Report SCARA workspace Lab

Emanuele BUA ODETTI

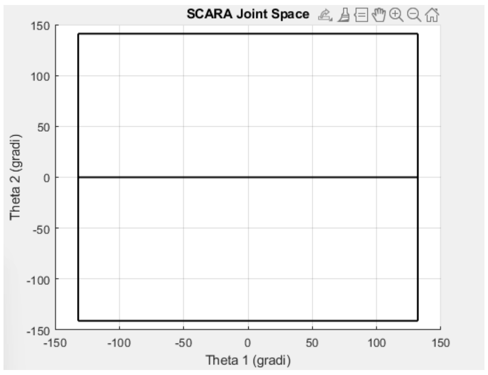
Enrico DONDERO

Exercise 1

This exercise was about developing a Matlab function to compute and plot both the cartesian and joint workspace of a reduced SCARA robot, given the link lengths, joint limits and obstacle position and radius.

To so so it was firstly necessary to define what the script had to do in order to achieve this task.

Starting from the computation of the workspace as a function of link lengths and joint limits. Given the simplicity of the system defining the joint space of the robot wasn't difficult, as we used a 2D plot to represent the two joint variables θ_1 and θ_2, the resulting plot is a rectangle, where the edges represent the joint limits and the line spanning on the 0 ordinate being the singularity across which the aspect of the SCARA robot changes.



To then plot the cartesian workspace of the robot were used the forward geometry equations, to associate each point on the edge of the joint space with a point in the xy plane.

$$\begin{cases} x_i = l_1 cos(\theta_{1i}) + l_2 cos(\theta_{1i} + \theta_{2i}) \\ y_i = l_1 sin(\theta_{1i}) + l_2 sin(\theta_{1i} + \theta_{2i}) \end{cases}$$

the obtained points were placed in two different bidimensional arrays for X and Y coordinates. Each array is made out of n_e columns corresponding to the number of edges used to trace the joint space, which vary depending on the presence of the obstacle.

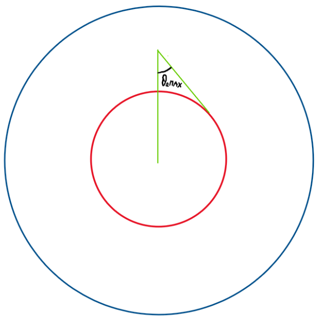
```
    for i = 1:size(edges, 3)
        for j = 1:num_points
            % Direct geometry equations
            X(j,i) = L1 * cos(edges(j, 1, i)) + ...
            ... L2 * cos(edges(j, 1, i) + edges(j, 2, i));
            Y(j,i) = L1 * sin(edges(j, 1, i)) + ...
            ... L2 * sin(edges(j, 1, i) + edges(j, 2, i));
        end
    end
```

where *num_points* is the resolution of the segments and was chosen to be 500.

Obstacle management

Joint 2 collision evaluation

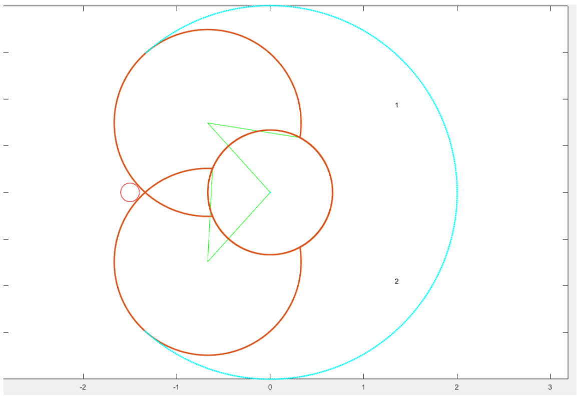
Before computing the updated workspace, it was necessary to check whether the obstacle could collide with the second joint, a case that wasn't supposed to be covered.

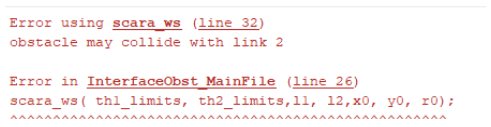


To do so, we verified the minimum reach of the second joint regardless of the angle of the first one. We used the closest point to the base of the SCARA from the EE to define the maximum radius for the allowable area of placement of the obstacle.

At the same time, it was implemented a check on the maximal reach of the EE to add the possibility of computing the workspace if the obstacle were to be placed outside of the working area.

This method of computing the collision space for joint 2 doesn’t take into account the possibility of placing the obstacle in the blind region of joint 2 caused by joint limits of joint 1. So this software is more cautious and doesn't allow the placement in that region, as a workaround to allow the computation of the whole joint space before computing the corresponding cartesian space points.





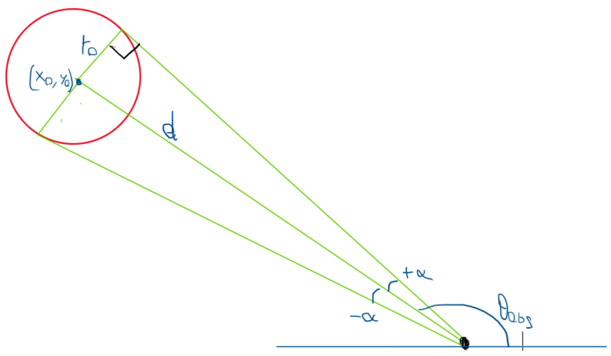
```
Error using scara_ws (line 32)
obstacle may collide with link 2

Error in InterfaceObst_MainFile (line 26)
scara_ws( th1_limits, th2_limits,l1, l2,x0, y0, r0);
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

on the left the workspace produced by the interface, while above the message provided by our script when the obstacle was placed in the blind spot of link 2

Obstacle in joint space

To compute the updated joint space, some geometrical considerations were made:



we used the known distance of the obstacle’s center computed as:

$$d = \sqrt{x_0^2 + y_0^2}$$

and the radius of the obstacle r_0, given as an argument of the function, to compute the angle α, that was summed and subtracted from the polar angle at which the obstacle was placed:

$$\theta_{obs\ lim} = \begin{bmatrix} \theta_{obs} - \alpha \\ \theta_{obs} + \alpha \end{bmatrix}$$

where

$$\theta_{obs} = atan2(y_0, x_0)$$

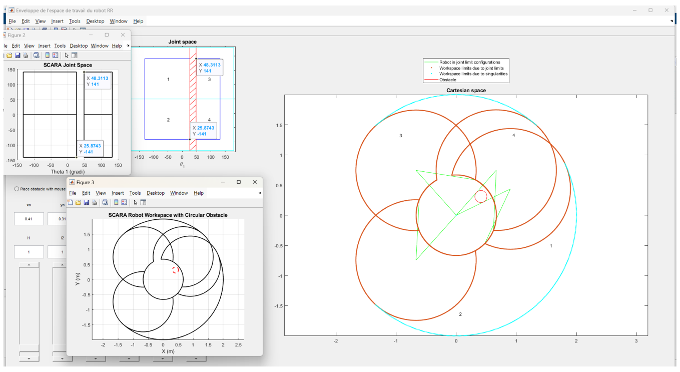
and

$$\alpha = arcsin(r_0/d)$$

thus obtaining the upper and lower limits for link 1 to avoid collision with the object. Now it was necessary to compute the joint space edges accordingly. This required the use of different *elseif* clauses to handle different scenarios.

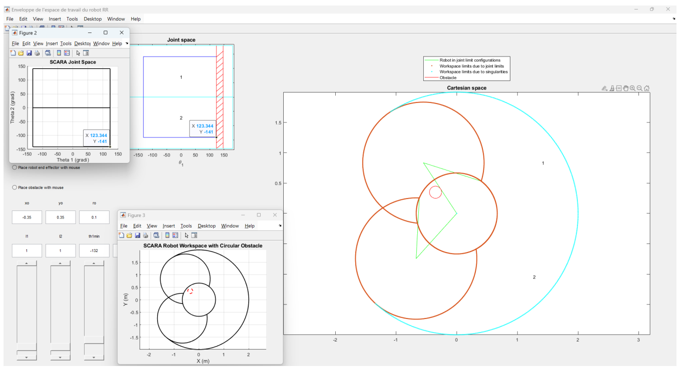
Obstacle angles completely inside the joint space

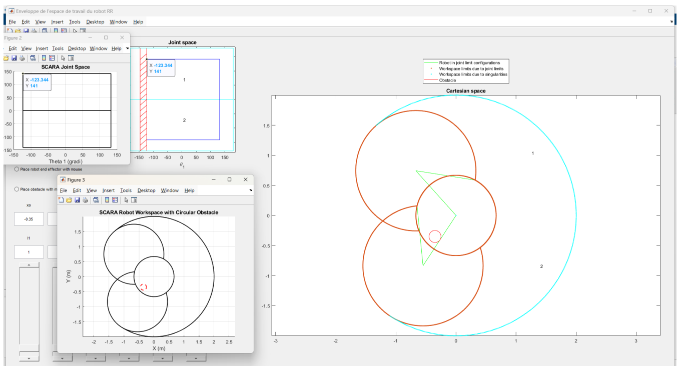
This case represents the main scenario, in which both upper and lower limits for the obstacle angles fall inside the joint space of the SCARA robot. This requires the use of 10 different segments to represent the whole joint space, divided in 4 vertical segments and 6 horizontal.



Obstacle partially overlapping joint space

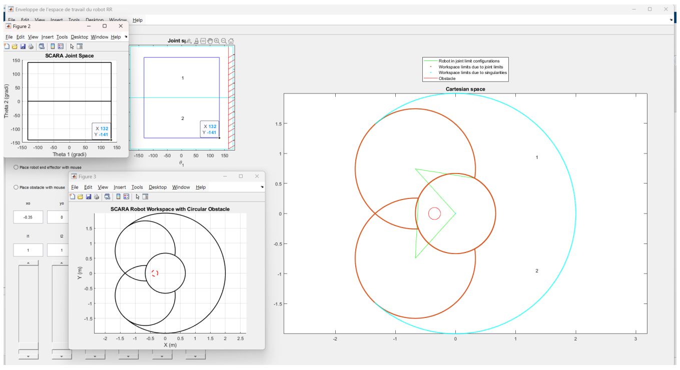
This case happens when just one of the obstacle limits fall inside the joint space, in this case, the number of segments remains five as the case with on obstacle, but one of the edges falls shorter than the joint limits of the robot





Obstacle outside of joint space

This happens when the obstacle is placed outside of joint 1 reach, but isn’t colliding with joint 2 either, thanks to initial evaluation.



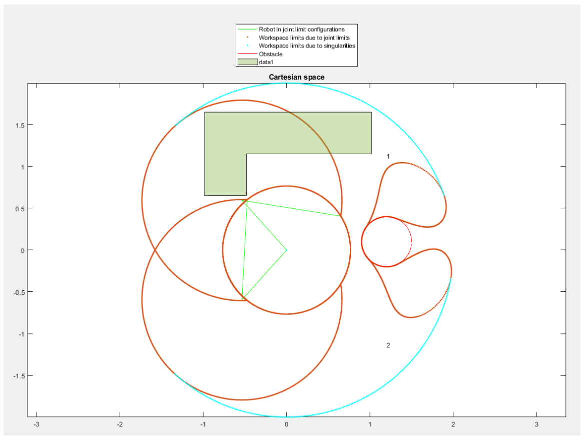
This case was handled using the *nargin* matlab variable, which counts the number of given inputs, in this way we computed the edges of joint space for this scenario as the obstacle was absent altogether

Exercise 2 & 3

These exercise were about pick-&-place and path-following respectively. The set-up was the same for the both, and it comprised the same simulator from the previous exercise with the addition of L-shaped platform.

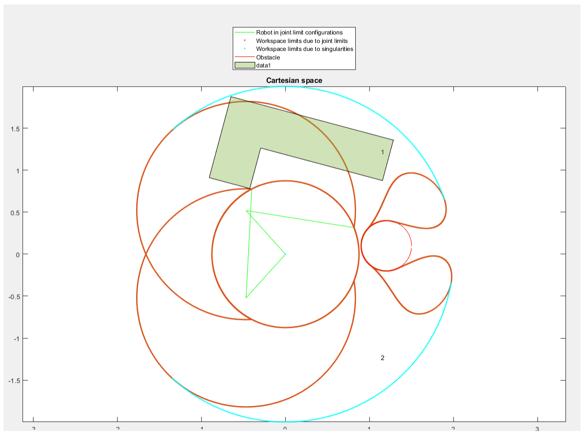
In the second exercise the task required find the most suitable ratio between the two link lengths such that the sum of them was fixed to 2.

Different combination of joint lengths where tested, and the best fitting lengths for the links were identified as 0.8 and 1.2 respectively



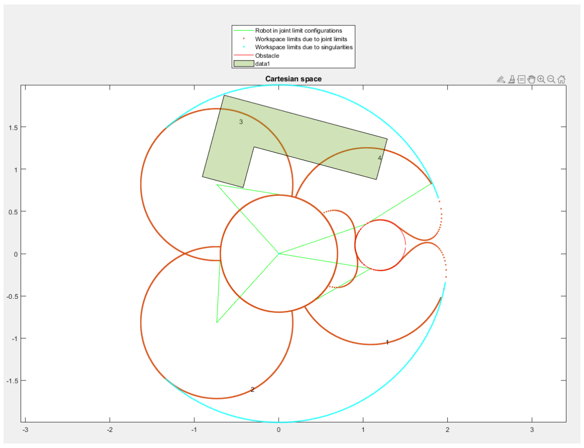
This was the best ratio for both tasks as the workspace surface was maximized guaranteeing a safe connection between the upper part and lower part of the workspace without changing aspect, allowing for flexible placement of the L-shaped table inside the workspace and at the same time the t-connected regions allowed for safe path-follow for the cutting task.

Smaller sizes for link one were also investigated, but they reduced the size of the passage from top to bottom of the t connected regions, and also reduced the flexibility of placement of the L-shaped object, possibly impairing the task of path following



length l1 = 0.7, l2 = 1.3

The last case was about using a longer link 1, in this scenario it is still possible to perform the pick-&-place task, but the different t-connected regions don’t allow for smooth path following tasks



lenght l1 = 1.1 ,l2 = 0.9