Homework 1

Student: Manzoni Antonio P38000234

Exercise 1

Since ATLAS is a real robot surely will have a physical limitation of its motors and so by definition will always be underactuated, so, in first, it is necessary to make the assumption that we have unbounded torques for the motors. ATLAS is a human-like robot with many motors, so we can assume that its mechanical behavior is identical to the human one.

First case: standing

While standing, we can say that the robot is **fully actuated** because we can provide any acceleration in any direction we want. So, the sentence A is true.

Second case: backflip

When the robot is doing backflip, there is no input that allows the robot to change the direction that is imposed when it detached the ground during the motion, so we can say that the robot is **underactuated**. So, the sentence B is false.

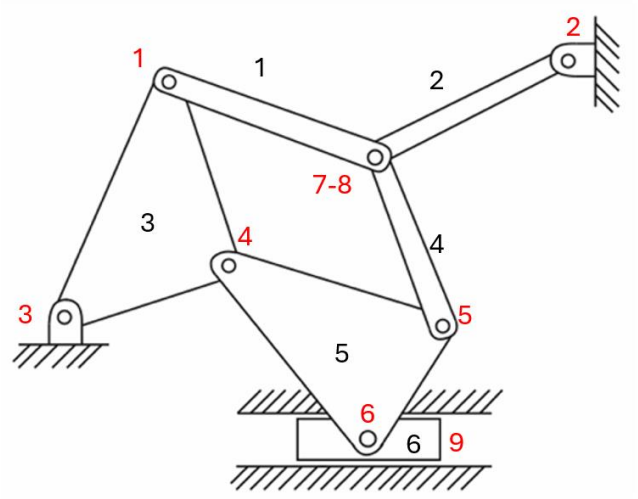
At the end, we can say that the same robot in two different configurations could change its property of underactuation or fully actuation depending on the dynamics and on the constraints that the robot has with the environment.

Exercise 2

In this exercise it is requested to determine the number of the degrees of freedom for each mechanism with the Gruebler's formula:

$$\boldsymbol{D.O.F.} = \text{m (N-1-J)} + \sum_{i=1}^{J} \boldsymbol{f_i}$$

First mechanism



Links	Joints
1	1 = Revolute
2	2 = Revolute
3	3 = Revolute
4	4 = Revolute
5	5 = Revolute
6	6 = Revolute
7 = Ground	7 = Revolute
	8 = Revolute
	9 = Prismatic

In this case m is set equal to 3 because the mechanism is planar, N (number of links) is equal to 7 (6 arms and the ground), J (number of joints) is equal to 9. Each joint provides 1 degree of freedom, so f_i is equal to 9. At first sight we probably could think that the mechanism has 1 D.O.F. due to the configuration of the joints 6 and 9 that reminds to a crank mechanism, but if we solve the Grubler's equation, we obtain:

$$3\,(7\text{-}1\text{-}9) + 9 = 0$$

So, the mechanism cannot move freely in the plane. We could think to eliminate the joint 3 (revolute) and what we obtain from the formula is:

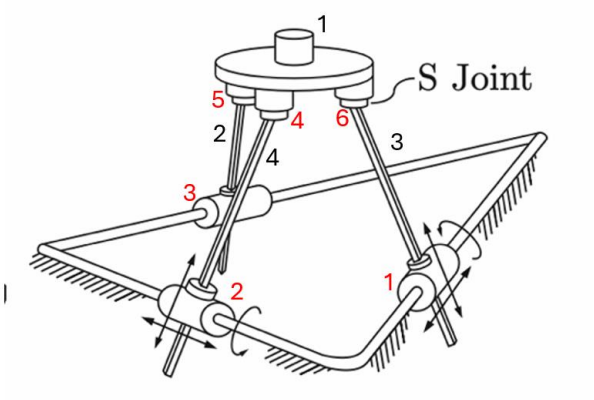
$$3\,(7\text{-}1\text{-}8) + 8 = 2$$

We know that the addition of a revolute joint eliminates 2 D.O.F. and so we obtain a mechanism with 0 D.O.F. Another way of thinking is to eliminate a link with 2 joints and obtain a crank mechanism with 1 D.O.F. (for example we choose to eliminate link 4 and joints 8 and 5)

$$3\,(6\text{-}1\text{-}7) + 7 = 1$$

Due the fact that we have 0 D.O.F. we can say that is meaningless to talk about Configuration Space.

Second mechanism



Links	Joints
1 = plate	1 = 3 D.O.F joint
2 = arm	2 = 3 D.O.F joint
3 = arm	3 = 3 D.O.F joint
4 = arm	4 = Spherical

In this mechanism m is set equal to 6 because we are in R^3 space, N is equal to 5, J is equal to 6. In this case we have f_i equal to 18 because we have three spherical joints and three joints that provide both 3 D.O.F.

If we solve the Grubler's equation, we have:

$$6\,(5\text{-}1\text{-}6) + 18 = 6$$

The mechanism is similar to a Stewart platform where the devices placed on the top plate can be moved in the six degrees of freedom in which it is possible for a freely-suspended body to move: three linear movements x, y, z (lateral, longitudinal, and vertical), and the three rotations (pitch, roll, and yaw). All the motions are produced by a combination of movements of multiple actuators.

Due the fact that the platform needs only 6 degrees of freedom, I can choose to actuate the three joints at the fixed base using only 2 D.O.F. per joint, for example I can choose to use the revolute motion and the translation along the rod. So, for the Configuration space I can consider:

$$\boldsymbol{R^3 x\, T^3}\ \textbf{(or } \boldsymbol{R^3 x\, I^3}\textbf{: since the rotations are limited)}$$

Exercise 3

A=TRUE.

The car has 7 D.O.F., but for control purposes we can consider only the D.O.F. that are relevant for the mathematical model. We choose to not care about the wheels' positions so my D.O.F. that are relevant for the model are only 3 (2 positions and the rotational angle of the chassis). The inputs of the system are 2 (steering angle and throttle), so the car will always have a number of inputs that is less than the number of D.O.F. So, we have $\dim[\tau] = 2$ and $rank[G(q, \dot{q},)] = 3$. We can say also by intuition that a car cannot accelerate in a direction perpendicular to the direction the wheels are facing, so it is **underactuated**.

B=TRUE.

The Kuka youBot is a mobile robot equipped with 4 mecanum wheels that provide an omnidirectional base and with a 5 D.O.F. manipulator mounted on it, so we have 12 D.O.F. in total. As we did already for the car, we can consider not all the degrees of freedom of the robot, and so we choose to consider only the D.O.F. of the chassis and the D.O.F. of the manipulator without counting the rotation of the wheels, in order to have in total 8 D.O.F. for our model. Thanks to the 4 motors of the wheels and 5 actuators for each joint of the manipulator we can affirm that the system is **fully actuated** since the chassis can easily move in any direction instantaneously. By intuition we can say that, thanks to the omnidirectional base, the youBot can overcome the issue of the car and move also in a direction perpendicular to the direction the wheels are facing.

C=FALSE.

Underactuation surely occurs when we have a number of inputs that is less than the number of degrees of freedom, but there could be cases where I have a number of inputs that is more or equal than D.O.F. and still be underactuated and this is one example. The hexarotor is equipped with six propellers displaced in the same plane and while flying the system has 6 D.O.F. Due the fact that all the propellers are co-planar the hexarotor can move laterally only if inclined (this happens if changing the velocity between the propellers), so while the system is perfectly horizontal to the ground (this happens when all the propellers have the same speed) it is not possible to find any configuration of the propellers that allows to fly horizontally while keeping the platform parallel to the ground, because in this configuration only three rotations and one vertical translation along the axis orthogonal to the base frame are admissible. So, in conclusion, we have $\dim[\tau] = 6$ and $rank[G(q, \dot{q},)] = 4$, it means that the hexarotor is **underactuated**.

D = It depends on task space

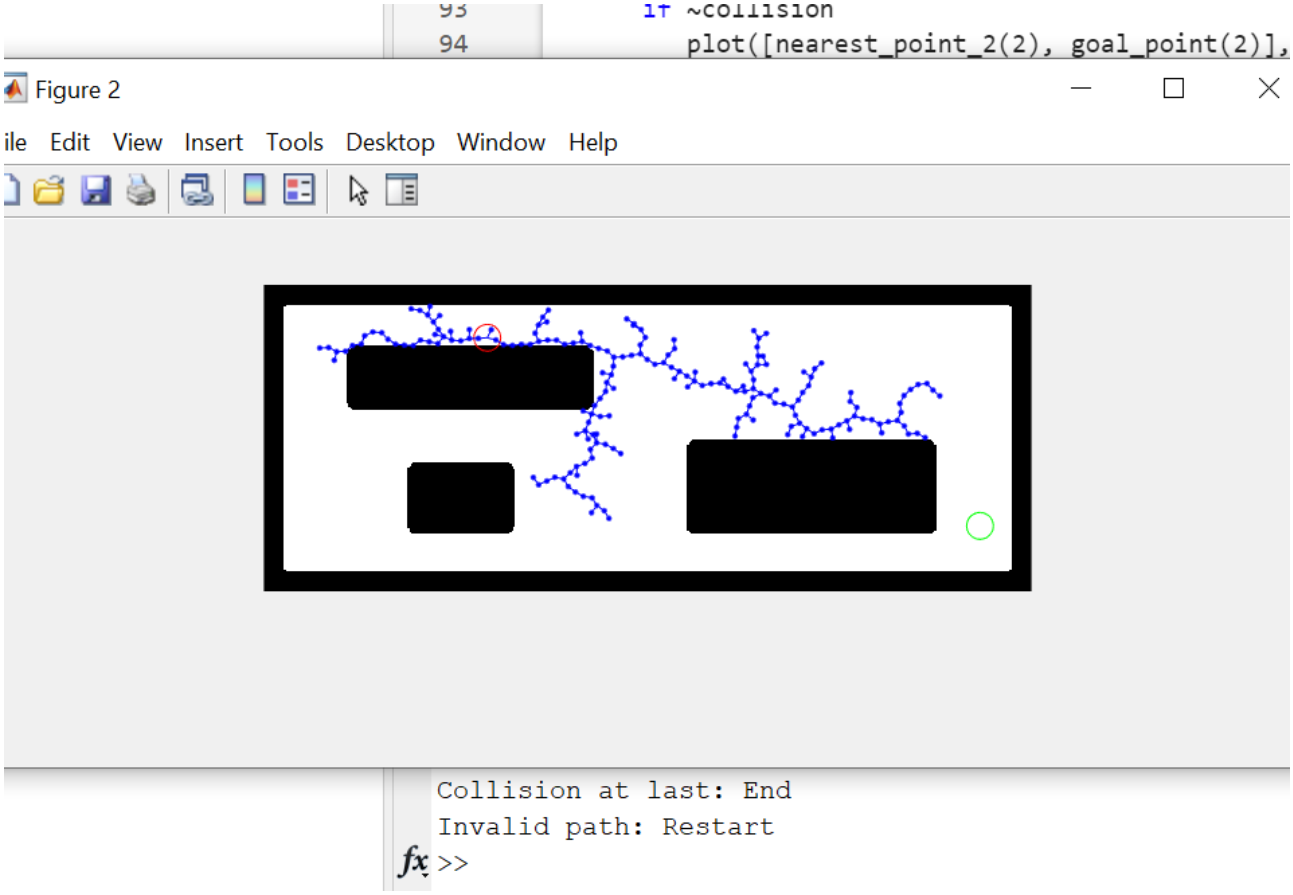
Redundancy is something related to the kinematics of the system and to the chosen task. We know that 7-Dof KUKA iiwa is a manipulator with a fixed base with 7 D.O.F., like the human arm. If we consider a tridimensional task space, only 6 D.O.F. are needed (three for the position and three for orientation of the end effector) and the additional D.O.F might be used to accomplish some secondary tasks, by changing the configuration of the robot, without moving the end effector. However, if we consider a task space with a larger dimension then the KUKA iiwa will not be redundant. In the end to give an answer to this question we need to know first the task that the robot must accomplish.

Exercise 4

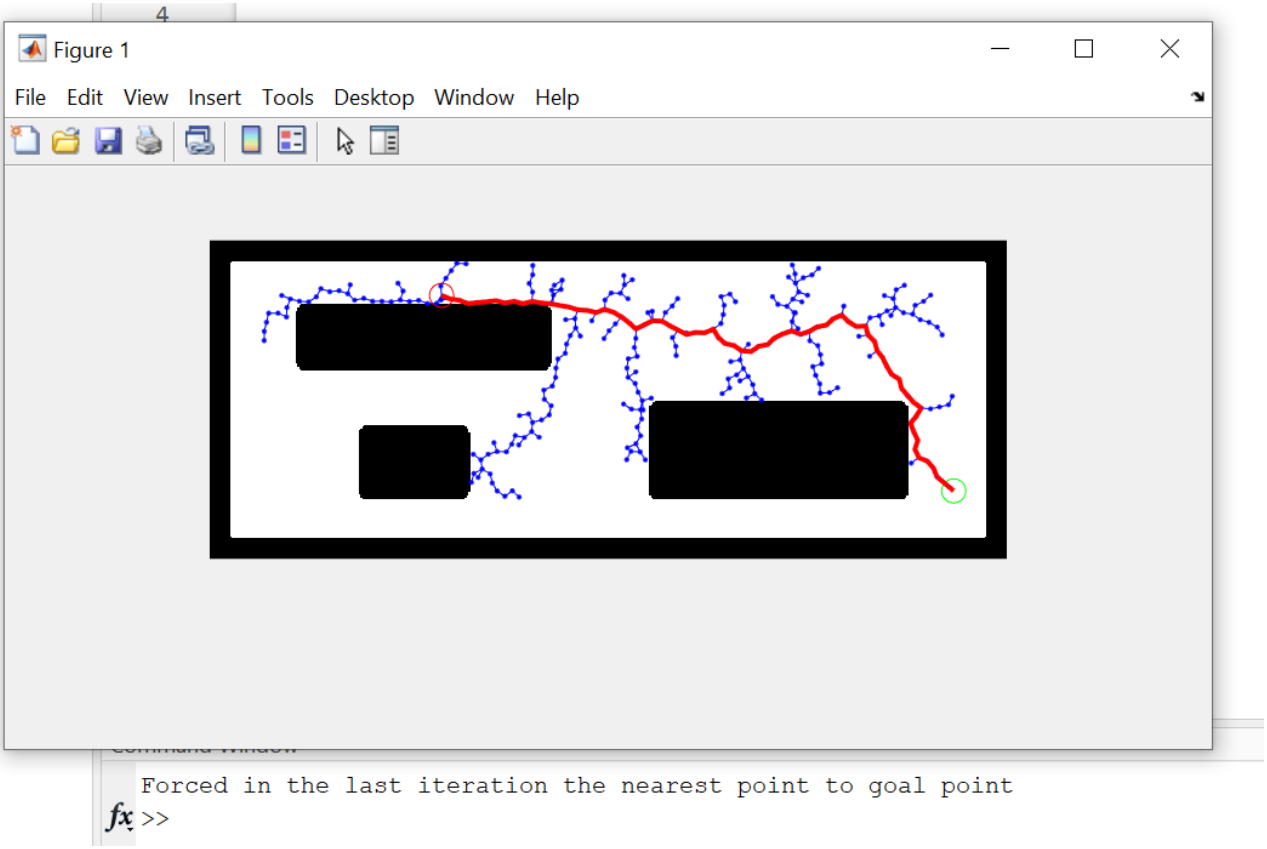
The observations are directed towards the outcome of various executions of the program. Clearly, it is not always possible to reach the goal point due to the presence of obstacles.

The program checks at the last iteration whether it is possible to connect the goal point with the nearest point to it. In all the elaborations with positive outcomes obtained, we have always found ourselves in this scenario.

The following picture shows what happens in a scenario with collisions during the path:



This other picture shows the scenario when the goal point is correctly reached:



Exercise 5

The program considers cells with a value of 100 as obstacles, while in the initial phase, the remaining cells all have a value of 0. The adopted solution involves establishing a relationship between the various nodes in a parent-child manner, meaning that for each parent cell, all child cells are tracked, and obviously, the relationship can also be interpreted in reverse. With this approach, all child cells will have their value incremented by one unit from the value contained in the parent cell. The final step of the program involves tracing the path starting from the destination cell and backtracking, loading a vector with the positions of adjacent cells with decreasing values until reaching the starting point.

By varying the adjacent cells from 8 to 4, it is ensured that no parent cell borders another parent cell with an equal value.

The following picture shows the final result considering 8 adjacent cells:

Command Window

```
map:
     0   100   100    10    10    10    11    12    13    14
     1     1   100     9     9    10    11   100   100    14
     2     2   100     8   100    10    11   100    13    13
     3     3   100     7   100    10    10   100    12    12
     4     4   100     6   100   100     9   100    11    12
     5     5     5     6     7     8     9    10    11    12
     6     6     6     6     7     8   100    10    11    12

Positions of cells with decreasing values:
     1     8
     1     7
     1     6
     2     5
     3     4
     4     4
     5     4
     6     3
     5     2
     4     2
     3     2
     2     2
     1     1

Path
    11   100   100     0     0    11    11    11     0     0
     0    11   100     0    11     0     0   100   100     0
     0    11   100    11   100     0     0   100     0     0
     0    11   100    11   100     0     0   100     0     0
     0    11   100    11   100   100     0   100     0     0
     0     0    11     0     0     0     0     0     0     0
     0     0     0     0     0     0   100     0     0     0

fx >>
```

The following picture shows the final result considering 4 adjacent cells:

Command Window

```
map:
     0   100   100    13    14    15    16    17    18    19
     1     2   100    12    13    14    15   100   100    18
     2     3   100    11   100    15    14   100    16    17
     3     4   100    10   100    14    13   100    15    16
     4     5   100     9   100   100    12   100    14    15
     5     6     7     8     9    10    11    12    13    14
     6     7     8     9    10    11   100    13    14    15

Positions of cells with decreasing values:
     1     8
     1     7
     2     7
     3     7
     4     7
     5     7
     6     7
     6     6
     6     5
     6     4
     6     3
     6     2
     5     2
     4     2
     3     2
     2     2
     2     1
     1     1

Path
    11   100   100     0     0     0    11    11     0     0
    11    11   100     0     0     0    11   100   100     0
     0    11   100     0   100     0    11   100     0     0
     0    11   100     0   100     0    11   100     0     0
     0    11   100     0   100   100    11   100     0     0
     0    11    11    11    11    11    11     0     0     0
     0     0     0     0     0     0   100     0     0     0
```