Cooperative Robotics

Authors: Cardano Matteo
EMAIL cardano.matte@gmail.com
Date: 21/05/2021

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 goal is to reach both the target position and attitude.They are expressed in global coordinates as requested.
Herein I define two task:

- the first to control the position.
- the second to control the attitude.

I consider this relation:

$$\dot{\mathbf{x}} = J\dot{\mathbf{y}}$$

with:

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{\mathbf{q}} \\ {}^v\dot{\mathbf{v}}_{v/w} \\ {}^v\dot{\omega}_{v/w} \end{bmatrix}$$

The vehicle position and the misalignment vector are chosen as variables because I consider goal and vehicle position/attitude expressed w.r.t world frame.

$$\dot{x} = D_w(P - O_w) =^w \mathbf{v}_{v/w}$$

(derivative of point w.r.t world frame)
I need to calculate the derivative of ρ for the other jacobian.

$$D_\alpha \boldsymbol{\rho} = \boldsymbol{n}\dot{\theta} + \theta D_\alpha \boldsymbol{n} = \boldsymbol{n}\boldsymbol{n} \bullet \boldsymbol{w}_{b/a} + \theta \boldsymbol{N}(\theta)\boldsymbol{w}_{b/\alpha} - \theta \boldsymbol{M}(\theta)\boldsymbol{w}_{\alpha/\alpha}$$

Considering our observer on the world frame the equation results:

$$D_w \boldsymbol{\rho} = \boldsymbol{n}\boldsymbol{n} \bullet \boldsymbol{w}_{v/W} + \theta \boldsymbol{N}(\theta)\boldsymbol{w}_{v/w}$$

To align to the versor n, the second term must be null because it is orthogonal to n. In this way I will be able to consider the derivative in term of ω only.

The Jacobians result:

$$JvehiclePos = \begin{bmatrix} zeros(3,7) & wRv & zeros(3,3) \end{bmatrix}$$

$$JvehicleAtt = \begin{bmatrix} zeros(3,7) & zeros(3,3) & wRv \end{bmatrix}$$

(wRv is the rotation matrix from frame v to frame w).

I need to project the variable on the world frame, because I calculated the derivative of the variable w.r.t. w.

The task reference needed has to have this form:

$$\dot{\overline{x}} = \lambda(\overline{x} - x)$$

Which represents the desired rate of variables'change.
I defined two task references computing the cartesian error between the vehicle frame and the goal frame (all projected in the world frame). The first is the reference for the position, the second is the reference for attitude.

```
%error between goal and vehicle position and orientation projected on <w>
[w_ang, w_lin] = CartError(uvms.wTgvehicle , uvms.wTv);
uvms.xdot.vehiclePos(1:3,:) = Saturate(0.5 * w_lin, 0.5);
uvms.xdot.vehicleAtt(1:3,:) = Saturate(0.2 * w_ang, 0.2);
```

Figure 1: taskReference vehicle position and attitude

The saturation function limits the max values returned as feedback to have an admissible operation with the physical properties of the system.

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 vehicle goes down to the target position, and it is free to move and rotate to the goal attitude through the Disabling of the horizontal attitude (safety task), without changing initial and goal coordinates.
In details, here I tried different combinations of initial vehicle position and goal position: in almost all cases the vehicle reaches the goal.

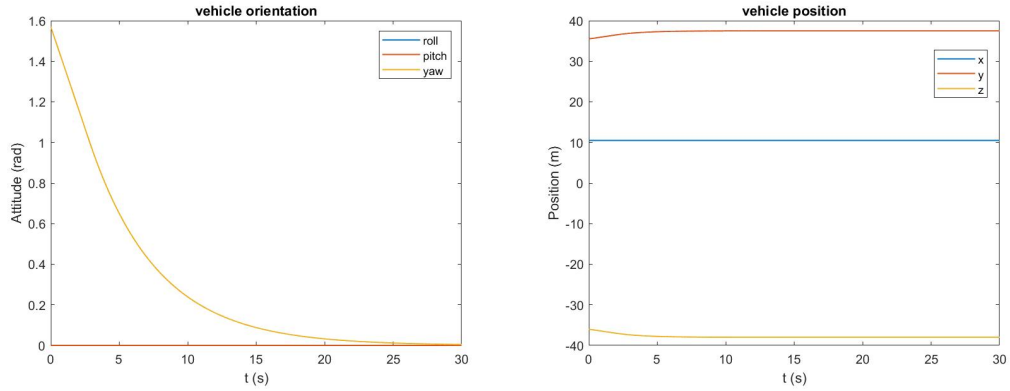


Figure 2: position and attitude

$$startPosition = \begin{bmatrix} 10.5000 \\ 35.5000 \\ -36.0000 \\ 0 \\ 0 \\ \pi/2 \end{bmatrix} \quad goalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad finalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ 0 \\ 0 \\ 0.0044 \end{bmatrix}$$

In this second case the vehicle reaches the target position without any problem as in case one:

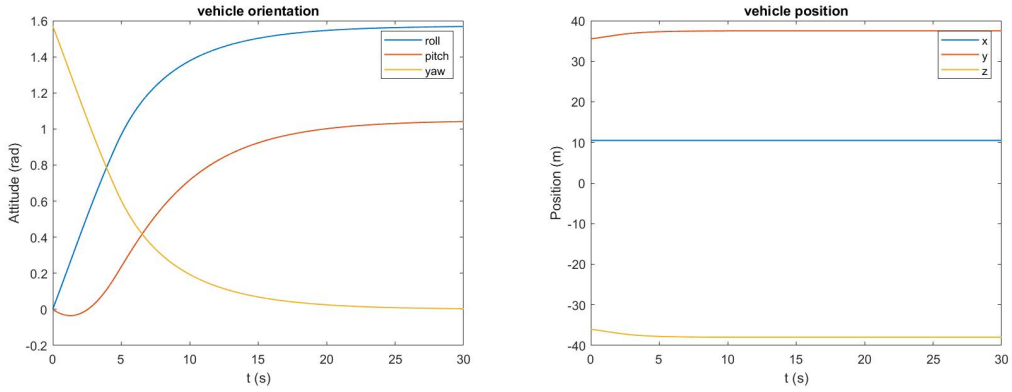


Figure 3: position and attitude

$$startPosition = \begin{bmatrix} 10.5000 \\ 35.5000 \\ -36.0000 \\ 0 \\ 0 \\ \pi/2 \end{bmatrix} goalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ \pi/2 \\ \pi/3 \\ 0 \end{bmatrix} finalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ 1.5675 \\ 1.0410 \\ 0.0033 \end{bmatrix}$$

The vehicle behaves strangely only in the cases where it lose at list one d.o.f (Gimbal lock), I.E. when it starts with this angle configuration:

$$(0, \pi/2, \pi/2)$$

A singularity occurs when the middle rotation in the sequence makes the rotation axes of the first and third rotations parallel.

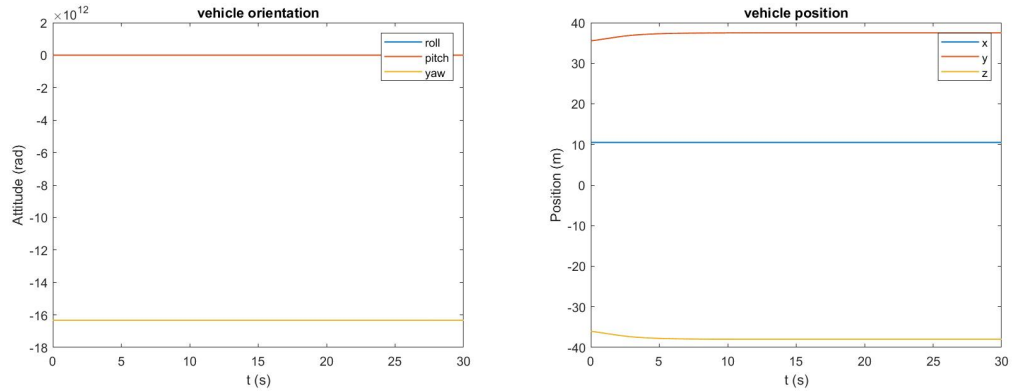


Figure 4: position and attitude

$$startPosition = \begin{bmatrix} 10.5000 \\ 35.5000 \\ -36.0000 \\ 0 \\ \pi/2 \\ \pi/2 \end{bmatrix} goalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ 0 \\ 0 \\ 0 \end{bmatrix} finalPosition = 1.0e+13 * \begin{bmatrix} 0.0000 \\ 0.0000 \\ -0.0000 \\ -1.6331 \\ -0.0000 \\ -1.6331 \end{bmatrix}$$

When the horizontal attitude is enabled with the following priority:

Task	Type
Horizontal attitude(S)	I
Vehicle position	E
Vehicle attitude	E

Table 2: single action, go to the target (max priority for safety task)

The vehicle starting point presents some pitch and roll. The objective is maintained as the defined target. The activation function is always enabled in the first phase. The vehicle changes its initial unusual configuration and reach the target (goal).

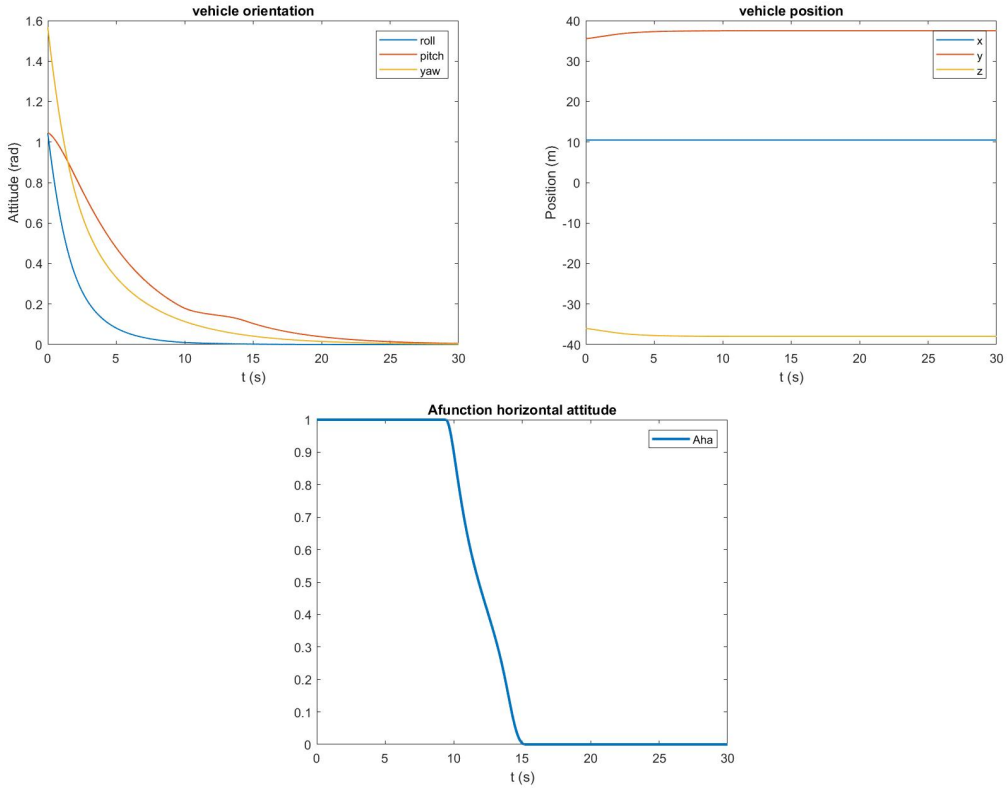


Figure 5: position attitude and activation function

$$startPosition = \begin{bmatrix} 10.5000 \\ 35.5000 \\ -36.0000 \\ \pi/3 \\ \pi/3 \\ \pi/2 \end{bmatrix} \quad goalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad finalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ 0.0000 \\ 0.0052 \\ 0.0021 \end{bmatrix}$$

Vehicle starts with basic configuration and the goal position with some pitch and roll. In this second case the control attitude task has major priority and hinders the goal achievement. The two objectives are indeed in conflict and the first, with higher priority, constraints the vehicle horizontally.

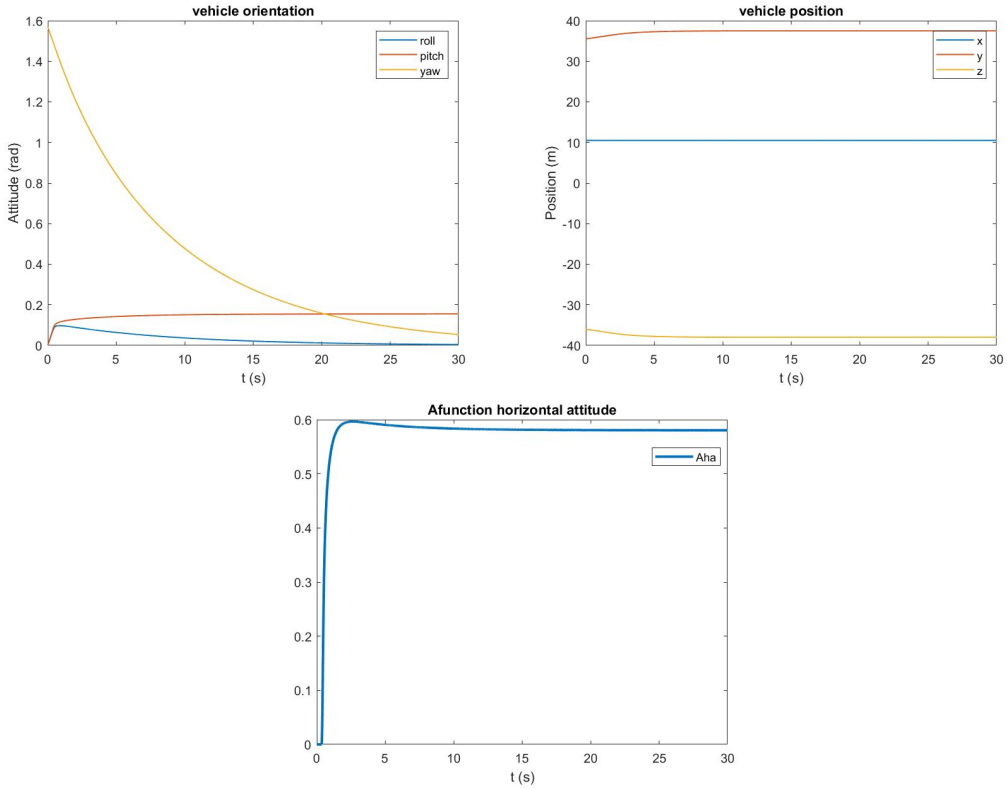


Figure 6: position attitude and activation function

$$startPosition = \begin{bmatrix} 10.5000 \\ 35.5000 \\ -36.0000 \\ 0 \\ 0 \\ \pi/2 \end{bmatrix} \quad goalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ 0 \\ \pi/2 \\ 0 \end{bmatrix} \quad finalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ 0.0041 \\ 0.1551 \\ 0.0533 \end{bmatrix}$$

Starting and goal position attitudes show high pitch and roll.The vehicle tries to move the attitude to a correct configuration but it can't reach the target (position and attitude).

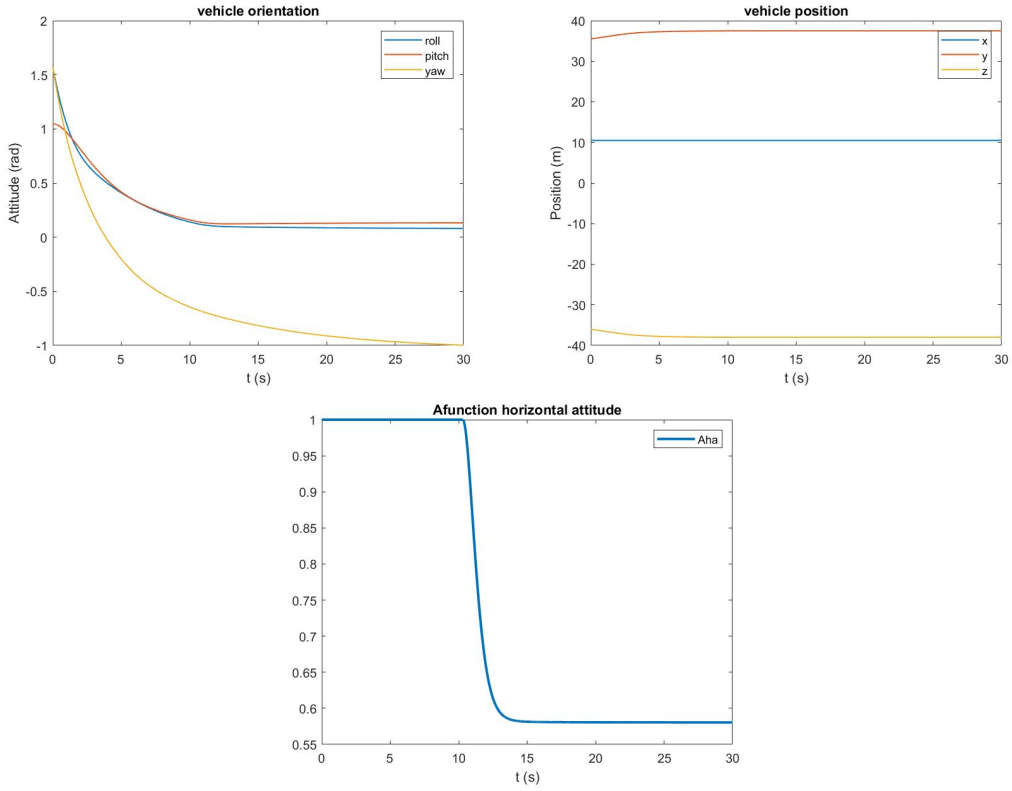


Figure 7: position attitude and activation function

$$startPosition = \begin{bmatrix} 10.5000 \\ 35.5000 \\ -36.0000 \\ \pi/2 \\ \pi/3 \\ \pi/2 \end{bmatrix} \quad goalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ \pi/2 \\ \pi/3 \\ 0 \end{bmatrix} \quad finalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ 0.0810 \\ 0.1325 \\ -0.9982 \end{bmatrix}$$

I did not enter a fourth case as the trivial one because it is in the ones already presented: when start and goal present small or null pitch and roll (which do not enable the horizontal attitude safety task), the vehicle reaches it's final destination without problem.

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

In this configuration the vehicle position tasks are more relevant. The horizontal attitude safety task is ignored also when the activation function is active.When the two objectives (goal attitude and horizontal attitude(S)) are in conflict, the second one is ignored. In this case the mission is more likely to: "go to the target position and, if you can respect the horizontal attitude task".
To complete the exercise showing it I report here just the example where the start and goal have high pitch and roll. In this case the vehicle reaches the goal position (attitude and position).

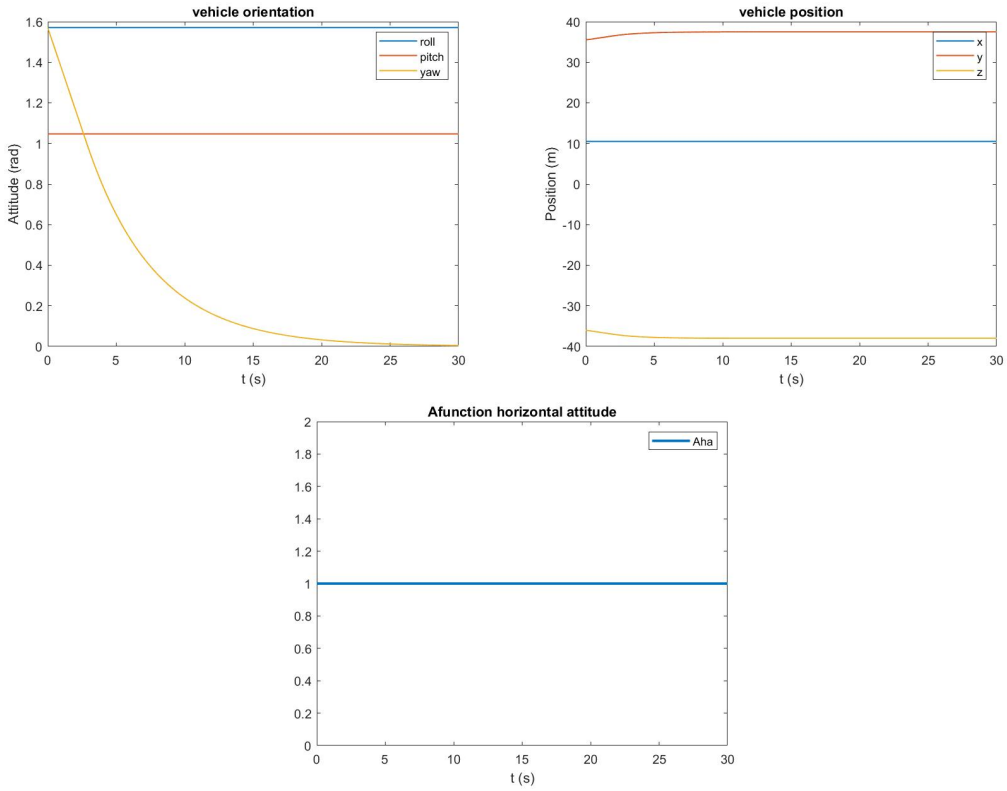


Figure 8: position attitude and activation function

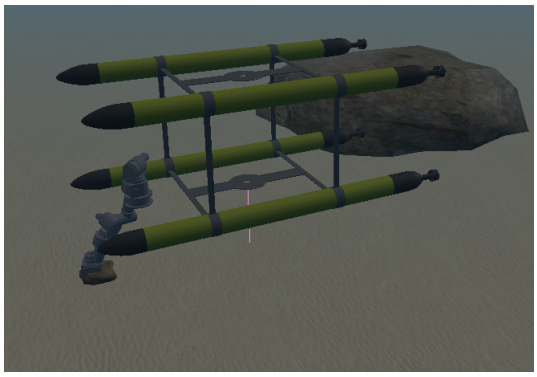
$$startPosition = \begin{bmatrix} 10.5000 \\ 35.5000 \\ -36.0000 \\ \pi/2 \\ \pi/3 \\ \pi/2 \end{bmatrix} \quad goalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ \pi/2 \\ \pi/3 \\ 0 \end{bmatrix} \quad finalPosition = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.000 \\ 1.5708 \\ 1.0472 \\ 0.0044 \end{bmatrix}$$

All the others configurations have the same results (the vehicle reach the target position).

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? Report the final hierarchy of tasks (and their priorities, see the template table in the introduction) which makes up the Safe Waypoint Navigation action.

The vehicle tries to reach its goal position and the tool does not move, when the tool Position control task is disabled, in order with the objective.
When the Tool Position control task is enabled also the tool tries to reach it objective. In this case there may be several scenarios:

- Tool task with higher priority. The vehicle goes down to the sea floor, initially it tries to respect all constraints and it reaches a very close position in respect to the goal one (for vehicle).This happen because they aren't in conflict. Then the tool position with an higher priority guides the robot to the goal position. In this case it goes close to the desired vehicle position only because the vehicle goal is near.

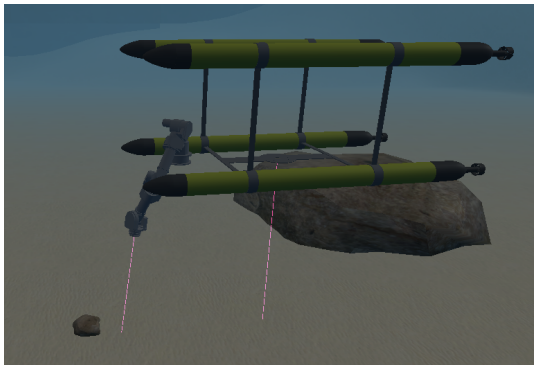


$$finalVehiclePos = \begin{bmatrix} 10.9594 \\ 37.4205 \\ -39.0323 \\ 0.0000 \\ -0.0000 \\ 0.0044 \end{bmatrix}$$

- Vehicle position and attitude with higher priority. The vehicle goes down to the vehicle position and tries to reach the tool objective moving only the arm and using the goal position (for the arm) as

$$(12.2025, 37.3748, -39.8860, 0, \pi, \pi/2)$$

the arm can't reach the goal (if the vehicle target position is the one of the exercise).



$$finalVehiclePos = \begin{bmatrix} 10.5000 \\ 37.5000 \\ -38.0000 \\ 0.0000 \\ 0.0000 \\ 0.0044 \end{bmatrix}$$

- Tool task with higher and very different arm and vehicle objectives (positions). The vehicle ignores its position tasks and it tries to reach the goal for the arm. The tool reaches its goal position.

I considered horizontal attitude task as a safety task, because of the various scenarios. Therefore this task has always been the priority.

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$
Horizontal attitude(S)	I	1	1
Vehicle position	E	2	
Vehicle attitude	E	3	
Tool objective	E		2

In the first action I only enabled the vehicle position and attitude. Now maintaining a correct attitude in the next hypothetical action I enable the tool control.

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 altitude(S)	I	1
Horizontal attitude(S)	I	2
Vehicle position	E	3
Vehicle attitude	E	4

To have a safety navigation, I decide that the first objective is to avoid floor collision (min altitude) the second is to have the horizontal attitude control and the last are the navigation tasks to the goal position and attitude. I made this hierarchy because I want to reach the goal target in a safe way without compromising the vehicle, without the risk of hitting the bottom (worst risk). In general is important to maintain an higher priority for safety tasks and later define the action task.

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 vehicle altitude a is selected as variable, For its determination I considered the vertical component of the vehicle position in the world frame (remind the time derivative of point) the Jacobian relationship result:

$$\dot{a} = \mathbf{k}_w \cdot D_w(P - O_w) = \mathbf{k}_w \cdot^w \mathbf{v}_{v/w}$$

$$JvehicleAlt = \begin{bmatrix}zeros(1,7) & \mathbf{k}'_w \cdot wRv & zeros(1,3)\end{bmatrix}$$

With:

$$\mathbf{k}_w = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

(the k axis of the world frame)

The scalar value used as task reference it is obtained with the difference between the sum of the lower value (safety altitude) and the range of interest, minus the current altitude (distance).

$$distance = \mathbf{k}'_w \cdot \mathbf{wSensorDist}$$

Task reference code:

```
%value t = 1m, 5m, 10m;range 0.5m
threshold = 1;
range = 0.5;
%reference for altittude task 1 m
uvms.xdot.vehicleAlt = Saturate(0.7 * ((threshold + range) - uvms.w_distance), 0.7);
```

Figure 9: task reference for vehicle altitude

The task reference plus the correct setting of the activation (considering (threshold + range)) guarantees the safe altitude. In this case I used an inequality task for activation function because some low altitude value need to be avoided. Indeed, if is greater than (threshold + range) the activation function isn't active. For the other cases it's enough change the threshold in the task reference and in the activation function to 5 or 10 (m).

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?

When the priority is set as 1.2.1, the vehicle can reach the goal position only in the first case (1m) safety altitude.In the two other cases the altitude task (having higher priority) imposes a safety altitude greater than the altitude of the target position.

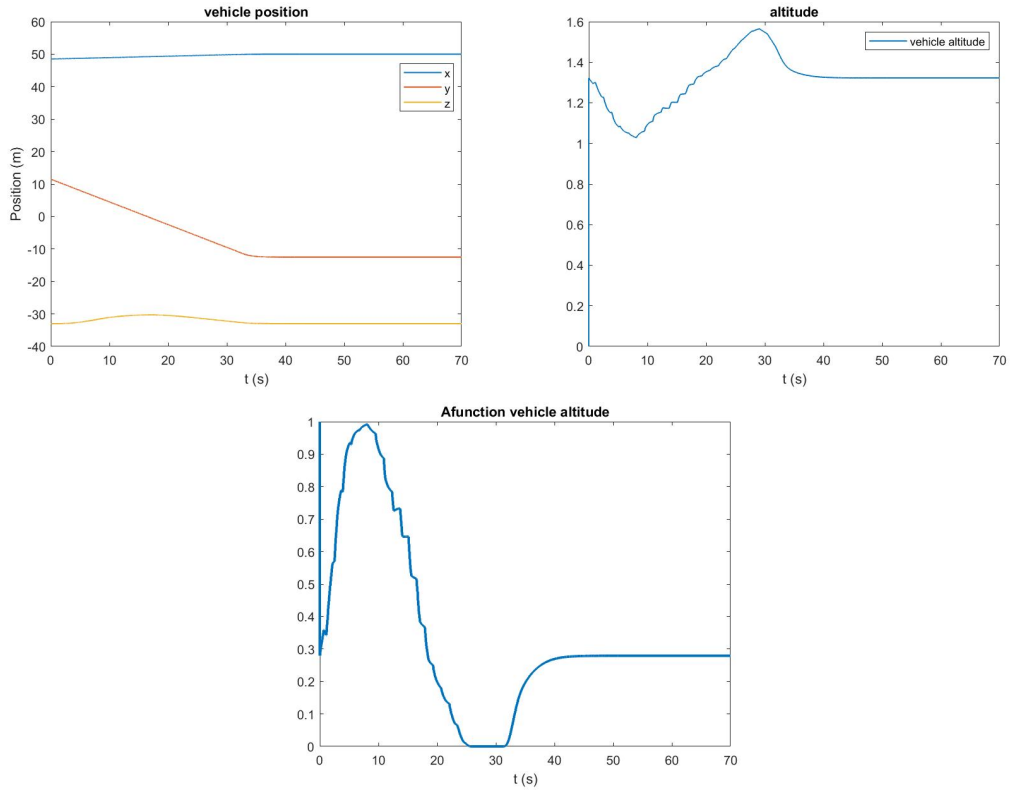


Figure 10: min altitude 1m

In this case the vehicle starts from an altitude lower than 5 meter. The safety task carries the vehicle to the desired safety altitude while it's moving to the target position. At the end it reaches only the goal (x,y) positions, but not z. Indeed the goal altitude is lower than 5m and the safety task kept the vehicle to a greater height.

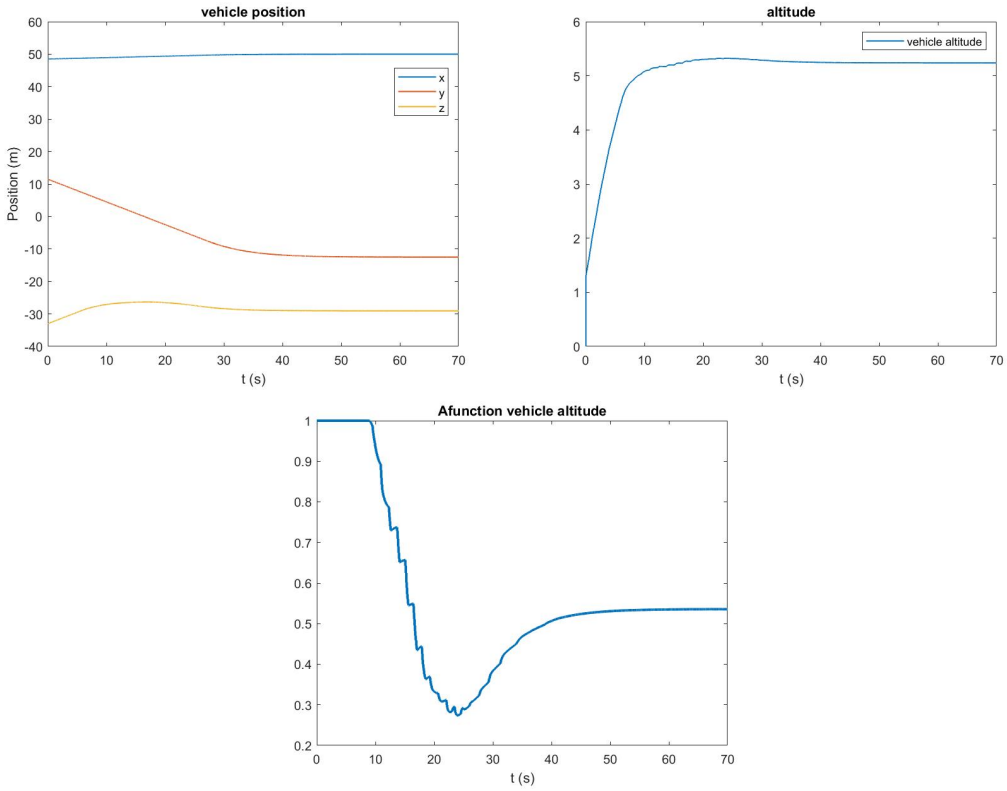


Figure 11: min altitude 5m

The behaviour with 10m min altitude is similar to the previous one. The initial altitude is lower than 10m and the task constraints the vehicle to move to the goal and to increase the altitude. At the end, the vehicle goes to the desired position but with an higher altitude. Because the safety altitude task with an higher priority constraints the vehicle to go up (as previous case).

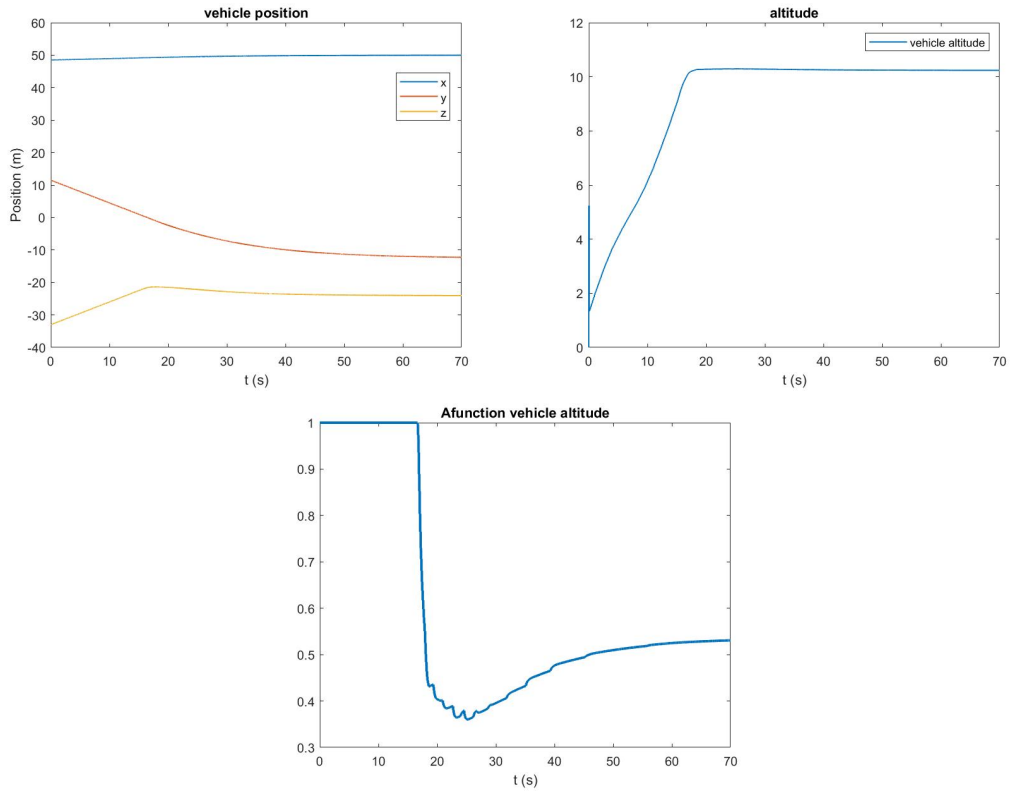


Figure 12: min altitude 10m

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?

The sensor is parallel with the z-axis of the vehicle frame. Considering a flat seafloor the altitude of the vehicle can be computed as the projection of the sensor value in the vertical direction of the world frame.
in formulas:

$$\mathbf{k}_w = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{vSensorDist} = \begin{bmatrix} 0 \\ 0 \\ uvms.sensorDistance \end{bmatrix}$$

$$\mathbf{wSensorDist} = wRv \cdot \mathbf{vSensorDist}$$

$$distance = \mathbf{k}'_w \cdot \mathbf{wSensorDist}$$

(The value provided by the sensor cannot be simply considered, because if the vehicle has a tilted position the sensor measures a wrong distance).
The assumption done (on the characteristics of the seabed) implies that with strange seafloor(high variability of the slope) the value computed results in an approximation of the real value.

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.

Task	Type	$\mathcal{A}_1$
Horizontal attitude	I	1
Vehicle altitude-landing	E	2

For a safe landing, it is important for the sub-marine to have a correct attitude. I don't now exactly the physical structure, but I suppose that to avoid structural damages is important to land in a parallel manner in respect to the seafloor and not upside-down (assuming that the seabed is flat or in any cases with a limited slopes) For this reason I preferred to maintain first the attitude task (as safety task) and then the landing one. I could also think of inserting a task (vehicle position) with a lower priority in order to guarantee landing in the correct x, y position even in the presence of disturbances.

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

The jacobian relationship for landing is the same of the previous exercise because I still want to control the altitude.

$$JvehicleAlt = \begin{bmatrix}zeros(1,7) & \mathbf{k}_w' \cdot wRv & zeros(1,3)\end{bmatrix}$$

with:

$$\mathbf{k}_w = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

here, instead the reference changes. I use the difference between 0 and the scalar value obtained as the scalar product of k_w and the sensor distance project in the world frame, to obtain its vertical component. Because I want to land in this mission.

```
%reference for landing action
uvms.xdot.vehicleAltLanding = Saturate(0.5 * (0 - uvms.w_distance), 0.5);
```

Figure 13: task reference altitude landing

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

With minimum altitude task I used an inequality task because I only want to keep the altitude over some "critical value". The task is active only when the vehicle is near to it. In this case instead, I want to reach a target (I use an equality task).
The task in the landing action is always active, while the reference and the jacobian have the same structure as the previous case. Only the threshold value changes, and it also have a lower priority now. In this case it is no more a safety task.

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.

Now I defined two action:

- Safety navigation to the given position.
- Landing.

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$
Minimum altitude(S)	I	1	
Horizontal attitude(S)	I	2	1
Vehicle altitude-landing	E		2
Task vehicle position	E	3	
Task vehicle attitude	E	4	

In the first action the vehicle is brought to the target position/att. while maintaining a safety height and a safety attitude. Laterit land to the seafloor maintaining a correct attitude to prevent damages.

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

When the vehicle is near to the vehicle goal position of the safe navigation action, the simulation pass to mission phase 2 and the transition begins. In 2 second time using the mission.phasetime and decreasing/increasing bell shaped function I disable the safety altitude task and simultaneously I enable the landing task. Horizontal attitude task remains active as first. I used the function (inc/dec. bell shaped) for a smooth transition.

```
switch mission.phase
    case 1
        %change mission phase
        [w_ang, w_lin] = CartError(uvms.wTgvehicle , uvms.wTv);
        if(norm(w_lin) < 0.1)
            mission.phase = 2;
            mission.phase_time = 0;
        end
     case 2

end
```

Figure 14: update mission phase

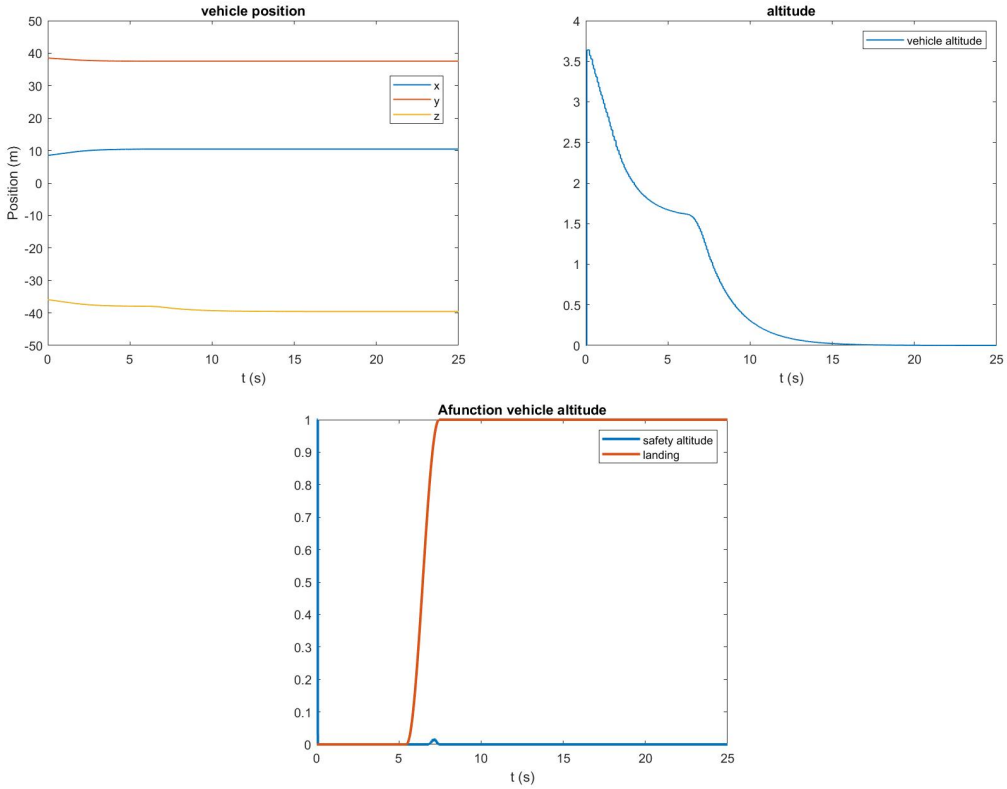


Figure 15: actions safe navigation and landing

The safety altitude task isn't active in the first action because the goal position it's over 1.5m to the floor. The inequality task isn't active for this reason. Around 5.4(s) the condition for change mission phase is satisfied. In two second, the tasks active in the first action but not in the next one are disabled, with a decreasing bell shape function. The same changes are done in the opposite way for the tasks that become active. The small blue shape(bell) around 7(s) is the residual component. It is due to the current altitude (under safety altitude) and the decreasing bell that deactivate them.

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.

New hierarchy:

- Safety navigation to the given position
- Land aligned

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$
Minimum altitude(S)	I	1	
Horizontal attitude(S)	I	2	1
Vehicle alignment	E		2
Vehicle altitude-landing	E		3
Vehicle position	E	3	
Vehicle attitude	E	4	

Here I defined two actions, the first is a safe navigation to the target point and then, when the vehicle is near to the goal position (linear distance), the execution switch to mission phase 2 where the vehicle tries to align to the target and land.

```
switch mission.phase
    case 1
        %change mission phase
        [w_ang, w_lin] = CartError(uvms.wTgvehicle , uvms.wTv);
        if(norm(w_lin) < 0.1)
            mission.phase = 2;
            mission.phase_time = 0;
        end
     case 2

end
```

Figure 16: update mission phase

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

I defined d as the distance vector between vehicle origin and goal origin.

$$\mathbf{d} :=^{w} P -^{w} O_v$$

I projected d on the inertial horizontal plane.

$${}^{w}\mathbf{d}_{\text{proj}} =^{w} \mathbf{d} - \mathbf{k}_w \cdot (\mathbf{k}'_w \cdot {}^{\mathbf{w}}\mathbf{d})$$

With:

$$\mathbf{k}_w = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

(of the world frame)
whit the projection of the vector joining vehicle with nodule frame. I compute the misalignment vector with the $reducedVersoLemma$.

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

between:

$${}^{v}\mathbf{d}_{\text{projVers}} = \frac{{}^{v}\mathbf{d}_{\text{proj}}}{\|{}^{v}\mathbf{d}_{\text{proj}}\|}$$

and the longitudinal axis of the vehicle:

$$\mathbf{i}_v = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

(all the parameters expressed w.r.t to the vehicle frame).
In this situations to evaluate the jacobian I need to calculate the derivative of ρ. In general it results:

$$D_\alpha \boldsymbol{\rho} = \boldsymbol{n}\dot{\theta} + \theta D_\alpha \boldsymbol{n} = \boldsymbol{nn} \bullet \boldsymbol{w}_{b/a} + \theta \boldsymbol{N}(\theta)\boldsymbol{w}_{b/\alpha} - \theta \boldsymbol{M}(\theta)\boldsymbol{w}_{\alpha/\alpha}$$

considering our observer on one of the two frame:

$$D_v \boldsymbol{\rho} = \boldsymbol{nn} \bullet \boldsymbol{w}_{b/a} + \theta \boldsymbol{N}(\theta)\boldsymbol{w}_{b/a}$$

rewriting it for the cases descripted before:

$$D_v \boldsymbol{\rho} = \boldsymbol{nn} \bullet \boldsymbol{w}_{d_{\text{projVers}}/i_v} + \theta \boldsymbol{N}(\theta)\boldsymbol{w}_{d_{\text{projVers}}/i_v}$$

where

$$\boldsymbol{\omega}_{d_{\text{projVers}}/i_v} = \boldsymbol{\omega}_{d_{\text{projVers}}/w} - \boldsymbol{\omega}_{i_v/w}$$

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

$$\boldsymbol{\omega}_{i_v/w} = \boldsymbol{\omega}_{v/w}$$

The other term can be computed as follow:

$$\boldsymbol{\omega}_{d_{\text{projVers}}/w} = \left(\frac{\boldsymbol{d}_{\text{proj}}}{\|\boldsymbol{d}_{\text{proj}}\|} \wedge \frac{\boldsymbol{v}_{d_{\text{proj}}/w}}{\left\|\boldsymbol{v}_{d_{\text{proj}}/w}\right\|} \right) \frac{\left\|\boldsymbol{v}_{d_{\text{proj}}/w}\right\|}{\|\boldsymbol{d}_{\text{proj}}\|} = \frac{1}{\|\boldsymbol{d}_{\text{proj}}\|^2} \left(\boldsymbol{d}_{\text{proj}} \wedge \boldsymbol{v}_{d_{\text{proj}}/w}\right)$$

with $\boldsymbol{v}_{d_{\text{proj}}/w} = -\boldsymbol{v}_{v/w}$ i can rewrite it as:

$$\boldsymbol{\omega}_{d_{\text{projVers}}/w} = -\frac{1}{\|\boldsymbol{d}_{\text{proj}}\|^2} \left(\boldsymbol{d}_{\text{proj}} \wedge \boldsymbol{v}_{v/w}\right)$$

and recalling that:

$$\theta \boldsymbol{N}(\theta) = -\frac{\theta}{\sin\theta} \boldsymbol{a} \wedge \{\boldsymbol{b} \wedge [(\boldsymbol{I}_{3\times 3} - \boldsymbol{n}(\boldsymbol{n}\bullet))]\} \boldsymbol{w}_{n_d/i_\varepsilon}$$

the derivative of ρ results:

$$D_v\boldsymbol{\rho} = \left(\boldsymbol{n}\boldsymbol{n}' - \frac{\theta}{\sin\theta}[\boldsymbol{a}\wedge][\boldsymbol{b}\wedge]\left(\boldsymbol{I}_{3\times3} - \boldsymbol{n}\boldsymbol{n}'\right)\right)\begin{bmatrix} \mathbf{0}_{3\times l} & -\frac{1}{\|\boldsymbol{d}_{proj}|^2}[\boldsymbol{d}_{proj}\wedge] & -\boldsymbol{I}_{3\times3} \end{bmatrix}\begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{v}_{v/w} \\ \boldsymbol{\omega}_{v/w} \end{bmatrix}$$

Looking only the derivative of θ in order to allign the vehicle the jacobian result:

$$JvehicleAllignement = \boldsymbol{n}'\begin{bmatrix} zeros(3,7) & -\frac{1}{\|\boldsymbol{d}_{proj}\|^2}[\boldsymbol{d}_{proj}\wedge] & -\boldsymbol{I}_{3\times3} \end{bmatrix}$$

As a task reference I defined the difference between the desired angle and the current value:

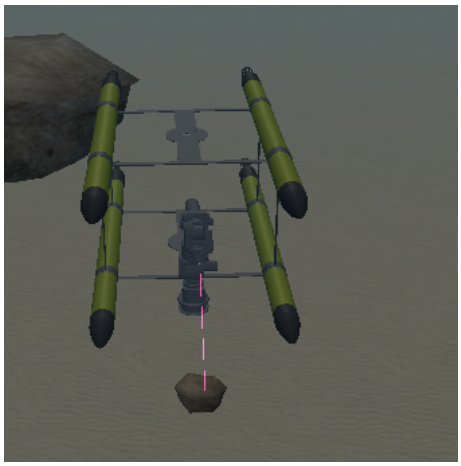
```
%reference for horizontal alignment
uvms.xdot.vehiclehorAlignement = Saturate(0.3 * (0 - uvms.theta), 0.3);
```

Figure 17: task reference vehicle alignment

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?

The solution developed depend on the gain of the task reference. Without changing the landing gain, the vehicle employs always the same time to land. While changing the alignment gain it employs different time to perform the required rotation from the safe goal attitude to the desired attitude to approach the stone.
In particular:

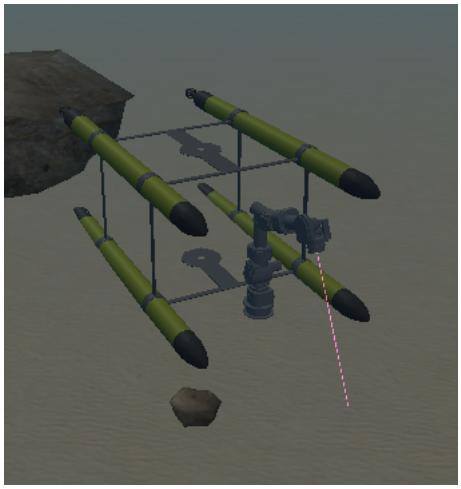
With this configuration the vehicle reaches the correct orientation when it is landed.



```
%reference for horizontal alignment
uvms.xdot.vehiclehorAlignement = Saturate(0.7 * (0 - uvms.theta), 0.7);
```

Figure 18: end position reached

With a small gain the vehicle land with a wrong attitude. The vehicle is severely misaligned. If I don't change the mission it tries to reduce the misalignment, scraping the frame with the seafloor. In a real case it could be dangerous or infeasible.



```
%reference for horizontal alignment
uvms.xdot.vehiclehorAlignement = Saturate(0.03 * (0 - uvms.theta), 0.03);
```

Figure 19: end position reached

The vehicle goes close to the alignment goal with an intermediate gain. But the scraping issue remain as before.

```
%reference for horizontal alignment
uvms.xdot.vehiclehorAlignement = Saturate(0.15 * (0 - uvms.theta), 0.15);
```

Figure 20: end position reached

To develop a solution that can work with different gains, I decided to subdivide the mission in three action. After a safe navigation to the vehicle goal position I have added a new action where I concentrate the effort to reduce the misalignment. The "update mission phase" wait until the misalignment is less than 4 degrees. When it is accomplished the action is changed to land with the same tasks as before.

- Safe navigation to the target
- Alignment with the goal
- Land while maintaining alignment

To explain the solution I report also the task hierarchy:

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
Minimum altitude(S)	I	1	1	
Horizontal attitude(S)	I	2	2	1
Vehicle alignment	E		3	2
Vehicle altitude-landing	E			3
Vehicle position	E	3	3	
Vehicle attitude	E	4		

In the action 2 I have kept the safety tasks active, because they not influence the alignment and take the vehicle in a safe configuration while the rotation is performing.
Whit this configuration, and all the previous gain, the achievement of the target is guaranteed.

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, how would you solve this problem to guarantee it?

If the arm reach the goal it is just a coincidence, because there is no task that constraints the vehicle to a correct distance. In this case we are lucky but for general cases I need to introduce new tasks. Looking the simulation it is also evident how without blocking the movement of the vehicle during the manipulation it tends to move, (this problem is covered in ex 4).

- Safe navigation to the target
- Alignment with the goal + target distance
- Land while maintaining alignment and target distance
- Tool control

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$
Minimum altitude(S)	I	1	1		
Horizontal attitude(S)	I	2	2	1	1
Vehicle alignment	E		3	2	
Vehicle altitude-landing	E			3	3
target distance U	I		4	4	
target distance L	I		5	5	
Vehicle position	E	3			
Vehicle attitude	E	4			
Tool control	E				2

I prefer to control the target distance in action two because it is safer to move with an higher altitude w.r.t obstacle. With this solution I can avoid making movements close to the bottom that could raise sand (in a real case).

I considered as variable the norm of the distance on the plane between the vehicle and the goal. The jacobian result:

$$\mathbf{targetDistance} = \mathbf{rockCenter}(x, y) - \mathbf{vehicleP}(x, y)$$

$$JtargetDistance = \left[zeros(1, 7) - (1/norm(\mathbf{targetDistance})) * [\mathbf{targetDistance}', 0] * wRv \quad zeros(1, 3)\right]$$

It is also important not to land on the target. I introduced two tasks, one to constraint the maximum distance "target distance U" and another for the minimum "target distance L". The goal is to bring the vehicle to a distance comprised below certain limits to guarantee the achievement of the goal and at the same time a minimum distance from the target.

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?

New hierarchy:

- Safe navigation to the target
- Alignment with the goal + target distance
- Land while maintaining alignment
- Tool control + stop vehicle motion

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$
Minimum altitude(S)	I	1	1		
Horizontal attitude(S)	I	2	2	1	
Vehicle stop	E				1
Vehicle alignment	E		3	2	
Vehicle altitude-landing	E			3	
target distance U	I		4	4	
target distance L	I		5	5	
Vehicle position	E	3			
Vehicle attitude	E	4			
Tool control	E				2

I always define 4 action, but in this fourth I added a new task to constraint the vehicle velocity to zero.

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

The Jacobian takes into account the linear and angular vehicle velocities of the vehicle itself.
The matrix is simple:

$$JvehicleStop = \begin{bmatrix} zeros(6,7) & I(6,6) \end{bmatrix}$$

I'm only interested on vehicle behaviour, and to keep it still, the task reference becomes:

```
%stop vehicle
uvms.xdot.vehicleStop = zeros(6,1);
```

Figure 21: vehicle stop reference

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

I modified the previous exercise deleting the landing action. The current mission results:

- Safe navigation to the target
- Alignment with the goal + target distance
- Floating manipulation

In the third action I added some noise simulating the effect of the current. The external factor influences the vehicle velocity, the *vehiclestop* it is not enough to keep the vehicle stationary. At any iteration the task constraint the vehicle to the position influenced by the disturbance. As you can see in the graph the vehicle is not stationary.

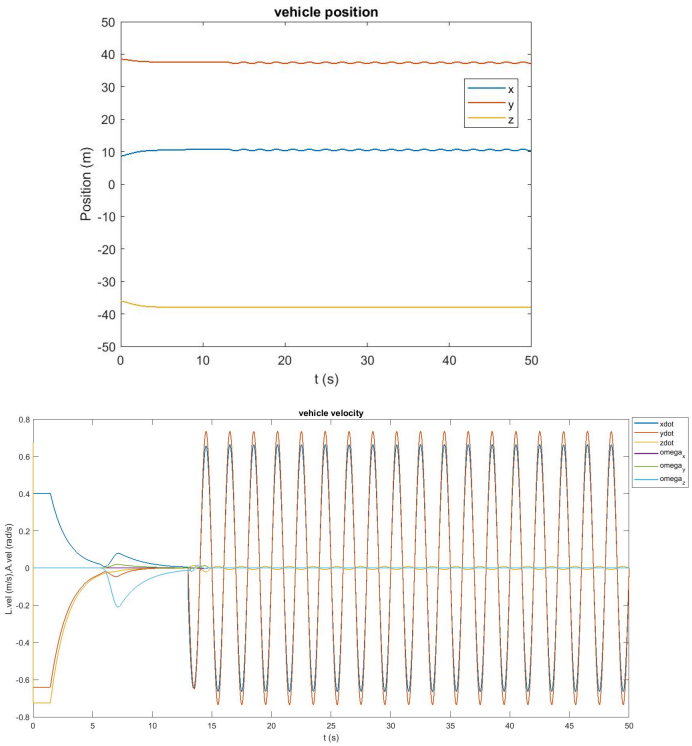


Figure 22: position and velocity with current noise

Code to generate the current noise:

```
%currents noise
if (mission.phase == 3)
    noise = sin(2 * pi * 0.5 * t) * [0.7 0.7 0 0 0 0]';
    noisePdot = [uvms.vTw(1:3,1:3)  zeros(3,3);zeros(3,3) uvms.vTw(1:3,1:3)]*noise;
    uvms.q_dot = ydotbar(1:7);
    uvms.p_dot = ydotbar(8:13) + noisePdot;
else
    uvms.q_dot = ydotbar(1:7);
    uvms.p_dot = ydotbar(8:13);
end
```

Figure 23: current noise

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?

The mission is composed by four actions:

- Safe navigation to the target
- Alignment with the goal + target distance
- Land while maintaining alignment
- Tool control + stop vehicle motion

When the vehicle is landed, in the manipulation action I have add the joint limit tasks. I considered them as a safety tasks. For this reason they have higher priority than the tool task.

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$
Minimum altitude(S)	I	1	1		
Horizontal attitude(S)	I	2	2	1	
Vehicle stop	E				1
Tool upper joint limits	I				2
Tool lower joint limits	I				3
Vehicle alignment	E		3	2	
Vehicle altitude-landing	E			3	
target distance U	I		4	4	
target distance L	I		5	5	
Vehicle position	E	3			
Vehicle attitude	E	4			
Tool control	E				4

The mission is like before, I have added two new tasks to keep the arm joint under some limits provided by the code.

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

I only consider the joint position in this case, forgetting the vehicle position and attitude,the jacobian become simple considering only the arm:

$$JjointLimits = \begin{bmatrix} I(7,7) & zeros(6,6) \end{bmatrix}$$

I have two task reference to carry on the arm under some constraint. In particular with the next two task I try to keep the joints values in the middle (for each joint) between min and max values. In particular I used two inequality tasks:
for lower bounds I use as a reference the difference between the lowerlimt plus some confidence range minus the current joint position.

```
%joint limits lower
uvms.xdot.jointLimitsL = Saturate(0.3 * ((uvms.jlmin + uvms.rangeJoint) -uvms.q), 0.3);
```

Figure 24: task reference joint lower bound

The lower limit plus the chosen range is the same value that I use in the activation function of this inequality task as the value from over them I disable the function. I want to keep the joint in the middle space defined between the lower and upper values. To accomplish I force the arm to change its values only when they are going near to the constraints. It's important that the range value isn't too big to interfere with the other constraints (upper limit).

The dual case is the upper bound:

```
%joint limits upper
uvms.xdot.jointLimitsU = Saturate(0.3 * ((uvms.jlmax - uvms.rangeJoint) -uvms.q), 0.3);
```

Figure 25: task reference joint upper bound

I tried some different goal orientations for the arm. In the example I impose as target orientation for the tool:

$$\begin{bmatrix} \pi/4 \\ \pi/3 \\ \pi/2 \end{bmatrix}$$

without the joint limits the arm reach the goal, with a specific value for the joints:

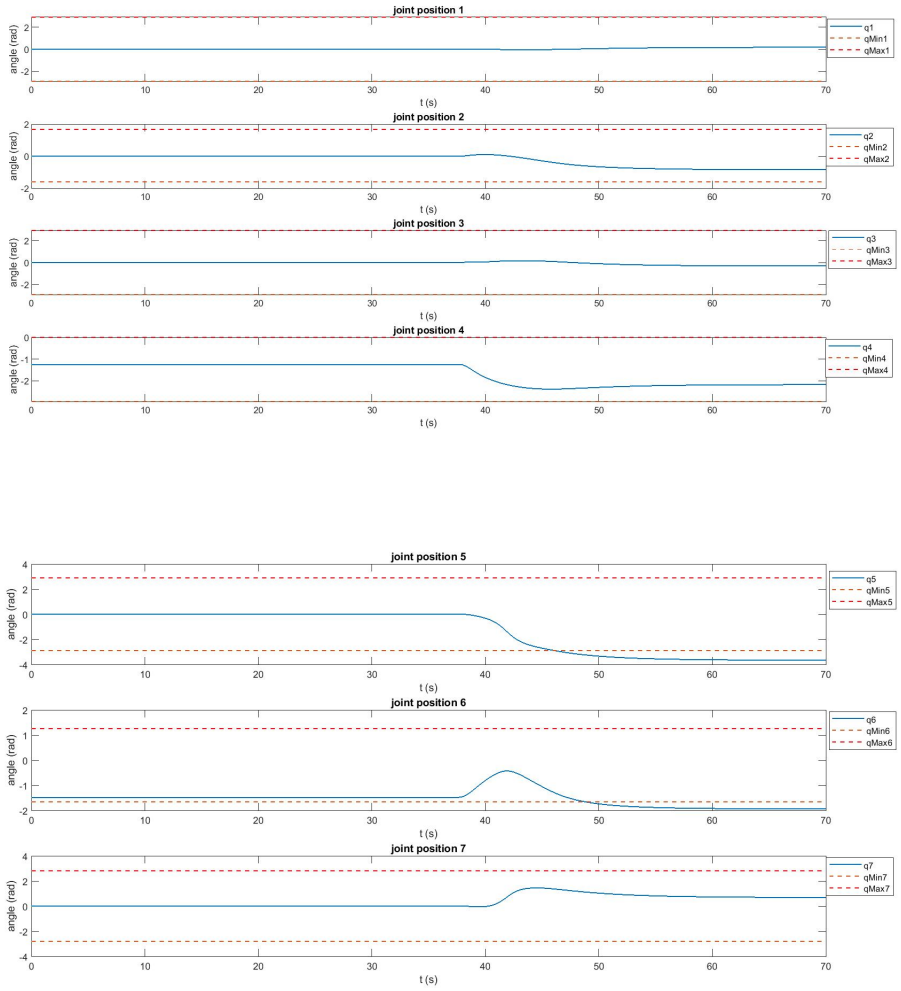


Figure 26: no limit joints behaviour

$$angulare rror = \begin{bmatrix} -0.0085 \\ -0.0041 \\ 0.0041 \end{bmatrix} \; linearerror = \begin{bmatrix} -0.0000 \\ -0.0000 \\ -0.0011 \end{bmatrix}$$

Angular and linear error between tool goal and position reached.

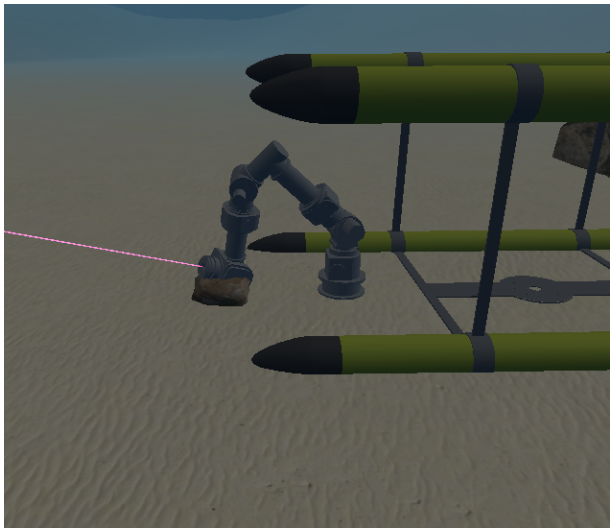


Figure 27: no limit tool behaviour

With the join limits the arm go close to the target position. The two task are safety ones with higher priority compared to tool task. The final target position is not guaranteed.it depends on the goal: in some cases the tool can reach the target as expected limiting strange behaviours. In others the safety task block extra movements, dangerous for the robot.

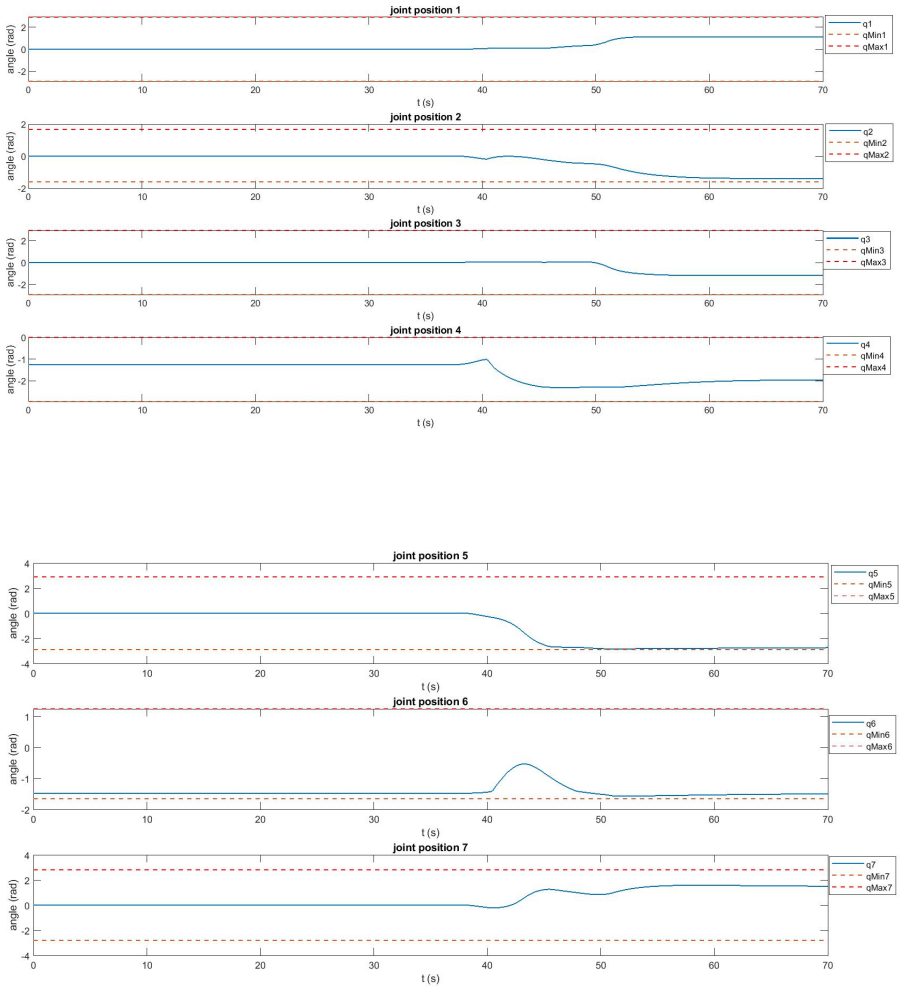


Figure 28: limit joints behaviour

$$angulararerror = \begin{bmatrix} -0.0109 \\ -0.0471 \\ 0.0770 \end{bmatrix} \; linearerror = \begin{bmatrix} -0.1171 \\ -0.1243 \\ -0.0641 \end{bmatrix}$$

Angular and linear error between tool goal and position reached.

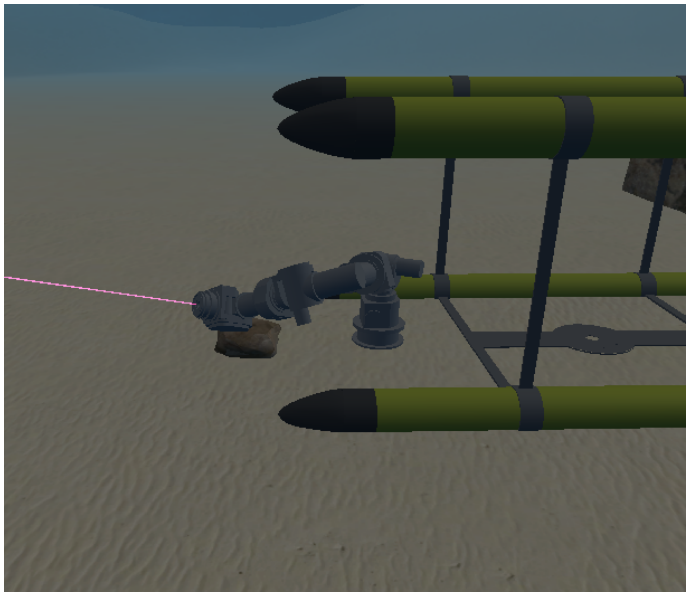


Figure 29: limit tool behaviour

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

Hierarchy:

Task	Type	$\mathcal{A}_1$
Horizontal attitude(S)	I	1
Tool upper joint limits	I	2
Tool lower joint limits	I	3
Tool	E	4
Prefered shape(O)	E	5

For the optimization tool task I selected the min priority. This is an optimization task with lower importance. Its Violation does not involve risks (it isn't a safety task). In others cases different from this one, where others tasks constrain the position of the vehicle (the seabed or obstacles). I prefer to reach the target by violating the optimal configuration rather than not reaching it. Otherwise if I can reach the goal I will prefer a specific shape of the manipulator. the violation of the preferred shape does not compromise the operation of the robot/vehicle, but it could still have less maneuverability in the end-effector.

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

Only the position of the first four joints need to be optimized. the jacobian for the arm optimization considering only the shape of the first four joints results:

$$JpreferedShape = \begin{bmatrix} I(4,4) zeros(4,9) \end{bmatrix}$$

I use the difference between the preferred position and the current value of the joint as reference. As a task reference I need something with the form:

$$\dot{\overline{q}} = \lambda(\overline{q} - q)$$

where $\overline{q}$ is the constant vector with the desired position of the first four joint called "prefSetting"

```
%prefered shapes
prefSetting = [-0.0031 1.2586 0.0128 -1.2460]';
uvms.xdot.preferedShape = Saturate(0.3 * (prefSetting- uvms.q(1:4)), 0.3);
```

Figure 30: joint optimization

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

Figure 31: no preferred shape

The vehicle assume this configuration if no optimization task is performed, all joints are free to move under up to the limits and the arm is very extended.

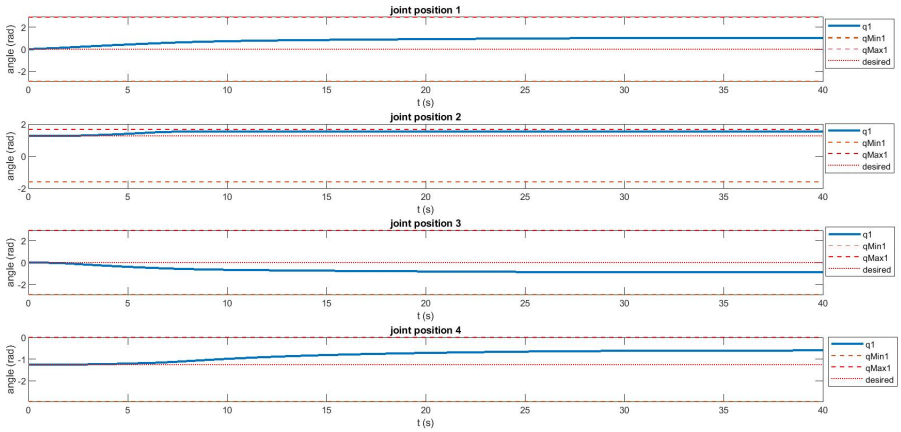


Figure 32: no preferred shape 4 interest joint

The arm respects the limits removing the optimization task, but leaving the safety task for the joints. Its position is different from the desired one.

With the optimization task:

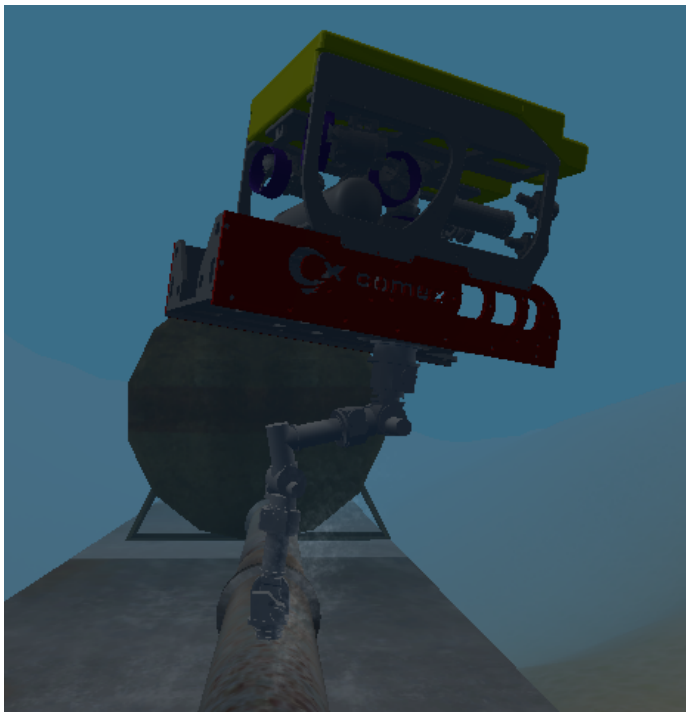


Figure 33: preferred shape

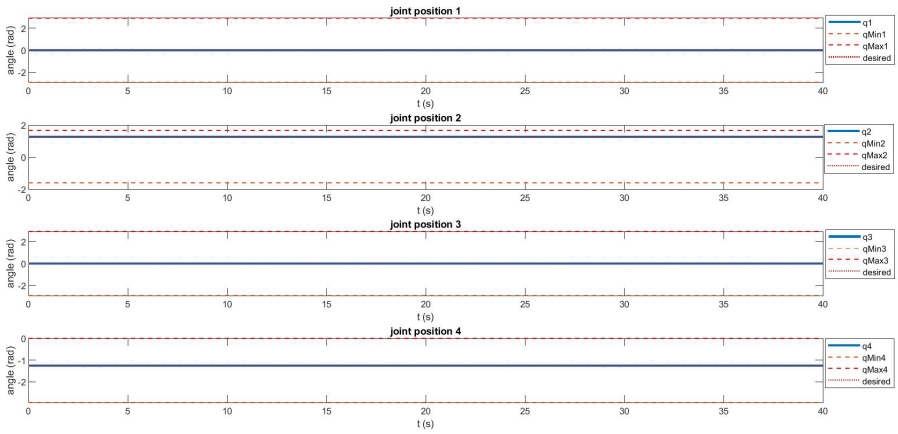


Figure 34: prefered shape 4 interest joint

In this case the optimization task force the arm to assume a determined configuration. The first four joint are fixed. The others joints are free to move under their limits (upper and lower).

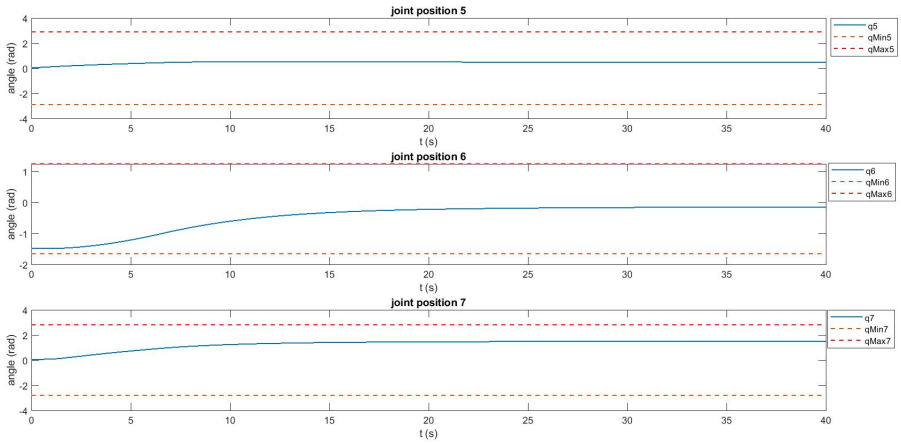


Figure 35: preferred shape others joint

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?

I defined two mission:

- Safe navigation up to to the goal position
- Go to the target with the arm

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$
Horizontal attitude(S)	I	1	1
Tool upper joint limits	I		2
Tool lower joint limits	I		3
Vehicle position	E	2	
Vehicle attitude	E	3	
Tool control	E		4
Preferred Shape(O)	E		5
Vehicle stop(O)	E		6

I use the vehicle stop position in a different way. In exercise 4 we land in the manipulability space of the arm.the movement of the vehicle on the seabed could be dangerous. Here instead the vehicle is floating and depending on the safe navigation altitude it may have to lower himself to reach the goal with the arm.I apply the vehicle stop motion as an optimization task. If the goal cant' be reached with the arm the vehicle can move down.

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

In the previous exercise I have only controlled the tool not the vehicle. The vehicle moves because of the objective of the toll. Now I have defined two actions: first a safe navigation controlling the vehicle position and attitude, second the manipulation, enabling the tool position control.It is not recommended to move the arm first.

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 I had defined two optimization problem:

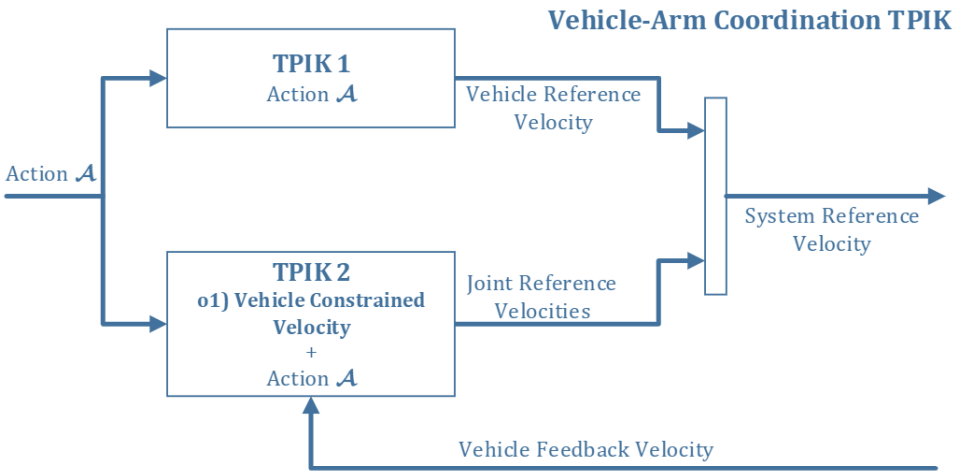


Figure 36: coordination scheme

- First TPIK has the tasks to control and reach the goal as:
  - horizontal attitude
  - upper and lower limit
  - tool control
  - preferred arm shape
- Second TPIK introduce also a new task with higher priority. In this second problem I consider the vehicle as totally non controllable. Only the manipulator variables are subject to be optimized.
  - vehicle non controllable
  - horizontal attitude
  - upper and lower limit
  - tool control
  - preferred arm shape

From the first TPIK I only considered the vehicle reference velocity. From the second TPIK (considering the vehicle totally non controllable) only the manipulator variables are optimized. For this reason I keep the joints velocity from the second.

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.

I added a noise with the form:

```
% add noise
% sinusoidal velocity disturbance * amplitude wrt world frame
dist = sin(2 * pi * t)*[0.0 0.1 0.3 0.0 0.0 0.0]';
% sinusoidal velocity disturbance wrt vehicle frame
noisePdot = [uvms.vTw(1:3,1:3)  zeros(3,3);zeros(3,3) uvms.vTw(1:3,1:3)]*dist;
uvms.q_dot = q_dot;
uvms.p_dot = p_dot + noisePdot;
```

With the amplitude specified as in the previous image the vehicle position and attitude result:

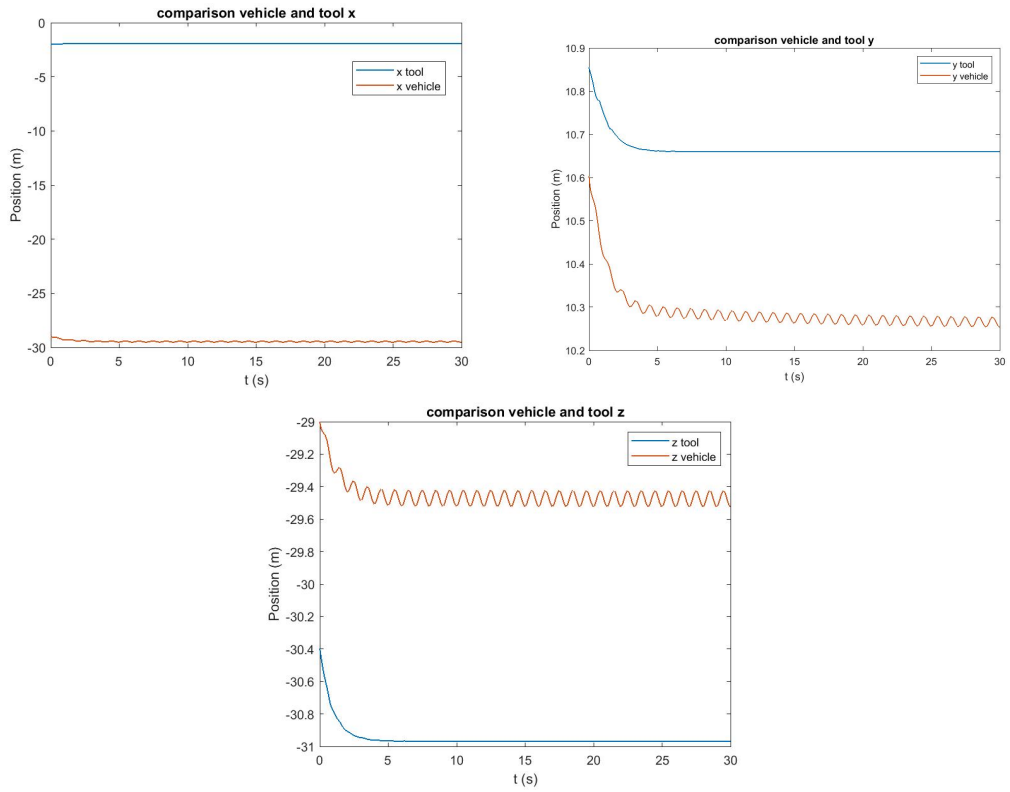


Figure 37: position vehicle and tool

(for the simulation I defined a starting position for the vehicle very close to the goal).

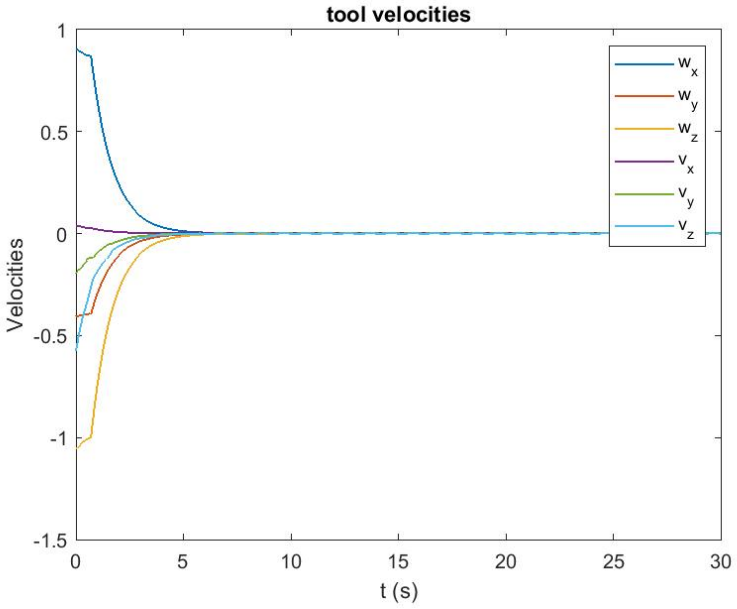


Figure 38: tool velocity

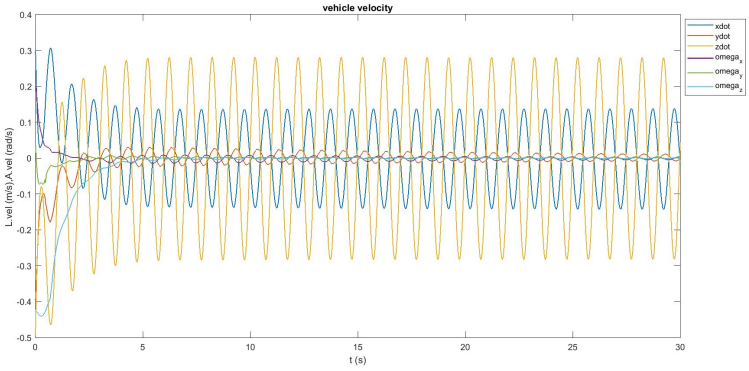


Figure 39: vehicle velocity

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.

If the disturbance increases too much, the arms cannot counteract the disturbance present on the vehicle. Even increasing the gain and the saturation value of the reference is not enough to limit the disturbance.

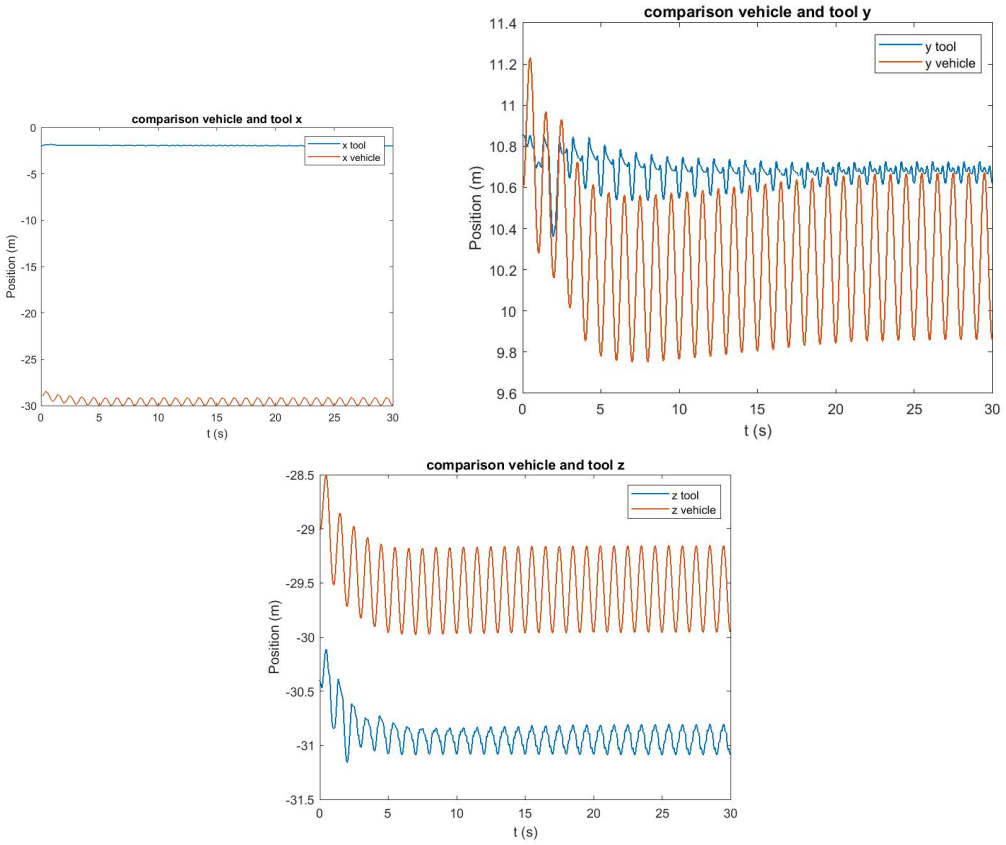


Figure 40: position vehicle and tool

In this specific case I have increased the amplitude of the noise (y and z component) up to 2.5. (the structure of the noise is like the previous example).

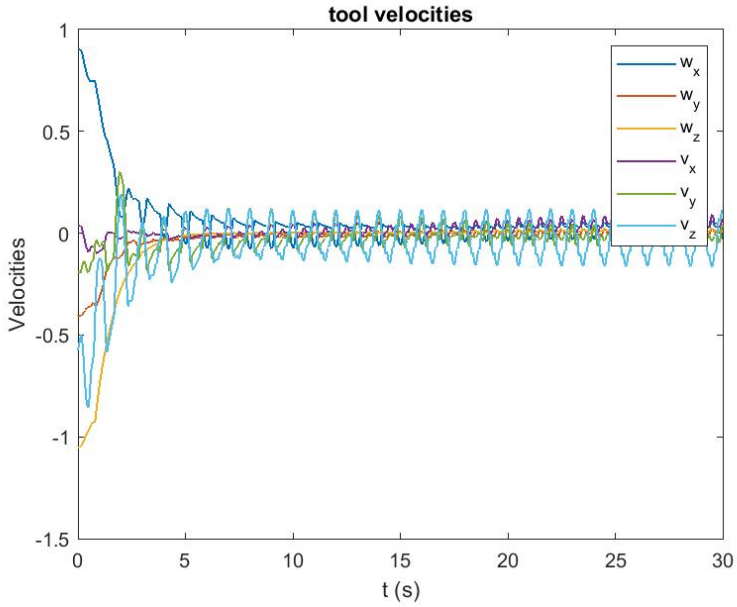


Figure 41: tool velocity

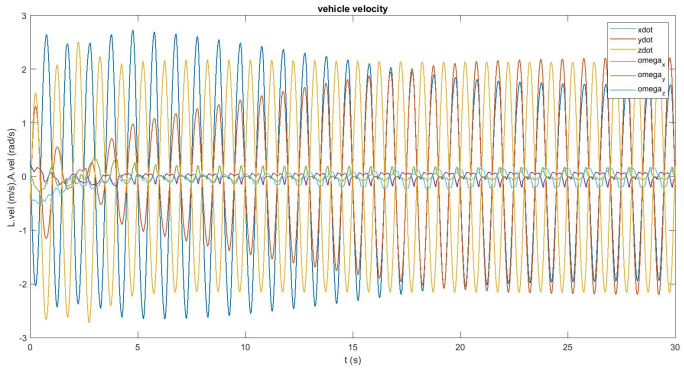


Figure 42: vehicle velocity

The coordination scheme works fairly well with limited currents, as the disturbance does not propagate towards the arm. In case of strong currents, however, it shows its limits as it cannot compensate these disturbances. (if the noise is removed, the vehicle behaves exactly as required and the joint speed provided by the second optimization problem is equal to the first).