**Exercise #4**

A planar 3R robot with unitary link lengths is commanded by a joint velocity $\dot{\boldsymbol{q}} \in \mathbb{R}^3$ with components bounded as $|\dot{q}_i| \le 2$ [rad/s], $i = 1, 2, 3$. The D-H joint variables have limited ranges specified by

$$q_1 \in [-\pi/2, \pi/2], \qquad q_2 \in [0, 2\pi/3], \qquad q_3 \in [-\pi/4, \pi/4].$$

At the configuration $\widehat{\boldsymbol{q}} = (2\pi/5, \pi/2, -\pi/4)$, the robot should move its end-effector horizontally with a speed $v_x = -3$ [m/s], while trying to keep the joints close to their midranges. Compute the value of the instantaneous joint velocity $\dot{\boldsymbol{q}}$ that performs the Cartesian task while improving at best the criterion $H_{range}(\boldsymbol{q})$. Check if this joint velocity is feasible and, if not, perform the least end-effector task scaling to recover feasibility.

**Exercise #4**

The planar 3R robot ($n = 3$) is redundant for the Cartesian position task ($m = 2$). When the joint limits are not regarded as hard constraints, the solution to the stated problem is

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}(\boldsymbol{q})\dot{\boldsymbol{r}} - \left(\boldsymbol{I} - \boldsymbol{J}^{\#}(\boldsymbol{q})\boldsymbol{J}(\boldsymbol{q})\right)\nabla_q H_{range}(\boldsymbol{q}),$$

where the task velocity is

$$\boldsymbol{r} = \begin{pmatrix} p_x \\ p_y \end{pmatrix} \quad \Rightarrow \quad \dot{\boldsymbol{r}} = \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} -3 \\ 0 \end{pmatrix},$$

and the associated Jacobian, evaluated at $\widehat{\boldsymbol{q}} = (2\pi/5, \pi/2, -\pi/4)$, is given by

$$\boldsymbol{J}(\boldsymbol{q}) = \begin{pmatrix} -(s_1 + s_{12} + s_{123}) & -(s_{12} + s_{123}) & -s_{123} \\ c_1 + c_{12} + c_{123} & c_{12} + c_{123} & c_{123} \end{pmatrix} \Rightarrow \boldsymbol{J} = \begin{pmatrix} -2.1511 & -1.2000 & -0.8910 \\ -1.0960 & -1.4050 & -0.4540 \end{pmatrix}.$$

For each joint $i$, we have a range $[q_{m,i}, q_{M,i}]$ and a midrange $\bar{q}_i = (q_{M,i} + q_{m,i})/2$. As a result, the objective function to be minimized is

$$H_{range}(\boldsymbol{q}) = \frac{1}{2n}\sum_{i=1}^{n} \frac{(q_i - \bar{q}_i)^2}{(q_{M,i} - q_{m,i})^2} = \frac{1}{6}\left(\frac{q_1^2}{\pi^2} + \frac{(q_2 - (\pi/3))^2}{(2\pi/3)^2} + \frac{q_3^2}{(\pi/2)^2}\right).$$

Its gradient evaluated at $\widehat{\boldsymbol{q}} = (2\pi/5, \pi/2, -\pi/4)$ is

$$\nabla_q H_{range}(\boldsymbol{q}) = \frac{1}{3}\begin{pmatrix} q_1/\pi^2 \\ (q_2 - \pi/3)/(2\pi/3)^2 \\ q_3/(\pi/2)^2 \end{pmatrix} \Rightarrow \nabla_q H_{range} = \begin{pmatrix} 0.0424 \\ 0.0398 \\ -0.1061 \end{pmatrix}.$$

As a result, the two terms of the solution are separately evaluated as

$$\dot{\boldsymbol{q}}_r = \boldsymbol{J}^{\#}\dot{\boldsymbol{r}} = \begin{pmatrix} 2.1076 \\ -1.9261 \\ 0.8730 \end{pmatrix}, \qquad \dot{\boldsymbol{q}}_n = -\left(\boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J}\right)\nabla_q H_{range} = \begin{pmatrix} -0.0437 \\ 0 \\ 0.1056 \end{pmatrix},$$

yielding thus

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_r + \dot{\boldsymbol{q}}_n = \begin{pmatrix} 2.0638 \\ -1.9261 \\ 0.9786 \end{pmatrix}. \tag{4}$$

The first component of the solution exceeds the (positive) velocity bound. This is true as well for the minimum norm solution $\dot{\boldsymbol{q}}_r$; the first component of the null space term $\dot{\boldsymbol{q}}_n$, being negative, mildens the situation but is not sufficient to recover feasibility. Therefore, the largest scaling factor $k < 1$ of the task velocity $\dot{\boldsymbol{r}}$ that allows to obtain a feasible solution w.r.t. the joint velocity bounds (uniformly equal to $\dot{q}_{max} = 2$ [rad/s] for all joints) is computed as follows:

$$\dot{\boldsymbol{r}} \to k\,\dot{\boldsymbol{r}} \Rightarrow \dot{\boldsymbol{q}} \to k\,\dot{\boldsymbol{q}}_r + \dot{\boldsymbol{q}}_n \Rightarrow k\,\dot{q}_{r,1} + \dot{q}_{n,1} \stackrel{\downarrow}{=} \dot{q}_{max} \Rightarrow k^* = \frac{\dot{q}_{max} - \dot{q}_{n,1}}{\dot{q}_{r,1}} = \frac{2 + 0.0437}{2.1076} = 0.9697.$$

Bounds on different components

Joint Variables are limited

We have a given configuration and starting from it, the Robot should move horizontally

We want to improve the criterion Hrange

We use the Projected Gradient method, which is the following:

$$\dot{q} = J^{\#}\dot{r} + (I - J^{\#}J)\dot{q}_0 \quad \text{and} \quad \dot{q}_0 = \nabla_q H(q)$$

$$\dot{q} = J^{\#}r + (I - J^{\#}J)\nabla_q H(q)$$

We have a starting configuration: $\hat{q} = \left(\frac{2\pi}{5}, \frac{\pi}{2}, -\frac{\pi}{4}\right)$ and $V_x = -3$ m/s

$$\dot{r} = \begin{pmatrix} -3 \\ 0 \end{pmatrix} \text{m/s}$$

$$\begin{pmatrix} -3 \\ 0 \end{pmatrix} = J(\hat{q})\begin{pmatrix} 2\pi/5 \\ \pi/2 \\ -\pi/4 \end{pmatrix} \Rightarrow r = \begin{cases} l_1c_1 + l_2c_{12} + l_3c_{123} \\ l_1s_1 + l_2s_{12} + l_3s_{123} \end{cases}$$

$$J = \begin{pmatrix} -2.1511 & -1.2 & -0.89 \\ -1.0960 & -1.405 & -0.454 \end{pmatrix}$$

$$\left(\frac{-\frac{\pi}{2} + \frac{\pi}{2}}{2}\right) = 0$$

$$H_{range}(q) = \underbrace{\frac{1}{2N}}_{\#\text{ of Joints}}\sum_{i=1}^{N}\left(\frac{q_i - \bar{q}_i}{q_{M,i} - q_{m,i}}\right)^2 \Rightarrow \bar{q}_i = \frac{q_{M,i} + q_{m,i}}{2} = \frac{0 + 2\pi/3}{2} = \frac{\pi}{3}$$

0

$$H_{range}(q) = \frac{1}{6}\left(\left(\frac{q_1}{\pi}\right)^2 + \left(\frac{q_2 - (\pi/3)}{2\pi/3}\right)^2 + \left(\frac{q_3}{(\pi/2)}\right)^2\right)$$

And in order to use this function for the projected gradient method, we need to differentiate it

We first compute the gradient, and then evaluate it at the $\hat{q}$ configuration

Thus we get: $\nabla_q H_{range}(q) = \begin{pmatrix} 0.0424 \\ 0.0398 \\ -0.1061 \end{pmatrix}$, and then combining all terms we get:

$\dot{q} = \begin{pmatrix} 2.0638 \\ -1.9261 \\ 0.9786 \end{pmatrix}$, which exceeds the bound on $\dot{q}_i \le 2$ rad/s on the first Joint

The problem asks to find the smallest coefficient needed in order to scale the velocity and making it feasible

We need to find a scaling coefficient $k$ such that: $\dot{r} = k \cdot \dot{r}$

The solution is given by 2 terms: $k\dot{q}_r + \dot{q}_n$

$\dot{q}_r$ is the term that violates the bounds, and thus:

$2 \text{ rad/s} = \dot{q}_{max} = k\dot{q}_r + \dot{q}_n \Rightarrow k = \frac{\dot{q}_{max} - \dot{q}_n}{\dot{q}_r}$, which gives us the correct Joint velocity value

The null Space component is used in order to execute secondary tasks, without affecting the primary task. The null Space velocity does not scale proportionally with $\dot{r}$ and moreover it is determined with the gradient, making it independent from other components.

SNS assumes to scale everything down, but this would not be the case since the null space component found with Hrange would still be there.

Let me clarify why SNS (Saturation in the Null Space) wouldn't work correctly for scaling in this situation.

The key issue is in how SNS typically works versus the nature of the solution in this problem.

In standard SNS, when a joint velocity limit is exceeded, the approach is to:

Identify which joints are saturated (at their limits)
Remove these joints from the solution by modifying the Jacobian
Recompute a new solution with the reduced Jacobian
This works well when you're only dealing with the minimum-norm solution, but it gets complicated when you have an additional null-space component that's independent of the task velocity.

Let's look at the specific numbers from the example:

The total solution is $\dot{q} = [2.0638, -1.9261, 0.9786]^T$
This consists of:
Minimum-norm part: $\dot{q}_r = [2.1076, -1.9261, 0.8730]^T$
Null-space part: $\dot{q}_n = [-0.0437, 0, 0.1056]^T$
The issue is that the first component (2.0638) exceeds the velocity bound of 2 rad/s.

If we tried to apply SNS directly, we'd be treating the entire solution $\dot{q}$ as if it could be scaled uniformly with the task velocity, but that's not true. The null-space component $\dot{q}_n$ is derived from the gradient of H_range, which is independent of the task velocity $\dot{r}$.

Put simply:

When you scale the task velocity $\dot{r} \to k \cdot \dot{r}$
The minimum-norm solution scales proportionally: $\dot{q}_r \to k \cdot \dot{q}_r$
But the null-space component remains unchanged: $\dot{q}_n \to \dot{q}_n$
So the correct approach (as shown in the slide) is to find the scaling factor k* that makes: $k^* \cdot \dot{q}_{r,1} + \dot{q}_{n,1} = \dot{q}_{max}$

This gives k* = 0.9697, which properly scales only the task-related component while preserving the null-space component that's working to optimize the secondary objective.

If we had incorrectly applied SNS by treating the entire solution as scalable with the task, we would have either:

Scaled everything (including the null-space part) which would not preserve optimality
Or failed to achieve feasibility because we wouldn't account for the proper relationship between scaling and the null-space contribution