
Vehicle Position Control	E,AD	2
Vehicle Orientation Control	E,AD	3

The horizontal attitude task adjusts the orientation of the vehicle, in order to make it parallel to the seafloor. To do so, it is necessary to reason on the orientation of the world frame $< w >$ with respect to the vehicle frame $< v >$. In particular, we need to reason on the misalignment between the $\hat{k}$ versor of the vehicle frame and the $\hat{k}$ versor of the world frame since, if the z-axis of the vehicle is equal to the z-axis of the world frame, it means that the vehicle is horizontal.
Indeed, the horizontal attitude will become active when the angle, between the two versors ${}^{v}k$ and ${}^{w}k$, is greater than a certain threshold θ_{min}.
The chosen activation function is the IncreasingBellShape() function, whose parameters are θ_{min}=0.2 and Δ=0.2.
To test this behavior, we have chosen as initial position $[10.5\ 35.5\ -36\ 0\ 0\ \pi/2]^T$ and as a target position $[10.5\ 37.5\ -38\ 0\ \pi/6\ 0]^T$. Giving a look to the graphs below:

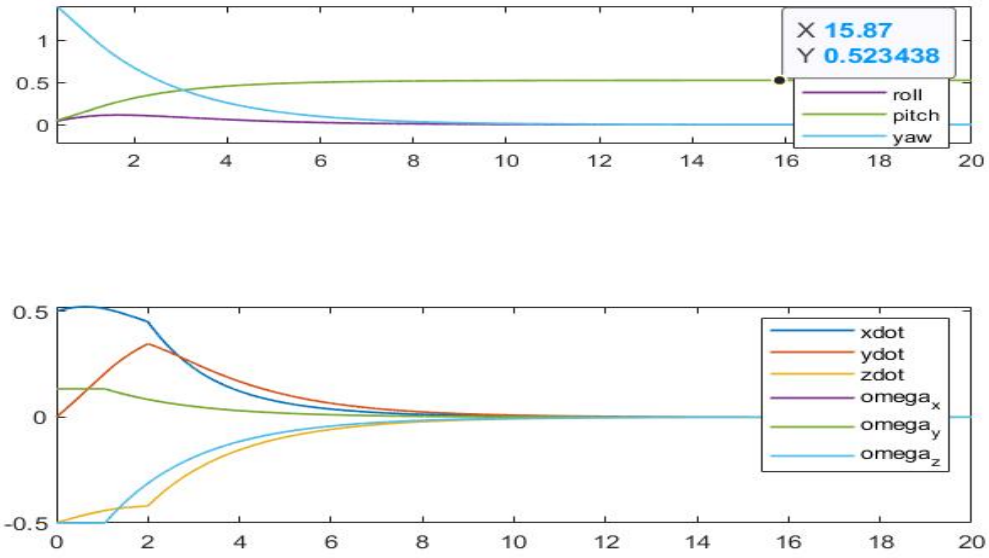


Figure 1: Motion of the vehicle when the horizontal attitude is not active

The behavior, shown in Figure 1, corresponds to the case in which the horizontal attitude is not enabled. We can see that the vehicle only tries to adjust itself to reach the target position and orientation. Moreover, if we focus on the graph which highlights the evolution of the angles, we can notice that, the pitch angle reaches a steady state of approximately $\pi/6 \approx 0.5236$.
Instead, if the horizontal attitude is enabled, the behaviour is the following:

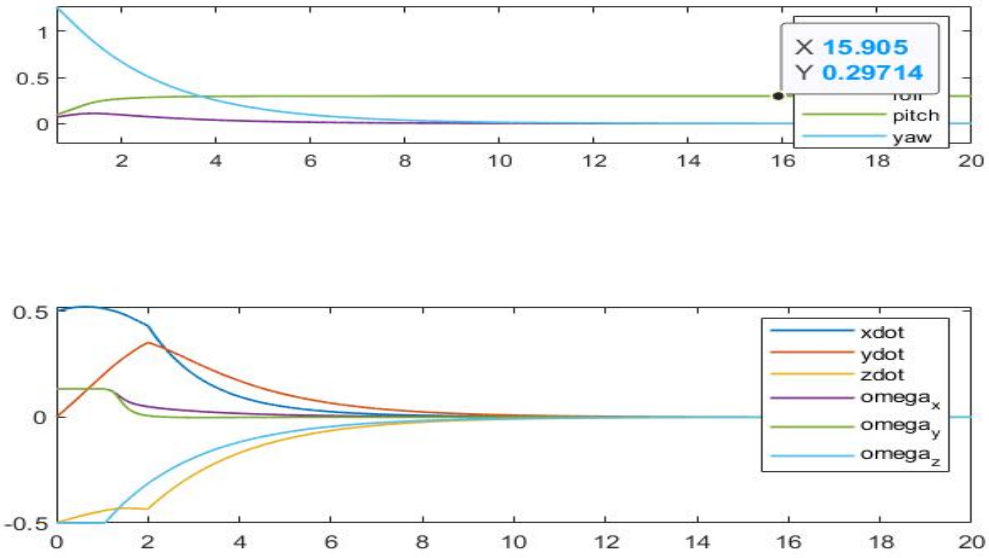


Figure 2: Motion of the vehicle when the horizontal attitude is active

Looking at figure 2, the vehicle needs to make a compromise on the angle and indeed the pitch angle of the vehicle does not reach the value of $\pi/6$ but, instead, it reaches a value between $\pi/11 \approx 0.2856$ and $\pi/10 \approx 0.3142$.

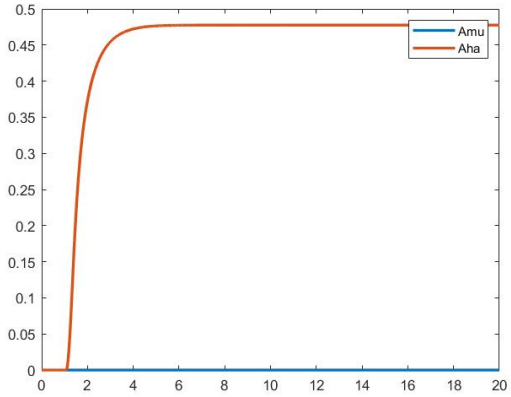


Figure 3: Activation function when the horizontal attitude is active

In Figure 3, we can notice that the activation function is off for some instances, meaning that the vehicle is horizontal with respect to the world frame, and then it activates. This happens since the vehicle is trying to reach the target, whose orientation is not parallel to the seafloor. The fulfilment of the action defining task, in this case, goes against the horizontal attitude task, whose priority in the hierarchy is higher with respect to the vehicle control orientation.

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

Task	Type	$\mathcal{A}_1$
Vehicle Position Control	E,AD	1
Vehicle Orientation Control	E,AD	2
Horizontal Attitude	I,S	3

If we swap the priorities in the hierarchy, and so, the safety task comes after the action defining task, we can notice from the graphs below that:

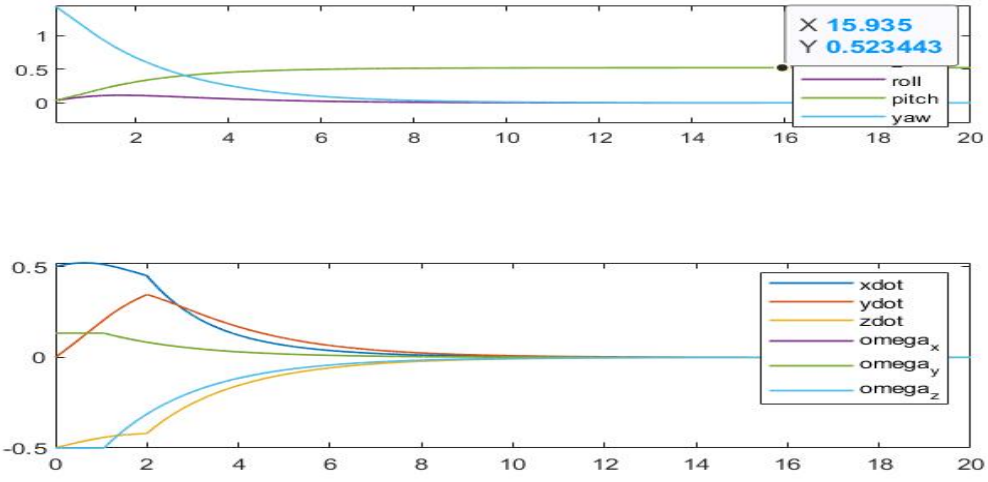


Figure 4: Motion of the vehicle when the horizontal attitude is at the lowest priority

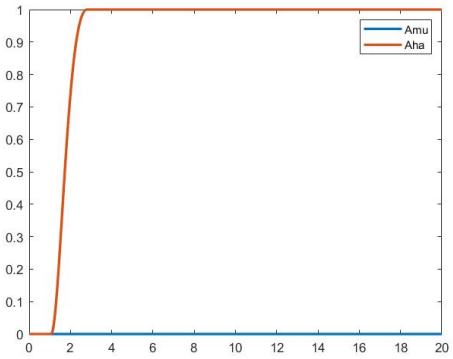


Figure 5: Activation function when the horizontal attitude is at the lowest priority

The behavior is very similar to the first one in which the horizontal attitude task was not enabled. Indeed, if we focus on the angles plot in Figure 4, we can clearly see that, the position control task is accomplished and the pitch angle reaches the value of $\pi/6$ as expected.
Since the horizontal attitude is no more at the highest priority level, we cannot ensure the same level of safety as before since, for the accomplishment of the mission, it's more relevant the reaching of the target than the safe way to get there.

Exercise 1 part 1 on GitHub

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 new hierarchy of tasks of the Safe Waypoint Navigation and their priorities. Comment how you choose the priority level for the minimum altitude.

Task	Type	$\mathcal{A}_1$
Minimum Attitude	I,S	1
Horizontal Attitude	I,S	2
Vehicle Position Control	E,AD	3
Vehicle Orientation Control	E,AD	4

The Minimum Attitude is an objective of inequality type and it belongs to the category of the safety tasks. For that reason, it needs to have an higher priority than the action defining tasks.

1.2.2 Q2: What is the Jacobian relationship for the Minimum Altitude control task? Report the formula for the desired task reference generation, and the activation thresholds.

The Minimum altitude control task employs, as its objective function, the projection of the measured altitude along the axis $\hat{k}_w$, which is assumed orthogonal to the plane of the seafloor. The scalar d is the

altitude detected by the vehicle, which is called *uvms.sensorDistance* in the simulation environment. Here's the objective function:

$$x(\underline{c}) = {}^{v}\,\underline{\hat{k}}_{w} \cdot \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix}$$

The derivative is simply the projection of the vehicle velocity along the same axis, so the Jacobian relationship can be written as:

$${}^{v}J_{\text{ma}} = \begin{bmatrix} O_{1\times 17} & {}^{v}\underline{\hat{k}}_{w} & O_{1\times 3} \end{bmatrix}$$

The task reference is the difference between the desired minimum altitude (for instance, 1 meter) and the current altitude of the vehicle, i.e. the projection of the detected distance from the seafloor and $\underline{\hat{k}}_{w}$. Here's the relationship:

$$\dot{\tilde{x}}_{ma} = \lambda_{ma} * (x_{ma} - x)$$

where x is the current "vertical altitude" of the vehicle, $\lambda_{ma} = 0.2$ and x_{ma} is the minimum altitude.

For sake of simplicity, let's consider, for instance, a minimum altitude of 1m. This is a inequality objective so, the task should be turned off when the current vertical altitude is greater than 1 meter so, 1 in $[0m, 1m]$ and 0 for $x \geq 1m$. But this threshold won't work well, even taking into account the slow dynamics of the system, hence, for $x \leq 2m$ works well, with a buffer zone of length 1m. With these values, the system tends to react "in advance", i.e. at a distance greater than the critical one. Applying this idea, we chose $x_{ma} = 3m$ instead of 1m.

1.2.3 Q3: Try imposing a minimum altitude of 1, 5, 10 m respectively. What is the behaviour? Does the vehicle reach its final goal in all cases?

- Minimum altitude = 1 meter

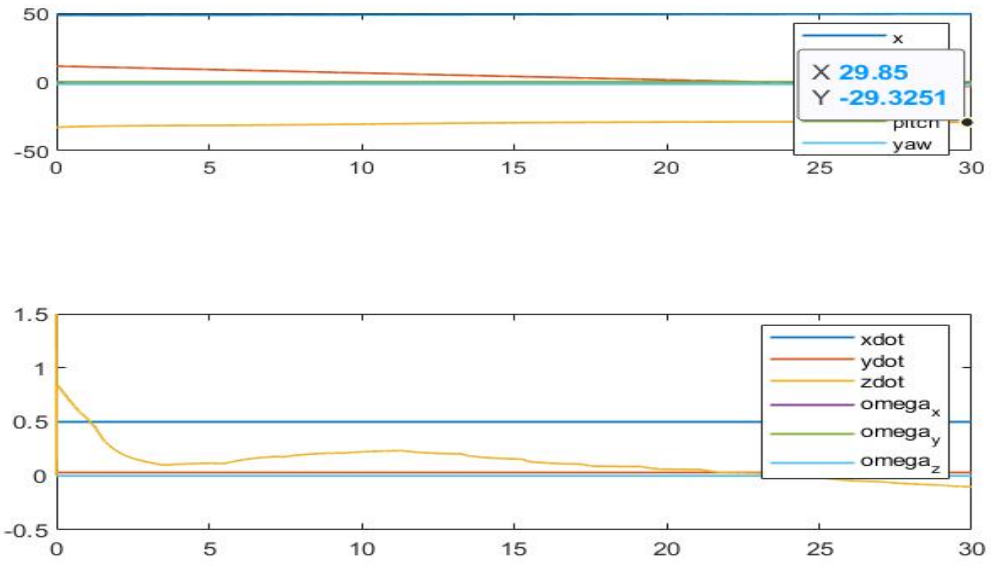


Figure 6: Motion of the vehicle when the minimum altitude is set at 1 meter

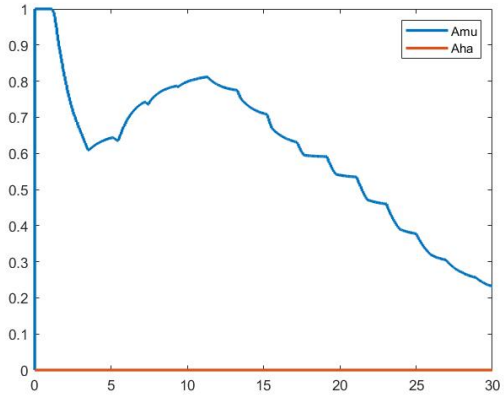


Figure 7: Activation function when the minimum altitude is set at 1 meter

In Figure 7, we can notice that, there is an impulse at t=0, meaning that the vehicle is too close to the seafloor, due to the initial position set at $[48.5\ 11.5\ -33\ 0\ 0\ -\pi/2]^T$. In Figure 6, we can notice that, due to the Minimum Altitude task, of course, the Vehicle Position Control objective (which correspond to the action defining task) is not completely fulfilled. Indeed, we can notice that, at the end of the simulation, the robot reaches a final position with a z$\simeq$-29m.

- Minimum altitude = 5 meters

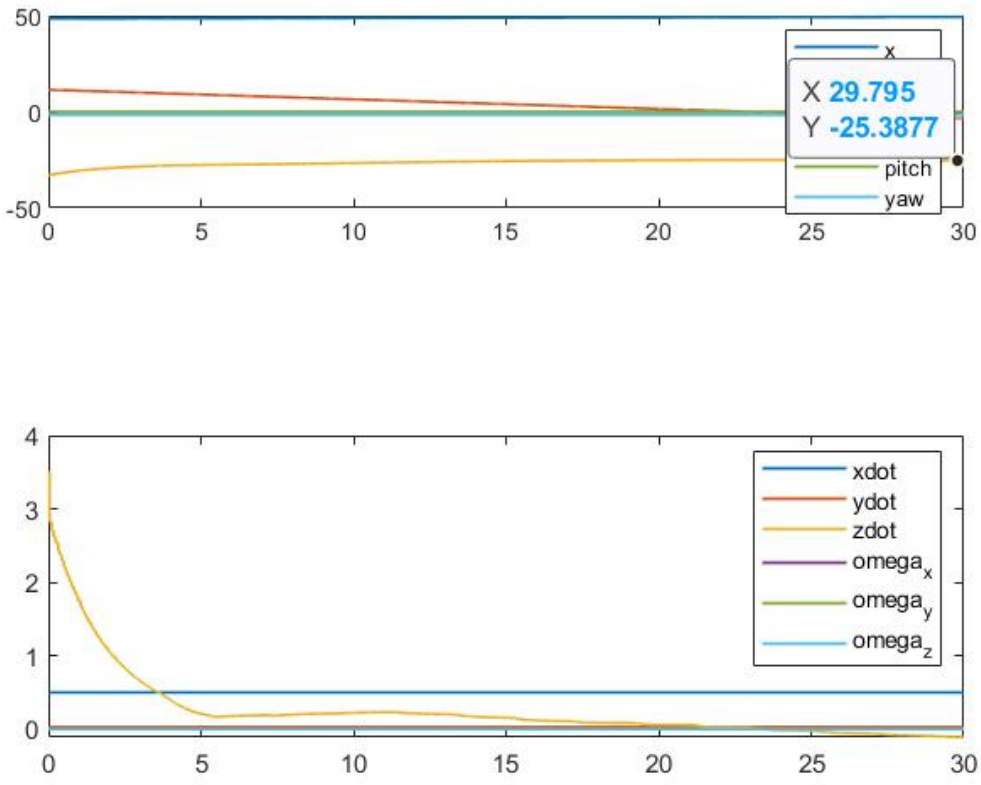


Figure 8: Motion of the vehicle when the minimum altitude is set at 5 meters

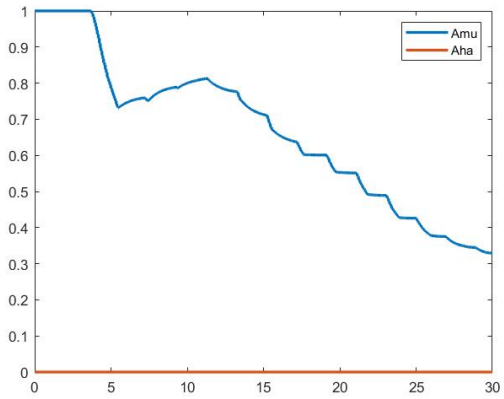


Figure 9: Activation function when the minimum altitude is set at 5 meters

In Figure 8, we can notice that again, due to the Minimum Altitude task, the robot is not able to reach completely its target position but, instead arrives in a final position with z≃-25m.

- Minimum altitude = 10 meters

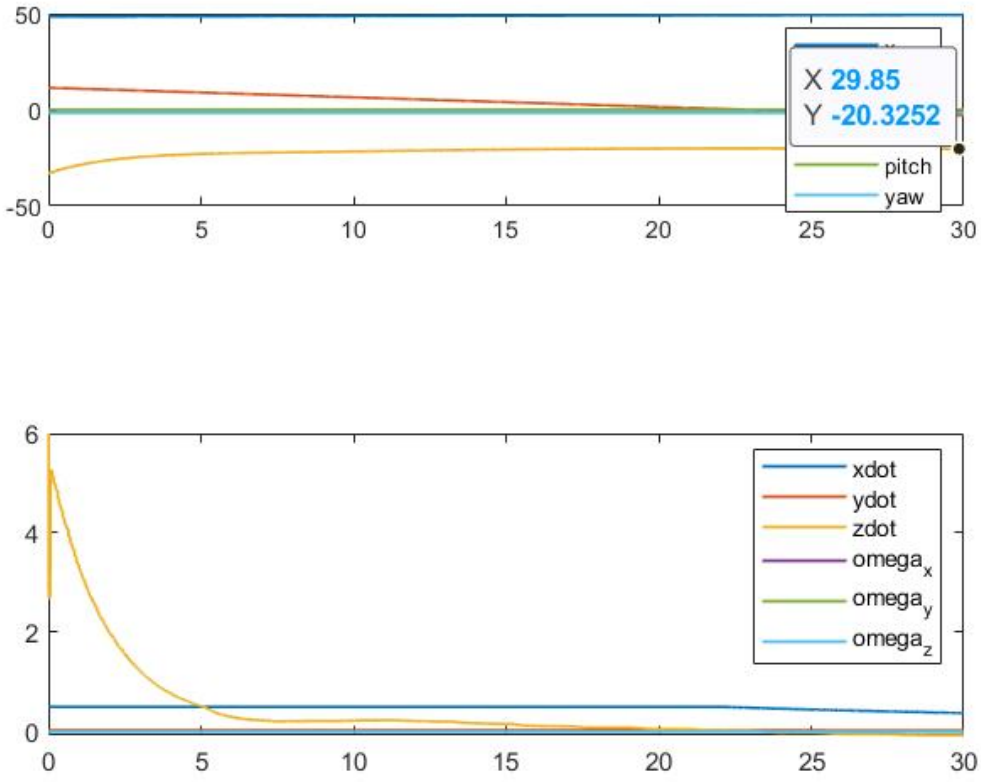


Figure 10: Motion of the vehicle when the minimum altitude is set at 10 meters

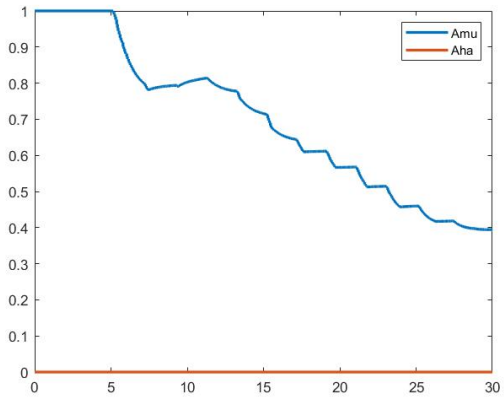


Figure 11: Activation function when the minimum altitude is set at 10 meters

Here again, in Figure 10, we can notice the conflict between these tasks, which is increasingly pronounced. Indeed, in this case the final position has the z component z≃-20m.

1.2.4 Q4: How was the sensor distance processed to obtain the altitude measurement? Does it work in all cases or some underlying assumptions are implicitly made?

As said in the question 1.2.2 above, the sensor distance is projected along the vertical axis of the world frame $< w >$ projected on $< v >$.

$$^{v}x(\underline{c}) = ^{v}\underline{\hat{k}}_{w} \cdot \begin{bmatrix} 0 \\ 0 \\ d \end{bmatrix}$$

Assume that, the sensor distance is a "ray" starting from the bottom of the vehicle and perpendicular to the plane $< i_v, j_v >$ of the vehicle frame (so, along the k axis of $< v >$); the altitude recorded from the vehicle, is the distance between the origin of the sensor attached to the bottom of the vehicle and the seafloor.

The main assumption is that, the altitude measurement only depends on the vertical motion of the vehicle, which is not generally true. In fact, the sensor distance can be represented using the Grassman rule:

$$\underline{d} = (P - O_s)$$

where P is the hit point of the "ray" with the seafloor, and O_s is the origin of the "ray". The derivative includes both the derivative of the origin of the sensor *and* the derivative of the point O_s, which should be estimated in some way.
The Jacobian relation used before is:

$$\dot{\underline{x}} = {}^{v}J_{\text{ma}}{}^{v}\underline{y}$$

This way to derive that vector doesn't take into account the other part of the derivative, because if the vertical velocity of the vehicle is not changing, but O_s is moving, the robot could crash on the seafloor.

Exercise 1 part 2 on GitHub

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 to implement the Landing Action. Comment how you choose the priority level for the altitude control task.

Here are the priorities assigned to each task in the set:

Task	Type	$\mathcal{A}_1$
Minimum Attitude	I,S	-
Horizontal Attitude	I,S	1
Zero Altitude Control	E,AD	2
Vehicle Position Control	E,AD	-
Vehicle Orientation Control	E,AD	-

The altitude control task is not a safety task as the horizontal attitude, so it should have a lower priority than the horizontal attitude.

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

The Jacobian is the same for the minimum altitude control task, shown above in the 1.2.2 .

$$^{v}J_{\text{ma}} = \begin{bmatrix} O_{1\times 17} & ^{v}\hat{\underline{k}}_{w} & O_{1\times 3} \end{bmatrix}$$

With the same limitations showed in the answer 1.2.4, negligible in this case, except when the system tries to perform a landing over a not horizontal seafloor (this is an implicit assumption).

The task reference is computed in a way similar to the one shown for the minimum altitude task, changing the "target altitude" to zero and with the activation function constantly 1 as equality task.

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

As said in the previous question, the two tasks are different in the "target altitude" which is zero for the altitude task and non zero for the minimum altitude task. Another difference lies in the activation function: the altitude task has an activation function which is always 1 (disregarding the mission control so far). The Jacobian relationship is the same in both cases.

Exercise 2 part 1 on GitHub

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.

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$
Minimum Attitude	I,S	1	-
Horizontal Attitude	I,S	2	1
Zero Altitude Control	E,AD	-	2
Vehicle Position Control	E,AD	3	-
Vehicle Orientation Control	E,AD	4	-

Since this mission is composed by two phases, which are the "safe waypoint navigation" and the "landing", we need to modify the hierarchy table: there will be two columns, instead of one, representing the two distinct actions of which the mission is composed. In particular, we denote the "safe waypoint navigation" phase as action $\mathcal{A}_1$ and the "landing" phase as action $\mathcal{A}_2$.
In the first phase of the mission, we need all the safety tasks that were previously defined, such as the Minimum Altitude and the Horizontal Attitude tasks and, of course, we must consider the action defining tasks, such as the Vehicle Position and Orientation Control, in order to make the vehicle able to reach the target point from where, the landing action, will start.
In the second phase instead, we need to disable the Minimum Altitude safety task since, we are starting to perform the landing action and we are no more interested in maintaining a minimum altitude from the seafloor. Of course, we are no longer interested in the Vehicle Position and Orientation Control tasks either, and those, will be substituted by the Zero Altitude Control objective, which represents the action defining task in this second phase.

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

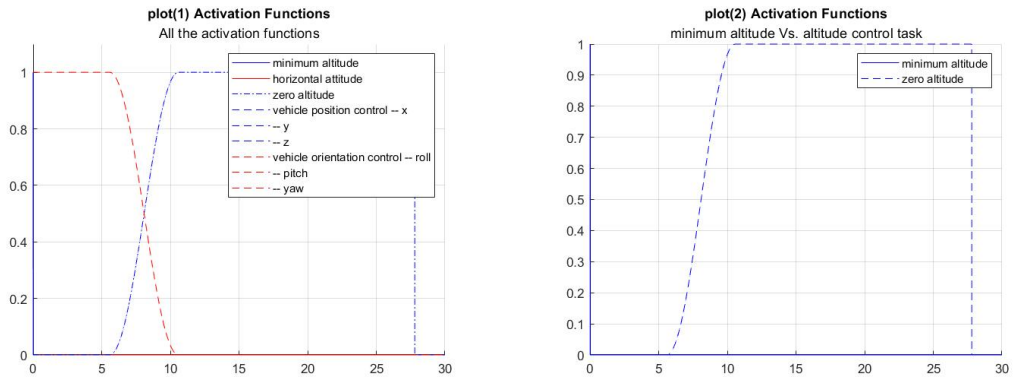
In this case, to ensure that the mission has been accomplished, the ideal would be to implement a state machine composed by four different states: the first state will coincide with the action $\mathcal{A}_1$, then in the second state we have a transition from the action $\mathcal{A}_1$ to the action $\mathcal{A}_2$, the third state coincides with action $\mathcal{A}_2$, and finally the fourth state is when the robot stops when it lands on the seafloor.
In particular, in the first state, we have implemented a control on the distance and the misalignment between the vehicle frame and the target frame, using the method CartError(), and imposing two thresholds which, once passed, ensure the attainment of the target position.
After that, there is the transition phase, in which the vehicle switches from action $\mathcal{A}_1$ to the action $\mathcal{A}_2$ in a smooth way. To enable such transition we need to modify the activation functions that needs to follow this rule:

$$a(x,p) = a^i(x)a^p(p)$$

in which $a^p(p)$ is a sigmoidal function used to obtain a smooth transition when a task is enabled or disabled, switching from an action to the other. In our case:

Task	Previous Action $\mathcal{A}_1$	Current Action $\mathcal{A}_2$	$a^p(p)$
Minimum Attitude	active	inactive	DecreasingBellShapedFunction()
Horizontal Attitude	active	active	1
Zero Altitude Control	inactive	active	IncreasingBellShapedFunction()
Vehicle Position Control	active	inactive	DecreasingBellShapedFunction()
Vehicle Orientation Control	active	inactive	DecreasingBellShapedFunction()

And this smooth transition from action $\mathcal{A}_1$ to action $\mathcal{A}_2$ can be observed in Figure 12:



(a) *All activation functions* (b) *Minimum Altitude and Zero Altitude*

Figure 12: Activation functions of the vehicle performing the mission

Moreover, it is shown the overall evolution of the mission in Figure 13 and 14:

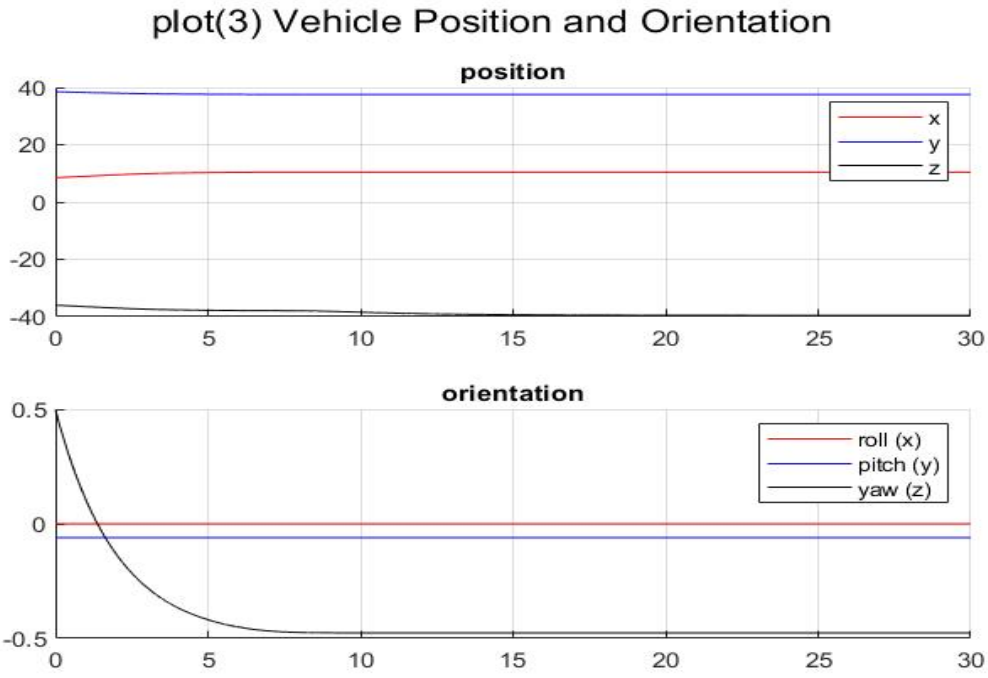


Figure 13: Motion of the vehicle performing the mission

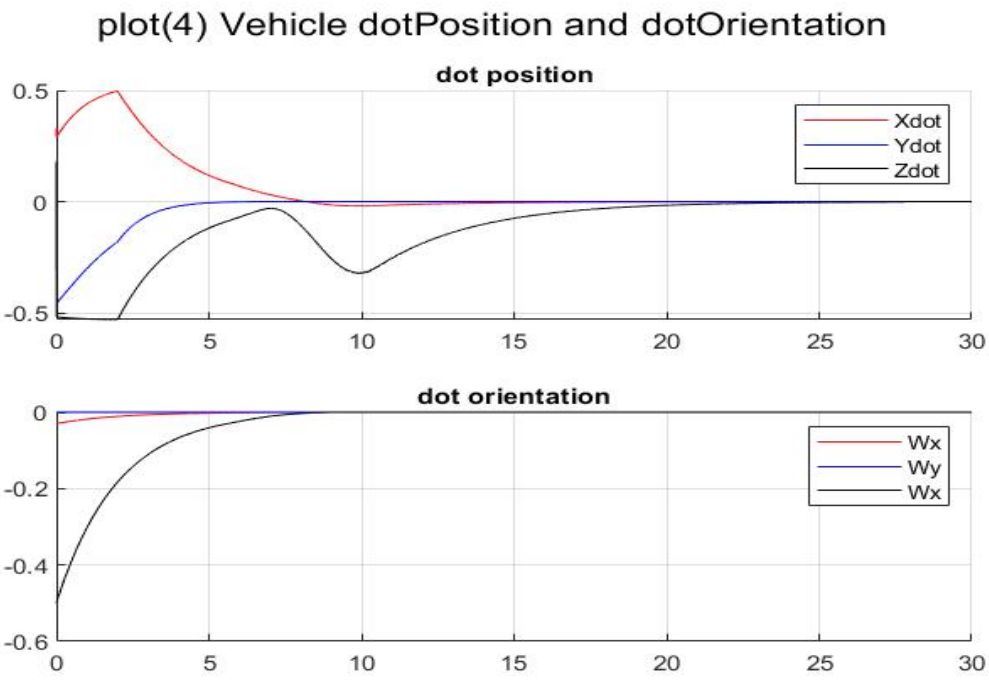


Figure 14: Velocity of the vehicle performing the mission

Exercise 2 part 2 on GitHub

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 from 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 hierarchy of tasks used and their priorities in each action. Comment the behaviour.

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$
Minimum Attitude	I,S	1	-
Horizontal Attitude	I,S	2	1
Vehicle Alignment to the Target	I,P	-	2
Zero Altitude Control	E,AD	-	3
Vehicle Position Control	E,AD	3	-
Vehicle Orientation Control	E,AD	4	-

In this case, we have the same behavior and the same sequence of tasks concerning the action $\mathcal{A}_1$ of the "safe waypoint navigation". Instead, if we focus on the action $\mathcal{A}_2$ of the "landing", we need to add a task that controls the alignment to the target that, in this case, is represented by the rock in the simulation. The "Vehicle Alignment to the Target" is an objective of inequality type and it belongs to the category of the prerequisite tasks. In the hierarchy those type of tasks, come after the safety tasks but, before the action defining tasks.
The behavior, after introducing this last task, would be again divided in two distinct phases.
In the first phase, we have as before the "safe waypoint navigation", until we reach the target point, where the robot will start performing the landing action from.
In the second phase, unlike before, the robot will try first to align its longitudinal axis to the target and then, starts to perform the landing action (due to their positions in the hierarchy).

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

In a nutshell, the main idea is to consider the projections of i_v and the distance vector from O_v to the rock center (in the code, *align_target*), in order to avoid undesired behaviours, such as the tilt of the vehicle.
Here are the steps to compute the Jacobian matrix. First of all, taking into account that the simulation environment makes available the projected point ${}^wO_{\text{rock}}$, we need the two projections along the plane with main axis k_w:

$${}^wi_{v,orth} = (I_{3\times3} - k_w \cdot k_w^T)\,{}^wR_v\,{}^vi_v$$

$${}^w\underline{d}_{orth} = (I_{3\times3} - k_w \cdot k_w^T)({}^wO_{\text{rock}} - {}^w\,O_v)$$

In particular, in order to apply the reduced versor lemma, we need the versor from ${}^w\underline{d}_{orth}$, which can be computed simply dividing ${}^w\underline{d}_{orth}$ for its modulus. Let's call it ${}^w\underline{n}$.

The system can consider the misalignment vector between the two vectors on the same plane. The misalignment will have this form: $\underline{\rho}_{i,n} = \underline{v}\theta$.
the two vectors belong to the same plane, so we can apply the reduced versor lemma to find out the axis of the vector $\underline{\rho}$:

$$^{w}\underline{v} = \frac{1}{sin(\theta)}(^{w}i_{v,orth} \wedge {}^{w}\underline{n})$$

where the angle θ can be gathered using the properties of the scalar product and the cross product. Notice that, the relation holds only if $\theta \neq 0, \pi, 2\pi, ..., k\pi$ ....
In order to obtain the task Jacobian, we can choose to control the angle only, using the parallel component of the derivative of $\underline{\rho}$ parallel:

$$\dot{\theta} = (^{w}\underline{v}\cdot)^{w}\underline{\omega}_{v/w}$$

In the end, the final Jacobian is the following, also taking into account that the system provides the projection of the $\underline{\dot{y}}$ on the vehicle frame:

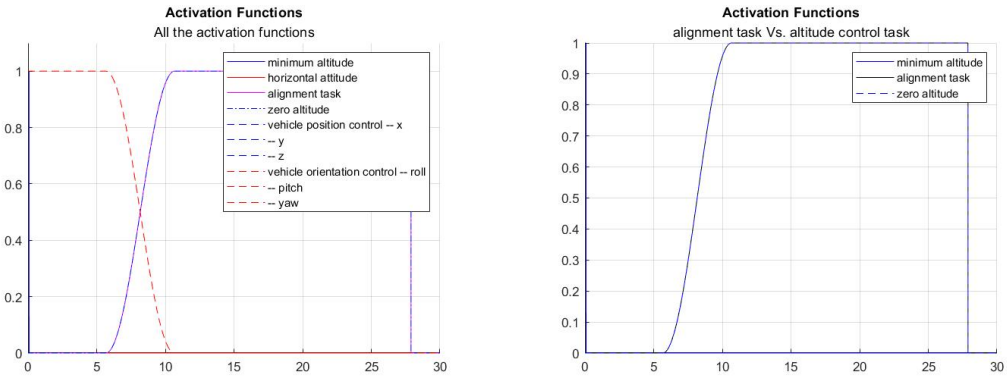
$$^{w}J_{rock} = \begin{bmatrix} O_{1\times 7} & O_{1\times 3} & ^{w}\underline{v}^{T\,w}R_{v} \end{bmatrix}$$

The task reference here is quite similar to the one employed for the horizontal attitude task, but evaluating a different angle, which is the one between the two projections on the plane orthogonal to k_w. The mathematical form is the following:

$$\dot{\bar{\theta}} = \lambda_{\text{align target}}(0 - |\underline{\rho}_{i,n}|)$$

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? Could you devise a solution that is gain-independent guaranteeing that the landing is accomplished aligned to the target?

- Gain = 0.01



(a) *All activation functions* (b) *Alignment task and Zero Altitude*

Figure 15: Activation functions of the vehicle performing the mission

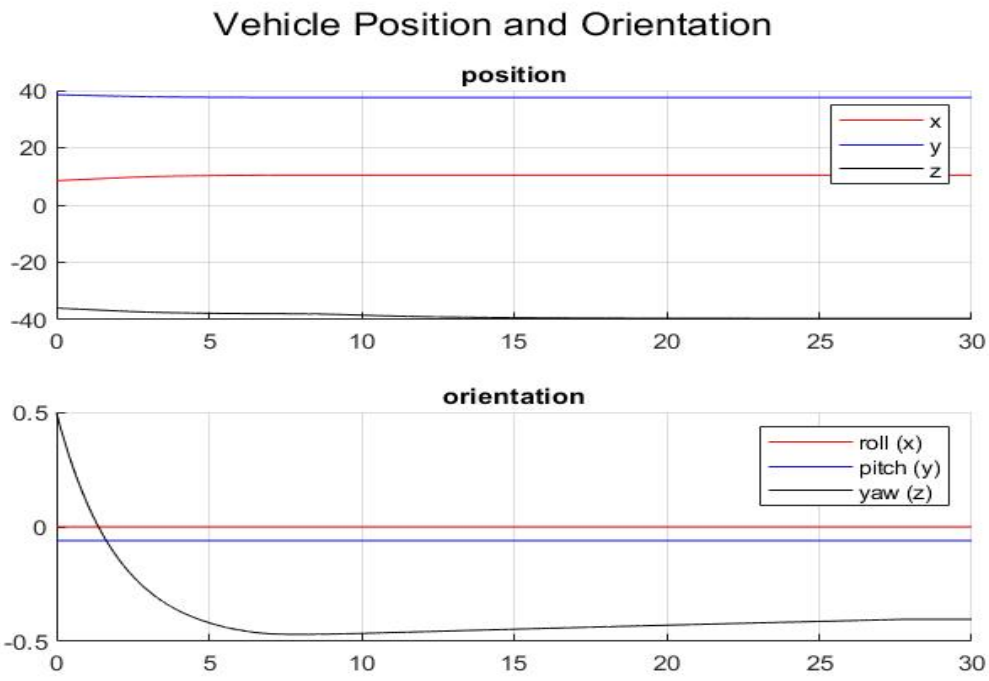


Figure 16: Motion of the vehicle performing the mission

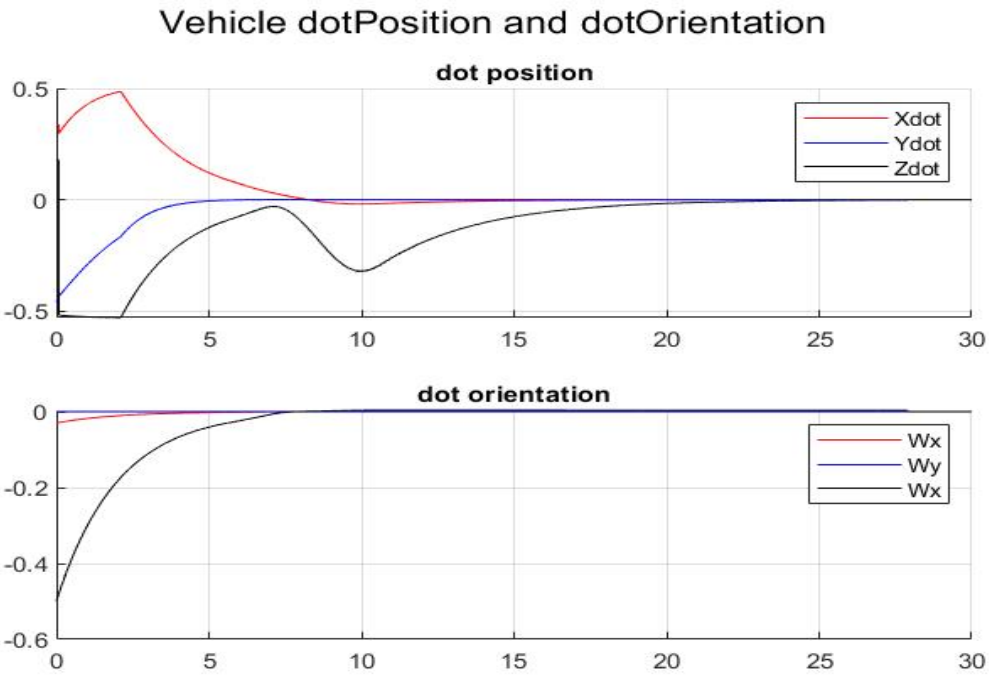
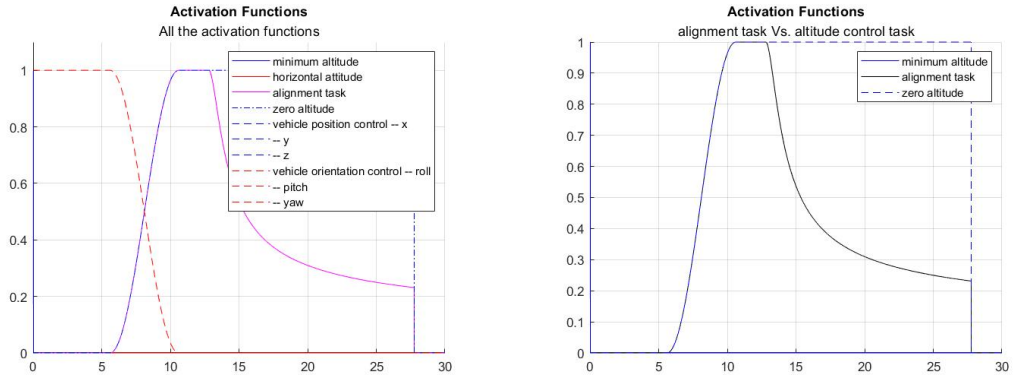


Figure 17: Motion of the vehicle performing the mission

In this case, since we have chosen a very small gain, the task of the vehicle alignment to the target is not even fulfilled. This slow trend can be noticed in all the above graphs. Indeed, in the activation function graphs (Figure 15), we can see that its value remains fixed at 1, when instead it should decrease. Also in Figure 16, we can notice that the yaw angle changes its value significantly only in the first phase, as well as it happens for its derivative in Figure 17.

- Gain = 0.5



(a) *All activation functions* (b) *Alignment task and Zero Altitude*

Figure 18: Activation functions of the vehicle performing the mission

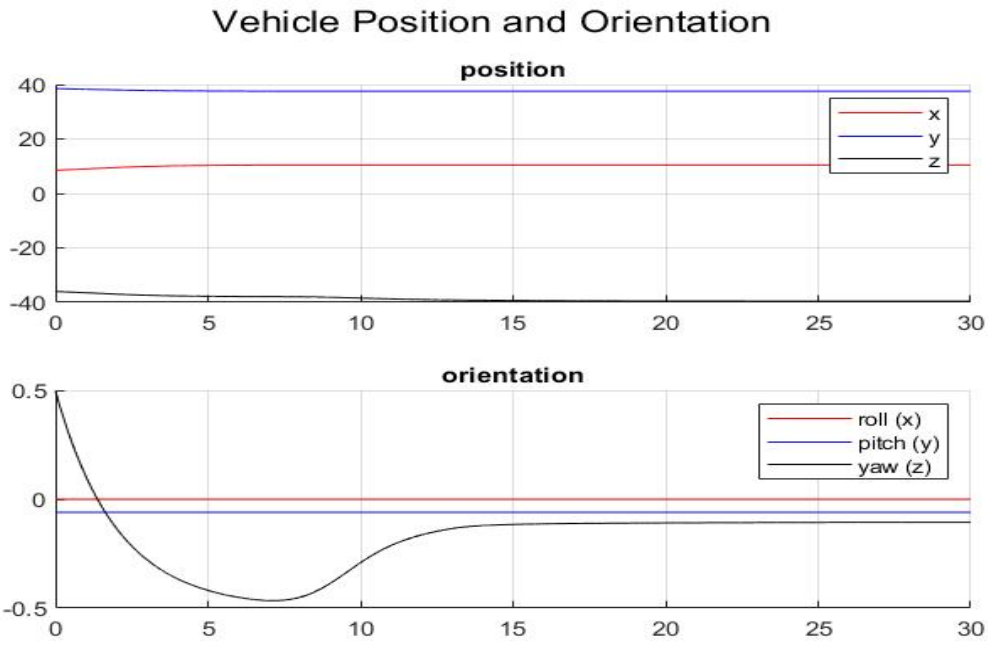


Figure 19: Motion of the vehicle performing the mission

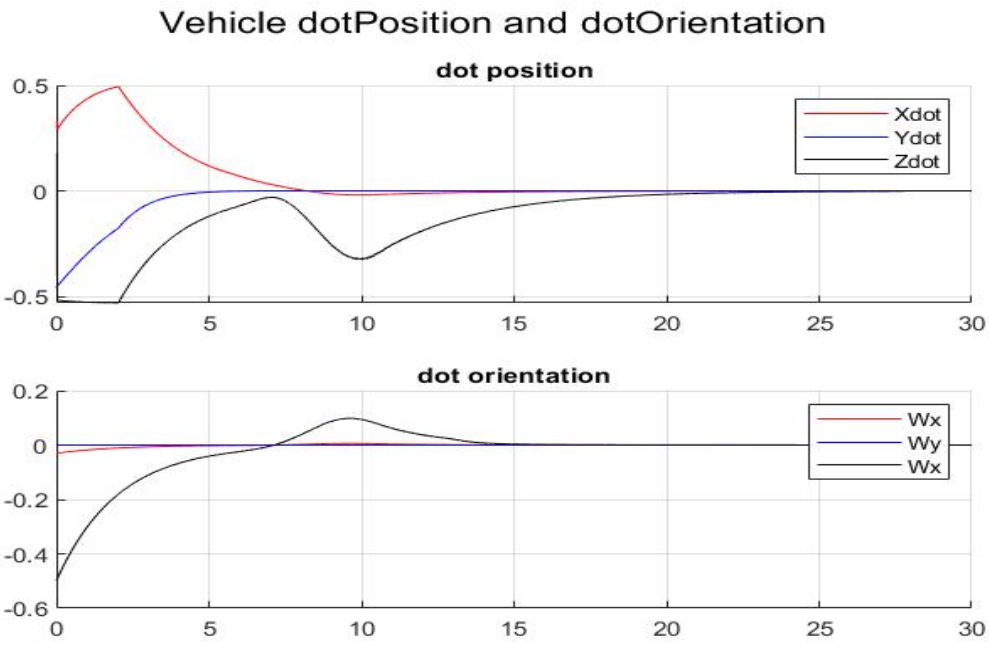
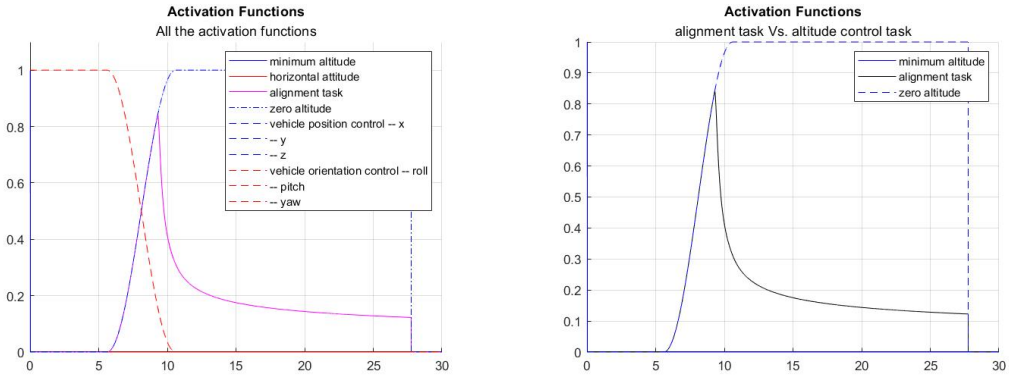


Figure 20: Motion of the vehicle performing the mission

In this case instead, since the value of the chosen gain is reasonable, we can see in the simulation that, indeed, the task is accomplished during the mission.
The behavior can be seen again looking at the above graphs.
In Figure 18, the activation function goes at 1 when the task needs to be active and then decreases smoothly. Then in figure 19 and 20, we can see the variation of the yaw angle and its derivative, witnessing the fact that, the vehicle aligns itself to the target during the landing action.

- Gain = 3



(a) *All activation functions* (b) *Alignment task and Zero Altitude*

Figure 21: Activation functions of the vehicle performing the mission

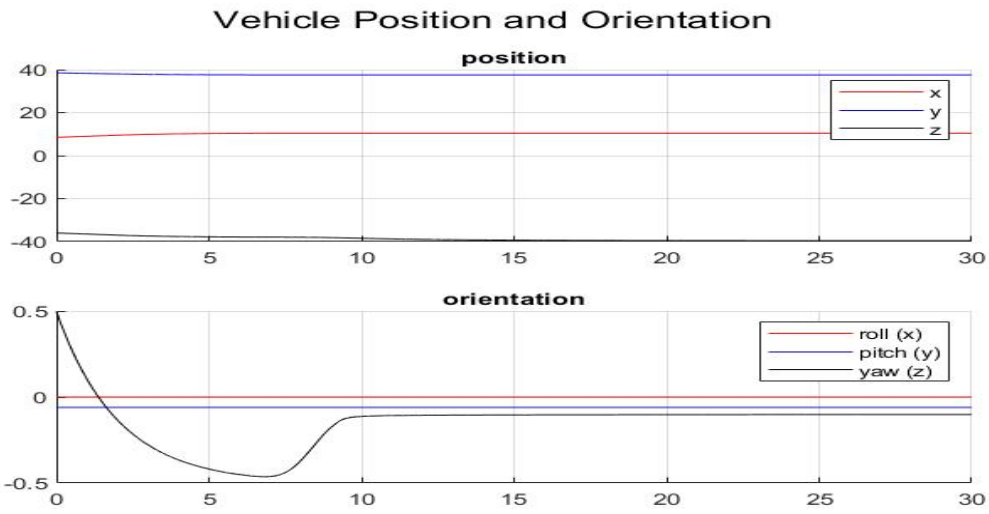


Figure 22: Motion of the vehicle performing the mission

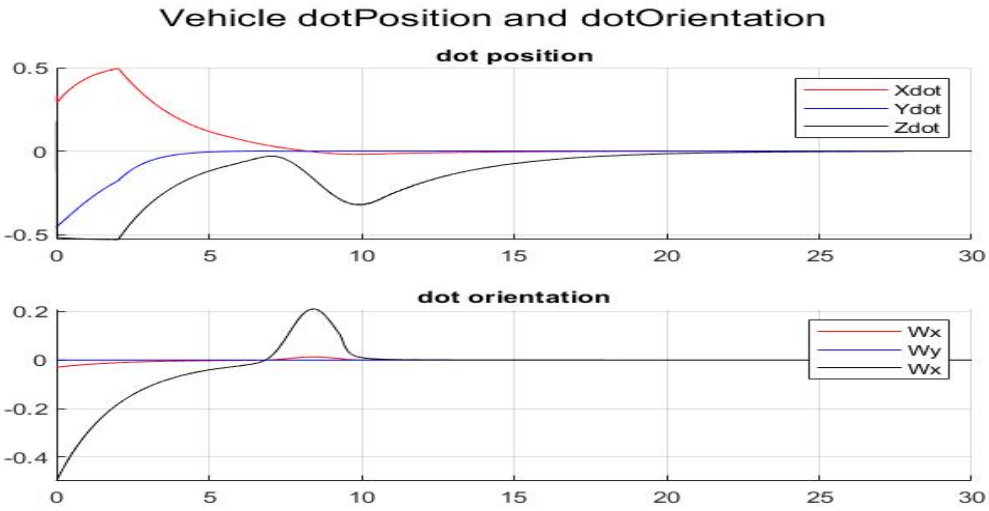


Figure 23: Motion of the vehicle performing the mission

In this final case, we have the same behavior as before since, also in this case, the task succeeds. The only difference is the speed with which this task is accomplished, indeed, in this case the execution is faster than before.The bigger the gain, the faster the evolution.

The only solution for which this task can be considered gain-independent, in the sense that the whole mission is accomplished, guaranteeing that the alignment objective is executed, even if it has a very small gain, would be to devote a whole action for this task. Indeed, there will be an action $\mathcal{A}_1$ in which the vehicle performs the "safe waypoint navigation", an action $\mathcal{A}_2$ in which the robot aligns its longitudinal axis to the target, and finally an action $\mathcal{A}_3$ in which the robot performs the landing phase.
With this mission structure the mission would have a hierarchy of this type:

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
Minimum Attitude	I,S	1	1	-
Horizontal Attitude	I,S	2	2	1
Vehicle Alignment to the Target	I,P	-	3	-
Zero Altitude Control	E,AD	-	-	2
Vehicle Position Control	E,AD	3	-	-
Vehicle Orientation Control	E,AD	4	-	-

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. If, after landing, the nodule is not in the manipulator's workspace, what was missing in the previous actions? How would you fix it?

(GitHub) Trying to move the end-effector

If we simply add an action $\mathcal{A}_3$, that consists of the movement of the manipulator after the landing phase, we can notice that the whole robot follows the motion of the manipulator. To avoid that kind of behaviour, we should add a task that would constraint the motion of the vehicle during the grasping phase, in a way that the action can be accomplished in a safer way and also more precisely.
As shown in the simulation, the distance between the point in which the robot lands and the one where the rock is located is sufficient to permit to the end-effector to reach the goal position. However, it is not always the case since, if we were in a real-case scenario, the position of the the rock will be unknown. In real cases like that, we should have an additional task that controls the distance between the robot and the target, after the desired target has been detected by the sensors of the robot.

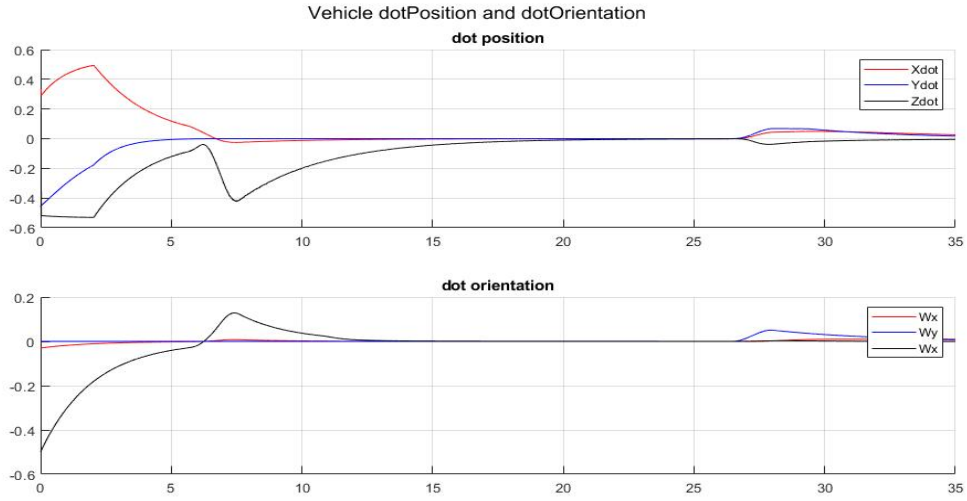


Figure 24: Motion of the vehicle performing the mission

The behavior that for sure needs to be correct, is the one related to the motion of the robot after the landing phase, as we can see in Figure 24.
If, after the landing phase, the rock is not in the manipulator workspace, it means that, in our case, we had given the wrong position where the robot should land at. In real-case scenario instead, as said before, since we don't know the position of the rock until we detect it with the robot sensor, we should add a control on the distance between the robot current position and the target that has been identified.

Exercise 3 part 1 on GitHub

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 hierarchy of tasks used and their priorities in each action. At which priority level did you add the constraint task?

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
Zero Velocity Constraint	E,C	-	-	1
Minimum Attitude	I,S	1	-	-
Horizontal Attitude	I,S	2	1	-
Vehicle Alignment to the Target	I,P	-	2	2
Arm Manipulation Task	E,AD	-	-	3
Zero Altitude Control	E,AD	-	3	-
Vehicle Position Control	E,AD	3	-	
Vehicle Orientation Control	E,AD	4	-	-

The task of Zero Velocity, that has been introduced, is an objective of constraint type. That means that its importance is even higher than the safety tasks and, for that reason, it would be placed in the highest position in the hierarchy. This constraint task should be considered only in action $\mathcal{A}_3$ of grasping since, in the previous phases of the mission, is not required a behaviour like that.

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

This constraint aims at making null both the linear and the angular velocity of the vehicle origin. We chose to compute the task Jacobian with respect to the world frame, hence the Jacobian is:

$$ {}^{w}J_{zvc} = \begin{bmatrix} O_{6\times 7} & \begin{bmatrix} {}^{w}R_{v} \\ O_{3\times 3} \end{bmatrix} & \begin{bmatrix} O_{3\times 3} \\ {}^{w}R_{v} \end{bmatrix} \end{bmatrix} $$

The task reference is computed as below:

$$ \dot{\tilde{x}} = \lambda_{zvc} \cdot (\underline{0} - \begin{bmatrix} {}^{w}\underline{v}_{v/w} \\ {}^{w}\underline{\omega}_{v/w} \end{bmatrix}) $$

where, in the simulation environment, $\lambda_{zvc} = 0.5$.

Exercise 4 part 1 on GitHub

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 hierarchy of tasks used and their priorities in each action. At which priority level did you add the joint limits task?

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
Zero Velocity Constraint	E,C	-	-	1
Joint Limit Constraint	I,C	-	-	2
Minimum Attitude	I,S	1	-	-
Horizontal Attitude	I,S	2	1	-
Vehicle Alignment to the Target	I,P	-	2	3
Arm Manipulation Task	E,AD	-	-	4
Zero Altitude Control	E,AD	-	3	-
Vehicle Position Control	E,AD	3	-	
Vehicle Orientation Control	E,AD	4	-	-

The task of Joint Limit, that has been introduced, is an objective of constraint type. That means that its importance is higher than the safety tasks and, for that reason, it would be placed in the highest position in the hierarchy. However, in action $\mathcal{A}_3$, where it should be taken in account, we already have a constraint task, concerning the zero velocity.

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

The main idea is to constraint thee joints to "orbit" about a average configuration, depending on the bounds given in the simulation. Let $q \in \mathbb{R}^7$ be the current configuration, and $\dot{q}$ its derivative, which is also part of the projected vector ${}^v\dot{y}$, rows from 1 to 7. In order to apply whatever control on the robotic manipulator, the simplest idea is to directly take the vector $\dot{q}$, hence the Task Jacobian referred to the Joint Limit Constraint task will be:

$$ {}^vJ_{jlc} = \begin{bmatrix} I_{6\times 7} & O_{6\times 6} \end{bmatrix} $$

Let q_m be the average configuration, which is a simple average between the vectors q_{max} and q_{min}, which are respectively the maximum values and the minimum values joint by joint. The average will be:

$$ q_m = \frac{1}{2}(q_{max} + q_{min}) $$

The objective is to keep the joint positions around q_m but without constraining too much the movements of the arm, as well as without letting it to overstep the limits. Hence, the chosen task reference in this case is the following:

$$ \dot{\tilde{x}}_{jlc} = \lambda_{jlc} \cdot (q_m - q) $$

Here is our choice for the activation function, which considers the module of the current configuration, component by component since each joint has its own excursion. Let ϵ be a $\mathbb{R}^7$ row vector, whose each coordinate is half of the radius of the neighborhood of $q_{m,i}$ the i-th component of the vector q_m. Let Δq be the difference between the two boundaries q_{max} and q_{min}; ϵ depends on Δq through the formula

$$ \epsilon = m \cdot \Delta q $$

where m is a percentage, tuned by trial and error; in our simulation, we chose $m = 0.9$, a pretty large percentage, since we don't want to constraint too much the motion of the system, even taking into account the priority the Joint Limit Task will have at execution time.

The shape of the activation function is quite similar to the following one:

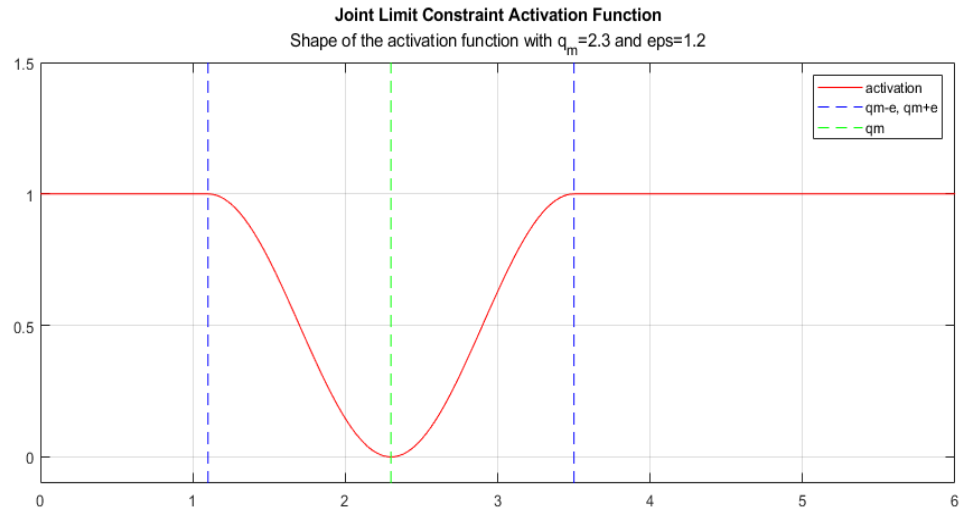


Figure 25: Motion of the vehicle performing the mission

and it is computed once for each component at each frame, since each joint has its own boundary using the absolute value of the given joint value.

Exercise 4 part 2 on GitHub

5 Exercise 5: Floating Manipulation

Use the DexROV simulation for this exercise.

5.1 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.

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

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$
Zero Velocity Constraint	E,C	-	1
Joint Limit Constraint	I,C	-	2
Horizontal Attitude	I,S	1	-
Arm Manipulation Task	E,AD	-	3
Vehicle Position Control	E,AD	2	-
Vehicle Orientation Control	E,AD	3	-

- Action $\mathcal{A}_1$: Safe Waypoint Navigation
  - Horizontal Attitude task is an objective of safety type that allows to maintain the robot horizontal with respect to the seafloor
  - Vehicle Position Control task allows the robot to reach the target position
  - Vehicle Orientation Control task allows the robot to reach the target orientation
- Action $\mathcal{A}_2$: Floating Manipulation
  - Zero Velocity task is an objective of constraint type that allows to maintain the robot still during the manipulating action
  - Joint Limit task is an objective of constraint type that avoid to stress the joints maintaining them between a minimum and a maximum range
  - Arm Manipulation task allows the robot's manipulator to move in order to reach the target

5.1.2 Q2: What is the difference if, from the very beginning, you use the action of floating manipulation (i.e. just a single action)?

Using the manipulation task only, while the manipulator is trying to reach the target, the robot tends both to tilt and to move according to the movement of the manipulator, which is definitely not the desired behaviour. In this case, there's no difference between DexRov and Robust: both tend to move as the manipulator tries to move towards a target. See the image below:

Figure 26: Manipulation Task without constraints.

Using the Floating Manipulation action showed above in the table of the previous question, the manipulator tries to reach the target, but the robot is constrained to keep the same position and to not tilt; therefore, the robot reaches a configuration in which only the arm moves, and the end effector is closer as much as possible to the target, but without being able to reach it.

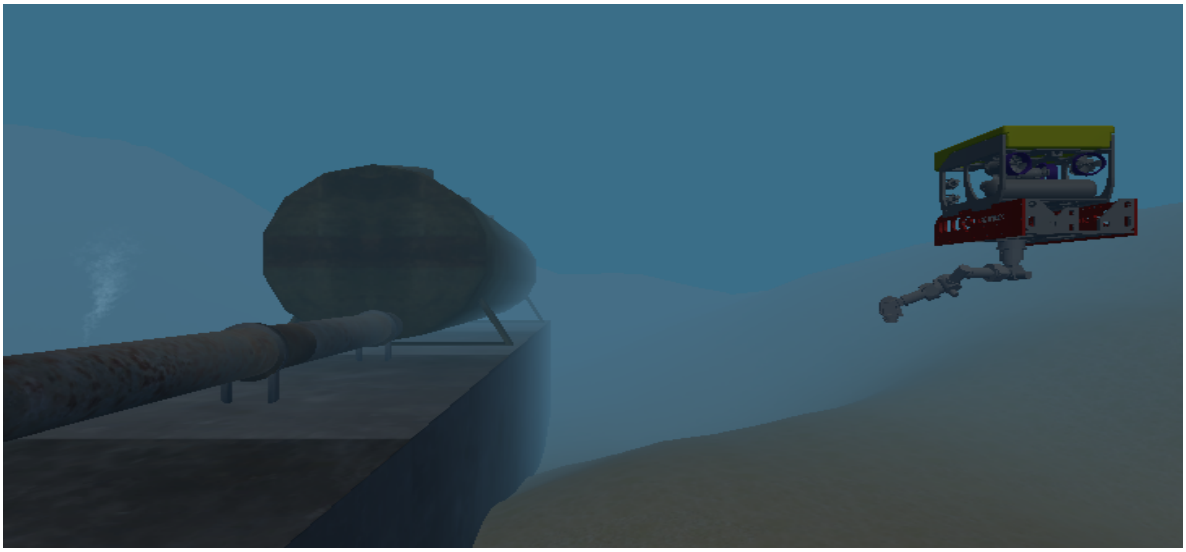


Figure 27: Manipulation Action: zero velocity constraint, joint limit constraint, and manipulation task. the robot is locked in that position, and the target is outside the manipulation manifold.

Exercise 5 part 1 on GitHub

5.2 Adding an optimization control objective

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

5.2.1 Q1: Report the hierarchy of tasks used and their priorities in each action. At which priority level did you add the optimization task?

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$
Zero Velocity Constraint	E,C	-	1
Joint Limit Constraint	I,C	-	2
Horizontal Attitude	I,S	1	-
Arm Manipulation Task	E,AD	-	3
Vehicle Position Control	E,AD	2	-
Vehicle Orientation Control	E,AD	3	-
Joint Preferred Shape Task	I,O	-	4

The Preferred Shape task is an objective of Optimization type and for this reason it should be placed in the lowest position of the hierarchy. This task should be activated only when the robot is performing the manipulation, so only during action $\mathcal{A}_2$.

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

The task Jacobian is quite similar to the one employed for the Arm Manipulation Task but controlling only the first four joints. The task Jacobian is this:

$$^{v}J_{sh} = \begin{bmatrix} I_{6\times 4} & O_{6\times 3} & O_{6\times 6} \end{bmatrix}$$

Also, the task reference is similar, as written here below:

$$\dot{\bar{x}}_{sh} = \lambda_{sh} \cdot (q_{sh} - q)$$

where q_{sh} is the preferred shape, and the vector q contains only the values for the first four joints.

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

If you have this kind of optimization task in your hierarchy, you are trying to force the joints to be in a chosen configuration. This may occur if you want, for instance, to keep your manipulator out of the camera field. Moreover, this can help in maintaining a good dexterity as well. This behavior can be noticed looking at the Figure 28 and Figure 29 below, and, in particular, focusing on the evolution of the first four joints.
Instead, if you do not have this kind of objective, the manipulator will arrive to the target position with the joint configuration that is more convenient and makes the end-effector following the trajectory it wants.

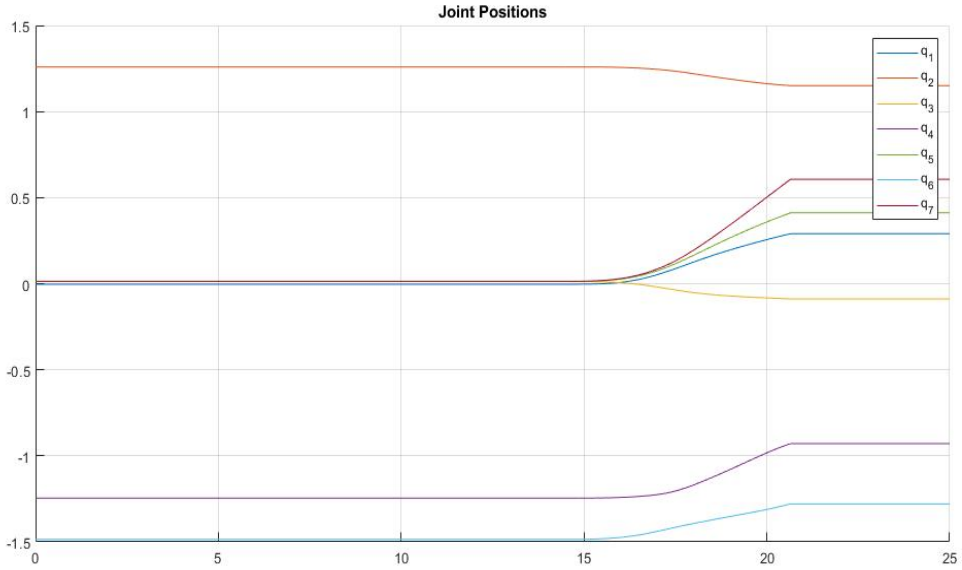


Figure 28: Joint position when Preferred Shape not enabled

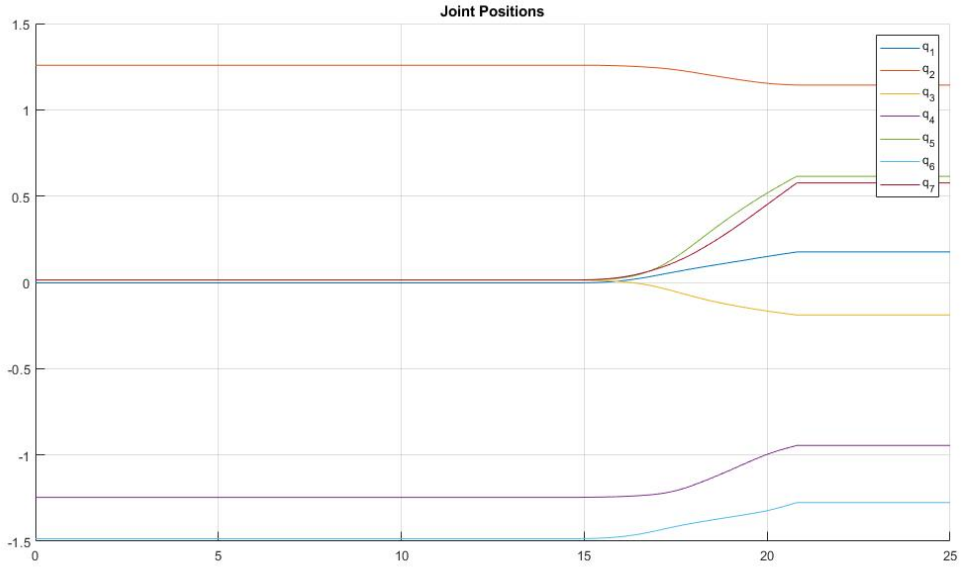


Figure 29: Joint position when Preferred Shape enabled

Exercise 5 part 2 on GitHub

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

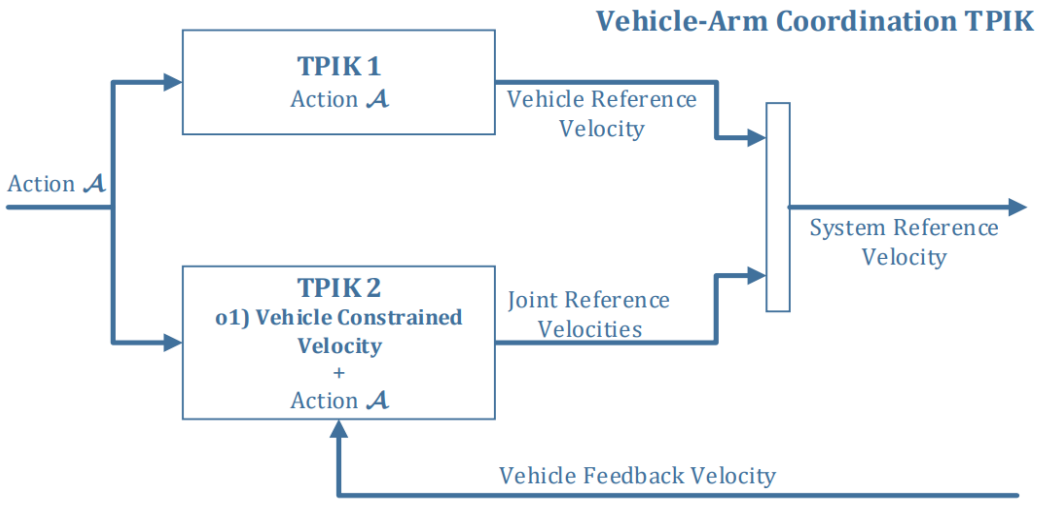
6.1 Adding the parallel arm-vehicle coordination scheme

Let us now see how the two different subsystems (arm and vehicle) can be properly coordinated. Introduce in the simulation a sinusoidal linear velocity disturbance acting on the vehicle (along a constant inertial x-y direction of your choice), 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?

Now the vehicle is affected by a regular sinusoidal disturbance acting on the pitch. In order to enable the robot to compensate this disturbance, in this situation, we have no longer an overall system, but the union of two systems, which have to collaborate each other. The new architecture is the one showed in the following schema:



The "upper" Task Priority Inverse Kinematics block (TPIK1) controls the vehicle only. In particular, the first one implements the tasks showed in the previous questions, plus a non reactive task set at maximum priority, which is in charge to compensate the vehicle oscillation.

The first block generates the velocity of the vehicle given the actual velocity, assumed to be observable, and the second one, TPIK2, controls the manipulator. The uppermost task constrains the arm to compensate the motion of the vehicle, establishing a collaboration between the two systems.

6.1.2 Q2: Show the plot of the position of the end-effector, showing that it is constant. Show also a plot of the velocities of the vehicle and of the arm.

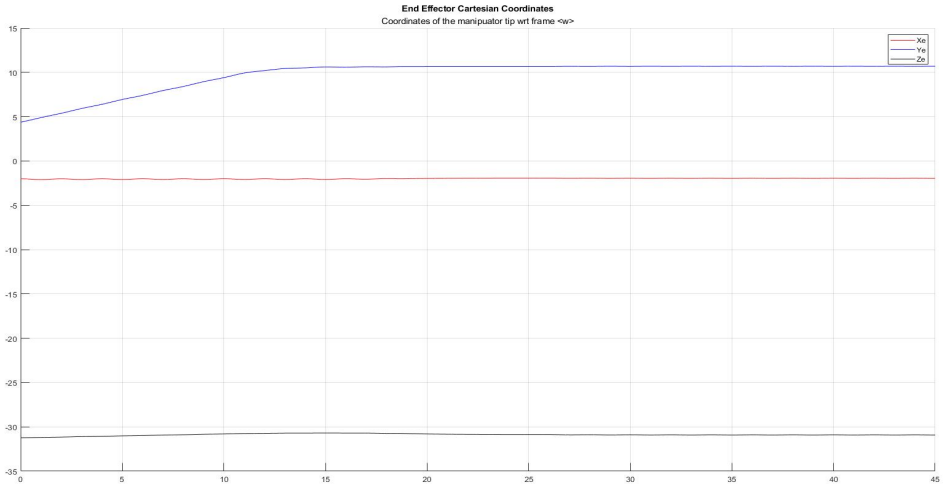


Figure 30: End-effector position during the mission

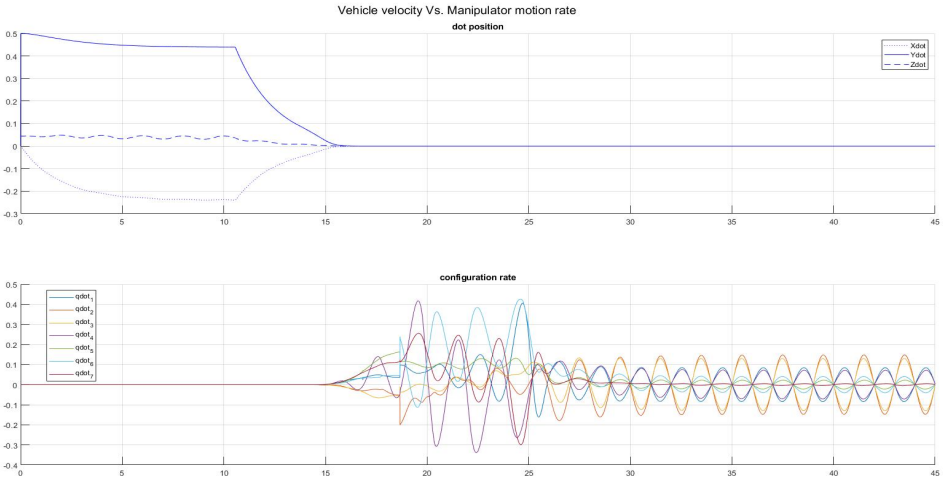


Figure 31: Linear velocity of the vehicle and velocities of the joints

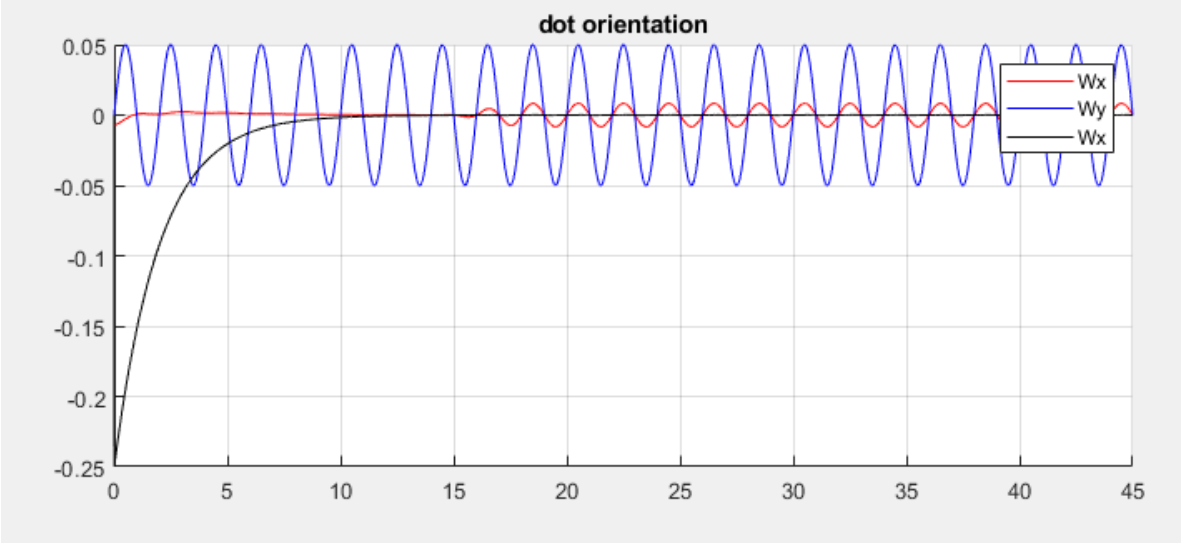


Figure 32: Angular velocity of the vehicle

In Figure 30, we can observe that the position of the end-effector becomes constant after about 25 seconds. Indeed, this is the amount of time needed by the robot to reach the target, accomplishing the mission. And, even if there is a disturbance, the position of the end-effector doesn't change.
Then, in Figure 31, the first graph shows that, after the safe waypoint navigation, the vehicle doesn't move from its target position. In the second graph, instead, there are shown the velocities of the joints during the mission. Noteworthy pointing out that, during the manipulation, these velocities assume a sinusoidal behavior, due to the fact that, the manipulator is trying to counterbalance the disturbance that has been added.
In the end, in Figure 32, looking at the angular velocity of the vehicle, we can see that only w_y presents a sinusoidal behavior, since the disturbance has been added to this particular component.

6.1.3 Q3: What happens if the sinusoidal disturbance becomes too big? Try increasing the saturation value of the end-effector task if it is too low.

As the intensity of the sinusoidal disturbance increases, the joints tend to oscillate following the disturbance, in order to keep the end effector as much near as possible to the point.
Let's suppose a big disturbance: beyond a certain intensity, the most immediate consequence is that the joint can no longer keep the position, because of, for instance, the joint limit constraint, which doesn't allow the arm to reach certain configurations needed to compensate the disturbance, hence also the end effector starts to oscillate, compromising the mission flow.
Another consequence is that, the arm could fall into singularity: consider for example the situation in which the robot is tilted about its pitch of 90 degrees; the arm is not geometrically able to reach the point.
In other words, the approach implemented before works well with low-intensity disturbances, but as the disturbance increases its intensity, the robot becomes less and less capable to make the end effector keeping a position, startig oscillating around it; and beyond a certain intensity, it becomes almost impossible to keep the end effector around the point.

Exercise 6 part 1 on GitHub