FSR HOMEWORK 1 2024

STUDENTE: PAOLO MAISTO
MATRICOLA: P38000191

Exercise 1:

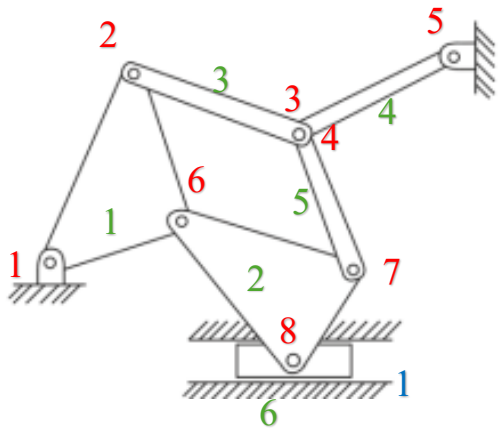
a) Analyzing the Atlas robot's numerous actuators while it's in a stationary position, along with the limitations inherent in this scenario, we can deduce that the system is fully actuated. This is because it has the ability to move instantly in any desired direction, such as the human body; that is, it is able to generate an input for any acceleration. Therefore, the statement ***a*** is **TRUE**.

b) While executing a backflip, we observe a certain coupling between orientation and position in this particular setup. Consequently, attempting to utilize the actuators does not allow for altering the ballistic trajectory of the robot's center of mass during the backflip. This leads us to classify the system in this setup as underactuated, so we can say that the statement ***b*** is **FALSE**.

In essence, despite both scenarios featuring an identical number of actuators, the distinction between full actuation and underactuation in this case derives from changes in the system's degrees of freedom (DOFs) over time. This underscores a property intricately linked to the evolving configurations and constraints.

Exercise 2:

FIRST MECHANISM:



The mechanism has:

- 6 links (green) + ground;
- 8 rotoidal joints (red);
- 1 prismatic joint (blue);

each joint provides a single degree of freedom, for a total of 9 degrees of freedom overall.

As you can see from the image, from the union of the links 3, 4 and 5 (represented in green) there is the overlap of 2 rotoidal joints, that is the joints 3 and 4 represented in red that have the same axis of rotation. Using the Grübler's formula we obtain:

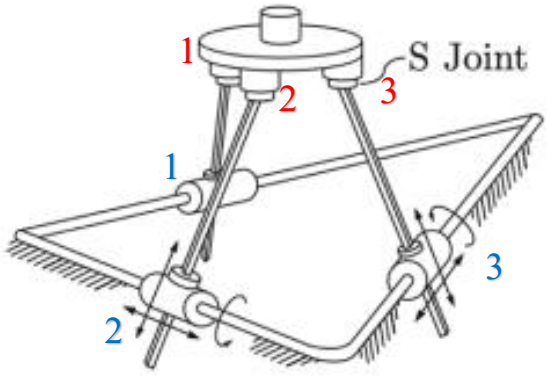
$$DoFs = 3(7 - 1 - 9) + 9 = 0$$

Structural limitations, i.e. constraints, strongly control link movements. As a result, the mechanical result is consistent, since the restrictions prevent any movement. So it is useless to analyze the configuration space topology given the lack of degrees of freedom.
Going to delete the overlap of the two joints, we will get that $N = 6$ and $J = 7$, we will have:

$$DoFs = 3(6 - 1 - 7) + 7 = 1$$

So, the Grübler's formula allows me to say I have 1 DoFs; just to underlining my initial intuition, to see the lower joint as a crank mechanism; so the rotation of a joint allows the translation of the only prismatic joint (in blue in the figure above), and vice versa its translation allows the rotation of the rotoidal joints.

SECOND MECHANISM:



The mechanism has:

- 3 spherical joints, each of which provides three degrees of freedom of rotation (red);
- 3 joints that allow the translation of the rods and the rotation and translation along the respective basic axes (blue);

for a total of 18 degrees of freedom overall.

Using the Grübler's formula we obtain:

$$DoFs = 6(5 - 1 - 6) + 18 = 6$$

So, we need 6 variables to describe all the possible configurations of the mechanism.
However, thanks to the characteristics of the spherical joints, they limit the movement of some joints without changing their position, thus ensuring their independence. As a result, the mechanical result is consistent, allowing different controlled movements.
While analyzing the topology of the configuration space of this mechanism, based on the spherical joints, can be defined as follows:

$$S^2 \times S^2 \times S^2$$

Exercise 3:

a. TRUE

A conventional car, which is based on steering angle and acceleration as inputs, cannot accelerate in a direction perpendicular to the orientation of its wheels, which means that acceleration in all directions is not possible. This limitation comes from the inherent mechanical design of standard vehicles and the specific characteristics of the wheels employed. So, the statement **a** is **TRUE.**

b. TRUE

The Kuka youBot consists of a base that can move in all directions, it features four mecanum wheels, each with its own motor, and considering the manipulator's 5 degrees of freedom (Dof) mounted on it, the entire robot has 12 Dofs. If we focus solely on the chassis and the manipulator without considering the wheels rotations, our model presents 8 Dof. With the 4 motors for the wheels and 5 actuators for each manipulator joint, we can affirm that the system is fully actuated. This is because the chassis can maneuver smoothly in any direction instantaneously, meaning that any acceleration can be achieved through a suitable combination of the system's inputs. So, the statement **b** is **TRUE.**

c. FALSE

The hexarotor is outfitted with six propellers, indicating a dimension of $\tau = 6$, and during flight the system operates with 6 degrees of freedom (Dofs). One might assume that if the number of inputs corresponds to the Dofs, then the system is fully activated, but it is not so. However, in this scenario, the six propellers are all on the same plane, meaning they cannot generate instantaneous acceleration in every direction with a specific input choice. It becomes evident in this setup that only three rotations and one vertical translation along the axis perpendicular to the base frame are feasible. Thus, $rank[G(q, \dot{q})] = 4$, which is less than 6, indicating that the system is underactuated. So, the statement **c** is **FALSE.**

d. TRUE

The 7-Dof KUKA iiwa represents a standard manipulator featuring a fixed base. This robot type boasts individual motors for each joint. Assuming infinite torques, one might conclude that the robot is fully actuated, so capable of generating any form of acceleration through torque input. Typically, with fixed robots reference is made to redundancy, which is influenced by the dimensions of the task space. In a three-dimensional task space, only 6-dof are essential (three for positional and three for orientational control of the end effector), leaving the extra degree of freedom available for secondary tasks such as optimizing manipulability indices or obstacle avoidance by altering the robot's configuration without moving the end effector. However, in scenarios involving task spaces of larger dimensions (e.g., collaborative tasks), the KUKA iiwa will not be redundant. Ultimately, before confirming the veracity of the statement, it's imperative to establish an assumption (as indicated in the text previously, that we are considering a task space of 6 DOF). So, the statement **d** is **TRUE.**

Exercise 4:

The implementation of the RRT method in a software program for a point robot moving in the map from the $q_i = [30\ 125]$ to $q_f = [135\ 400]$ was performed using the matlab software. The code generated for this point is called *"Es_4"*.
The logic with which it is implemented follows the RRT algorithm. As required in the track, a first number of iterations are defined, during which the software generates random points, then derives a line of defined delta length 10, starting from the starting point along the direction towards the random point, then check the collision of the generated line, if there is no collision adds this point to my branch. So we can deduce that if you have as number of iterations 150 we will have a number of generating nodes equal to or less than 150, because many can be eliminated because there is collision. At the end of this iteration check which is the point closest to the goal, to try to join and draw the line with the color magenta, then check if there is collision, if there is no collision, it draws with the red line the path from q_i to q_f; otherwise it means that it does not find the correct path, so repeat this procedure 5 times, number chosen by me and required in the track. So being random the generation of points and defined the number of iterations manually the code repeats the procedure for a maximum of 5 times in case it does not find a solution, the fifth repetition without solution reports error. At this point the maximum number of iterations must be changed manually.

Below some examples are shown, as you can see the starting point is bounded by a red circle, while the goal from a blue circle:

1. But since the generation of random points the result depends on case to case, that is in the following case shown with 50 iterations does not report a solution, but maybe by trying several times at some point you can reach a solution.

```
Es_4.m
25  max_iterations_per_attempt = 100;
26  max_attempts = 5;
27  delta = 10;

Command Window
Attempt 1:
Intersection detected in magenta line. NEXT ATTEMPT!
Attempt 2:
Intersection detected in magenta line. NEXT ATTEMPT!
Attempt 3:
Intersection detected in magenta line. NEXT ATTEMPT!
Attempt 4:
Intersection detected in magenta line. NEXT ATTEMPT!
Attempt 5:
Intersection detected in magenta line. NEXT ATTEMPT!
FAILURE! Optimal path is not found with 5 Attempts.
fx >>
```

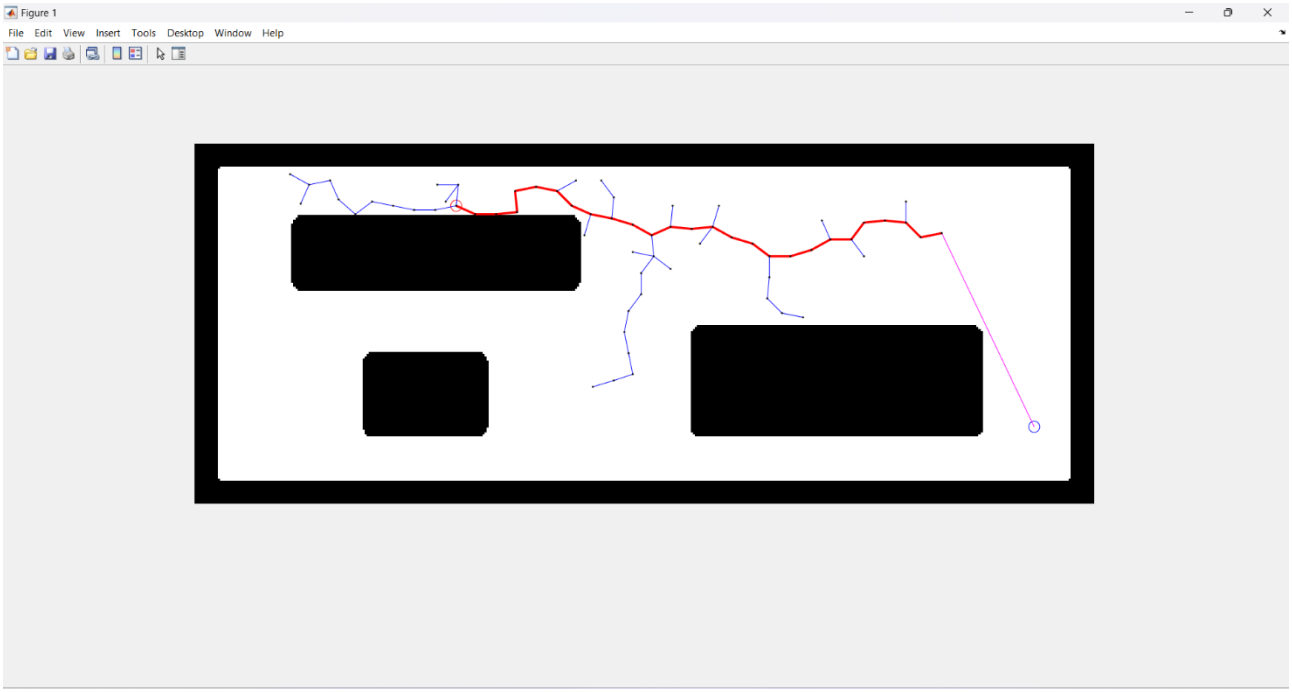
nodes 64x2 double

2. In fact below with the same number of iterations the software finds a solution and it also generates a different number of nodes compared to the previous example:

```
Es_4.m
25  max_iterations_per_attempt = 100;
26  max_attempts = 5;
27  delta = 10;

nodes   65x2 double

Command Window
Attempt 1:
SOLUTION FOUND!
Optimal path found at the Attempt number: 1.
fx >>
```



3. Obviously the larger the number of iterations the easier it will be to find a solution to the first attempts, as it will generate more nodes of our structure. 150 iterations were used below, as we can see with 150 iterations the software generates only 95 nodes and it founds a solution at the second attempt:

```
Es_4.m
25  max_iterations_per_attempt = 150;
26  max_attempts = 5;
27  delta = 10;
```

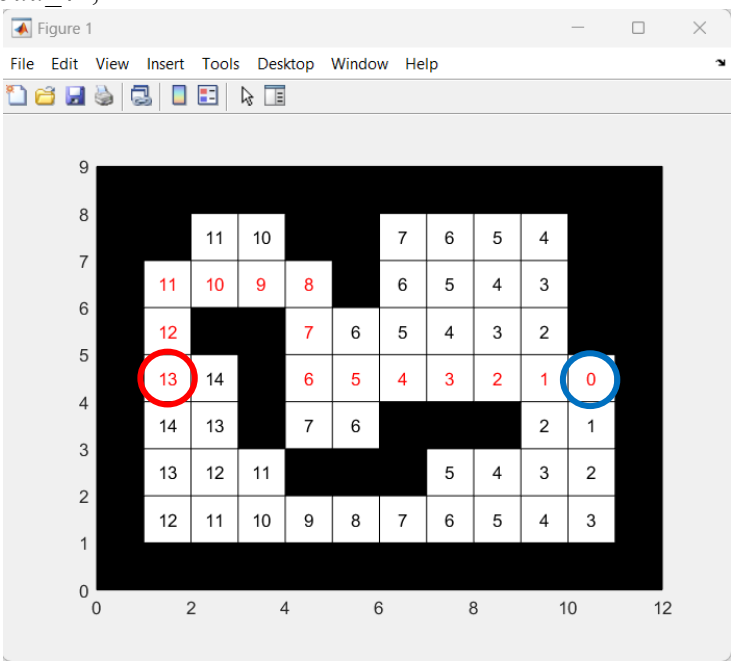
nodes 95x2 double

```
Command Window
Attempt 1:
Intersection detected in magenta line. NEXT ATTEMPT!
Attempt 2:
SOLUTION FOUND!
Optimal path found at the Attempt number: 2.
fx >>
```

Exercise 5:

The motion planning method based on the numerical navigation function is implemented using matlab software. Being my odd serial number (P38000191) the table on the left has been used.
The logic behind the code is as follows: the matrix G is initialized and then filled through a while, in which the final goal is defined and the obstacles present with the number 99 creates the matrix by numbering the remaining cells based on the number of adjacent cells, and then print it on video, also added a kind of frame created by all 99 numbers in the G matrix.
Then defined the starting and final point through always a while recreates the path to reach the goal; and then print on the screen the table of nodes visited, that is, assigns 0 to cells not seen and therefore correspond to obstacles, 1 to the visited cells and 10 to the visited cells that are part of the path, and then print the cell sequence on screen.
Two codes have been created based on the number of adjacent cells used; in both cases the goal is circled in blue, while in red the start cell, similarly the numbers of the cells that are part of the path are written in red, as shown in the following figures:

- In the following picture is represented the case with 4 adjacent cells, whose file of the code is called "*Es_5_odd_4*";

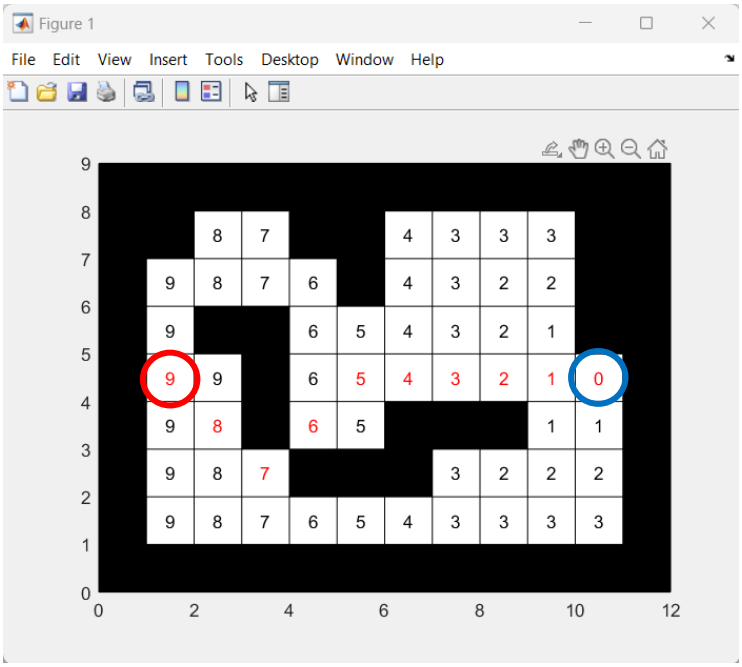



```
Matrix G:
    99    99    99    99    99    99    99    99    99    99    99    99
    99    99    11    10    99    99     7     6     5     4    99    99
    99    11    10     9     8    99     6     5     4     3    99    99
    99    12    99    99     7     6     5     4     3     2    99    99
    99    13    14    99     6     5     4     3     2     1     0    99
    99    14    13    99     7     6    99    99    99     2     1    99
    99    13    12    11    99    99    99     5     4     3     2    99
    99    12    11    10     9     8     7     6     5     4     3    99
    99    99    99    99    99    99    99    99    99    99    99    99

Matrix of visited cells:
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     1     1     0     0     1     1     1     1     0     0
     0    10    10    10    10     0     1     1     1     1     0     0
     0    10     0     0    10     1     1     1     1     1     0     0
     0    10     0     0    10    10    10    10    10    10    10     0
     0     0     1     0     1     1     0     0     0     1     1     0
     0     1     1     1     0     0     0     1     1     1     1     0
     0     1     1     1     1     1     1     1     1     1     1     0
     0     0     0     0     0     0     0     0     0     0     0     0
```

```
Path from q_g to q_s:
     5    11
     5    10
     5     9
     5     8
     5     7
     5     6
     5     5
     4     5
     3     5
     3     4
     3     3
     3     2
     4     2
     5     2
```

- In the following picture is represented the case with 8 adjacent cells, whose code is called "*Es_5_odd_8*";




```
Matrix G:
    99    99    99    99    99    99    99    99    99    99    99    99
    99    99     8     7    99    99     4     3     3     3    99    99
    99     9     8     7     6    99     4     3     2     2    99    99
    99     9    99    99     6     5     4     3     2     1    99    99
    99     9     9    99     6     5     4     3     2     1     0    99
    99     9     8    99     6     5    99    99    99     1     1    99
    99     9     8     7    99    99    99     3     2     2     2    99
    99     9     8     7     6     5     4     3     3     3     3    99
    99    99    99    99    99    99    99    99    99    99    99    99

Matrix of visited cells:
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     1     1     0     0     1     1     1     1     0     0
     0     1     1     1     1     0     1     1     1     1     0     0
     0     1     0     0     1     1     1     1     1     1     0     0
     0    10     0     0     1    10    10    10    10    10    10     0
     0     0    10     0    10     1     0     0     0     1     1     0
     0     0     1    10     0     0     0     1     1     1     1     0
     0     0     1     1     1     1     1     1     1     1     1     0
     0     0     0     0     0     0     0     0     0     0     0     0
```

```
Path from q_g to q_s:
     5    11
     5    10
     5     9
     5     8
     5     7
     5     6
     6     5
     7     4
     6     3
     5     2
```

As you can see from the two pictures emphasize the difference of the two cases. First of all in the creation of the G matrix it is noticed that the maximum number reached in the case of 8 adjacent cells (9) is lower than in the case of 4 adjacent cells (14), which in turn leads me to a considerable difference in the path, in which the one with 8 adjacent cells the path is much shorter, having in some cases the possibility to move diagonally.