Università degli Studi di Genova

CORSO DI LAUREA MAGISTRALE

Robotics Engineering

COOPERATIVE ROBOTICS

Authors:

Giacomo Lugano
Youssef Attia
Hussein Fouad

Email:

jek.lugano@yahoo.com

Youssef-attia@live.com

hussienfouad90@gmail.com

Date: 02/02/2023

I. Robust Nodule Inspection Mission

1. Tasks unified hierarchy

As shown from the table below the Task Priority Control hierarchy is composed of three actions. Firstly, the vehicle is spawned in:

$$[8.5 \quad 38.5 \quad -36 \quad 0 \quad -0.06 \quad 0.5]^T$$

and navigates to:

$$[10.5 \quad 37.5 \quad -38 \quad 0 \quad -0.06 \quad 0.5]^T$$

keeping into account a minimum altitude to the see bed "assuming it's flat" of 1 meter and maintaining a proper pitch, roll and yaw presentenced in the so-called vehicle horizontal attitude control task.

Secondly, the landing action starts in which the vehicle lands to the seabed and concurrently "with higher priority" aligned itself to the nodule to be inspected and keeping a desired distance from the nodule.

Lastly, the inspection actions start maintaining the vehicle position in the so-called Vehicle null velocity task and guiding the end-effector to the nodule keeping into account "with higher priority" the arm joint limits and manipulability to avoid singularity. Furthermore, the vehicle null velocity task is crucial in this action to avoid the vehicle motion due to the forces and moments generated by the end effector motion.

Table 1: Robust Nodule Inspection Mission Task Priority Control hierarchy

Task	Type	Navigate to goal	Landing	Inspection
Arm Joints Limits	I			1
Arm Manipulability	I			2
Vehicle Null Velocity	E			3
End Effector Linear Position	E			4
End Effector Angular Position	E			5
Vehicle minimum altitude	I	1		
Vehicle horizontal attitude	I	2	1	
Vehicle heading control	I	3		
Vehicle position control	E	4		
Vehicle alignment to nodule	I		2	
Vehicle distance to nodule	I		3	
Vehicle altitude control	E		4	

2. Robot's behavior

As per the discussion of the robot's task priority control hierarchy in section I.1. This section will discuss the robot behavior proving it is following the proposed hierarchy.

Starting from the vehicle altitude and attitude throughout its mission which is composed of the three actions. It can be seen from the following graph [Fig.1] of the vehicle altitude that as the vehicle is spawned it has relatively high altitude and moving to the goal with the action *Navigate to waypoint* the altitude starts converging to the goal height. However, the *Vehicle minimum altitude* task should prevent the vehicle from going beneath 1 meter of altitude, but it is not the case here as the goal height is higher than task's threshold. Moreover, as the vehicle have reached its goal and action *Landing* takes over the altitude starts converging to zero and stays there due to the task "Vehicle null velocity" in action *Inspection*.

Similar behavior can be seen in the graph of the vehicle attitude for the misalignment that is used for the *Horizontal attitude* task. However, a small change can be seen between the transition of action *Landing* to action *Inspection*. This is due to the relatively long period of transition between the two actions.

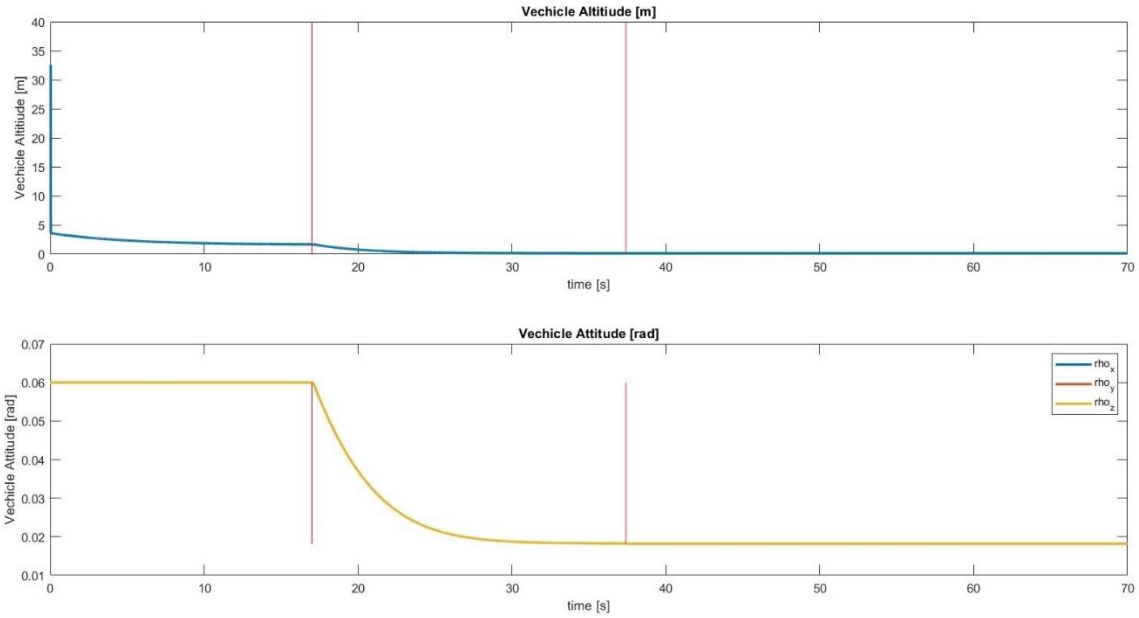


Figure 1:Vehicle Altitude and Attitude

Moreover, the following graphs in Fig.2 shows how the action *Navigate to waypoint* takes effect. In the beginning and as the simulation starts the vehicle is spawned away from the goal. And, as the action starts can be seen that the distance vector starts to converge to zero and same goes to the misalignment vector of the vehicle to the goal. As the Vehicle reaches its goal, the action *Landing* two takes over and the vehicle goes further from the original goal. It can be noted here that the change is mainly on the Z-axis as the vehicle is Landing.

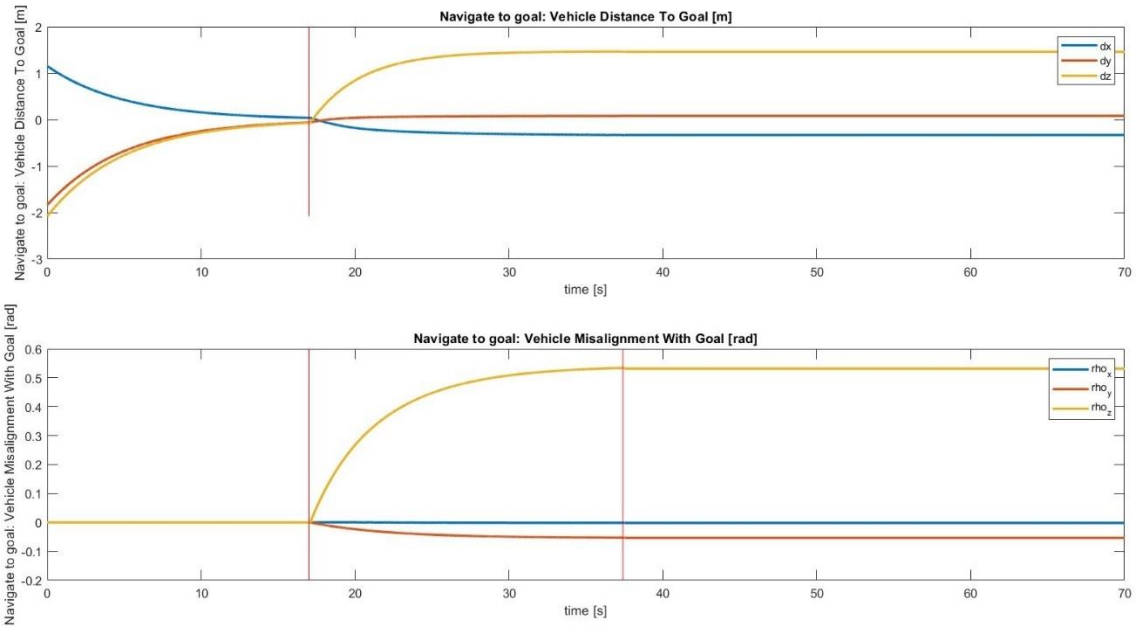


Figure 2: Vehicle distance and misalignment to goal

Moving to the action *Landing* the main goal here is to land making sure that the vehicle is aligned to the nodule and that nodule is in the manipulator's workspace. Furthermore, the robot's behavior in this case can be studied from the following graphs in Fig.3 showing the distance between the vehicle and the nodule and also the misalignment vector between the vehicle and the nodule. It can be noted here that after the action *Navigate to waypoint* has ended the vehicle is not aligned to the nodule and as the action *Landing* aligns the vehicle the misalignment vector converges to zero. Increasing the gain makes the vehicle align faster and vice versa. This is the task *Vehicle longitudinal alignment to nodule* is further discussed in section 3 and section 4.

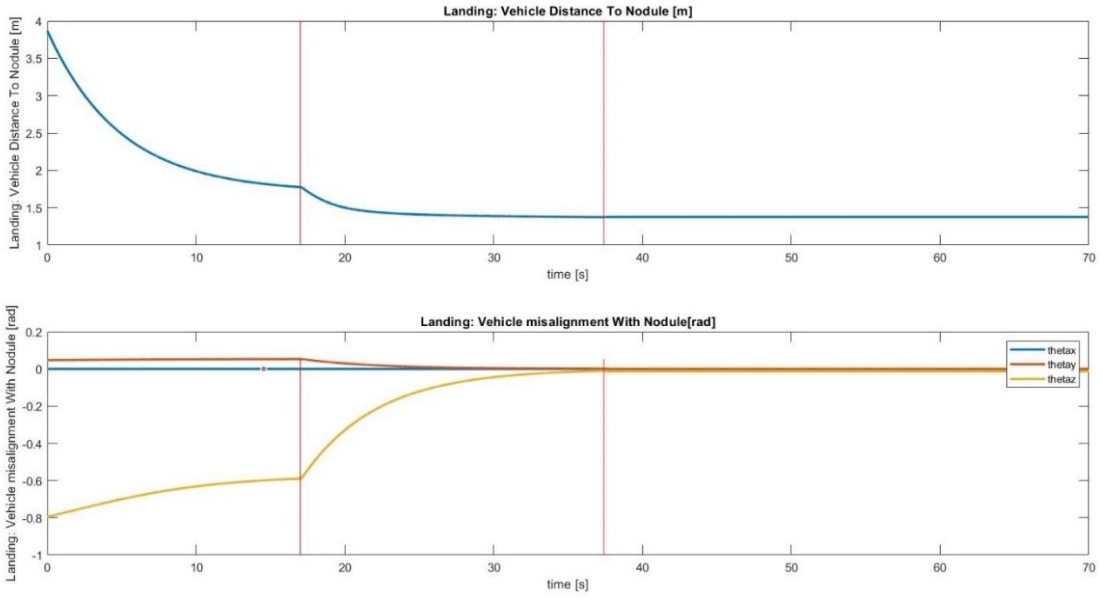


Figure 3: Vehicle distance and misalignment to nodule

Lastly, from *Inspection* action point of view both the Arm distance and misalignment to the nodule should be studied, and it can be seen in Figure 4 that as soon as the *Inspection* action takes place both the distance and the misalignment start to converge to zero and they arrive to zero around t = 60 seconds and this can be considered the end of the mission.

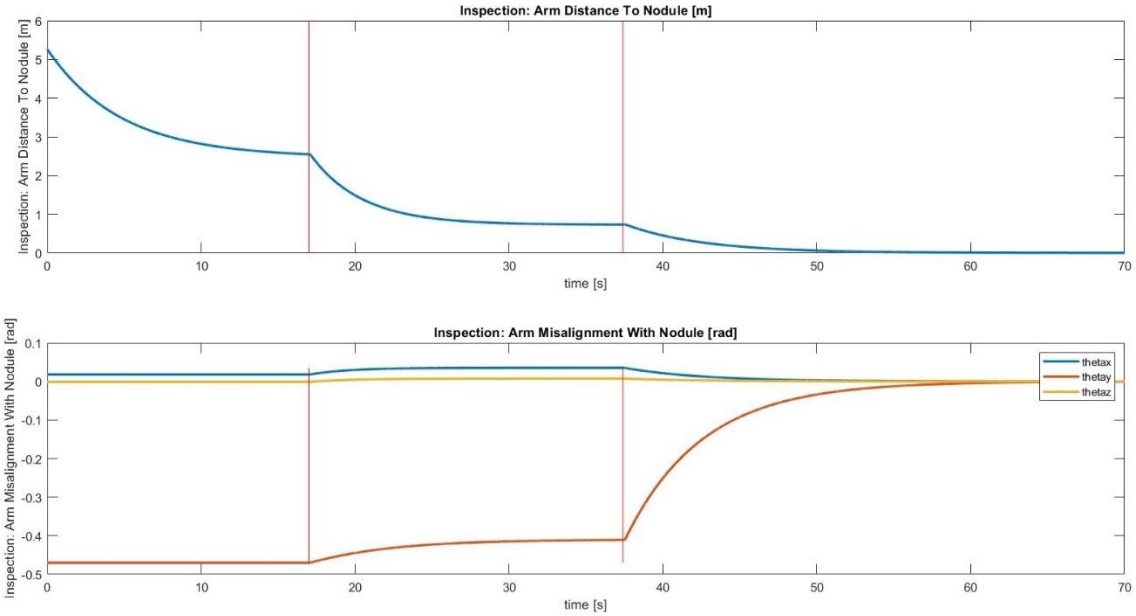


Figure 4: Arm distance and misalignment to nodule

3. Vehicle alignment to nodule task Jacobian

This objective requires that the vehicle aligns to a nodule in the space, the nodule, but only in the horizontal part, as it can be seen from the figure below:

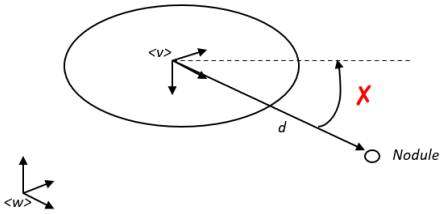


Figure 5: Faulty misalignment evaluation

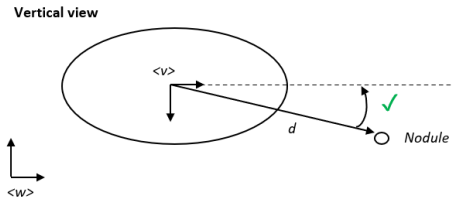


Figure 6: Proper misalignment evaluation

In other word the following equality objective must be satisfied:

$$\boldsymbol{\rho}_{v/g} = 0$$

Where $\boldsymbol{\rho}_{v/g}$ is the misalignment vector between the vehicle $\boldsymbol{i}_v$-unit vector and the projection of the distance vector $\boldsymbol{d}$ on the world horizontal plane.

As it can be seen, this task is structured as an equality task and acquire the attribute of operational prerequisite:

$$\boldsymbol{x} = \bar{\boldsymbol{x}} \qquad \bar{\boldsymbol{x}} = \text{desired misalignment}$$

Therefore, the activation function will be an identity matrix, and since it is desired to control three scalars (the component of the error along x, y, z), of dimension 3x3:

$$\boldsymbol{A} = \boldsymbol{I}_{3\times 3}$$

From here the evaluation of the task Jacobian can take place, let's start by evaluating the distance vector between the vehicle position $\boldsymbol{O}_v$ and the nodule $\boldsymbol{N}$, which can be called $\boldsymbol{d}$.

$$\boldsymbol{d} = (\boldsymbol{N} - \boldsymbol{O}_v)$$

Then project such vector on the world horizontal plane as:

$$\boldsymbol{d1} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\boldsymbol{d} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)](\boldsymbol{N} - \boldsymbol{O}_v) = P(\boldsymbol{N} - \boldsymbol{O}_v)$$

Where $\boldsymbol{k}_w$ is the vertical unit vector of the world reference frame, $\boldsymbol{I}$ the identity matrix.

Thus, the problem would be solved if the $\boldsymbol{i}_v$-unit vector of the vehicle reference frame align to unit vector $\boldsymbol{n}_d = \boldsymbol{d1}/\|\boldsymbol{d1}\|$, let's consider the misalignment vector between the two, computed with the reduced versor lemma:

$$\boldsymbol{\rho} = \theta \boldsymbol{n} = \frac{\theta}{\sin\,\theta}\boldsymbol{i}_v \times \boldsymbol{n}_d$$

To achieve that the two-unit vector align it is needed to find the relationship between how $\boldsymbol{\rho}$ evolves in time and the system control variables $\dot{\boldsymbol{y}}$ .The derivative of $\boldsymbol{\rho}$ with respect to an observer α is given by:

$$D_\alpha \boldsymbol{\rho} = \boldsymbol{n}\dot{\theta} + \theta D_\alpha \boldsymbol{n} = \boldsymbol{n}\boldsymbol{n} \cdot \boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v} + \theta N(\theta)\boldsymbol{\omega}_{\boldsymbol{n}_d/\alpha} - \theta M(\theta)\boldsymbol{\omega}_{\boldsymbol{i}_v/\alpha}$$

If the observer α is considered to be on one of the two vectors, for example considering it on the vehicle reference frame <v>, the equation simplifies as:

$$D_\alpha \boldsymbol{\rho} = \boldsymbol{n}\boldsymbol{n} \cdot \boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v} + \theta N(\theta)\boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v}$$

With the goal of expressing such equation in terms of the system control variables $\dot{\boldsymbol{y}}$, $\boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v}$ can be expressed as:

$$\boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v} = \boldsymbol{\omega}_{\boldsymbol{n}_d/w} - \boldsymbol{\omega}_{\boldsymbol{i}_v/w}$$

Where $\boldsymbol{\omega}_{\boldsymbol{i}_v/w}$ is the angular velocity between the $\boldsymbol{i}_v$-unit vector of the vehicle reference frame and the world reference frame and belong to the same class of the vehicle rotational velocity so it can be concluded that:

$$\boldsymbol{\omega}_{\boldsymbol{i}_v/w} = \boldsymbol{\omega}_{v/w}$$

Concerning $\boldsymbol{\omega}_{\boldsymbol{n}_d/w}$ it is known that the tip of the vector $\boldsymbol{d1}$ moves with a velocity which satisfies:

$$\boldsymbol{v}_{d1/w} = \boldsymbol{\omega}_{d/w} \times \boldsymbol{d1}$$

where $\boldsymbol{v}_{d1/w}$ can be evaluated as the derivative of $\boldsymbol{d1}$ as:

$$D_w\boldsymbol{d1} = D_w[\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)](\boldsymbol{N} - \boldsymbol{O}_v) + [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]D_w(\boldsymbol{N} - \boldsymbol{O}_v)$$

Where $D_w[\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]$ is equal to zero since all its terms are fixed in the world reference frame, obtaining:

$$D_w\boldsymbol{d1} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]D_w(\boldsymbol{N} - \boldsymbol{O}_v) = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)](\boldsymbol{v}_{\frac{n}{w}} - \boldsymbol{v}_{\frac{v}{w}})$$

In this study case, the nodule is fixed in space, thus the equation simplifies as:

$$D_w\boldsymbol{d1} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\left(-\boldsymbol{v}_{\frac{v}{w}}\right) = \boldsymbol{v}_{d1/w}$$

Where $\boldsymbol{v}_{\frac{v}{w}}$ is part of the control variables of the vector $\dot{\boldsymbol{y}}$.

The inverse cross product problem can be solved by finding $\boldsymbol{\omega}_{d/w}$ which is a vector orthogonal to both $\boldsymbol{v}_{d1/w}$ and $\boldsymbol{d1}$ whose length is equal to $\|\boldsymbol{v}_{d1/w}\|/\|\boldsymbol{d1}\|$ as:

$$\boldsymbol{\omega}_{d/w} = \left(\frac{\boldsymbol{d1}}{\|\boldsymbol{d1}\|} \times \frac{\boldsymbol{v}_{d1/w}}{\|\boldsymbol{v}_{d1/w}\|}\right)\frac{\|\boldsymbol{v}_{d1/w}\|}{\|\boldsymbol{d1}\|} = (\boldsymbol{d1} \times \boldsymbol{v}_{\frac{d1}{w}})\frac{1}{\|\boldsymbol{d1}\|^2}$$

$$\boldsymbol{\omega}_{d/w} = (\boldsymbol{d1} \times [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\left(-\boldsymbol{v}_{\frac{v}{w}}\right))\frac{1}{\|\boldsymbol{d1}\|^2}$$

$$\boldsymbol{\omega}_{d/w} = -\frac{1}{\|\boldsymbol{d1}\|^2}(\boldsymbol{d1} \times [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\left(\boldsymbol{v}_{\frac{v}{w}}\right))$$

Inserted in the equation of $\boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v}$ gives:

$$\boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v} = -\frac{1}{\|\boldsymbol{d1}\|^2}(\boldsymbol{d1} \times [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\left(\boldsymbol{v}_{\frac{v}{w}}\right)) - \boldsymbol{\omega}_{\boldsymbol{i}_v/w}$$

and by recalling that:

$$\theta N(\theta) = -\frac{\theta}{sin\theta}\boldsymbol{n}_d \times \{\boldsymbol{i}_v \times [\boldsymbol{I} - \boldsymbol{n}(\boldsymbol{n} \cdot))]\}\boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v}$$

The final relation between the control's variables and the derivative of $\boldsymbol{\rho}$ can be obtained as:

$$D_v\boldsymbol{\rho} = \left(\boldsymbol{n}\boldsymbol{n}^T - \frac{\theta}{sin\theta}[\boldsymbol{n}_d \times][\boldsymbol{i}_v \times][\boldsymbol{I}_{3\times3} - \boldsymbol{n}\boldsymbol{n}^T]\right)\left[\boldsymbol{0}_{3\times7} \ -\frac{1}{\|\boldsymbol{d1}\|^2}[\boldsymbol{d1} \times][\boldsymbol{I}_{3\times3} - \boldsymbol{k}_w\boldsymbol{k}_w^T] \ -\boldsymbol{I}_{3\times3}\right]\dot{\boldsymbol{y}}$$

Therefore, the final task Jacobian can be obtained as:

$$\boldsymbol{J}_n = \left(\boldsymbol{n}\boldsymbol{n}^T - \frac{\theta}{sin\theta}[\boldsymbol{n}_d \times][\boldsymbol{i}_v \times][\boldsymbol{I}_{3\times3} - \boldsymbol{n}\boldsymbol{n}^T]\right)\left[\boldsymbol{0}_{3\times7} \ -\frac{1}{\|\boldsymbol{d1}\|^2}[\boldsymbol{d1} \times][\boldsymbol{I}_{3\times3} - \boldsymbol{k}_w\boldsymbol{k}_w^T] \ -\boldsymbol{I}_{3\times3}\right]$$

To solve the problem, it will need to create a task which is able to bring $\boldsymbol{\rho}$ to zero, among many possible choices, one requires the derivative of $\boldsymbol{\rho}$ to be proportional to the misalignment vector itself:

$$\dot{\bar{\boldsymbol{\rho}}} = \lambda(\bar{\boldsymbol{\rho}} - \boldsymbol{\rho}) = -\lambda\boldsymbol{\rho}$$

Where $\bar{\boldsymbol{\rho}}$ is the desired misalignment and it's equal to zero, if the true derivative $D\boldsymbol{\rho}$ follows the desired value, this would guarantee an asymptotic convergence of $\boldsymbol{\rho} \rightarrow \boldsymbol{0}$.

4. Alerting gain & observation

In this section it is discussed how the gain of the task *Vehicle alignment to the nodule* takes effect on the robot's behavior. For this case three different values of gain were chosen to observe the robot's behavior difference. It was expected that having a lower gain makes the vehicle align to nodule in longer period and vice versa. However, there three gains value that were chosen are 0.01, 0.2 and 0.8. They were chosen like that as the first value is relatively very low and with this gain it's expected that the vehicle should not finish its alignment even after touch the seabed. While the second value is usually the same used for most tasks of the vehicle tasks. Lastly, the gain of 0.8 is relatively high value and it is expected that the vehicle will align with nodule even before the landing occurs.

Furthermore, the different values were tested, and the following graphs [Fig.7] was the result. And it can be seen here that the hypothesis of the experiment that was made can be considered as achieved. As for the first gain the vehicle was never able to finish the action *Landing* on the time of the simulation. While for the gain the vehicle was able to finish the action *Landing* and move to the action *Inspection* around t = 35 seconds. Finally for the gain of 0.8 the vehicle finished the action *Landing* around t = 25 seconds.

In conclusion, it can be reported that altering the gain has a directly proportional effect on the time needed for achieving the task and therefore the action. Keep in mind that this test is done on the simulation which has a max time of 70 seconds, this is relatively far from the real-life implementation and the gains must be tuned according to the vehicle specifications.

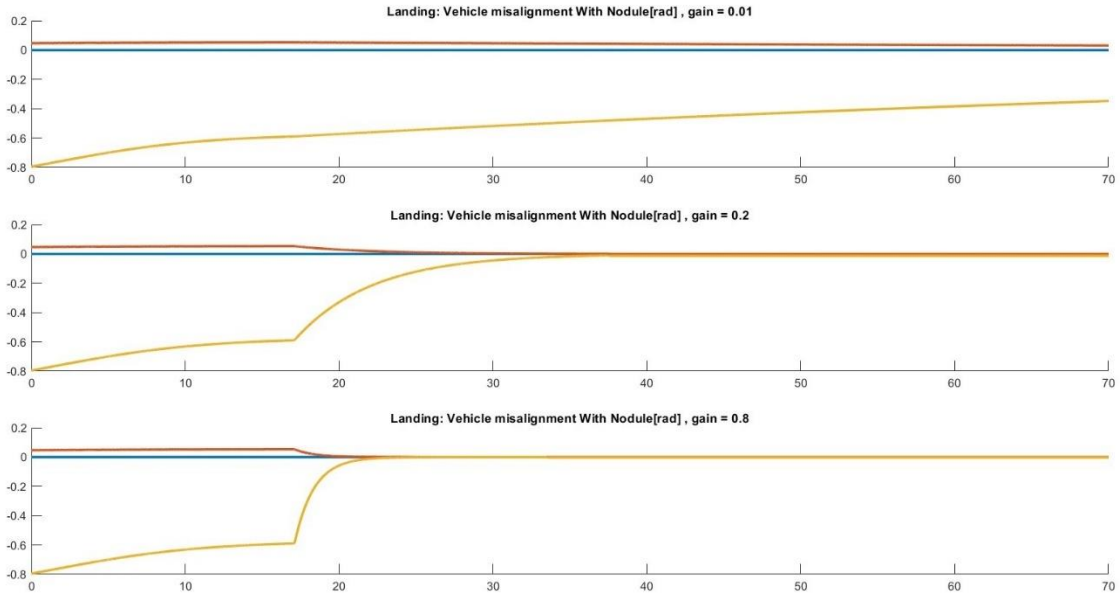


Figure 7: Different gains for Vehicle alignment to nodule task

On the other hand, this implementation can not be granted 100% to finish the action of *Landing* on time. A proper solution for this problem is to divide the action of *Landing* into two separate actions in which the vehicle first aligns to the nodule as one action and once done it moves to the second action which is *Landing*. A modification of the proposed hierarchy in section I.1 can be seen in the following table.

Table 2: Robust Nodule Inspection Mission modified Task Priority Control hierarchy

Task	Type	Navigate to goal	Alignment	Landing	Inspection
Arm Joints Limits	I				1
Arm Manipulability	I				2
Vehicle Null Velocity	E				3
End Effector Linear Position	E				4
End Effector Angular Position	E				5
Vehicle minimum altitude	I	1			
Vehicle horizontal attitude	I	2	1	1	
Vehicle heading control	I	3			
Vehicle position control	E	4			
Vehicle alignment to nodule	I		2	2	
Vehicle distance to nodule	I			3	
Vehicle altitude control	E			4	

Having such a hierarchy ensures that the vehicle will never land without aligning to the nodules prior to the *Landing* action. This is also crucial in real life situation as it was observed in the test of gain 0.01 that the vehicle laned touching the nodule which can cause leakage in an AUV (Autonomous Underwater Vehicle) if there were no obstacle avoidance task with higher priority. Please find the proposed hierarchy solution as version 2 of the MATLAB scripts.

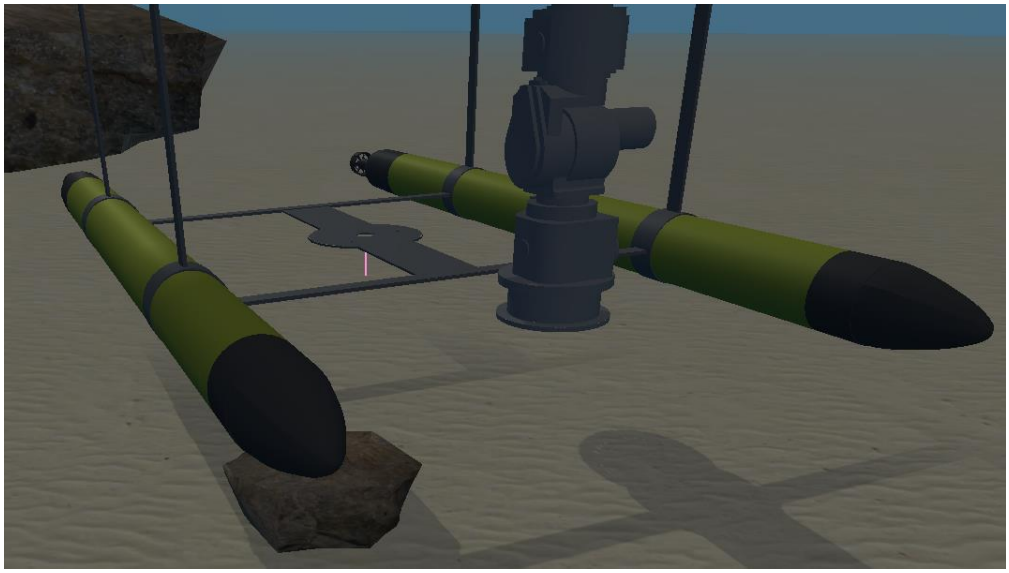


Figure 8: Vehicle landing while low gain of alignment to nodule task, hitting the rock.

5. Landing and Inspection actions

After the vehicle has landed and the *Inspection* action takes over, the magnitude of the distance between the Arm and the nodules was less than 1 meter as seen before from figure 4 which is enough to do the *Inspection* action without having the task of the *Arm Joint Limits* stopping the action defining task from being archived.

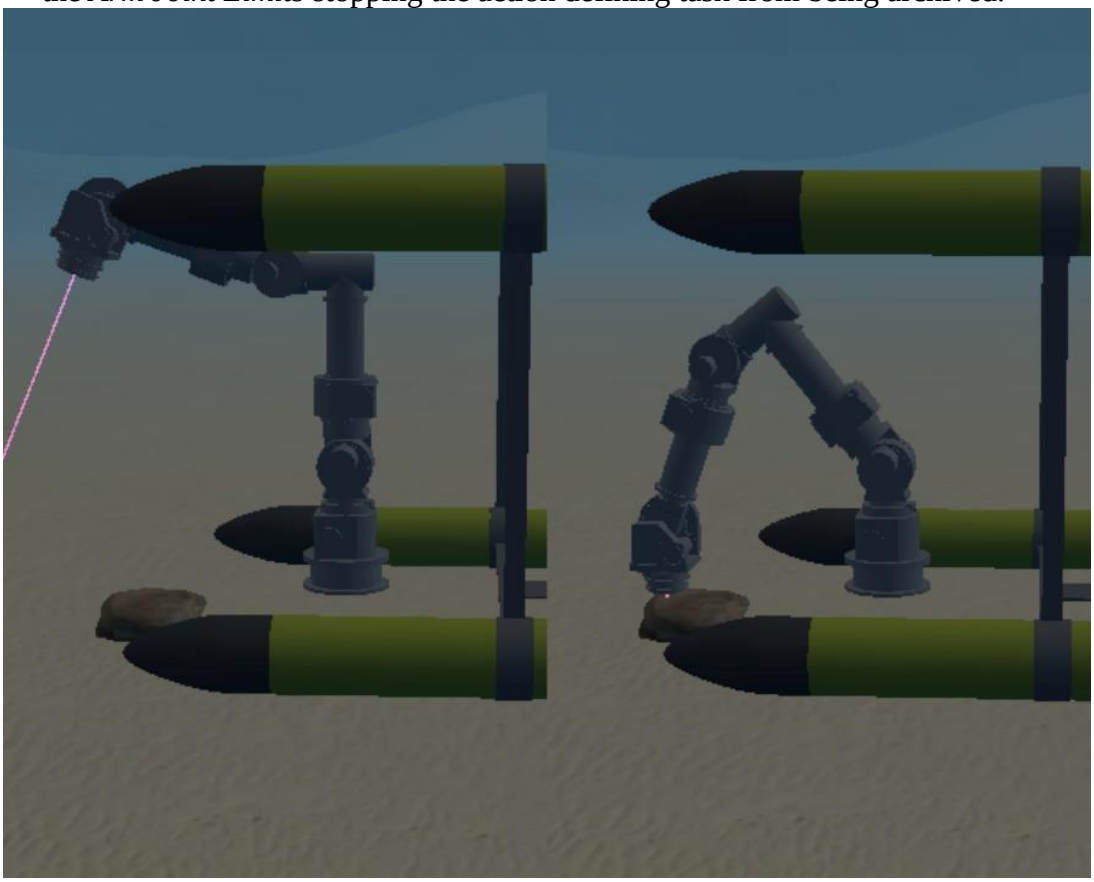


Figure 9: Vehicle between Landing and Inspection actions

6. Proof that nodule is in manipulator's workspace

To guarantee that after the landing is finished the nodule is in the manipulator's workspace. The task Jacobian of the task *Vehicle distance to nodule* should be computed with respect to the arm base frame and not the vehicle frame. As for this vehicle the distance between the base of the arm is not relatively far from the vehicle frame. However, the following solution was proposed to ensure a generic architecture.

The objective requires that the distance between the manipulator base $\boldsymbol{B}$ and the nodule $\boldsymbol{N}$ projected on the horizontal plane goes under a desired threshold t in such a way that the nodule will be in the workspace of the arm. In other word the following equality objective must be satisfied:

$$d_h < t$$

Where t is the desired scalar position error between the manipulator base and the nodule position projected on the horizontal plane.

As seen this task is structured as an inequality task and acquire the attribute of operational prerequisite task.

$$\boldsymbol{x} < \overline{\boldsymbol{x}} \qquad \overline{\boldsymbol{x}} = \text{desired distance}$$

So, the activation function will be an increasing bell-shaped function, and since control takes place over three scalars (the component of the error along x, y, z), of dimension 3x3:

$$\boldsymbol{A = IncreasingBellShapedFunction}(\boldsymbol{t - \Delta, t, 0, 1})_{3\times 3}$$

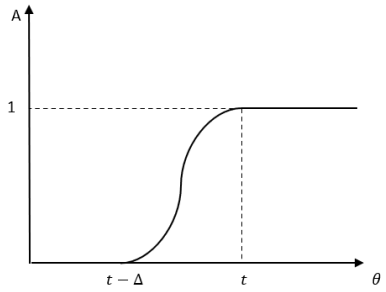


Figure 10: Increasing bell shape activation function.

Moving to the evaluation of the task Jacobian, the variable that must be controlled is the projection of the distance vector between the arm base and the nodule, which is equal to:

$$\boldsymbol{d} = (\boldsymbol{N} - \boldsymbol{B})$$

In which the distance in terms of also the vehicle reference frame origin can be expressed as:

$$\boldsymbol{d} = (\boldsymbol{N} - (\boldsymbol{O}_v + \boldsymbol{r}_{O-B}))$$

Where $\boldsymbol{r}_{O-B}$ is a constant vector expressing the position of the arm base $\boldsymbol{B}$ with respect to the origin of the vehicle reference frame $\boldsymbol{O}_v$.

Now the evaluation of the projection of such a vector on the horizontal plane, obtaining the vector $\boldsymbol{d1}$ is:

$$\boldsymbol{d1} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)](\boldsymbol{N} - (\boldsymbol{O}_v + \boldsymbol{r}_{O-B}))$$

To control that variable, it is needed to compute its derivative and express it in terms of the control variables embedded in the vector $\dot{\boldsymbol{y}}$.

$$D_w \boldsymbol{d1} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)] D_w(\boldsymbol{N} - \boldsymbol{O}_v - \boldsymbol{r}_{O-B}) = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)](\boldsymbol{v}_{\frac{n}{w}} - \boldsymbol{v}_{\frac{v}{w}} - \boldsymbol{v}_{\frac{O}{B}})$$

Where both $\boldsymbol{v}_{\frac{n}{w}}$ and $\boldsymbol{v}_{\frac{O}{B}}$ are equal to zero since the nodule is fixed in space during time as the distance between the origin of the vehicle reference frame $\boldsymbol{O}_v$ and the arm base $\boldsymbol{B}$, obtaining:

$$D_w \boldsymbol{d1} = -[\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\dot{\boldsymbol{y}}$$

Obtaining the task Jacobian:

$$\boldsymbol{J}_b = [\boldsymbol{0}_{3\times 7} \quad -[\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)] \quad \boldsymbol{0}_{3\times 7}]$$

Then the task reference, which is the controlling law that drive the control variable, can be obtained by modifying a bit the above relation as:

$$d_h < t$$

$$\|\boldsymbol{d1}\| < t$$

$$\boldsymbol{d1} < \frac{\boldsymbol{d1}}{\|\boldsymbol{d1}\|} t$$

$$\bar{\boldsymbol{t}} = \frac{\boldsymbol{d1}}{\|\boldsymbol{d1}\|} t$$

$$\dot{\bar{\boldsymbol{x}}} = \lambda(\bar{\boldsymbol{t}} - \boldsymbol{d1}) \qquad \lambda > 0$$

In which t is the desired distance between the base and the nodule as described before.

II. Franka Panda Bimanual Manipulation Mission

1. Tasks unified hierarchy

As shown in the table below the Task Priority Control hierarchy is composed of four actions but before proceeding with the explanation of the control hierarchy it must be mentioned the calculations that were made for computing the transformation between the world and the base of each robot knowing that the left arm base coincides with the world frame. While the right arm base is rotated around the z-axis of the world frame with 180 degrees and is translated along it's x-axis with 1.05 meters.

Moving on to the control hierarchy, regarding the *Reaching goal* action which controls the two arms to move to the common target to be grasped, this action is composed of two tasks, which are the *Joints limits* "having the highest priority" and *reaching goal 1*.

Secondly, the *Grasping* action starts once the robots reaches the first goal, this action consists of the *kinematic Constraint* imposed by the object. The *Joints limits* task and *Reaching goal 2* task which moves the object to the desired new goal knowing that the tasks are written with respect to the priority level.

Finally, the robotics manipulators starts the third action called *Reaching goal 2* guiding the manipulators to a goal to invoke the joint limits of the left to activate. Note that the last introduced action called *Finish* has no effect on the designed mission and its just there to make sure the robot performances safely in real life.

Table 3: Franka Panda Bimanual Manipulation Mission Task Priority Control hierarchy

Task	Type	Reaching goal 1	Grasping	Reaching goal 2	Finish
Kinematic Constrain	E		1	1	1
Joint limits	I	1	2	2	1
Finish Task	E				1
Reaching goal 1	E	2			
Reaching goal 2	E		3		
Reaching goal 3	E			3	

2. Joint Limits task Jacobian and task reference

This objective requires that the joint configuration remain bounded by an upper and lower limit:

$$q_i > q_i^{min} \qquad q_i < q_i^{max} \qquad i = 1, 2, 3, .., n$$

Where q_i^{min} and q_i^{max} are respectively the lower and upper bound of the i-joint configuration and n is the number of joints of the manipulator.

As seen this task is designed as an inequality task and acquire the attribute of safety task. Normally this task can be subdivided in two tasks, one for the upper limit and one for the lower limit and then merge them together to obtain the same priority, with this goal in mind, noticing also the structure of the inequality the activation functions will be a diagonal matrix of dimension equal to the number of joints in which for the lower joint limit each element will be a decreasing bell shaped function. While for the upper joint limit an increasing bell-shaped function, both related to the actual value of the configuration and the relative limit:

$$\boldsymbol{A_{min}} = diag\left(DecreasingBellShapedFunction\left(q_i^{min}, q_i^{min} + \Delta, 0, 1\right)\right)$$
$$\boldsymbol{A_{max}} = diag\left(IncreasingBellShapedFunction\left(q_i^{max} - \Delta, q_i^{max}, 0, 1\right)\right)$$

We can move to the evaluation of the task Jacobian, since it is needed to directly control the joint configuration q_i , the control can take place over their derivatives that can be easily obtained from the control variables vector $\dot{\boldsymbol{y}}$ as:

$$\boldsymbol{J_{jl}^{min}} = \boldsymbol{J_{jl}^{max}} = [\boldsymbol{I}_{7\times7}\ \boldsymbol{0}_{7\times7}\ \boldsymbol{0}_{7\times7}]$$

Then the task reference, which is the controlling law that drive the control variable, can simply be expressed as:

$$\dot{\boldsymbol{x}}_{\boldsymbol{min}} = \lambda\left((\boldsymbol{q}^{min} + \Delta) - \boldsymbol{q}\right) \qquad \lambda > 0$$
$$\dot{\boldsymbol{x}}_{\boldsymbol{max}} = \lambda((\boldsymbol{q}^{max} - \Delta) - \boldsymbol{q}) \qquad \lambda > 0\backslash$$

- Implementation remark:
  The above relationship can be exploited in a smarter solution stacking the task references as:

```
% reference for the joint limits task:
uvms.xdot.jl = zeros(7, 1);
for i = 1:size(uvms.jlmin)
    mean(i) = (uvms.jlmin(i) + uvms.jlmax(i))/2;
    if uvms.q(i) <= mean(i)
        uvms.xdot.jl(i) = 0.2 * (uvms.jlmin(i) - uvms.q(i) +0.01);
    else
        uvms.xdot.jl(i) = 0.2 * (uvms.jlmax(i) - uvms.q(i) -0.01);
    end
end
```

Where uvms.xdot.jl is the global joint limit task reference, uvms.jlmin is the vector of the lower joint limits , mean is the vector of the mean value of the configuration and uvms.q is the configuration vector of the manipulator.

Moreover, the proposed logic is to add in each row of the task reference vector the task reference of the lower limit or of the upper limit based on the actual joint configurations, it's nearer to the lower limit then its task reference will be inserted in the vector, otherwise the upper limit one will be inserted.

And by adding together the activation functions as:

```
ujl = [2.9;1.65;2.9;0.01;2.9;1.25;2.8];
ljl = [-2.9;-1.6;-2.9;-2.95;-2.9;-1.65;-2.8];
for i = 1:size(ujl, 1)
    ArrayActivationFunction(i, i) =
IncreasingBellShapedFunction(uvms.jlmax(i) - 0.01, uvms.jlmax(i), 0,
1, uvms.q(i)) + DecreasingBellShapedFunction(uvms.jlmin(i),
uvms.jlmin(i) + 0.01, 0, 1, uvms.q(i));
end
uvms.A.jl = ArrayActivationFunction .* ActionTransition("JL",
previous_tasks, current_tasks, mission.phase_time);
```

3. Kinematic Constrain task Jacobian and task reference

This objective takes into consideration the constraint imposed by the manipulation of an object by two different manipulators mounted on the same base, avoiding the introduction of stresses on the object. To do that it is needed firstly to compute for each arm the basic Jacobian for each arm respect a common reference frame, in this case the object one <o>:

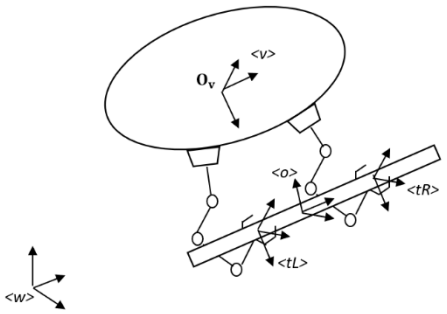


Figure 11: Bimanual manipulator and tool reference frames

$$\boldsymbol{J}_o = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ [\boldsymbol{r}_{t/o}\times] & \boldsymbol{I}_{3\times3} \end{bmatrix} \boldsymbol{J}_t$$

Where:

$\boldsymbol{J}_o$ is the Jacobian respect the object reference frame.

$[\boldsymbol{r}_{t/o}\times]$ is the distance vector between the tool frame and the object frame, projected in the world reference frame.

$\boldsymbol{J}_t$ is the basic Jacobian of the arm with respect to the tool frame expressed in the world reference frame.

The two Jacobians are then merged as:

$$\boldsymbol{J}_o^{kc} = [\boldsymbol{J}_o^L \quad \boldsymbol{J}_o^R]$$

Which is the task Jacobian for that task. Then the task reference requires that all the velocities obtained with this Jacobian are equal to zero for each arm:

$$0 = \boldsymbol{J}_o^{kc}\dot{\boldsymbol{y}}$$

Finally obtaining:

$$\dot{\boldsymbol{x}} = \boldsymbol{0}$$

This task should be always active while the object is grasped, for this reason the activation function will be an identity matrix of size 6 by 6:

$$\boldsymbol{A} = \boldsymbol{I}_{6\times6}$$

For its importance the action will acquire the attribute of constraint and safety task.

4. Robot's behavior

Moving forward from the designed control hierarchy, this section reports the robot behavior to observer whether it follows the designed architecture or not. Firstly, it is important to compare the object desired velocity and the two end-effectors actual velocity to be able to say that the cooperation part is successful and that there is no forces or torques introduced on the object while moving it.

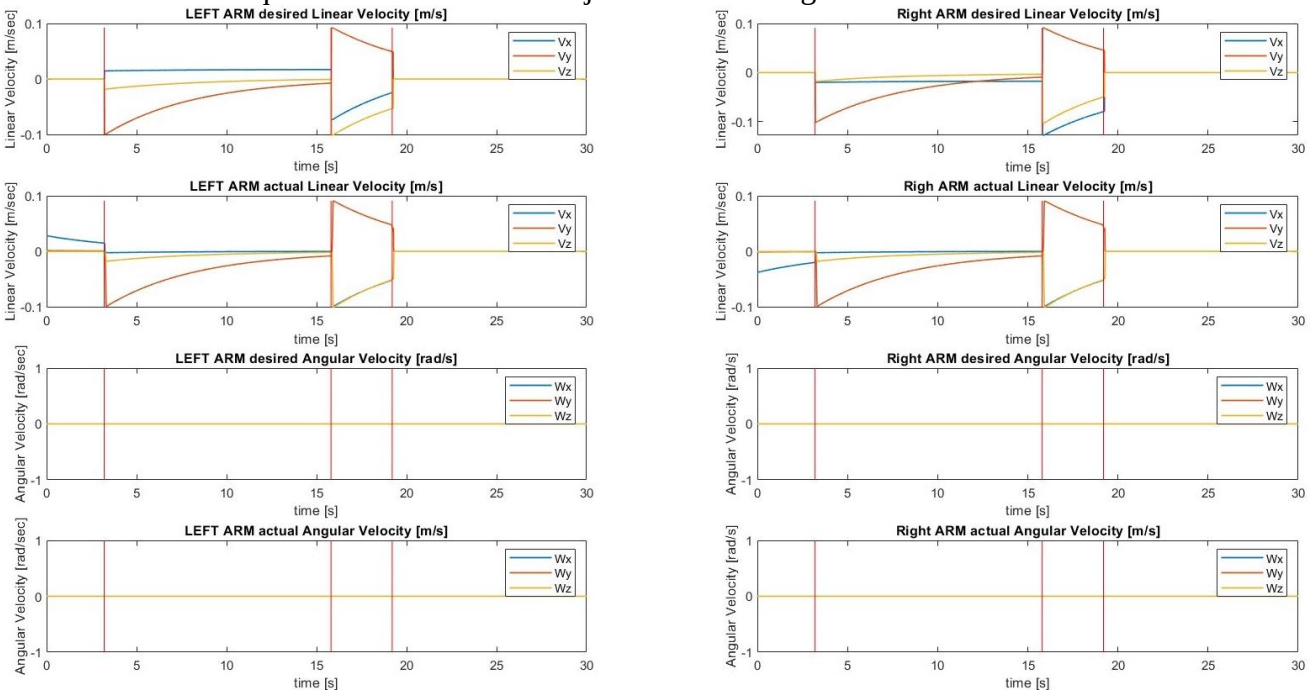


Figure 12: Left and Right arms desired and actual linear, angular velocities

As shown from Figure 12 and expect from the angular velocities that is not controlled. It can be said for both arms the desired Liner Velocities are slightly different then the actual ones, this is due to the *Kinematic Constrain* which prevents the introduction of forces/ torques on the tool. However, the velocities are converging to zero as when the manipulators reach the goals in each action it stops and then the transition of the following action takes place and moves the manipulators again.

Having a closer look at the first action which is *Reaching goal 1*, it can be seen from the following figure [Fig.13] that the magnitude of the distance between the Arm and the goal converges to 0.1 meters and same behavior goes to the misalignment with the goal. Furthermore, as soon as the manipulators have finished the Reaching goal 1 action the following action takes place guiding the manipulators away from the first goal.

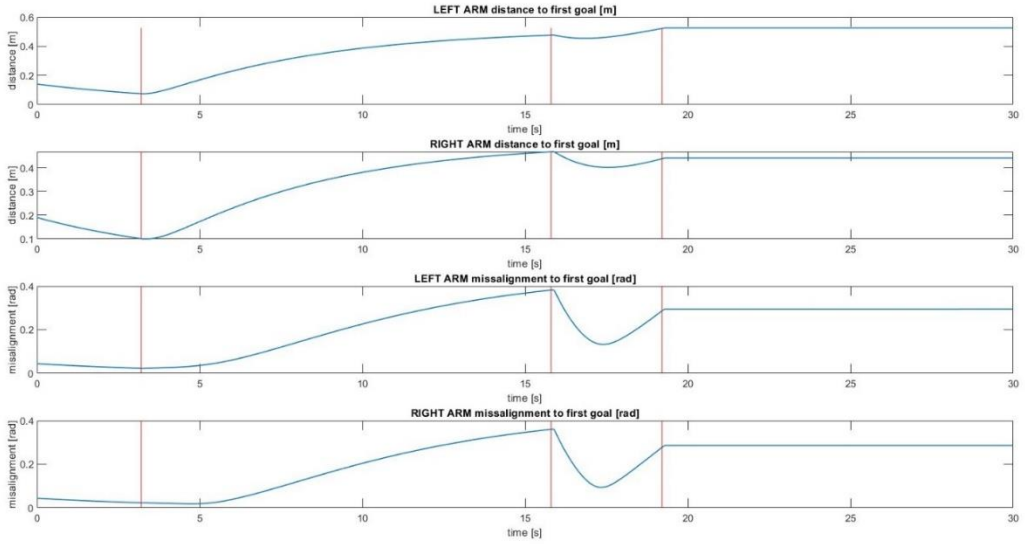


Figure 13: Reaching goal 1 action

A more interesting action to give care to is the action called *Reaching goal 2*, where the main goal here is to guide the manipulators to a position in which one of the manipulators activates is *Joint limits* task. In this case the position $[0 \quad 0 \quad 0]$ was chosen as it is the base frame of the left arm with respect to the world frame. So theoretically, the left arm end effector should never be able to reach that goal due to the *Joint limits* task.

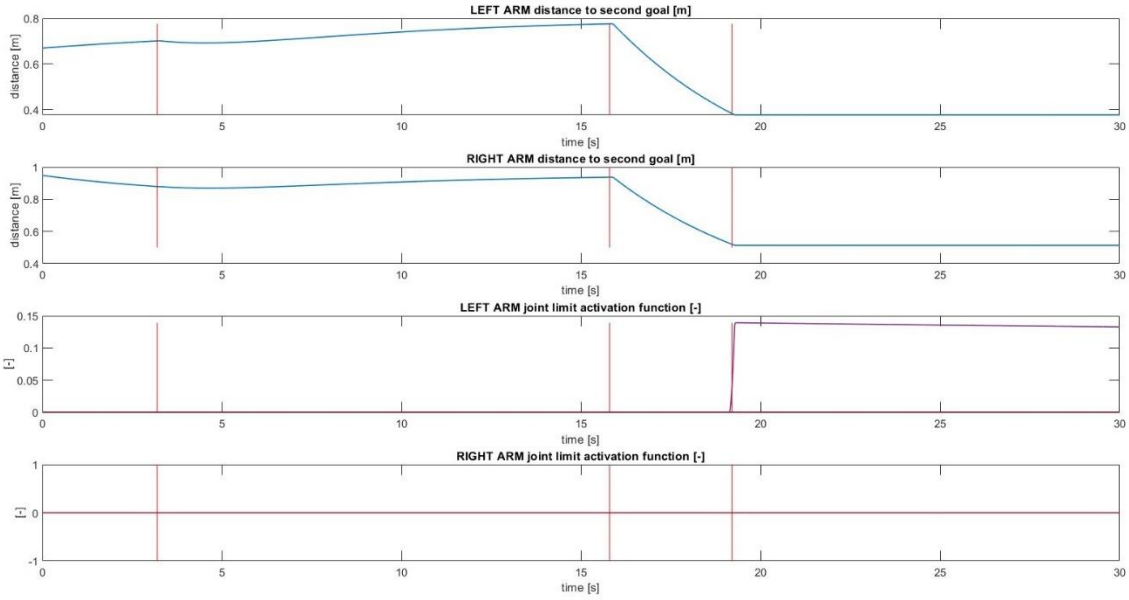


Figure 14: Reaching goal 2 action

And as expected and as seen from Figure 14 that in the action *Reaching goal 2* the distance to the goal starts converging to zero. However, before the manipulators can reach its goal the *Joint Limits* task goes active and prevents the manipulators from reaching that goal and finally the transition between that action and the action *Finish* takes place and the manipulators stops its motion. And with that it can concluded the end of the mission as success. Please note the end of the action *Reaching goal 2* is decided when the task Joint

Limits goes on. However, this action transition does not make much sense but it's designed like that just for testing purposes. So, the manipulator never reaches the second goal.

III. Franka Panda Cooperative Manipulation Mission

1. Tasks unified hierarchy

Firstly, in Table 4 the *Reaching goal* action (non-cooperative action) consists of joints limits tasks for both arms separately having the highest priority, then tasks for reaching goals for both the right and left arms independently, which also are needed for calculations to operate the tasks cooperatively like the non-cooperative tool frame velocities $\dot{x}$ to calculate the assignments of weights μ to get the feasible cooperative velocity vector $\hat{\dot{x}}$.

Table 4: Franka Panda Non-Cooperative Manipulation Mission Task Priority Control hierarchy

Task	Type	Reaching goal	Grasping	Grasping 2	Finish
Joints Limits Arm Left	I	1	1	1	1
Joints Limits Arm Right	I	2	2	2	2
Reaching Goal Left Arm	E	3			
Reaching Goal Right Arm	E	4			
Reaching Goal_1 Left Arm	E		3		
Reaching Goal_1 Right Arm			4		
Reaching Goal_2 Left Arm				3	
Reaching Goal_2 Right Arm				4	
Finish	E				3

Secondly, in Table 5 the *Grasping* action (cooperative action) is responsible for grasping the object then move it to a desired position. The priorities of tasks in this action are changed as the reaching new desired goal tasks have more priority than the joints limits tasks.

Table 5: Franka Panda Cooperative Manipulation Mission Task Priority Control hierarchy

Task	Type	Reaching goal	Grasping	Grasping 2	Finish
Reaching Goal_1 Left Arm	E		1		
Reaching Goal_1 Right Arm	E		2		
Reaching Goal_2 Left Arm	E			1	
Reaching Goal_2 Right Arm	E			2	
Joints Limits Arm Left	I	1	3	3	1
Joints Limits Arm Right	I	2	4	4	2
Reaching Goal Left Arm	E	3			
Reaching Goal Right Arm	E	4			
Finish	E				3

Finally, the *Finish* action contains one task responsible for stopping the whole movement of the end effectors for both arms.

2. Discussion of the two hierarchies

There are two hierarchies for this mission:

- Non-cooperative tasks hierarchy is used in the reaching goal action, in which both arms are controlled independently. The aim of this action is: to move the arms to reach the point where the object is placed, to grasp with respect to the joint limits for each arm (since it is the highest priority task) and to move the object. All this is needed for the next hierarchy, to compute the assignments of weights μ to get feasible cooperative tool frame velocity vector $\dot{\hat{x}}$.

- Cooperative tasks hierarchy is used in grasping the object and move it to another desired position by controlling the two arms cooperatively. This hierarchy is reached after the calculation of velocities $\dot{\hat{x}}$ for the left and right arms, knowing that the reaching of the new desired point task has the highest priority than the joint limits tasks priority.

3. Robot's Behavior

As shown from the following figure for the left and right non-cooperative angular velocities they are trying to reach a different velocity in the y component. Since this is not possible as the object is grasped between the two manipulators the architecture tries to find a mean between the two manipulators desired velocity and the output can be seen in the cooperative output of both manipulators' angular velocity plots. The same concept is implemented for Linear velocities of both manipulators, it's always the non-cooperative "the desired reference velocity" is changed to the cooperative values "actual velocities".

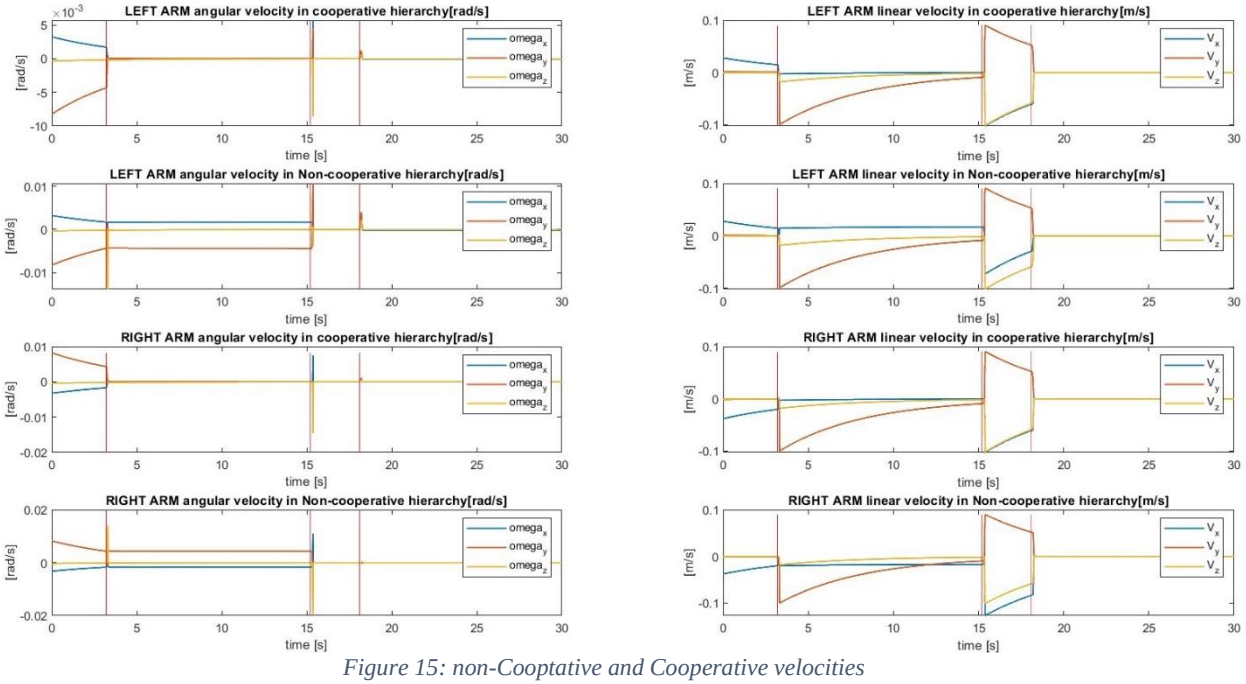


Figure 15: non-Cooptative and Cooperative velocities

Other than that, the robot behavior can be studied as the same of section II. However, an interesting action to study is *Grasping 2*.

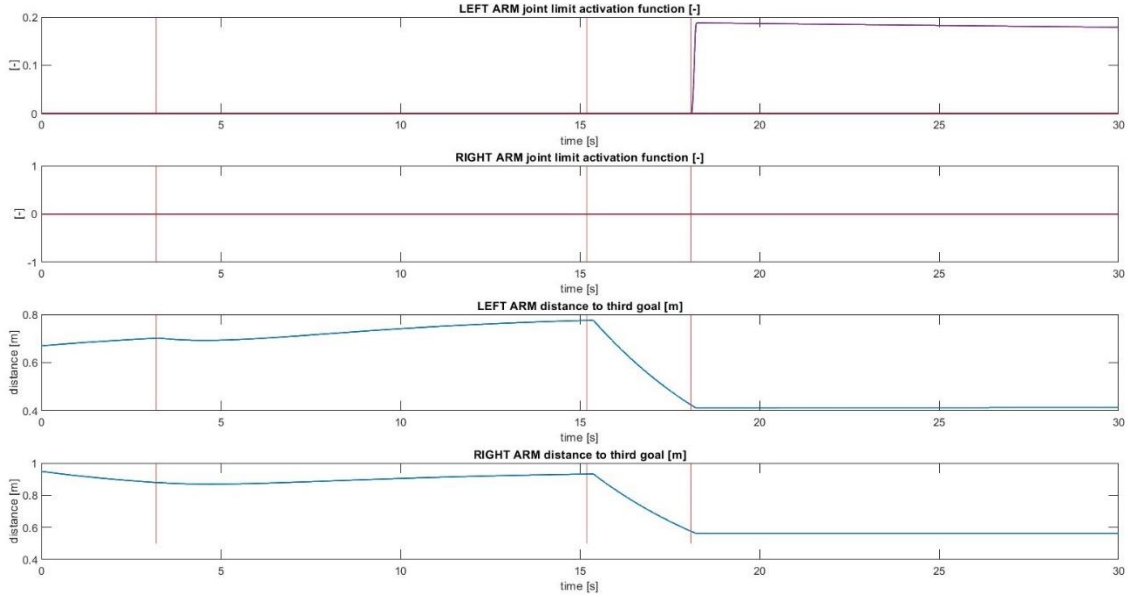
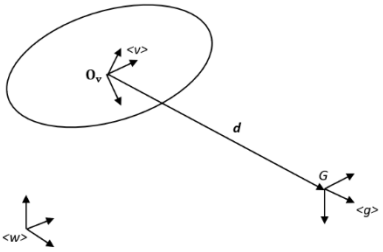


Figure 16: Grasping 2 action

As discussed before in section II it can be seen here how the *Joint limits* tasks for both manipulators prevents them from reaching there goal. The only difference here that the *Joint limits* are associated for each manipulator while in section II arm had the same task.

Appendix

1. Task: Reaching a Goal position:



This objective requires that the vehicle frame origin O_v converges to a given goal frame <g> origin *G.*
In other word the following equality objective must be satisfied:

$$r_i = \mathbf{0} \quad i = 1,2,3$$

Where **r** is the position error between the vehicle and goal frame.
As we can see this task is structured as an equality task and acquire the attribute of action defining task.

$$\boldsymbol{x} = \bar{\boldsymbol{x}} \qquad \bar{\boldsymbol{x}} = \text{desired goal position}$$

So the activation function will be an identity matrix, and since we are controlling three scalar (the component of the error along x, y, z), of dimension 3x3:

$$\boldsymbol{A} = \boldsymbol{I}_{3\times3}$$

We can move to the evaluation of the task Jacobian, the variable we want to control is the distance vector between the vehicle and goal reference frame origin:

$$\boldsymbol{d} = (\boldsymbol{G} - \boldsymbol{O}_v)$$

In order to control its behavior we need to compute its time derivative:

$$D\boldsymbol{d} = D(\boldsymbol{G} - \boldsymbol{O}_v) = D\boldsymbol{G} - D\boldsymbol{O}_v = \boldsymbol{v}_g - \boldsymbol{v}_v$$

In our case we're considering the goal as fixed in space, obtaining a derivative equal to:

$$D\boldsymbol{d} = -\boldsymbol{v}_v$$

Which is already expressed respect one of our control variables, allowing us to define directly the task Jacobian as:

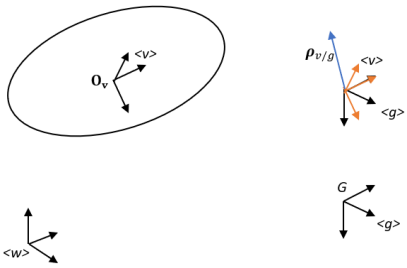
$$D\boldsymbol{d} = [\mathbf{0}_{3\times7} \; -\boldsymbol{I}_{3\times3} \; \mathbf{0}_{3\times3}]\,\dot{\boldsymbol{y}}$$
$$\boldsymbol{J}_d = [\mathbf{0}_{3\times7} \; -\boldsymbol{I}_{3\times3} \; \mathbf{0}_{3\times3}]$$

Then the task reference, which is the controlling law that drive the control variable, can simply be expressed as:

$$\dot{\bar{\boldsymbol{x}}} = \lambda(\bar{\boldsymbol{d}} - \boldsymbol{d}) = -\lambda\boldsymbol{d} \qquad \lambda > 0$$

Where since our desired distance to the goal $\bar{d}$ is equal to zero the formula simply become the one above.

2. Task: Align to a Goal reference frame:



This objective requires that the vehicle frame <v> converges to a given goal frame <g>.
In other word the following equality objective must be satisfied:

$$\theta_i = 0 \qquad i = 1,2,3$$

Where $\boldsymbol{\theta}$ is the misalignment error between the vehicle and goal frames.
As we can see this task is structured as an equality task and acquire the attribute of action defining since it defines the action itself.

$$\boldsymbol{x} = \overline{\boldsymbol{x}} \qquad \overline{\boldsymbol{x}} \text{ = desired goal orientation}$$

So the activation function will be an identity matrix, and since we are controlling three scalar (the component of the error along x, y, z), of dimension 3x3:

$$\boldsymbol{A} = \boldsymbol{I}_{3\times3}$$

We can move to the evaluation of the task Jacobian, the variable we want to control is the misalignment vector between the vehicle and goal reference frame origin $\boldsymbol{\rho}_{v/g}$ that can be easily obtained from the unit vector lemma.

$$\boldsymbol{\rho}_{v/g} = \theta \boldsymbol{n}$$

In order to control its behavior we need to compute its time derivative:

$$D\boldsymbol{\rho} = \boldsymbol{n}\dot{\theta} + \theta D\boldsymbol{n} = [\boldsymbol{n}(\boldsymbol{n}\,\cdot) + \boldsymbol{N}(\theta)]\boldsymbol{\omega}_{v/g}$$

In which since we assume the goal reference frame <*g*> fixed we can simplify the equation as:

$$D_w\boldsymbol{\rho} = \boldsymbol{n}(\boldsymbol{n}\,\cdot)\boldsymbol{\omega}_{v/w} = \boldsymbol{\omega}_{v/w}$$

Which is already expressed respect one of our control variables, allowing us to define directly the task Jacobian as:

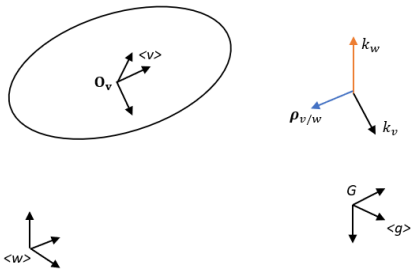
$$D\boldsymbol{\rho} = [\boldsymbol{0}_{3\times7}\ \boldsymbol{0}_{3\times3}\ \boldsymbol{I}_{3\times3}]\,\dot{\boldsymbol{y}}$$
$$\boldsymbol{J}_{\boldsymbol{\rho}} = [\boldsymbol{0}_{3\times7}\ \boldsymbol{0}_{3\times3}\ \boldsymbol{I}_{3\times3}]$$

Then the task reference, which is the controlling law that drive the control variable, can simply be expressed as:

$$\dot{\bar{\boldsymbol{x}}} = \lambda(\overline{\boldsymbol{\rho}} - \boldsymbol{\rho}) = -\lambda\boldsymbol{\rho} \qquad \lambda > 0$$

Where since our desired misalignment to the goal $\overline{\boldsymbol{\rho}}$ is equal to zero the formula simply become the one above.

3. Task: Horizontal attitude control



This objective requires that the vehicle orientation of one of the vehicle reference frame axis, in our case k_v, align to a given direction, expressed by another unit vector, in our case k_w or better the misalignment between the two must stay under a certain threshold.

In other word the following inequality objective must be satisfied:

$$\theta_i < t \qquad i = 1,2,3 \qquad t_i = threshold\ i \qquad i = 1,2,3$$

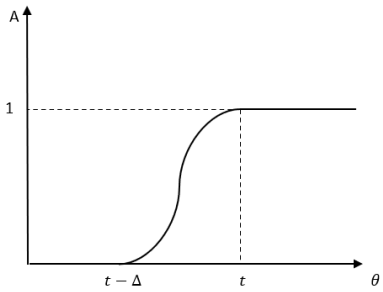
Where $\boldsymbol{\theta}$ is the misalignment error between the vehicle k_v unit vector and the world k_w unit vector.

As we can see this task is structured as an inequality task and acquire the attribute of operational prerequisite task.

$$\boldsymbol{x} < \boldsymbol{t} \qquad \boldsymbol{t} \text{ = misalignment threshold}$$

So since the misalignment must remain under a certain threshold the activation function will be an increasing bell shaped function that will reach the pick when the misalignment reach the threshold, and since we are controlling three scalar (the component of the error along x, y, z), of dimension 3x3:

$$\boldsymbol{A = IncreasingBellShapedFunction}(\boldsymbol{t} - \Delta, \boldsymbol{t}, \mathbf{0}, \mathbf{1})_{3\times3}$$



We can move to the evaluation of the task Jacobian, the variable we want to control is the misalignment vector between the vehicle and goal reference frame origin $\boldsymbol{\rho}_{v/g}$ that can be easily obtained from the unit vector lemma, a less constraining solution can be computed by just controlling the norm of such vector θ:

$$\boldsymbol{\rho}_{v/g} = \theta \boldsymbol{n}$$

In order to control its behavior we need to compute its time derivative:

$$\dot{\theta} = \boldsymbol{n} \cdot \boldsymbol{\omega}_{v/w}$$

Which is already expressed respect one of our control variables, allowing us to define directly the task Jacobian as:

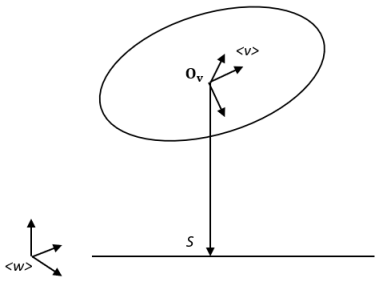
$$\dot{\theta} = [\mathbf{0}_{1\times7}\ \mathbf{0}_{1\times3}\ {}^{v}\boldsymbol{n}^{T}]\,\dot{\boldsymbol{y}}$$
$$\boldsymbol{J}_{\theta} = [\mathbf{0}_{1\times7}\ \mathbf{0}_{1\times3}\ {}^{v}\boldsymbol{n}^{T}]$$

Then the task reference, which is the controlling law that drive the control variable, can simply be expressed as:

$$\dot{\bar{\boldsymbol{x}}} = \lambda(\bar{\theta} - \theta) = -\lambda\theta \qquad \lambda > 0$$

Where our desired misalignment to the goal $\bar{\theta}$ is equal to zero the formula simply become the one above.

4. Task: Minimum altitude control:



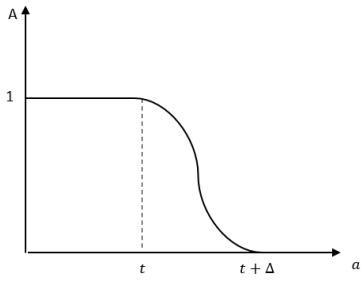
This objective requires that the vehicle altitude remain over a certain threshold $\boldsymbol{t}$ for the whole time of the action, considering the seabed flat.
In other word the following inequality objective must be satisfied:

$$a > t \qquad t = altitude\ threshold$$

Where the altitude is described as the vertical distance between the vehicle reference frame origin O_v and the seabed S.
As we can see this task is structured as an inequality task and acquire the attribute of safety task.

$$x > t \qquad t = \text{altitude threshold}$$

So since the altitude must remain over a certain threshold the activation function will be a decreasing bell shaped function that will reach the pick when the altitude reach the threshold, and since we are controlling one scalar, of dimension 1:



$$A = DecreasingBellShapedFunction(\ t, t + \Delta, 0, 1)$$

We can move to the evaluation of the task Jacobian, the variable we want to control is the distance between the vehicle O_v and the seabed S, expressed by the distance vector $\boldsymbol{d}$:

$$\boldsymbol{d} = (\boldsymbol{S} - \boldsymbol{O}_v)$$

In order to control its behavior we need to compute its time derivative:

$$D\boldsymbol{d} = D(\boldsymbol{S} - \boldsymbol{O}_v) = D\boldsymbol{S} - D\boldsymbol{O}_v = \boldsymbol{P}_{xy}(\boldsymbol{v}_v) - \boldsymbol{v}_v$$

Where $\boldsymbol{P}_{xy}(\boldsymbol{v}_v)$ is the projection of the vehicle on the horizontal plane which can be evaluated considering the plane normal which is the $\boldsymbol{k}_w$ unit vector of the world reference frame <*w*>:

$$\boldsymbol{P}_{xy}(\boldsymbol{v}_v) = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\boldsymbol{v}_v$$

Obtaining:

$$D\boldsymbol{d} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\boldsymbol{v}_v - \boldsymbol{v}_v = \boldsymbol{v}_v - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)\boldsymbol{v}_v - \boldsymbol{v}_v = -\boldsymbol{k}_w(\boldsymbol{k}_w \cdot)\boldsymbol{v}_v$$

As before with the intent of constraint less the motion of the vehicle we can just observe the derivative of the module of $\boldsymbol{d}$:

$$\|\boldsymbol{d}\| = a = \|-\boldsymbol{k}_w(\boldsymbol{k}_w \cdot)\boldsymbol{v}_v\| = -\boldsymbol{k}_w \cdot \boldsymbol{v}_v$$

Obtaining the relative Jacobian matrix:

$$\boldsymbol{J}_a = \left[\boldsymbol{0}_{1\times7}\ -\boldsymbol{k}_w{}^{\boldsymbol{T}}\boldsymbol{0}_{1\times3}\right]$$

In which the – in front of $\boldsymbol{k}_w$ depends on the orientation of the distance vector $\boldsymbol{d}$.

Then the task reference, which is the controlling law that drive the control variable, can simply be expressed as:

$$\dot{\boldsymbol{x}} = \lambda(\bar{a} - a) \qquad \lambda > 0$$

Where $\bar{a}$ is our desired minimum altitude $\bar{a}$ and a is the actual one.

5. **Task: altitude control (landing):**

This objective requires that the vehicle altitude will converge to a specified value.

In other word the following equality objective must be satisfied:

$$a = \bar{a} \qquad \bar{a} = desired\ altitude$$

Where the altitude is described as the vertical distance between the vehicle reference frame origin O_v and the seabed S.

As we can see this task is structured as an equality task and acquire the attribute of action defining.

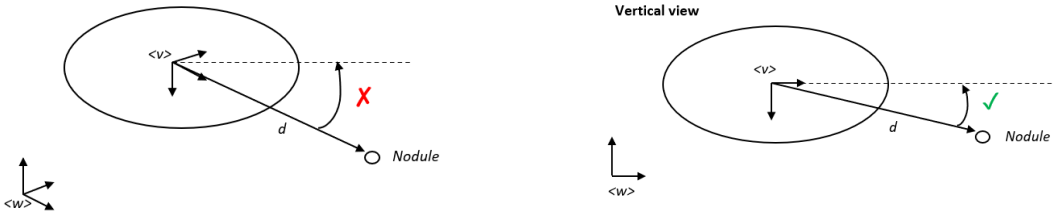
$$x = \bar{a} \qquad\qquad \bar{a} = desired\ altitude$$

So the activation function will be a constant equal to 1 since it will stay always active and we're controlling only a scalar.

The rest of the discussion for the evaluation of the task Jacobian and the task reference is the same as the one of the minimum altitude control.

6. **Task: Heading to nodule:**

This objective requires that the vehicle align to an object in the space, the nodule, but only in the horizontal part, as we can see from the figure below:



In other word the following equality objective must be satisfied:

$$\boldsymbol{\rho}_{v/g} = 0$$

Where $\boldsymbol{\rho}_{v/g}$ is the misalignment vector between the vehicle $\boldsymbol{i}_v$-unit vector and the projection of the distance vector $\boldsymbol{d}$ on the world horizontal plane.
As we can see this task is structured as an equality task and acquire the attribute of operational prerequisite.

$$\boldsymbol{x} = \overline{\boldsymbol{x}} \qquad \overline{\boldsymbol{x}} = \text{desired misalignment}$$

So the activation function will be an identity matrix, and since we are controlling three scalar (the component of the error along x, y, z), of dimension 3x3:

$$\boldsymbol{A} = \boldsymbol{I}_{3\times 3}$$

We can move to the evaluation of the task Jacobian, let's start by evaluating the distance vector between the vehicle position $\boldsymbol{O}_v$ and the nodule $\boldsymbol{N}$, that we will call $\boldsymbol{d}$.

$$\boldsymbol{d} = (\boldsymbol{N} - \boldsymbol{O}_v)$$

We can now project such vector on the world horizontal plane as:

$$\boldsymbol{d1} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\boldsymbol{d} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)](\boldsymbol{N} - \boldsymbol{O}_v) = P(\boldsymbol{N} - \boldsymbol{O}_v)$$

Where $\boldsymbol{k}_w$ is the vertical unit vector of the world reference frame, $\boldsymbol{I}$ the identity matrix.
Thus the problem would be solved if the $\boldsymbol{i}_v$-unit vector of the vehicle reference frame align to unit vector $\boldsymbol{n}_d = \boldsymbol{d1}/\|\boldsymbol{d1}\|$, let's us consider the misalignment vector between the two, computed with the reduced verso lemma:

$$\boldsymbol{\rho} = \theta\boldsymbol{n} = \frac{\theta}{\sin\theta}\boldsymbol{i}_v \times \boldsymbol{n}_d$$

To make the two unit vector align we need to find out the relationship between how $\boldsymbol{\rho}$ evolves in time and our system control variables $\dot{\boldsymbol{y}}$.
The derivative of $\boldsymbol{\rho}$ with respect to an observer α is given by:

$$D_\alpha\boldsymbol{\rho} = \boldsymbol{n}\dot{\theta} + \theta D_\alpha\boldsymbol{n} = \boldsymbol{n}\boldsymbol{n} \cdot \boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v} + \theta N(\theta)\boldsymbol{\omega}_{\boldsymbol{n}_d/\alpha} - \theta M(\theta)\boldsymbol{\omega}_{\boldsymbol{i}_v/\alpha}$$

If we consider our observer α on one of the two vectors, for example considering it on the vehicle reference frame <*v*>, the equation simplifies as:

$$D_\alpha\boldsymbol{\rho} = \boldsymbol{n}\boldsymbol{n} \cdot \boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v} + \theta N(\theta)\boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v}$$

With the intent of expressing such equation in terms of the system control variables $\dot{\boldsymbol{y}}$ we can express $\boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v}$ as:

$$\boldsymbol{\omega}_{\boldsymbol{n}_d/\boldsymbol{i}_v} = \boldsymbol{\omega}_{\boldsymbol{n}_d/w} - \boldsymbol{\omega}_{\boldsymbol{i}_v/w}$$

Where $\boldsymbol{\omega_{i_v/_w}}$ is the angular velocity between the $\boldsymbol{i_v}$-unit vector of the vehicle reference frame and the world reference frame and belong to the same class of the vehicle rotational velocity so we can say:

$$\boldsymbol{\omega_{i_v/_w}} = \boldsymbol{\omega_{v/_w}}$$

Concerning $\boldsymbol{\omega n_{d/_w}}$ we know that the tip of the vector $\boldsymbol{d1}$ moves with a velocity which satisfies:

$$\boldsymbol{v}_{d1/w} = \boldsymbol{\omega}_{d/w} \times \boldsymbol{d1}$$

where $\boldsymbol{v}_{d1/w}$ can be evaluated as the derivative of $\boldsymbol{d1}$ as:

$$D_w \boldsymbol{d1} = D_w[\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)](\boldsymbol{N} - \boldsymbol{O}_v) + [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]D_w(\boldsymbol{N} - \boldsymbol{O}_v)$$

Where $D_w[\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]$ is equal to zero since all its terms are fixed in the world reference frame, obtaining:

$$D_w \boldsymbol{d1} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]D_w(\boldsymbol{N} - \boldsymbol{O}_v) = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)](\boldsymbol{v}_{\frac{n}{w}} - \boldsymbol{v}_{\frac{v}{w}})$$

In our study case we are considering a fixed nodule in space, thus the equation simplifies as:

$$D_w \boldsymbol{d1} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\left(-\boldsymbol{v}_{\frac{v}{w}}\right) = \boldsymbol{v}_{d1/w}$$

Where $\boldsymbol{v}_{\frac{v}{w}}$ is part of the control variables of the vector $\dot{\boldsymbol{y}}$.

We can then solve the inverse cross product problem, finding out $\boldsymbol{\omega}_{d/w}$ which is a vector orthogonal to both $\boldsymbol{v}_{d1/w}$ and $\boldsymbol{d1}$ whose length is equal to $\|\boldsymbol{v}_{d1/w}\|/\|\boldsymbol{d1}\|$ as:

$$\boldsymbol{\omega}_{d/w} = \left(\frac{\boldsymbol{d1}}{\|\boldsymbol{d1}\|} \times \frac{\boldsymbol{v}_{d1/w}}{\|\boldsymbol{v}_{d1/w}\|}\right)\frac{\|\boldsymbol{v}_{d1/w}\|}{\|\boldsymbol{d1}\|} = (\boldsymbol{d1} \times \boldsymbol{v}_{\frac{d1}{w}})\frac{1}{\|\boldsymbol{d1}\|^2}$$

$$\boldsymbol{\omega}_{d/w} = (\boldsymbol{d1} \times [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\left(-\boldsymbol{v}_{\frac{v}{w}}\right))\frac{1}{\|\boldsymbol{d1}\|^2}$$

$$\boldsymbol{\omega}_{d/w} = -\frac{1}{\|\boldsymbol{d1}\|^2}(\boldsymbol{d1} \times [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\left(\boldsymbol{v}_{\frac{v}{w}}\right))$$

Which putted back In the equation of $\boldsymbol{\omega n_{d/_{i_v}}}$ gives us:

$$\boldsymbol{\omega n_{d/_{i_v}}} = -\frac{1}{\|\boldsymbol{d1}\|^2}(\boldsymbol{d1} \times [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)]\left(\boldsymbol{v}_{\frac{v}{w}}\right)) - \boldsymbol{\omega_{i_v/_w}}$$

by recalling that:

$$\theta N(\theta) = -\frac{\theta}{sin\theta}\boldsymbol{n}_d \times \{\boldsymbol{i}_v \times [\boldsymbol{I} - \boldsymbol{n}(\boldsymbol{n} \cdot))]\}\boldsymbol{\omega n_{d/_{i_v}}}$$

We can obtain the final relation between our control's variables and the derivative of $\boldsymbol{\rho}$:

$$D_v\boldsymbol{\rho} = \left(\boldsymbol{n}\boldsymbol{n}^T - \frac{\theta}{sin\theta}[\boldsymbol{n}_d \times][\boldsymbol{i}_v \times][\boldsymbol{I}_{3\times3} - \boldsymbol{n}\boldsymbol{n}^T]\right)\left[\boldsymbol{0_{3\times7}}\right.$$
$$\left. -\frac{1}{\|\boldsymbol{d1}\|^2}[\boldsymbol{d1} \times][\boldsymbol{I}_{3\times3} - \boldsymbol{k}_w\boldsymbol{k}_w^T] \quad -\boldsymbol{I}_{3\times3}\right]\dot{\boldsymbol{y}}$$

From which we can finally extract the task Jacobian:

$$\boldsymbol{J}_n = \left(\boldsymbol{n}\boldsymbol{n}^T - \frac{\theta}{sin\theta}[\boldsymbol{n}_d \times][\boldsymbol{i}_v \times][\boldsymbol{I}_{3\times3} - \boldsymbol{n}\boldsymbol{n}^T]\right)\left[\boldsymbol{0_{3\times7}}\right.$$
$$\left. -\frac{1}{\|\boldsymbol{d1}\|^2}[\boldsymbol{d1} \times][\boldsymbol{I}_{3\times3} - \boldsymbol{k}_w\boldsymbol{k}_w^T] \quad -\boldsymbol{I}_{3\times3}\right]$$

To solve the problem we will need to create a task which is able to bring $\boldsymbol{\rho}$ to zero, among many possible choices, one is required the derivative of $\boldsymbol{\rho}$ to be proportional to the misalignment vector itself:

$$\dot{\bar{\boldsymbol{\rho}}} = \lambda(\bar{\boldsymbol{\rho}} - \boldsymbol{\rho}) = -\lambda\boldsymbol{\rho}$$

Where $\bar{\boldsymbol{\rho}}$ is our desired misalignment and it's equal to zero, if the true derivative $D\boldsymbol{\rho}$ follows the desired value, this would guarantee an asymptotic convergence of $\boldsymbol{\rho} \to \mathbf{0}$.

7. Task: distance to nodule:

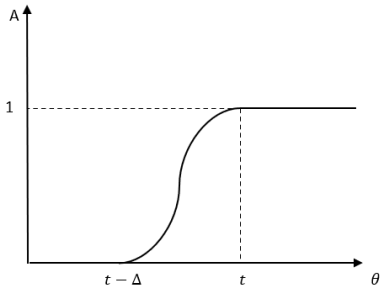
This objective requires that the distance between the manipulator base $\boldsymbol{B}$ and the nodule $\boldsymbol{N}$ projected on the horizontal plane goes under a desired threshold t in such a way that the nodule will be in the workspace of the arm.
In other word the following equality objective must be satisfied:

$$d_h < t$$

Where t is the desired scalar position error between the manipulator base and the nodule position projected on the horizontal plane.
As we can see this task is structured as an inequality task and acquire the attribute of operational prerequisite task.

$$\boldsymbol{x} < \bar{\boldsymbol{x}} \qquad\qquad \bar{\boldsymbol{x}} = \text{desired distance}$$

So the activation function will be an increasing bell shaped function, and since we are controlling three scalar (the component of the error along x, y, z), of dimension 3x3:



$$\boldsymbol{A} = \boldsymbol{IncreasingBellShapedFunction}(\boldsymbol{t} - \Delta, \boldsymbol{t}, \mathbf{0}, \mathbf{1})_{3\times3}$$

We can move to the evaluation of the task Jacobian, the variable we want to control is the projection of the distance vector between the arm base and the nodule, which is equal to:

$$\boldsymbol{d} = (\boldsymbol{N} - \boldsymbol{B})$$

Where we can express the distance in terms of also the vehicle reference frame origin as:

$$\boldsymbol{d} = (\boldsymbol{N} - (\boldsymbol{O}_v + \boldsymbol{r}_{O-B}))$$

Where $\boldsymbol{r}_{O-B}$ is a constant vector expressing the position of the arm base $\boldsymbol{B}$ with respect to the origin of the vehicle reference frame $\boldsymbol{O}_v$.
We may now pass to the evaluation of the projection of such a vector on the horizontal plane, obtaining the vector $\boldsymbol{d1}$:

$$\boldsymbol{d1} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w\,\cdot)](\boldsymbol{N} - (\boldsymbol{O}_v + \boldsymbol{r}_{O-B}))$$

To control that variable we need to compute its derivative and express it in terms of the control variables embedded in the vector $\dot{\boldsymbol{y}}$.

$$D_w\boldsymbol{d1} = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w\,\cdot)]D_w(\boldsymbol{N} - \boldsymbol{O}_v - \boldsymbol{r}_{O-B}) = [\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w\,\cdot)](\boldsymbol{v}_{\frac{n}{w}} - \boldsymbol{v}_{\frac{v}{w}} - \boldsymbol{v}_{\frac{O}{B}})$$

Where both $\boldsymbol{v}_{\frac{n}{w}}$ and $\boldsymbol{v}_{\frac{O}{B}}$ are equal to zero since the nodule is fixed in space during time as the distance between the origin of the vehicle reference frame $\boldsymbol{O}_v$ and the arm base $\boldsymbol{B}$, obtaining:

$$D_w\boldsymbol{d1} = -[\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w\,\cdot)]\dot{\boldsymbol{y}}$$

Obtaining the task Jacobian:

$$\boldsymbol{J}_b = [\mathbf{0}_{3\times7} \quad -[\boldsymbol{I} - \boldsymbol{k}_w(\boldsymbol{k}_w \cdot)] \ \mathbf{0}_{3\times7}]$$

Then the task reference, which is the controlling law that drive the control variable, can be obtained by modifying a bit the above relation as:

$$d_h < t$$
$$\|\boldsymbol{d1}\| < t$$
$$\boldsymbol{d1} < \frac{\boldsymbol{d1}}{\|\boldsymbol{d1}\|} t$$
$$\bar{\boldsymbol{t}} = \frac{\boldsymbol{d1}}{\|\boldsymbol{d1}\|} t$$
$$\dot{\bar{\boldsymbol{x}}} = \lambda(\bar{\boldsymbol{t}} - \boldsymbol{d1}) \qquad \lambda > 0$$

In which t is the desired distance between the base and the nodule as described before.

8. Task: fixed base:

This objective requires that the vehicle maintain its base fixed, both in the rotational and translational movement, this means that we will control directly the velocities of the base that are already in the control variables vector $\dot{\boldsymbol{y}}$.

$$\boldsymbol{v}_v = 0 \qquad \boldsymbol{\omega}_v = 0$$

As we can see this task is structured as an equality task and acquire the attribute of operational prerequisite.

So the activation function will be an identity matrix, and since we are controlling three scalar (the component of the error along x, y, z), of dimension 3x3:

$$\boldsymbol{A} = \boldsymbol{I}_{3\times3}$$

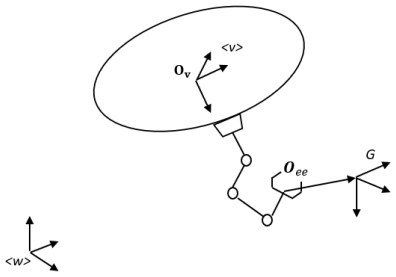
We can move to the evaluation of the task Jacobian, the variable we want to control are the velocities of the base, so the task Jacobian should extract them from the control variable vector $\dot{\boldsymbol{y}}$ obtaining:

$$\boldsymbol{J}_{fb} = [\mathbf{0}_{6\times7} \ \boldsymbol{I}_{6\times6}]$$

Then the task reference, which is the controlling law that drive the control variable, can simply be expressed as:

$$\dot{\bar{\boldsymbol{x}}} = 0$$

9. Task: End-effector position control:



This objective requires, under the assumption of having a fixed base, that the end-effector frame origin O_{ee} converges to a given goal frame <*g*> origin *G.*

In other word the following equality objective must be satisfied:

$$r_i = \mathbf{0} \quad i = 1, 2, 3$$

Where **r** is the position error between the end-effector and the goal frame.

As we can see this task is structured as an equality task and acquire the attribute of action defining task.

$$\boldsymbol{x} = \bar{\boldsymbol{x}} \qquad \bar{\boldsymbol{x}} = \text{desired goal position}$$

So the activation function will be an identity matrix, and since we are controlling three scalar (the component of the error along x, y, z), of dimension 3x3:

$$\boldsymbol{A} = \boldsymbol{I}_{3\times3}$$

We can move to the evaluation of the task Jacobian, the variable we want to control is the distance vector between the end-effector and goal reference frame origin:

$$\boldsymbol{d} = (\boldsymbol{G} - \boldsymbol{O}_{ee})$$

In order to control its behavior we need to compute its time derivative:

$$D\boldsymbol{d} = D(\boldsymbol{G} - \boldsymbol{O}_{ee}) = D\boldsymbol{G} - D\boldsymbol{O}_{ee} = \boldsymbol{v}_g - \boldsymbol{v}_{ee}$$

In our case we're considering the goal as fixed in space, obtaining a derivative equal to:

$$D\boldsymbol{d} = -\boldsymbol{v}_{ee}$$

Where the linear velocity of the end-effector can be obtained with the first three rows of the basic Jacobian of the manipulator in terms of the joint configuration velocities $\dot{q}_{\boldsymbol{i}}$, which are part of our control variables $\dot{\boldsymbol{y}}$.

$$\boldsymbol{v}_{ee} = \boldsymbol{J}_b(1{:}3, 1{:}7)\dot{\boldsymbol{y}}$$

Allowing us to define the task Jacobian for this task as:

$$D\boldsymbol{d} = [-\boldsymbol{J}_b(1{:}3, 1{:}7)\ \boldsymbol{0}_{3\times3}\ \boldsymbol{0}_{3\times3}]\ \dot{\boldsymbol{y}}$$

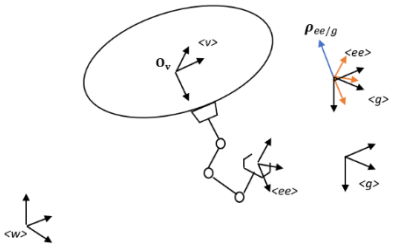
$$\boldsymbol{J}_d = [-\boldsymbol{J}_b(1{:}3, 1{:}7)\ \boldsymbol{0}_{3\times3}\ \boldsymbol{0}_{3\times3}]$$

Then the task reference, which is the controlling law that drive the control variable, can simply be expressed as:

$$\dot{\bar{\boldsymbol{x}}} = \lambda(\bar{\boldsymbol{d}} - \boldsymbol{d}) = -\lambda\boldsymbol{d} \qquad \lambda > 0$$

Where since our desired distance to the goal $\bar{d}$ is equal to zero the formula simply become the one above.

10. Task: End-effector orientation control:



This objective requires that, under the assumption of having a fixed base, the end-effector frame *<ee>* converges to a given goal frame *<g>.*
In other word the following equality objective must be satisfied:

$$\theta_i = 0 \qquad i = 1, 2, 3$$

Where $\boldsymbol{\theta}$ is the misalignment error between the vehicle and goal frames.
As we can see this task is structured as an equality task and acquire the attribute of action defining since it defines the action itself.

$$\boldsymbol{x} = \bar{\boldsymbol{x}} \qquad \bar{\boldsymbol{x}} = \text{desired goal orientation}$$

So the activation function will be an identity matrix, and since we are controlling three scalar (the component of the error along x, y, z), of dimension 3x3:

$$\boldsymbol{A} = \boldsymbol{I}_{3\times3}$$

We can move to the evaluation of the task Jacobian, the variable we want to control is the misalignment vector between the end-effector and goal reference frame origin $\boldsymbol{\rho}_{ee/g}$ that can be easily obtained from the unit vector lemma.

$$\boldsymbol{\rho}_{ee/g} = \theta\boldsymbol{n}$$

In order to control its behavior we need to compute its time derivative:

$$D\boldsymbol{\rho} = \boldsymbol{n}\dot{\theta} + \theta D\boldsymbol{n} = [\boldsymbol{n}(\boldsymbol{n}\,\cdot) + \boldsymbol{N}(\theta)]\boldsymbol{\omega}_{ee/g}$$

In which since we assume the goal reference frame <g> fixed we can simplify the equation as:

$$D_w\boldsymbol{\rho} = \boldsymbol{n}(\boldsymbol{n}\,\cdot)\boldsymbol{\omega}_{ee/w} = \boldsymbol{\omega}_{ee/w}$$

Moreover the angular velocity of the end effector can be expressed in terms of the configuration velocities $\dot{q}_{\boldsymbol{\iota}}$ by the lastly three rows of the basic Jacobian of the manipulator as:

$$\boldsymbol{\omega}_{ee} = \boldsymbol{J}_b(4{:}\,6, 1{:}\,7)\dot{\boldsymbol{y}}$$

Allowing us to define the task Jacobian for this task as:

$$D\boldsymbol{d} = [-\boldsymbol{J}_b(4{:}\,6, 1{:}\,7)\ \mathbf{0}_{3\times3}\ \mathbf{0}_{3\times3}]\ \dot{\boldsymbol{y}}$$
$$\boldsymbol{J}_\rho = [-\boldsymbol{J}_b(4{:}\,6, 1{:}\,7)\ \mathbf{0}_{3\times3}\ \mathbf{0}_{3\times3}]$$

Then the task reference, which is the controlling law that drive the control variable, can simply be expressed as:

$$\dot{\bar{\boldsymbol{x}}} = \lambda(\bar{\boldsymbol{\rho}} - \boldsymbol{\rho}) = -\lambda\boldsymbol{\rho} \qquad \lambda > 0$$

Where since our desired misalignment to the goal $\bar{\boldsymbol{\rho}}$ is equal to zero the formula simply become the one above.

11. Task: Joint limit:

This objective requires that the joint configuration remain bounded by an upper and lower limit:

$$q_i > q_i^{min} \qquad q_i < q_i^{max} \qquad i = 1, 2, 3, .., n$$

Where q_i^{min} and q_i^{max} are respectively the lower and upper bound of the i-joint configuration and n is the number of joint of the manipulator.

As we can see this task is designed as an inequality one, and acquire the attribute of safety task.

Normally this task can be subdivided in two tasks, one for the upper limit and one for the lower one and then stuck them together to obtain the same priority, with this goal in mind, noticing also the structure of the inequality the activation functions will be a diagonal matrix of dimension equal to the number of joints in which for the lower joint limit each element will be a decreasing bell shaped function, while for the upper joint limit an increasing bell shaped function, both related to the actual value of the configuration and the relative limit:

$$\boldsymbol{A_{min}} = diag\left(DecreasingBellShapedFunction\left(\ q_i^{min}, q_i^{min} + \Delta, 0, 1\right)\right)$$
$$\boldsymbol{A_{max}} = diag\left(IncreasingBellShapedFunction\left(\ q_i^{max} - \Delta, q_i^{max}, 0, 1\right)\right)$$

We can move to the evaluation of the task Jacobian, since we want to directly control the joint configuration q_i we can act on their derivatives that can be easily obtained from the control variables vector $\dot{\boldsymbol{y}}$ as:

$$\boldsymbol{J}_{jl}^{min} = \boldsymbol{J}_{jl}^{max} = [\,\boldsymbol{I}_{7\times7}\ \mathbf{0}_{7\times7}\ \mathbf{0}_{7\times7}]$$

Then the task reference, which is the controlling law that drive the control variable, can simply be expressed as:

$$\dot{\bar{\boldsymbol{x}}}_{\boldsymbol{min}} = \lambda\left((\boldsymbol{q}^{min} + \Delta) - \boldsymbol{q}\right) \qquad \lambda > 0$$
$$\dot{\bar{\boldsymbol{x}}}_{\boldsymbol{max}} = \lambda((\boldsymbol{q}^{max} - \Delta) - \boldsymbol{q}) \qquad \lambda > 0$$

- Implementation remark:

The above relationship can be exploited in a smarter solution stacking the task references as:

```
% reference for the joint limits task:
uvms.xdot.jl = zeros(7, 1);
for i = 1:size(uvms.jlmin)
    mean(i) = (uvms.jlmin(i) + uvms.jlmax(i))/2;
    if uvms.q(i) <= mean(i)
        uvms.xdot.jl(i) = 0.2 * (uvms.jlmin(i) - uvms.q(i) +0.01);
    else
        uvms.xdot.jl(i) = 0.2 * (uvms.jlmax(i) - uvms.q(i) -0.01);
    end
end
```

where:

`uvms.xdot.jl` is the global joint limit task reference.

`uvms.jlmin` is the vector of the lower joint limits.

`mean` is the vector of the mean value of the configuration.

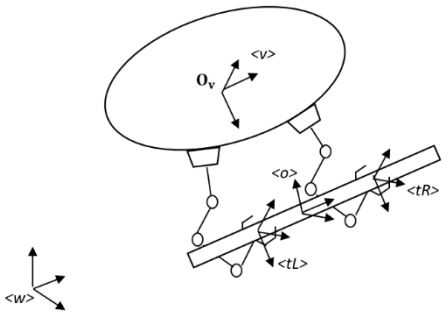
`uvms.q` is the configuration vector of the manipulator.

and then the proposed logic is to put in each row of the task reference vector the task reference of the lower limit or of the upper limit based on the actual joint configurations, it it's nearer to the lower limit then its task reference will be inserted in the vector, otherwise the upper limit one will be inserted.

And putting together the activation functions as:

```
ujl = [2.9;1.65;2.9;0.01;2.9;1.25;2.8];
ljl = [-2.9;-1.6;-2.9;-2.95;-2.9;-1.65;-2.8];
for i = 1:size(ujl, 1)
    ArrayActivationFunction(i, i) =
IncreasingBellShapedFunction(uvms.jlmax(i) - 0.01, uvms.jlmax(i), 0,
1, uvms.q(i)) + DecreasingBellShapedFunction(uvms.jlmin(i),
uvms.jlmin(i) + 0.01, 0, 1, uvms.q(i));
end
uvms.A.jl = ArrayActivationFunction .* ActionTransition("JL",
previous_tasks, current_tasks, mission.phase_time);
```

12. Task: Kinematic constraint:



This objective takes into consideration the constraint imposed by the manipulation of an object by two different manipulators mounted on the same base, avoiding to introduce stresses on it.

To do that we need first to compute for each arm the basic Jacobian for each arm respect a common reference frame, in our case the object one *<o>:*

$$\boldsymbol{J}_o = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ [\boldsymbol{r}_{t/o}\times] & \boldsymbol{I}_{3\times3} \end{bmatrix} \boldsymbol{J}_t$$

Where:

$\boldsymbol{J}_o$ is the Jacobian respect the object reference frame.

$[\boldsymbol{r}_{t/o}\times]$ is the distance vector between the tool frame and the object frame, projected in the world reference frame.

$\boldsymbol{J}_t$ is the basic Jacobian of the arm respect the tool frame expressed in the world reference frame.

The two Jacobians are then stacked together as:

$$\boldsymbol{J}_o^{kc} = [\boldsymbol{J}_o^L \quad \boldsymbol{J}_o^R]$$

Which is our task Jacobian for that task.

Then the task reference asks that all the velocities obtained with this Jacobian are equal to zero for each arm:

$$0 = \boldsymbol{J}_o^{kc}\dot{\boldsymbol{y}}$$

Finally obtaining:

$$\dot{\bar{\boldsymbol{x}}} = \boldsymbol{0}$$

This task should be always active while the object is grasped, for this reason the activation function will be an identity matrix of size 6 by 6:

$$\boldsymbol{A} = \boldsymbol{I}_{6\times6}$$

For its importance the action will acquire the attribute of constraint and safety task.