Cooperative Robotics

Authors: Astha Gupta, Francesca Cantoni
EMAILs: astha736@gmail.com, francescacantoni95@gmail.com
Date: January 27, 2020

General concepts and notations

The configuration of the system is the vector of its DOF parameters.

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

where $\mathbf{q}$ is the configuration of the arm

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

and $\boldsymbol{\eta}$ is the configuration of the vehicle:

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

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

$$\dot{\boldsymbol{y}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{\nu} \end{bmatrix} \quad \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{\nu_1} \\ \boldsymbol{\nu_2} \end{bmatrix} \quad \begin{matrix} \boldsymbol{\nu_1} =^v \boldsymbol{v} \\ \boldsymbol{\nu_2} =^v \boldsymbol{w} \end{matrix} \tag{5}$$

The relationship between the derivative of Euler (RPY) angles and vehicle-base's angular velocity is given the below equation:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan(\theta)\sin(\phi) & \tan(\theta)\cos(\phi) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{bmatrix} \boldsymbol{\nu}_2 \tag{6}$$

- Thus in the code the Jacobian expected is between ${}^w\dot{\boldsymbol{x}}$, in the world frame $< w >$ and ${}^v\dot{\boldsymbol{y}}$, in the vehicle frame $< v >$.

$${}^w\dot{\boldsymbol{x}} = \boldsymbol{J}{}^v\dot{\boldsymbol{y}} \tag{7}$$

- The basic functionality listed below has already been implemented.
  - *"ha"*: Horizontal Altitude
  - *"mu"*: Manipulability
  - *"t"*: Tool Position Control

Structure of the task priority approach

The considered Matlab simulation is based on the following general architecture:

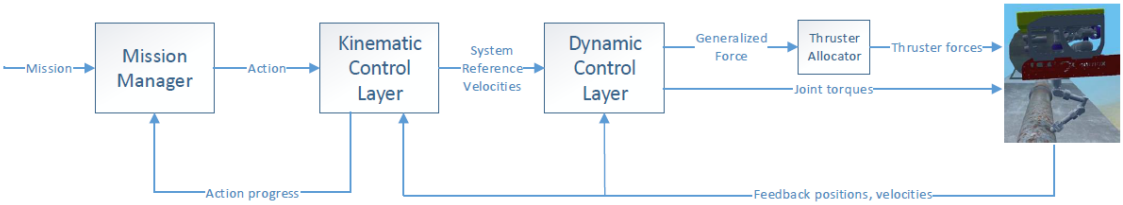


Figure 1: General architecture of the task priority approach

1. **Mission Manager**: Notifies the Kinematic Control Layer about the *actions* that must be executed based on the current *mission*
2. **Kinematic Control Layer**: Implements the task priority control framework and generates the system reference velocities
3. **Dynamic Control Layer**: Tracks the system reference velocities by generating appropriate force/torques references for the vehicle and manipulator

In the simulation we develop the code for the Mission Manager and parts of Kinematic Control Layer.

Control objective

- **Types of Objective**: We have two different types of objectives:
  - *Equality control objective(E)*: Given by $x(\boldsymbol{c}) = x_0$
  - *Inequality control objective(I)*: Given by $x(\boldsymbol{c}) \geq x_{min}$ or $x(\boldsymbol{c}) \leq x_{max}$
- **Categories of objectives**: The objectives can be divided into these five main categories:
  - *Action Defining(AD)*: Action oriented objectives
  - *Safety(S)*: Objectives related to the safety of the robot
  - *Operational Prerequisite(P)*: Objective that are prerequisite for the given action
  - *Optimization(O)*: Optimization objective
  - *Constraints(C)*: Objective related to the physical constraints of the system

Control task

We have two categories of control task:

- *Reactive control task(R)*: Capable of tacking the reference rate $\dot{\boldsymbol{x}}$ generated by a feedback.
- *Non-Reactive control task(NR)*: Defined directly in certain task velocity space, thus the reference velocity tracked is not generated by a feedback.

REMARK: In order to better explain this project the following notation is used:

$$[\text{R/NR},\quad \text{I/E}\quad \text{C/S/P/AD/O}]\quad \text{Name of the task/objective}$$

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. What is the Jacobian relationship for the Vehicle Position control task? How was the task reference computed?

A1: To implement **safe waypoint navigation** the system needs to satisfy the **vehicle position control** (*posc*) objective and some additional tasks. Considering the main categories of the objective the preferable priority for each category is listed below (highest to lowest priority).

1. safety objective
2. operational prerequisite objective
3. action defining objective
4. optimization objective

The hierarchy of the tasks for exercise 1.1 is:

Priority	Category	Description	Name of the objective
1	Prerequisite	[R, I, P]	ha
2	Action-defining	[R, E, AD]	posc

Table 1: Hierarchy of the tasks for "Safe Waypoint Navigation"

Giving the highest priority to *"ha"* asserts that the robot does not flip over given desired reference velocities. This is possible because of the manifold of the solutions for the second minimization problem (*"posc"*) is already constrained by the higher priority task (*"ha"*) given:

$$\dot{\boldsymbol{y}} = \boldsymbol{\rho_1} + \boldsymbol{Q_1}\dot{\boldsymbol{z}}_\mathbf{1} \qquad \forall \dot{\boldsymbol{z}}_\mathbf{1} \tag{8}$$

where the *"posc"* task can only act on the residual arbitrariness given by $\dot{\boldsymbol{z}}_\mathbf{1}$.

If the second task is not able to consume all the residual arbitrariness of $\dot{\boldsymbol{y}}$, the default task (*minimization of the control vector* $\dot{\boldsymbol{y}}$) always at the lowest priority consumes all the residual arbitrariness [1]. Thus the final control law reduces to:

$$\dot{\boldsymbol{y}} = \boldsymbol{\rho_3} \tag{9}$$

Consumption of all the residual arbitrariness for the minimization of the control vector ensures the continuity during task activations.

REMARK: To accomplish *"posc"* objective the Manipulability *"mu"* and the Tool Position-Control *"t"* tasks are not necessary. Hence, they have been disabled. More details can be found in section 1.1.4.

[1] This task has $\boldsymbol{J_k} = \boldsymbol{I}_{13\times13}$, $\boldsymbol{A_k} = \boldsymbol{I}_{13\times13}$ and $\dot{\tilde{\boldsymbol{x}}}_\boldsymbol{k} = 0_{13\times1}$ with k=3 in this specific case.

A2: The Jacobia for *"posc"* task is:

$$ {}^{w}\dot{\boldsymbol{x}}_{\boldsymbol{2}} = {}^{w}\dot{\boldsymbol{x}}_{\boldsymbol{posc}} = \boldsymbol{J}_{\boldsymbol{posc}} \; {}^{v}\dot{\boldsymbol{y}} \tag{10} $$

where $\dot{\boldsymbol{x}}_{\boldsymbol{posc}} \in \mathbb{R}^6$, $\boldsymbol{J}_{\boldsymbol{posc}} \in \mathbb{R}^{6\times 13}$ and $\dot{\boldsymbol{y}} \in \mathbb{R}^{13}$; Jacobian has 13 columns corresponding to the dimension of *control variables* at the kinematic level[2], and 6 rows corresponding to the dimension of *reference velocity*.

$$ {}^{w}\dot{\boldsymbol{x}}_{\boldsymbol{posc}\ 6\times 1} = \begin{bmatrix} {}^{w}\boldsymbol{v}_{\ 3\times 1} \\ {}^{w}\boldsymbol{\omega}_{\ 3\times 1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0}_{\ 3\times 7} & {}^{w}\boldsymbol{R}_{v\ 3\times 3} & \boldsymbol{0}_{\ 3\times 3} \\ \boldsymbol{0}_{\ 3\times 7} & \boldsymbol{0}_{\ 3\times 3} & {}^{w}\boldsymbol{R}_{v\ 3\times 3} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\ 7\times 1} \\ {}^{v}\boldsymbol{v}_{\ 3\times 1} \\ {}^{v}\boldsymbol{\omega}_{\ 3\times 1} \end{bmatrix} \tag{11} $$

A3: Since the *"posc"* task is a *reactive control task*, the task reference is compute using the formula of the *closed loop feedback reference rate* such that:

- required position:

$$ {}^{w}\ \overline{\boldsymbol{v}} = \lambda_l({}^{w}\ \overline{\boldsymbol{x}}_{\boldsymbol{position\_goal}} - {}^{w}\ \boldsymbol{x}_{\boldsymbol{actual\_position}}) \tag{12} $$

- required orientation:

$$ {}^{w}\ \overline{\boldsymbol{\omega}} = \lambda_a \ \ VersorLemma({}^{w}\ \overline{\boldsymbol{x}}_{\boldsymbol{orientation\_goal}}, {}^{w}\ \boldsymbol{x}_{\boldsymbol{actual\_orientation}}) \tag{13} $$

and in a compact form:

$$ {}^{w}\ \dot{\overline{\boldsymbol{x}}} = \begin{bmatrix} \lambda_l & \lambda_r \end{bmatrix} \begin{bmatrix} {}^{w}\delta\ \boldsymbol{r} \\ {}^{w}\delta\ \boldsymbol{\theta} \end{bmatrix} \tag{14} $$

with:

$$ \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{\theta} \end{bmatrix} = CartError\ ({}^{w}\ \overline{\boldsymbol{x}}_{\boldsymbol{position\_goal}},\ {}^{w}\ \boldsymbol{x}_{\boldsymbol{actual\_position}}) \tag{15} $$

We use "*CartError*" function to calculate $\boldsymbol{r}$ and $\boldsymbol{\theta}$ and we set the two gains equal to $\lambda_l = 0.2$ and $\lambda_a = 0.5$.

Conclusion: as expected, Fig. 2, after a reasonable amount of time[3], the position error $\boldsymbol{r}$ and the orientation error $\boldsymbol{\theta}$ converge to zero.

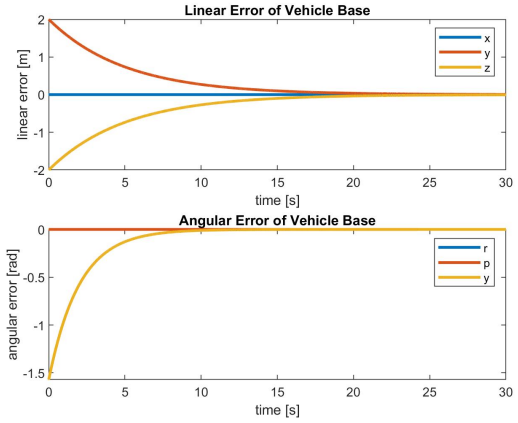


Figure 2: Position error and orientation error of the vehicle

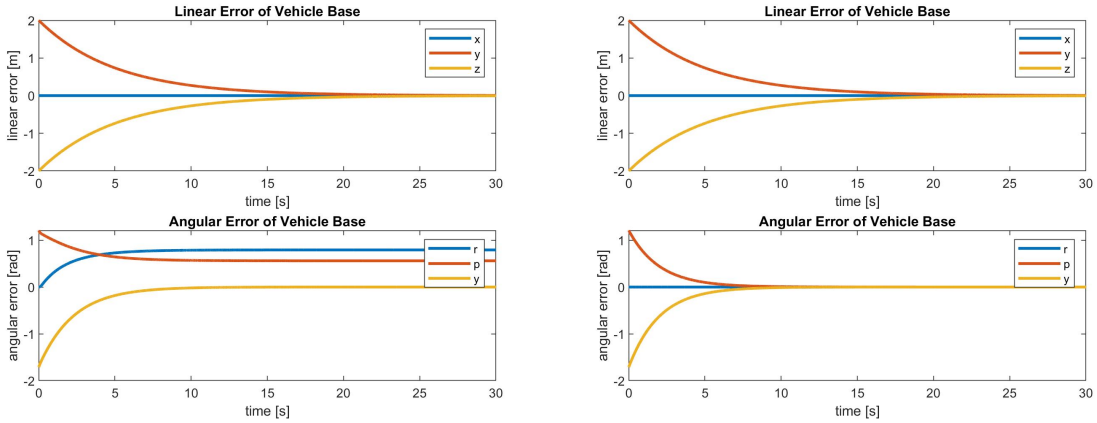
[2]The control variables are 7 joints of the manipulator and 6 D.O.F for base of the vehicle.
[3]The amount of time is function of: difference between the initial and the target positions or λ value.

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.

A: Although the hierarchy described in Table 1 remains the same, by disabling task *"ha"* (setting $A_{ha} = 0$), the task is unable to add any constraints on the velocities of the vehicle. Therefore, the solutions manifold of the *"posc"* minimization problem behaves as if it has the highest priority.

We are now going to compare the two different behaviours by considering:

- initial position: $\begin{bmatrix}10.5 & 35.5 & -36 & 0 & 0 & \pi/2\end{bmatrix}^\top$
- new target position: $\begin{bmatrix}10.5 & 37.5 & -38 & 0 & \pi/4 & \pi/4\end{bmatrix}^\top$



(a) (b)

Figure 3: Position and orientation errors with "ha" active (a) and non active (b)

From Fig. 3 above we notice that:

Fig.3 (a) *"ha"* active: the vehicle is able to achieve goal position but not the goal orientation. This is because the "ha" task constraints the *z-axis* of the vehicle to be aligned with the *z-axis* of the world frame.

Fig.3 (b) *"ha"* non active: since "ha" is not active, the vehicle is able to reach the goal position and goal orientation. Which flips the robot as shown in Fig. 4.

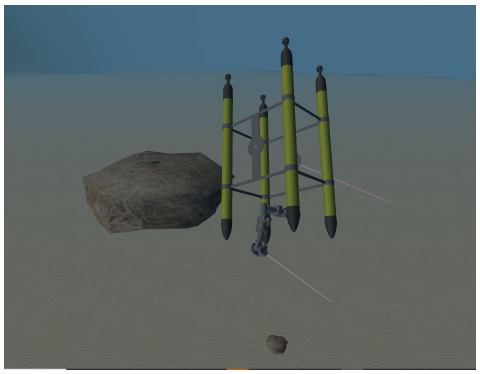


Figure 4: Final position of the robot with "ha" off

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

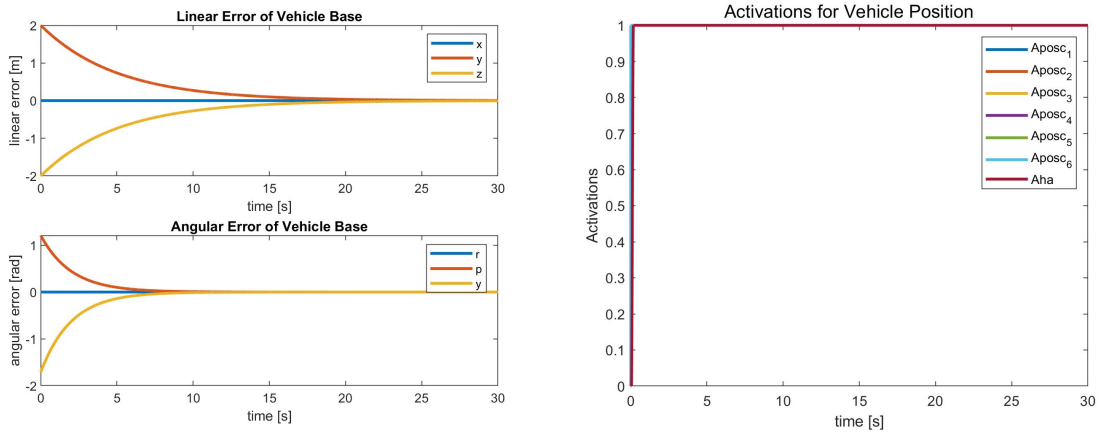
A: Considering the new case we have that Table 1 changes as follow:

Priority	Category	Description	Name of the objective
1	Action-defining	[R, E, AD]	posc
2	Prerequisite	[R, I, P]	ha

Table 2: "Safe Waypoint Navigation" action with swapped priorities of the control tasks

Owing to the inversion of priority between *"posc"* and *"ha"*, the manifold of solutions for the highest priority task, given by eq. (8), agrees with *"posc"* task. Therefore, even when both the tasks are enabled, *"ha"* can only act on the residual arbitrariness left by *"posc"*. This implies that the robot can not ensure that the vehicle fulfills the *"ha"* objective while trying to minimize the orientation error $\boldsymbol{\theta}$. The final behaviour is the same as the one shown in Fig. 4.

Considering the same values for *target position* as in the exercise 1.1.2, we have the following error and activation:



(a) (b)

Figure 5: (a) Position and orientation errors considering swapped priorities and (b) activation functions

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?

A: For this exercise we activate the *"mu"* and *"t"* task, because if the Tool Position Control Task is active we would also like the arm to maintain a good manipulability.

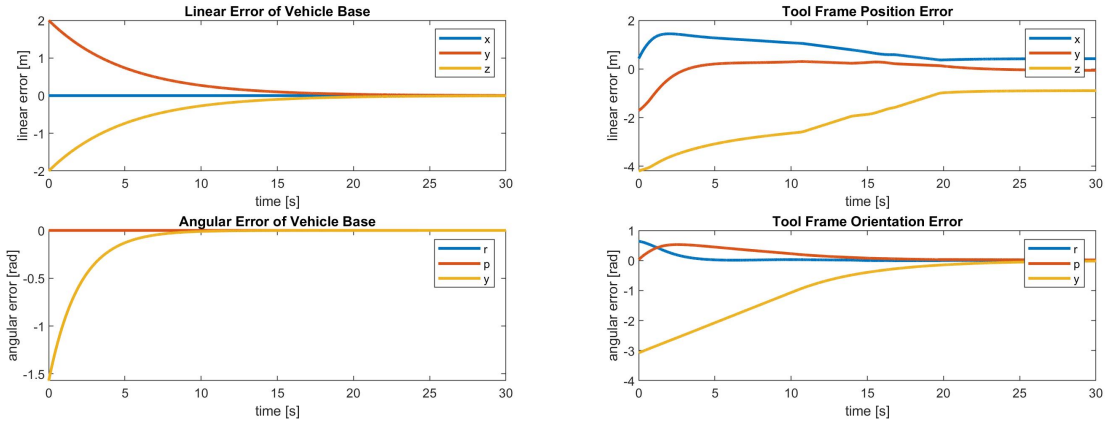
Priority	Category	Description	Name of the objective
1	Prerequisite	[R, I, P]	ha
2	Action-defining	[R, E, AD]	posc
3	Prerequisite	[R, I, P]	mu
4	Action-defining	[R, E, AD]	t

Table 3: "Safe Waypoint Navigation" action hierarchy with "t" and "mu" control tasks

The robot achieves the desired goal position and orientation. The graph for errors for vehicle base given by Fig. 6(a) is identical to the Fig. 2 for exercise 1.1.1 . This suggests that indeed lower

priority tasks do not affect the output of the higher priority tasks. Meanwhile the manipulator tries to reduce the error as much as possible converges to a constant error just few seconds after the vehicle reaches its goal.

Although the Tool Position control ("t") task does not affect the Vehicle Position control ("$posc$") due to the their priorities, it should be disabled for Safe Waypoint navigation to avoid possible collision and other unanticipated behaviour.



(a) (b)

Figure 6: (a) Position and orientation errors for vehicle base and (b) Position and orientation errors for tool frame

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.

A: The **Minimum Altitude control objective** *"mac"* is a safety control objective. Therefore, it has to have higher priority than action defining objective such as *"posc"* (explained in subsection 1.1.1). Thus we choose *"mac"* to have the highest priority. Our hierarchy is as follows:

Priority	Category	Description	Name of the objective
1	Safety	[R, I, S]	mac
2	Prerequisite	[R, I, P]	ha
3	Action-defining	[R, E, AD]	posc

Table 4: Hierarchy of the objectives for the improved "Safe Waypoint Navigation"

The *"mac"* task is given the highest priority because it is more important to avoid collision with the seafloor than maintaining vehicle's horizontal altitude objective.

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

A1: The *"mac"* task is characterized by the following Jacobian relationship:

$$ {}^{w}\dot{\boldsymbol{x}}_{\boldsymbol{mac}} = \boldsymbol{J}_{\boldsymbol{mac}} \ {}^{v}\dot{\boldsymbol{y}} \tag{16} $$

where $\dot{\boldsymbol{x}}_{\boldsymbol{mac}} \in \mathbb{R}^3$, $\boldsymbol{J}_{\boldsymbol{mac}} \in \mathbb{R}^{3\times 13}$ and $\dot{\boldsymbol{y}} \in \mathbb{R}^{13}$; $\boldsymbol{J}_{\boldsymbol{mac}}$ has three rows corresponding to the dimension of the reference velocity($\dot{\boldsymbol{x}}_{\boldsymbol{mac}}$), where $\dot{\boldsymbol{x}}_{\boldsymbol{mac}}$ has only the linear components.

$$ {}^{w}\dot{\boldsymbol{x}}_{\boldsymbol{mac}} \ = \ {}^{w}\boldsymbol{v}_{\ 3\times 1} \ = \ \begin{bmatrix} \mathbf{0}_{\ 3\times 7} & {}^{w}\boldsymbol{R}_{v\ 3\times 3} & \mathbf{0}_{\ 3\times 3} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\ 7\times 1} \\ {}^{v}\boldsymbol{v}_{\ 3\times 1} \\ {}^{v}\boldsymbol{\omega}_{\ 3\times 1} \end{bmatrix} \tag{17} $$

A2: The *"mac"* task is based on an inequality objective with the main goal of ensuring that the vehicle maintains the altitude above a certain minimum threshold:

$$ h_{actual} \ \geq \ h_{min\_thresh} \tag{18} $$

The task reference has the following structure:

$$ {}^{w}\dot{\bar{\boldsymbol{x}}}_{\boldsymbol{mac}} \ = \ \begin{bmatrix} 0 \\ 0 \\ {}^{w}v_z \end{bmatrix} \tag{19} $$

Where we provide a velocity in the z direction to ensure that the robot achieves its "mac" objective.

Activation for this task, $\boldsymbol{A}_{\boldsymbol{mac}}$, has the below structure:

$$ \boldsymbol{A}_{\boldsymbol{mac}} = \begin{bmatrix} a_{1,1} & 0 & 0 \\ 0 & a_{2,2} & 0 \\ 0 & 0 & a_{3,3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & a_{3,3} \end{bmatrix} \tag{20} $$

$a_{1,1}$ and $a_{2,2}$ are equal to zero because the intent is to constrain the system for only the velocity in z direction of the world frame, given by eq. (19). Thus, for x and y component the activation should always remain zero.

The desired behaviour for reactive control of our inequality objective is:

- The task should be fully active only when the inequality is false.
- The transition for activation should be smooth.

Therefore, we use *DecreasingBellShapedFunction* such that:

$$ a_{3,3} \triangleq \begin{cases} 1 & h_{actual} < h_{min\_thresh} \\ \text{decreasing bell} & h_{min\_thresh} \leq h_{actual} \leq h_{min\_thresh} + \Delta \\ 0 & h_{actual} > h_{min\_thresh} + \Delta \end{cases} \tag{21} $$

We compute variable ${}^{w}v_z$ using the below equation:

$$ {}^{w}v_z = \lambda \ ({}^{w}\bar{h} \ - \ {}^{w}h_{actual}) \tag{22} $$

Where ${}^{w}h_{actual}$ is the distance given by the ^{v}*sensorDistance*[4] project on the *z-axis* of the world frame.

[4]This quantity represents the distance of the vehicle from the seafloor seen from the vehicle itself.

CASES:

- **case 1**: ${}^w\overline{h} = {}^w h_{min\_thresh}$
  - $h_{actual} < h_{min\_thresh}$: The reference velocity (eq. 43) will be positive and will drive the robot to h_{min_thresh} with activation $a_{3,3} = 1$
  - $h_{min\_thresh} < h_{actual} < h_{min\_thresh} + \Delta$: The reference velocity (eq. 43) will be negative and will drive the robot to h_{min_thresh} with activation $a_{3,3} < 1$ & $a_{3,3} > 0$ (transition region)
  - $h_{actual} > h_{min\_thresh} + \Delta$: The reference velocity (eq. 43) will be negative, but the activation $a_{3,3} = 0$ and therefore does not have any affect on the uvms.
- **case 2**: ${}^w\overline{h} = {}^w h_{min\_thresh} + \Delta$
  - $h_{actual} < h_{min\_thresh}$: The reference velocity (eq. 43) will be positive and will drive the robot towards $h_{min\_thresh} + \Delta$ with activation $a_{3,3} = 1$
  - $h_{min\_thresh} < h_{actual} < h_{min\_thresh} + \Delta$: The reference velocity (eq. 43) will be positive and will drive the robot towards $h_{min\_thresh} + \Delta$ with activation $a_{3,3} < 1$ & $a_{3,3} > 0$
  - $h_{actual} > h_{min\_thresh} + \Delta$: The reference velocity (eq. 43) will be negative, but the activation $a_{3,3} = 0$ and therefore does not have any affect on the uvms.

Therefore, **case 1** forces the robot to achieve an altitude of h_{min_thresh} for $h_{min\_thresh} < h_{actual} < h_{min\_thresh} + \Delta$. Which is not a desirable behaviour. For this reason, we choose to implement **case 2** which helps us to avoid over-constraining the system.

$$ {}^w\overline{h} \;=\; {}^w h_{min\_thresh} + \Delta \tag{23} $$

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

CASE 1: $h_{min\_thresh} = 1$ and $\Delta = 1$

- Since the sensor is in the middle of the vehicle, the front portion for the vehicle is closer to the seafloor then the distance measured by the sensor.
- The *z-axis* threshold is not large enough to avoid the seamount in time.
- Once the robot has collided with the seamount, the sensor distance information is not accurate due to the limitations of the simulation. Hence we observe a brief period of inactive *"mac"* task, until the robot comes out of the seamount on the other side.

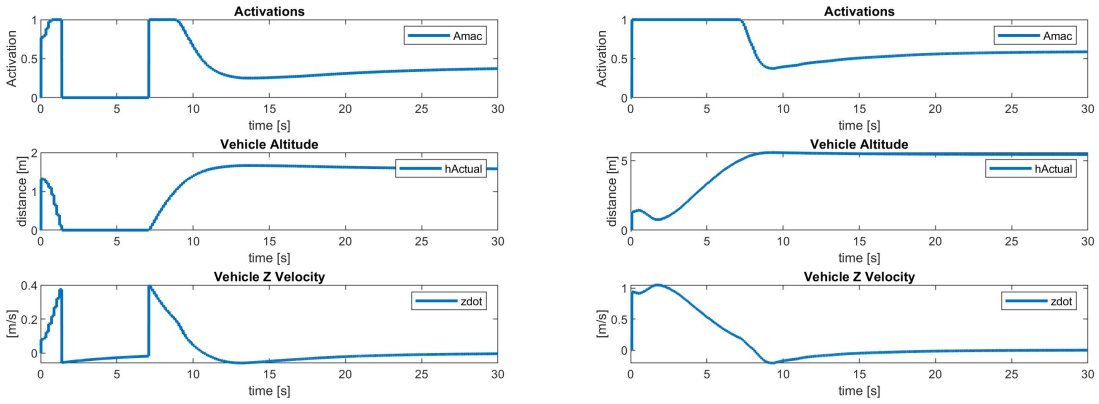
CASE 2: $h_{min\_thresh} = 5$ and $\Delta = 1$

- The threshold is sufficiently big to avoid the seamount.
- Even though the robot starts very close to the seafloor and there is a sharp slope ahead of it, the robot successfully avoids collision because of the sufficiently large threshold and the generated upward reference velocity v_z.

CASE 3: $h_{min\_thresh} = 10$ and $\Delta = 1$

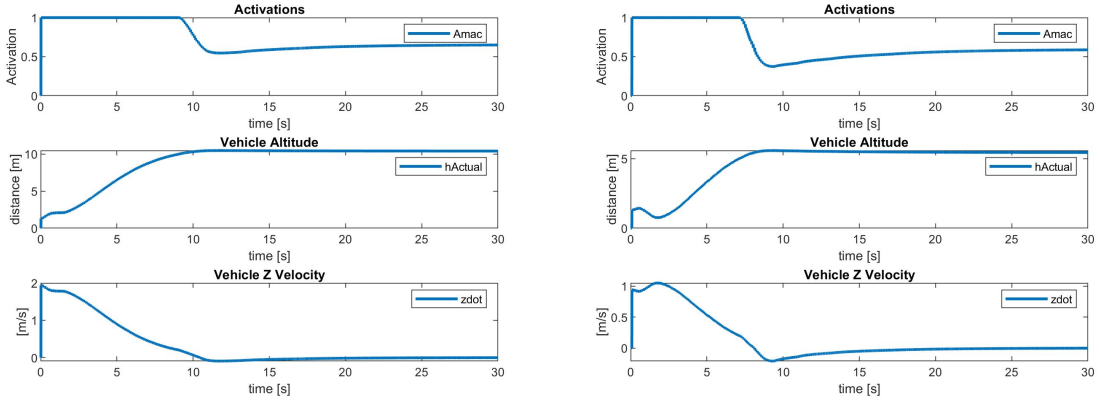
- The threshold is very large for the given task.
- The uvms is able to avoid the obstacles but the final error for position control is too big.

These are the graphs related to the cases above:



(a) (b)

Figure 7: Activation, vehicle altitude and vehicle z velocity for (a) $h_{min\_thresh} = 1$ (b) $h_{min\_thresh} = 5$



(a) (b)

Figure 8: Activation, vehicle altitude and vehicle z velocity for (a) $h_{min\_thresh} = 10$ (b) $h_{min\_thresh} = 5$

Conclusion: Different threshold is suitable for different types of sea floor. We summarise our thoughts in the table below.

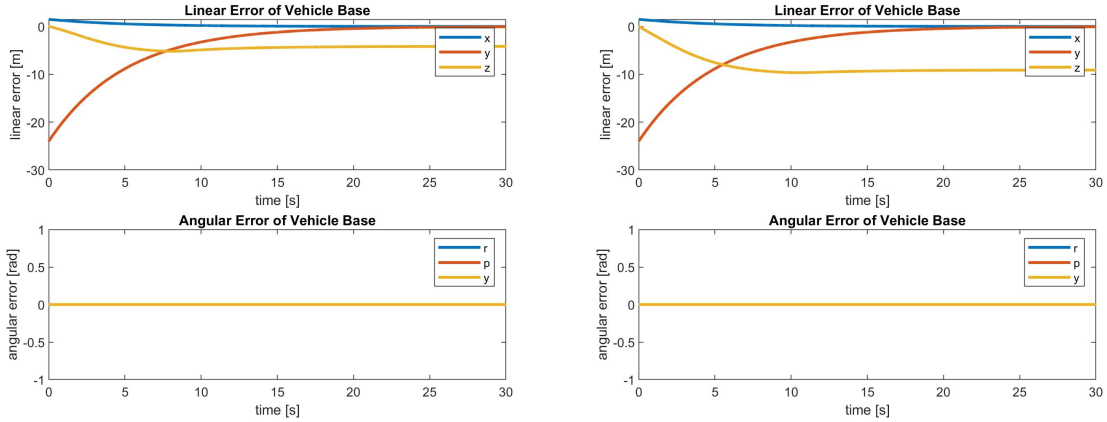
Type of seafloor	$\boldsymbol{h_{min\_thresh}}$**=1**	$\boldsymbol{h_{min\_thresh}}$**=5**	$\boldsymbol{h_{min\_thresh}}$**=10**
Almost flat	safe	safe	safe
Small protuberances	not completely safe	safe	safe
Large protuberances	not safe	not always safe	safe

Table 5: Comparison between behaviour for different threshold for different types of seafloor

From Fig. 9 it is also possible to extract the following considerations:

- The UVMS is unable to satisfy the *"posc"* task as it has conflicting objective with the higher priority task, *"mac"*.
- The final *"posc"* error is proportional to h_{min_thresh}.

- When $h_{min\_thresh} < h_{actual} < h_{min\_thresh}+\Delta$, the *"posc"* is actively trying to push the robot downwards (*-z*) to achieve the desired goal position, where as the *"mac"* task provides an upward velocity (*+z*) with activation between 0 and 1. Thus, the robot's velocities become zero when the two objective nullify each other.



(a) (b)

Figure 9: Error in position and orientation for (a) $h_{min\_thresh} = 5$ (b) $h_{min\_thresh} = 10$

1.2.4 Q4: How was the sensor distance processed?

A: The *sensorDistance* is the distance measured by the sensor on the UVMS along the *z-axis* of the sensor frame $< s >$.

$$^{s}\textbf{distance vector} = \begin{bmatrix} 0 \\ 0 \\ ^{s}\text{sensorDistance} \end{bmatrix}$$

To obtain the distance between seafloor and robot in the world frame ($< w >$) we use the below equation:

$$^{w}\textbf{distance vector} = \begin{bmatrix} ^{w}\text{x-component} \\ ^{w}\text{y-component} \\ ^{w}h_{actual} \\ 0 \end{bmatrix} = {}^{w}\boldsymbol{T}_{v}\ {}^{v}\textbf{distance vector} \tag{24}$$

with "h_{actual}" extracted by the z component of the *sensorDistance* projected in the world frame.

REMARK: We assume that sensor frame ($< s >$) and vehicle frame ($< v >$) coincide.

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.

A: Borrowing from the previous exercises, the hierarchy of the tasks is given by the table below. The only addition that has been made in this exercise is the Landing Objective (*"la"*). Since *"la"* is a Action-defining (AD) objective it has been placed after the safety tasks. It's important also to remark that *"la"* and *"posc"* have very different objectives, so it's unlikely to have them active at the same time. For that reason their relative priority does not affect the solution.

The Last column ("Landing") indicates the whether the tasks were enable or disabled during the landing phase.

Priority	Category	Description	Objective	Landing
1	Safety	[R, I, S]	mac	0
2	Prerequisite	[R, I, P]	ha	1
3	Action-defining	[R, E, AD]	la	1
4	Action-defining	[R, E, AD]	posc	0
5	Prerequisite	[R, I, P]	mu	1
6	Action-defining	[R, E, AD]	t	0

Table 6: Control tasks and priorities for "Landing" action

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

A1: Jacobian is computed such that the reference velocities only affects the linear attributes of the vehicle velocities.

$$ {}^{w}\dot{\boldsymbol{x}}_{\boldsymbol{la}} \;=\; {}^{w}\boldsymbol{v}_{\;3\times1} \;=\; \begin{bmatrix} \mathbf{0}_{\;3\times7} & {}^{w}\boldsymbol{R}_{v\;3\times3} & \mathbf{0}_{\;3\times3} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\;7\times1} \\ {}^{v}\boldsymbol{v}_{\;3\times1} \\ {}^{v}\boldsymbol{\omega}_{\;3\times1} \end{bmatrix} \tag{25} $$

A2: As we would like to drive the distance between vehicle and seafloor to zero the task reference for landing action is computed as below:

$$ {}^{w}v_z \;= \lambda\;(0 \;-\; {}^{w}h_{actual}) \tag{26} $$

$$ {}^{w}\dot{\boldsymbol{x}}_{\boldsymbol{la}} \;= [0 \;\; 0 \;\; {}^{w}v_z]^T \tag{27} $$

where ${}^{w}h_{actual}$ is the same quantity used in eq. 24 (exercise 1.2.4).

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

A: The main differences are that the reference velocity is given downwards on the *z-axis* and it is calculating using the distance between the robot and seafloor using the sensor data [5]. Another important aspect is that *"la"* is an equality objective, whereas *"mac"* is a inequality one. Thus, for this task we do not use any function for smooth transition of activation function.

[5]The sensor data is projected on to the world frame and the magnitude of the *z-component* is considered as the distance between the vehicle and seafloor.

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.

A: Hierarchy of the task is the same as Table 6. *Way Point Navigation* and *Landing* are the two phases that needs to be executed in this exercise. Depending on the phase of the mission, different tasks are either enabled or disabled. The table below specifies the task's external activation depending on the phase of the mission.

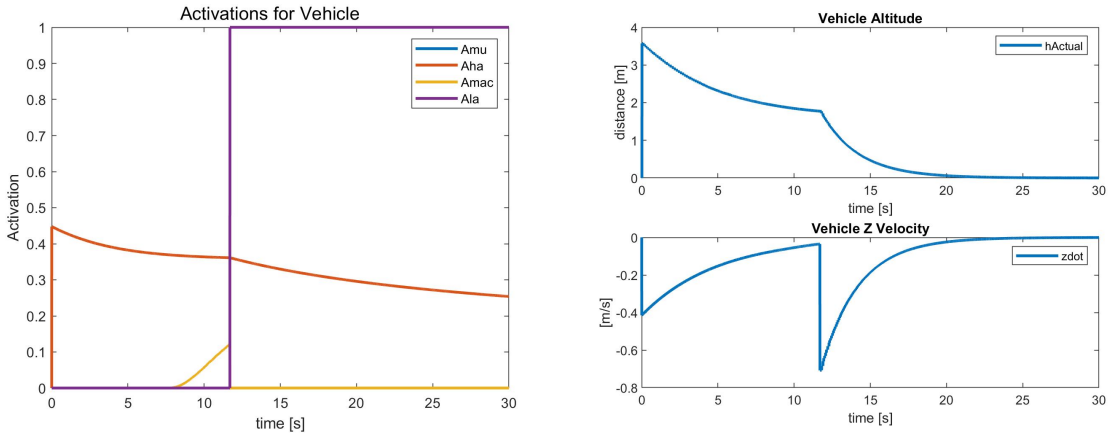
Priority	Tasks	Way Point Navigation	Landing
1	mac	1	0
2	ha	1	1
3	la	0	1
4	posc	1	0
5	mu	1	1
6	t	0	0

Table 7: External activation status for different tasks in different phases

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

A: The transition from one phase (action) to another was implemented using "phase" variable of "mission" Matlab structure. The "phase" variable is updated when the previous action has been completed with desired accuracy. For this exercise the phase was updated when the robot was near ($||error_{lin}|| < 0.4$ and $||error_{ang}|| < 0.2$) the desired goal location of the way point navigation. The checks for phase update conditions are performed in the "UpdateMissionPhase" in every loop.

REMARK: The transition between one phase to another can be made seamless by applying bell curves (increasig or decreasing) for activation by considering buffer time.



(a) (b)

Figure 10: Behaviour for *Way Point Navigation* and *Landing* task using external activation for phase change

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 unified hierarchy of tasks used and their priorities. Comment the behaviour.

A: We again use the same hierarchy as developed in exercise 2, Table 6. To improve out landing we add Alignment to Target Control (*"at"*) objective after landing in our task hierarchy. We decide to place *"at"* below *"la"* to take advantage of residual arbitrariness to keep aligning the robot to our desired orientation in the landing phase.

Priority	Category	Description	Objective
1	Safety	[R, I, S]	mac
2	Prerequisite	[R, I, P]	ha
3	Action-defining	[R, E, AD]	la
4	Prerequisite	[R, E, P]	at
5	Action-defining	[R, E, AD]	posc
6	Prerequisite	[R, I, P]	mu
7	Action-defining	[R, E, AD]	t

Table 8: Control tasks and priorities form improved landing using "at" task

The below table lists the external activation for different tasks in different phases:

Priority	Tasks	Way Point	Alignment	Landing
1	mac	1	1	0
2	ha	1	1	1
3	la	0	0	1
4	at	0	1	1
5	posc	1	0	0
6	mu	1	1	1
7	t	0	0	0

Table 9: External activation for different tasks during different mission phases in this exercise

Before landing the robot tries to align itself to the rock, when it has reached a sufficient alignment ($\theta < 0.5$), the landing phase is activated during which the robot lands as well as aligns as much as possible .

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

A: Before computing the Jacobian relationship we have to define the parameters of interest.

Recalling that main goal of this objective is to align the vehicle (especially the x direction) toward the target (the small rock). To do so we have to compute the rotation vector $\boldsymbol{\rho}$ between the two vector of interest.

The two vectors are:

- **a**: *x-axis* of the vehicle frame

$$^{v}\boldsymbol{a} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{28}$$

- **b**: vector between the vehicle frame and nodule projected onto the inertial horizontal plane and expressed in vehicle frame.

$$^{v}\boldsymbol{b} \;=\; ^{v}\boldsymbol{R}_{w}\;(\boldsymbol{I} - \boldsymbol{k}\boldsymbol{k}^{\top})\;\;^{w}Distance\_rock \tag{29}$$

Now we can proceed to compute $\boldsymbol{\rho}$ vector in the following way:

$$\boldsymbol{a} \wedge \boldsymbol{b} \;=\; \boldsymbol{n}\;sin(\theta) \tag{30}$$

$$\boldsymbol{\rho} \;=\; \boldsymbol{n}\;\theta \tag{31}$$

that returns the direction **n** and the magnitude θ that vector **a** must perform to be aligned with **b**.

Since our final goal is to have *x-axis* of the vehicle aligned with the target we have to study the behaviour of this resulting vector during time, thus we want:

$$\dot{\boldsymbol{x}}_{\boldsymbol{ref}} \;=\; \dot{\boldsymbol{\rho}} \;=\; \gamma \boldsymbol{n}\theta \tag{32}$$

Considering a generic observer we have:

$$D_\alpha \boldsymbol{\rho} = \dot{\theta}\boldsymbol{n} + \theta D_{\boldsymbol{a}}(\boldsymbol{n}) = \boldsymbol{n}\boldsymbol{n}\boldsymbol{\omega}_{b/\alpha} + \boldsymbol{N}_\alpha(\theta)(\boldsymbol{\omega}_{b/\alpha}, \boldsymbol{\omega}_{a/\alpha}) \tag{33}$$

Considering an observer inside the rigid space of the vehicle frame $< v >$ on which there is vector **a**:

$$D_{\boldsymbol{a}} \boldsymbol{\rho} = \boldsymbol{n}\boldsymbol{n}\boldsymbol{\omega}_{b/a} + \boldsymbol{N}_\alpha(\theta)\boldsymbol{\omega}_{b/a} = \boldsymbol{n}\boldsymbol{n}\boldsymbol{\omega}_{b/a} \tag{34}$$

REMARK: the second term is exactly equal to zero due to the fact that we have $\boldsymbol{\rho}$ and $\dot{\boldsymbol{\rho}}$ aligned.

Finally we want:

$$\gamma \;\; \boldsymbol{n}\theta \;\; = \;\; \boldsymbol{\omega}_{b/a} \tag{35}$$

Since the quantity $\boldsymbol{\omega}_{b/a}$ is not easy to compute we can compute it using the *law of addition of angular velocity vectors*:

$$\boldsymbol{\omega}_{b/a} \;\; = \;\; \boldsymbol{\omega}_{b/w} - \boldsymbol{\omega}_{a/w} \tag{36}$$

where:

- $\boldsymbol{\omega}_{a/w}$ is the angular velocity of the vehicle with respect to the world.
- $\boldsymbol{\omega}_{b/w}$ is the angular velocity given by the movement of the vehicle with respect to the node that produces a change of direction of unit vector joining the vehicle frame to the nodule frame.

NOTE : It is important to compute this quantities using the same observer as for ρ, computed in eq. (31), thus in our case:

$$^v\boldsymbol{\omega}_{b/a} \;\; = \;\; ^v\boldsymbol{\omega}_{b/w} - ^v\boldsymbol{\omega}_{a/w} \tag{37}$$

in particular:

$$^v\boldsymbol{\omega}_{b/w} = \left(\frac{^v\boldsymbol{b}}{||^v\boldsymbol{b}||} \wedge \frac{-^v\boldsymbol{vp}}{||^v\boldsymbol{vp}||} \right) \;\; \frac{||^v\boldsymbol{vp}||}{||^v\boldsymbol{b}||} \tag{38}$$

in which $^v\boldsymbol{vp}$ is the projection of the linear velocity of the vehicle on the inertial horizontal plane expressed in vehicle frame thus, $\quad ^v\boldsymbol{vp} \;\; = \;\; ^v\boldsymbol{R}_w \;\; (\boldsymbol{I} - \boldsymbol{k}\boldsymbol{k}^T) \;\; ^w\boldsymbol{R}_v \;\; ^v\boldsymbol{v}$.

We can now proceed to compute the desired Jacobian matrix that, according to the equations (5) (6), must be the following one:

$$\dot{\boldsymbol{x}} = \dot{\boldsymbol{\rho}} = \left[\begin{array}{ccc} \boldsymbol{0}_{3\times 7} & -\frac{||^v\boldsymbol{vp}||}{||^v\boldsymbol{b}||}\left(\frac{^v\boldsymbol{b}}{||^v\boldsymbol{b}||} \wedge \frac{^v\boldsymbol{R}_w(\boldsymbol{I}-\boldsymbol{k}\boldsymbol{k}^T)^w\boldsymbol{R}_v}{||^v\boldsymbol{vp}||}\right) & 1 \end{array} \right] \left[\begin{array}{c} \dot{\boldsymbol{q}} \\ ^v\boldsymbol{v} \\ ^v\boldsymbol{w} \end{array} \right] \tag{39}$$

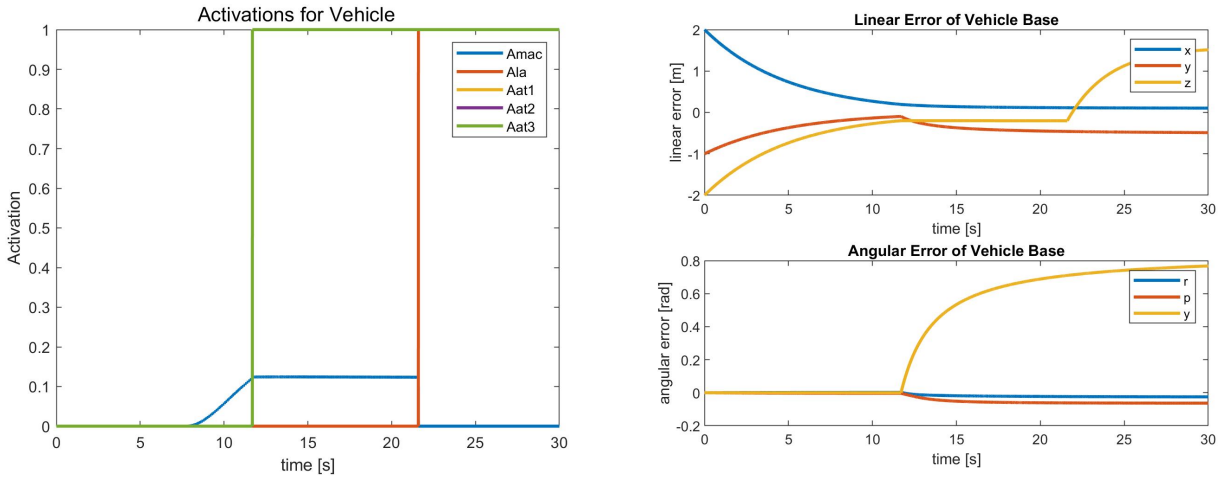
3.1.3 Q3: Try changing the gain of the alignment task. Try at least three different values, where one is very small. What is the observed behaviour?

We observe:

CASE 1: When gain = 0.01, the angular velocity is very small and thus the robot is unable to complete its alignment task in desired time.

CASE 2: When gain = 1.5, the robot is able to align itself to the stone in appropriate time.

CASE 3: When gain = 5, the robot is able to align itself with a very high angular velocity.



(a) (b)

Figure 11: Activations(a) and Vehicle base's error(b) during different phases for case 2

From Fig.11 we observe that, when the task *"at"* is active, around $t = 12s$, the error in orientation starts increasing, while the linear error remains the almost same almost. This happens because the vehicle base's errors are being computed with respect to the goal position given for way point navigation. At round $t = 22s$, the landing task is also activated, and thus we see an increase of error in z direction. Since the *"la"* and *"at"* task is active simultaneously (after about $t = 22s$), both the linear error and angular error increases.

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.

After landing is accomplished, if we enable the tool frame control task, the tool frame tries to reach the desired goal position (on top of the stone) by sinking the robot in the seafloor. To improve upon this behaviour we implement fixed-based manipulation described in the next exercise.

REMARK: If we activate the non-reactive task (*"nr"*) implemented for exercise 4, and given sufficient time, the tool frame almost reaches the desired goal position for end-effector without sinking into the seafloor.

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?

Added the Non-Reactive Task (*"nr"*) at the top of the hierarchy as described in the below table:

Priority	Category	Description	Objective
1	Constraint	[NR, E, C]	nr
2	Safety	[R, I, S]	mac
3	Prerequisite	[R, I, P]	ha
4	Action-defining	[R, E, AD]	la
5	Prerequisite	[R, E, P]	at
6	Action-defining	[R, E, AD]	posc
7	Prerequisite	[R, I, P]	mu
8	Action-defining	[R, E, AD]	t

Table 10: Control tasks and priorities for fixed base manipulation action

For each phase we have the following tasks active/inactive:

Priority	Tasks	Way Point	Alignment	Landing	Tool Frame
1	nr	0	0	0	1
2	mac	1	1	0	0
3	ha	1	1	1	1
4	la	0	0	1	0
5	at	0	1	1	0
6	posc	1	0	0	0
7	mu	1	1	1	1
8	t	0	0	0	1

Table 11: External activation status for different tasks in different phases

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

A1: Since we are interested only in controlling the vehicle velocity (${}^{w}\nu$) the Jacobian Relationship used is the following:

$$ {}^{w}\dot{\boldsymbol{x}}_{\boldsymbol{nr}\ 6\times1} = \begin{bmatrix} {}^{w}\boldsymbol{v}\ _{3\times1} \\ {}^{w}\boldsymbol{\omega}\ _{3\times1} \end{bmatrix} = \begin{bmatrix} \mathbf{0}\ _{3\times7} & {}^{w}\boldsymbol{R}_{v}\ _{3\times3} & \mathbf{0}\ _{3\times3} \\ \mathbf{0}\ _{3\times7} & \mathbf{0}\ _{3\times3} & {}^{w}\boldsymbol{R}_{v}\ _{3\times3} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}\ _{7\times1} \\ {}^{v}\boldsymbol{v}\ _{3\times1} \\ {}^{v}\boldsymbol{\omega}\ _{3\times1} \end{bmatrix} \tag{40} $$

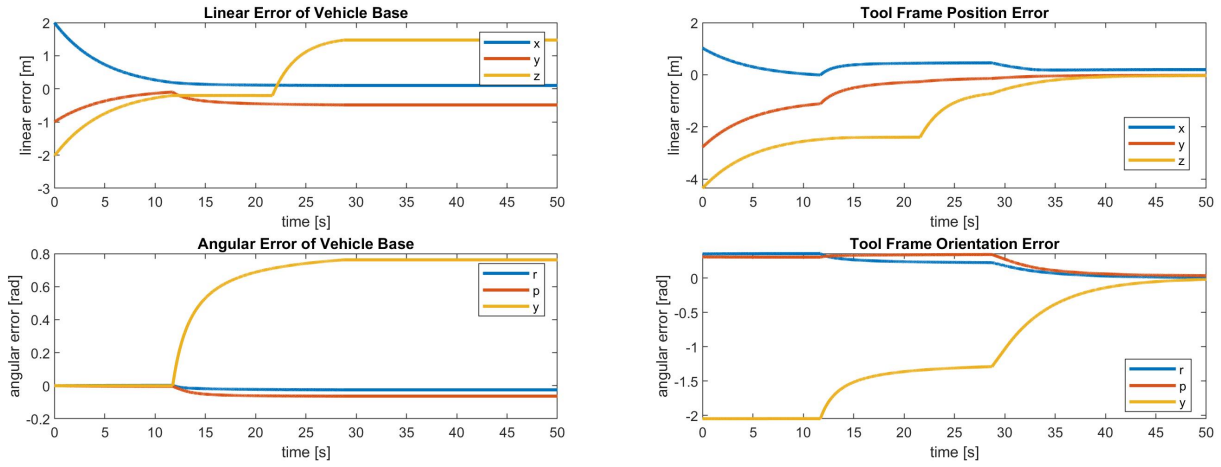
A2: Vehicle Null Velocity task is a non-reactive one, so we can not provide reference velocity as a error feedback but it has to be provided in the task velocity space. Therefore we simply set the reference as zeros.

$$ {}^{w}\dot{\tilde{\boldsymbol{x}}}_{\boldsymbol{nr}\ 6\times1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\top} \tag{41} $$

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?

A: Given the Jacobian (eq. 40), the ${}^{v}\boldsymbol{\omega}$ and ${}^{v}\boldsymbol{v}$ will be set as zeros. Thus, the thrusters of the vehicle base will be off. Hence, if there are any external disturbances, such as currents, the thrusters will be off inorder to fulfill the constraint set by the *"nr"* task, but the robot will not be able to maintain a zero velocity.

Conclusion: From Fig. 12 we observe that the error in vehicle base's position and orientation remains almost constant after *30*s of simulation, this is when the robot has completed Landing phase and started the Tool Frame phase. Therefore, active *"nr"* task in Tool Frame phase helps us achieve fixed base manipulation. The tool frame error reduces to almost zero while the error on vehicle base's position remains the same (no changes in the error).



(a) (b)

Figure 12: Fig (a) illustrates vehicle base's error computed against the goal position given by way point navigation, and Fig (b) displays tool frame's error.

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 joint limit task?

The Joint limit (*"jl"*) task is added after the Non-Reactive constraint one, as its a safety task, and does not have any conflicting interest with *"mac"*.

Priority	Category	Description	Objective
1	Constraint	[NR, E, C]	nr
2	Safety	[R, I, S]	jl
3	Safety	[R, I, S]	mac
4	Prerequisite	[R, I, P]	ha
5	Action-defining	[R, E, AD]	la
6	Prerequisite	[R, E, P]	at
7	Action-defining	[R, E, AD]	posc
8	Prerequisite	[R, I, P]	mu
9	Action-defining	[R, E, AD]	t

Table 12: Control tasks and priorities for fixed base manipulation action with joint limits

Priority	Tasks	Way Point	Alignment	Landing	Tool Frame
1	nr	0	0	0	1
2	jl	1	1	1	1
3	mac	1	1	0	0
4	ha	1	1	1	1
5	la	0	0	1	0
6	at	0	1	1	0
7	posc	1	0	0	0
8	mu	1	1	1	1
9	t	0	0	0	1

Table 13: External activation for different tasks during different mission phases in this exercise

It's important to highlight that *"jl"* task is always active in such a way that, given any phase, we would want the manipulator to be within its limit in order to avoid to damage the manipulator.

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

A1: The Jacobian relationship is simply identity for the first seven elements, as we are specifying the velocities of each joint directly. While the rest is set to zeros, as joint velocities do no impact the vehicle base's velocities.

$$\dot{\boldsymbol{x}}_{\boldsymbol{jt}} \;=\; \dot{\boldsymbol{q}}_{\;7\times1} \;=\; \begin{bmatrix} \mathbf{I}_{\;7\times7} & \mathbf{0}_{\;7\times3} & \mathbf{0}_{\;7\times3} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\;7\times1} \\ {}^{v}\boldsymbol{v}_{\;3\times1} \\ {}^{v}\boldsymbol{\omega}_{\;3\times1} \end{bmatrix} \tag{42}$$

A2: The task reference is computed as follow:

$$\dot{\overline{\boldsymbol{x}}}_{\boldsymbol{jt}} = \lambda\ (\overline{\boldsymbol{q}} \;-\; \boldsymbol{q}_{\boldsymbol{actual}}) \tag{43}$$

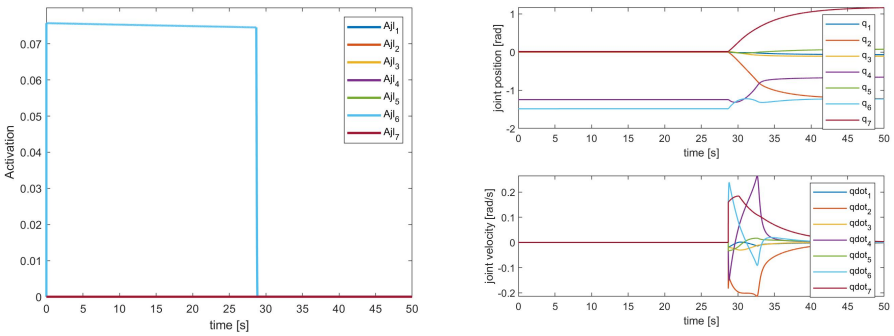
Let the $q_{i\ \ actual}$ correspond to the i^{th} element of the vector containing information for the i^{th} joint of the manipulator we have these two possible cases:

CASE 1: if $q_{l\ \ min} < q_{i\ \ actual} < q_{l\ \ min} + \Delta$:

$$q_i \;=\; q_{l\ \ min} + \Delta \tag{44}$$

CASE 2: if $q_{l\ \ max} - \Delta < q_{i\ \ actual} < q_{l\ \ max}$:

$$q_{i} \ = \ q_{l\ \ max} - \Delta \tag{45}$$



(a) (b)

Figure 13: (a) Activation for Joint limits (b) Joint Velocities

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 for DexROV is a little different from previous robot simulation.

Priority	Category	Description	Objective	Tool frame control
1	Constraint	[NR, E, C]	nr	0
2	Safety	[R, I, S]	jl	1
3	Prerequisite	[R, I, P]	ha	1
4	Action-defining	[R, E, AD]	posc	0
5	Prerequisite	[R, I, P]	mu	1
6	Action-defining	[R, E, AD]	t	1
7	Optimization	[R, I, O]	mp	1

Table 14: Floating manipulation action hierarchy. The last column "Tool Frame Control" states the external activation used for this exercise.

"mac" and *"at"* tasks are missing in this table for the following reasons:

- *"mac"*: because there is no sensor on the robot.
- *"at"*: because there is no requirement for alignment of the robot in this and following exercises.

For the rest of the tasks we kept the same hierarchy as the previous exercises with the only addition of *"mp"* task. This task has lower priority wrt *"t"* task because we have to keep the preferred shape **only** when it is possible[6].

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

A1: The Jacobian for this task is similar to the *"jl"* task. For this exercise we are interested in maintaining the shape of the first four joints thus, only the first four value of the diagonal of *"mp"* activation function are equal to 1.

$$\dot{\boldsymbol{x}}_{\boldsymbol{mp}} \;=\; \dot{\boldsymbol{q}}_{\;4\times1} \;=\; \begin{bmatrix} \mathbf{I}_{\;4\times4} & \mathbf{0}_{\;4\times9} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}_{\;7\times1} \\ \boldsymbol{v}_{\;3\times1} \\ \boldsymbol{\omega}_{\;3\times1} \end{bmatrix} \tag{46}$$

A2: The reference velocity is proportional to the error between the joint values and the desired ones and each element of $\dot{\boldsymbol{x}}_{\boldsymbol{mp}}$ corresponds to the reference velocity of the respective joint.

$$\dot{\bar{\boldsymbol{x}}}_{\boldsymbol{mp}} = \lambda \;(\boldsymbol{preferred\_shape} - \boldsymbol{actual\_shape}) \tag{47}$$

where in our case $\lambda = 0.2$.

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

Both the cases allow the tool frame to reach the final position. Additionally, we observe the following behaviour:

CASE 1: *"mp"* inactive

The tool is able to reach the desired final position but the shape is not the desired one

CASE 2: *"mp"* active

This task allows to choose the solution, within the manifold of solutions restricted step by step by the tasks up to *"mp"* one, that best accomplish the required preferred shape.

REMARK: since task *"mp"* is conflicting with *"mu"* and *"jl"* tasks, the activation function of *"jl"* remains non-zero during all the simulation (especially for joints 4 and 5) as shown in Fig. 14.

[6] In this way we are not over-constraining the system.

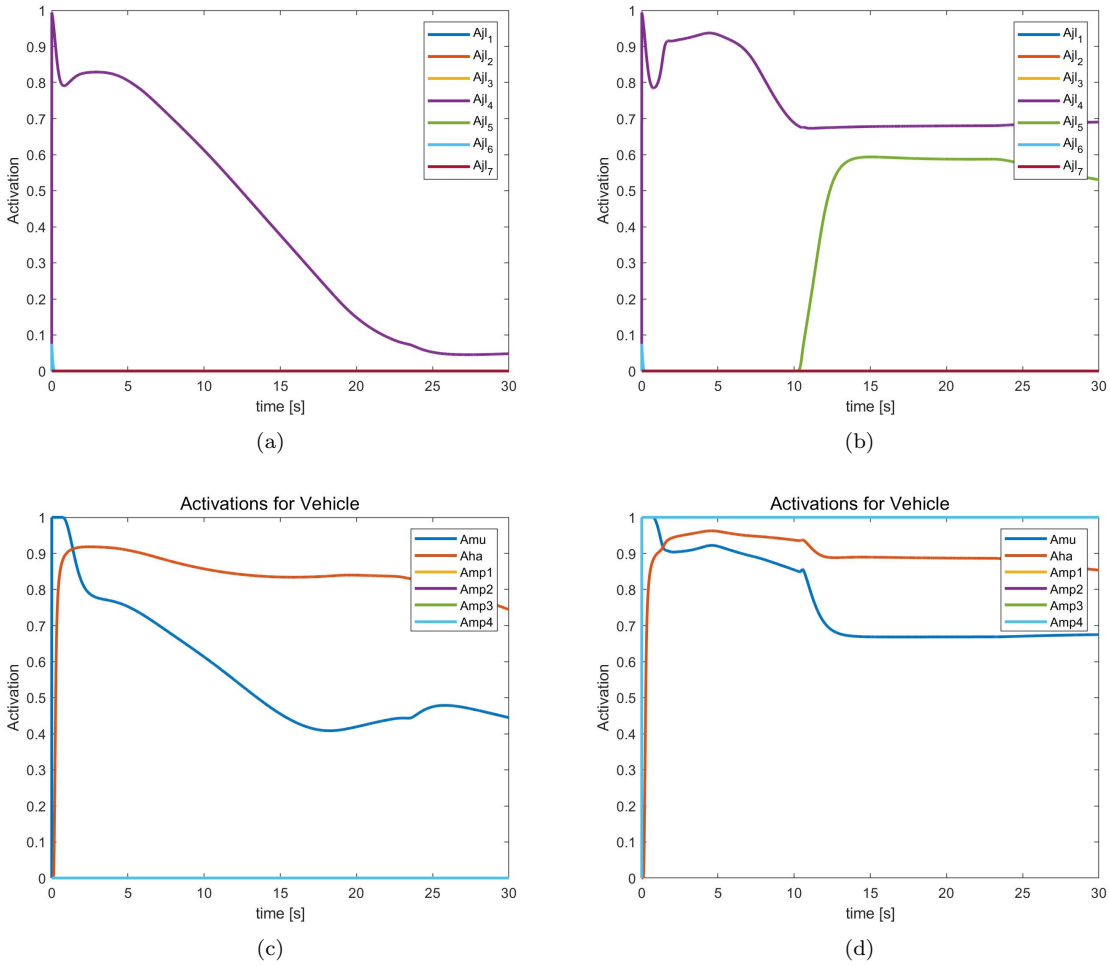


Figure 14: (a) and (c) show the behaviour for "mp" inactive. (b) and (d) show the behaviour for "mp" active

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?

The hierarchy is same as the previous exercise, given by Tab. 14. The robot has two different phases, as detailed in the table below:

Priority	Tasks	Way Point	Tool Frame
1	nr	0	0
2	jl	1	1
3	ha	1	1
4	posc	1	0
5	mu	1	1
6	t	0	1
7	mp	1	1

Table 15: External activations for different tasks in different phase for this exercise

phase 1: Way Point Navigation

Since the robot needs to perform safe way point navigation, tasks *"t"* and *"nr"* are not required in this phase.

phase 2: Tool Frame Control

In this phase, we do not consider any disturbances and we assume that robot is able to provide the desired velocities without any delay so, *"nr"* has activation function equal to zero. Also, once the robot has reached the desired position, its *"t"* task is responsible for reaching the desired goal position and orientation for end-effector, hence *"posc"* task is also switched off.

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

The difference between the previous simulation and this one can be understood by observing the behaviour of tool frame error and activations of the different tasks.

- **Tool frame position error:** For this exercise the curves converge almost asymptotically to the desired goal position (Fig. 15(a)). Whereas in the previous case, on which there is only a single action, it is possible to notice a deviation on the x-component (Fig. 15(b)).
- **Tool frame orientation error:** For this exercise, there is a sharp change when the second phase is activated (Fig. 15(a)) whereas in the previous case there is a smooth curve (Fig. 15(b)).
- **Activation of *"ha"*:** For this exercise *"ha"* remains inactive during the Way Point navigation (Fig. 15(c)) as the reference velocities generated during *"posc"* task do not affect the horizontal balance of the robot. Whereas, in the previous exercise *"ha"* remains active throughout the simulation (Fig. 15(d)). The latter behaviour is due to the reference velocities generated for *"t"* task. The Jacobian relationship of the *"t"* task affects the joint velocities and the base velocities. The velocities of the vehicle are generated with the aim of reducing tool frame error(linear and angular) in an optimized manner. Which causes the robot to be very close to its horizontal balance threshold. Therefore, *"ha"* task compromises(high activation) to allow the robot achieve the demanded velocities, by *"t"* task for navigation, as much as possible.

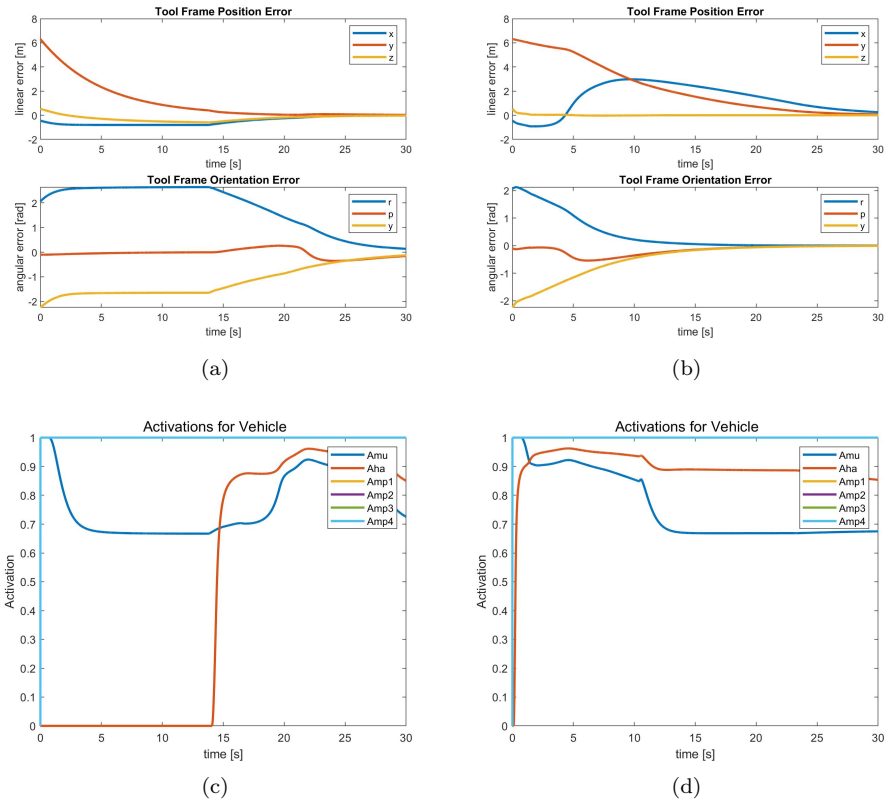


Figure 15: (a) and (c) show the behaviour of tool frame and activations for phase implementation, instead (b) and (d) show the behaviour for single activation (just tool frame control)

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?

To implement the parallel arm-vehicle coordination scheme we introduced a higher level block "$TPIK$". This block contains the same "$iCAT\_task$" with the same hierarchy developed up till previous exercise (Tab. 14).

The approach allow us to optimize the outputs obtained during the *"Tool Frame"* phase based on two different sub-blocks: *TPIK 1* and *TPIK 2*.

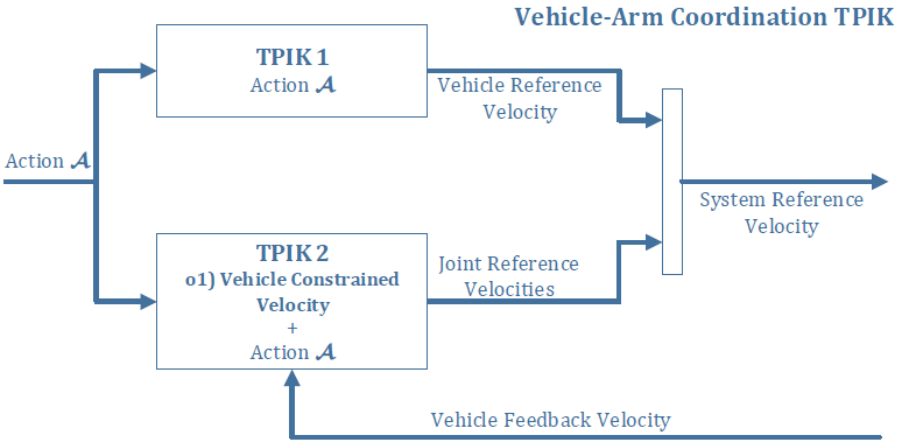


Figure 16: Parallel coordination scheme

TPIK 1: in this case we are perform the calculations for both vehicle base's velocity and joint velocities. After this procedure we discard the joint velocities and save vehicle base's velocity ($\overline{\boldsymbol{\nu}}$). The $\overline{\boldsymbol{\nu}}$ value will be the input of the *DCL* layer.

TPIK 2: we use again TPIK to decouple the velocities between manipulator and vehicle base. We perform a second optimization with the assumption that the vehicle's base is completely non-controllable. To be able to simulate that we use *"nr"* task in that way: we set *"nr"* activation to true and we use the true velocity of the robot's base[7] as reference value. This allow us to re-optimize the velocities of the manipulator by constraining the vehicle velocities to be equal to the actual velocities (so the one with disturbances). Thus, the solution of joint velocities $\dot{\boldsymbol{q}}$ given by TPIK 2 tries reach the goal for tool frame, while compensating for disturbances in vehicle's base velocities.

Finally, the reference velocity provided as input to the *DCL* layer is given by:

$$\dot{\bar{\boldsymbol{y}}} = \begin{bmatrix} \dot{\boldsymbol{q}} & \text{TPIK 2} \\ \overline{\boldsymbol{\nu}} & \text{TPIK 1} \end{bmatrix} \tag{48}$$

These following table summarises the activation of the two sub-blocks:

Priority	Tasks	TPIK 1	TPIK 2
1	nr	0	1
2	jl	1	1
3	ha	1	1
4	posc	0	0
5	mu	1	1
6	t	1	1
7	mp	1	1

Table 16: External activation for different tasks for TPIK 1 and TPIK 2

[7] In simulation we use uvms.p, but in real scenarios TPIK 2 will use velocities measure by the sensors.

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

If the gain is too high, the manipulator is unable to compensate the disturbances thus, the tool fame error (especially for linear component) does not converge anymore to zero.

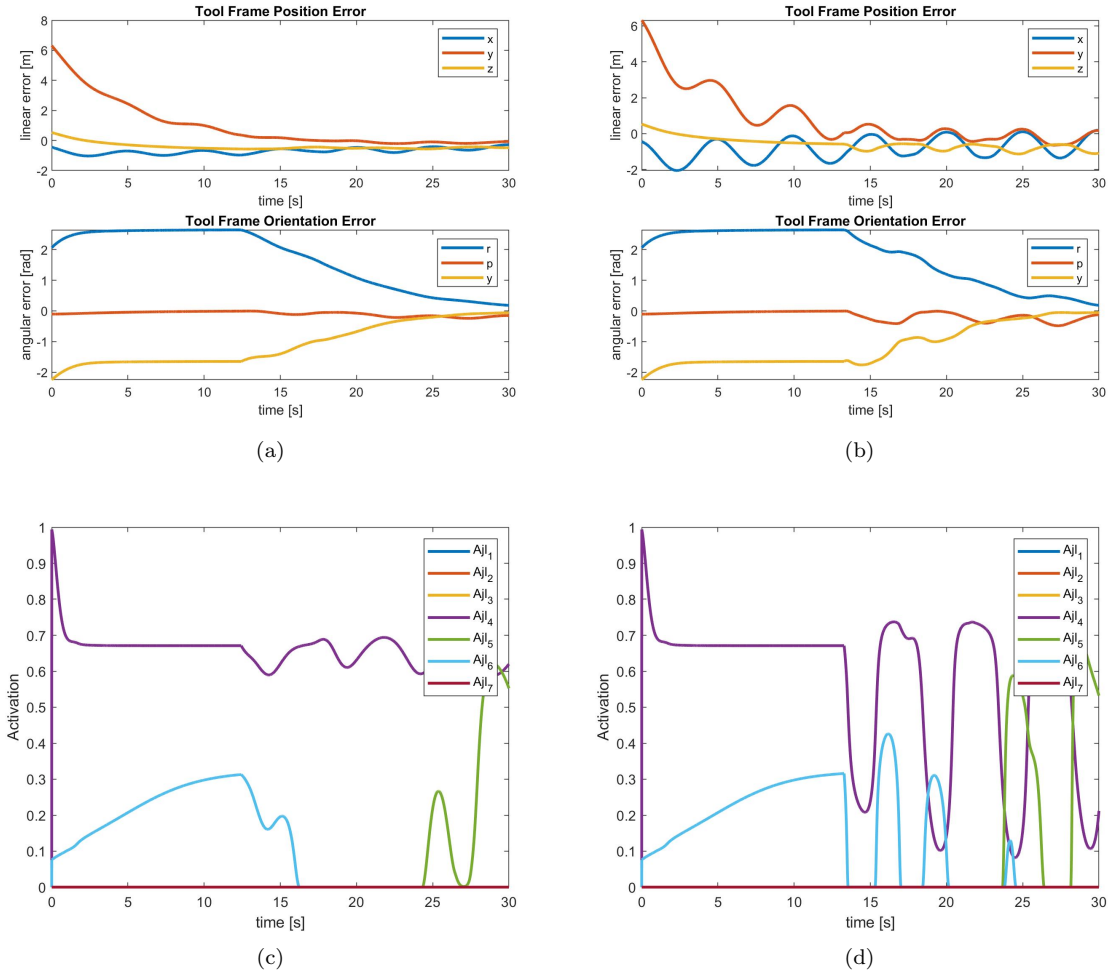


Figure 17: (a)and (c) is the behaviour of manipulator when the gain of the disturbances is small (0.2), instead (b) and (d) when the gain of the disturbances is high (1)

Conclusion: The below plots illustrates the impact of using parallel coordination scheme; to demonstrate the behaviour clearly we use high gains. (G = 3 and frequency = 0.2 Hz)

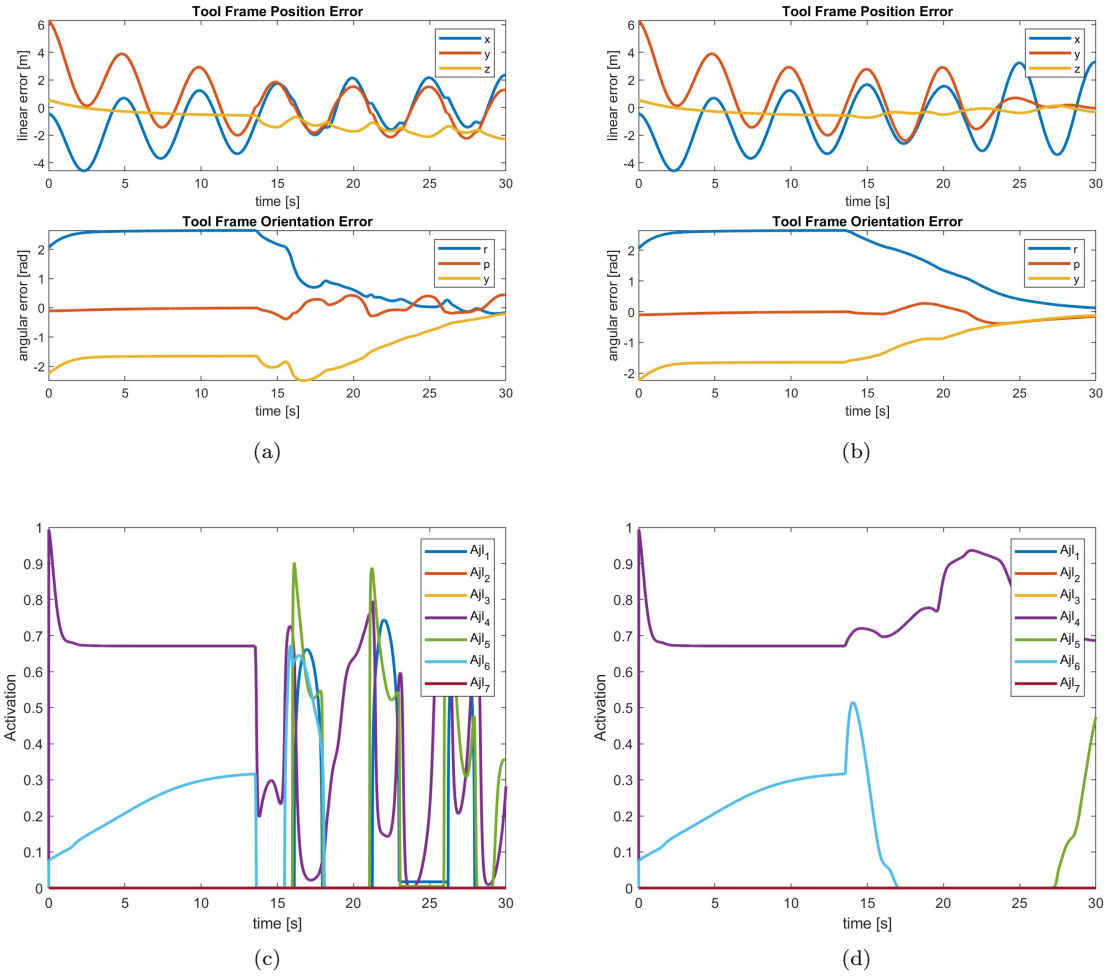


Figure 18: (a) and (c) when *TPIK 2* is active. (b) and (d) when *TPIK 2* is inactive

From the plots above we can say that:

CASE 1: *TPIK 2* is active

The manipulator tries to compensate the error (as show by position error of tool frame in part (a)).

CASE 2: *TPIK 2* is inactive

The error is propagated to the tool frame (as show in part (b)).