**FSR HOMEWORK 2 2024**

**STUDENTE:** PAOLO MAISTO

**MATRICOLA:** P38000191

**Exercise 1:**

1. Considering the following distribution:

Immediately we can say that this distribution is **involutive** because any one-dimensional distribution is involutive and the zero vector belongs to any distribution by default.

Now, we aim to determine the annihilator for the distribution ​, which involves finding a set of covectors such that . It's worth noting that if and , then we can infer that . So

Imposing for the independent variables: . We achieve:

At this stage, we need to demonstrate that these two covectors are linearly independent. To achieve this, we construct the following matrix. Subsequently, we verify that this matrix is full rank (rank = 2) by considering a minor matrix of size . We must ensure that the determinant of this minor is non-zero:

At the end, if we select parameters to satisfy the previous relation, then is the annihilator for the distribution ​.

1. Considering the following distribution:

First of all I construct the matrix F:

Considering the first matrix, the determinant is therefore the dimension of this distribution is , thus the distribution is not singular.

The distribution has dimension , now I proceed to compute the lie bracket:

The next step is verify that lie bracket belongs to span genereted by vectors . To accomplish this, we construct the following  matrix, where its columns are the vectors

So this matrix is always full rank:

Thus, we dedude that the vector is not a linear cobination of and . So we can come to conclusion that distribution is **not involutive**.

Now, we aim to determine the annihilator for the distribution ​, which involves finding a set of covectors such that . It's worth noting that if and , then we can infer that . So:

Imposing for the independent variables that . Thus, we obtain the following annihilator:

It's worth noting that to avoid considering the banal solution , it's necessary to impose the constraint .

1. Considering the following distribution:

First of all I construct the matrix F:

Considering the last matrix, the determinant is therefore the and the dimension of this distribution is , thus the distribution is not singular*.*

The distribution has dimension , now I proceed to compute the lie bracket:

The next step is verify that lie bracket belongs to span genereted by vectors . To accomplish this, we construct the following matrix, where its columns are the vectors

So, I can deduce that the matrix is always full rank by the fact that:

Thus, we dedude that the vector is not a linear cobination of and . So we can come to conclusion that distribution is **not involutive**.

Now, we aim to determine the annihilator for the distribution ​, which involves finding a set of covectors such that . It's worth noting that if and , then we can infer that . So:

Imposing for the independent variables that . Thus, we obtain the following annihilator:

It's worth noting that to avoid considering the banal solution , it's necessary to impose the constraint .

**Exercise 2:**

The omnidirectional mobile robot we're looking at has these generalized coordinates:

Where:

* are centre coordinate of robot;
* is orientation of robot;
* are angular position each wheel;

And I define:

* is radius of the wheels;
* is distance between wheels and centre of robot.

To establish the kinematic model for this robot, we must determine the matrix such that

. To achieve this, we employ the MATLAB command null: .

is a matrix with three rows and six columns, therefore is a matrix with six rows and three columns. The kinematic model is:

Defining each column of like The accessibility distribution is constructed with these three vectors, where each lie bracket is comprised of , , and .

The Jacobian of these vectors needs to be calculated first for this reason:

Immagine che contiene testo, Carattere, schermata

Descrizione generata automaticamente

And then, we can compute the lie bracket as:

Immagine che contiene testo, Carattere, bianco, tipografia

Descrizione generata automaticamente

Notice that if we construct a matrix whose columns are vectors of , , , , , and , its rank is five. Therefore, we can infer that one vector is linearly dependent. By removing this vector, the accessibility distribution has a dimension . Consequently, we can say:

The system is still **nonholonomic** and have only partially integrable constraints (as it is printed on the screen by the code in matlab, the script is called *‘Es\_2’*).

**Exercise 3:**

We employ a cubic Cartesian polynomial to execute a path planning algorithm, aiming to guide the unicycle from configuration to the random final point (at each run the final point coordinates are printed on the screen as shown in the following image). The file called ‘Es\_3\_1’ implements only the cubic Cartesian polynomial, the results are shown in the following graphics. It is evident that the unicycle goes from the origin to the final point which has the following coordinates with an orientation of.

Immagine che contiene testo, diagramma, schermata, linea

Descrizione generata automaticamente

However, if we encounter velocity constraints as follows, it becomes necessary to implement a specific time law to adhere to these constraints.

The ‘Es\_3\_2’ file implements the second part of this exercise. In which I define a third order polynomial time law with the following conditions:

* Start instant and final instant is made to choose through terminal, to every iteration if the is too small for the constraints to be respected, it returns error with the graphs that emphasize that the constraints are exceeded, for then to ask to insert a greater always through terminal;
* Start position and random final position;
* Start velocity and final velocity ;

Where:

As shown in the following pictures initially I put as input a time interval of but velocity bounds are not respected.

Immagine che contiene testo, linea, diagramma, Diagramma

Descrizione generata automaticamente

So I inserted a higher final time and consequently velocity constraints are respected.

Immagine che contiene testo, linea, Diagramma, diagramma

Descrizione generata automaticamente

The following terminal first shows the coordinates of the final point and then shows the requests for the final time. Obviously the end point of the two cases (‘Es\_3\_1’ and ‘Es\_3\_2’) differs because they are different simulations and in both simulations the end point is generated randomly.

Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente

In the same way as in the previous case, the trajectory and the evolution of theta are reported as shown in the following figure.

Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

**Exercise 4:**

In order to implement input/output linearization control we utilize the trajectory from the previous exercise as a reference. This approach allows us to control the position of the unicycle. The method relies on feedback from the following outputs, representing the Cartesian coordinates of a point B situated along the sagittal axis of the unicycle at a distance . I have opted to set to position it forward.

In the Simulink file’Es\_4’, I've implemented two blocks to compute these outputs. The first block calculates the reference signals ​ and ​, which correspond to the trajectory generated in the previous exercise. The second block computes ​ and ​ for the unicycle. Subsequently, the controller calculates ​ and ​ using predefined formulas, where I defined the gain values . Finally, the last block computes the heading and angular velocity.

First of all, we observe the simulink results in the first image, while in the second are reported the results obtained by simulating the matlab file from which then the trajectory is resumed as a reference, so we can say that the trajectory is executed correctly (also because it was done in such a way that when the simulink file starts running, it automatically runs first the matlab file and then continue only after it has taken the data interessed).

Immagine che contiene linea, Diagramma, diagramma, pendio

Descrizione generata automaticamenteImmagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

In fact as shown by the following images all variables correctly follow the respective references:

Immagine che contiene linea, Diagramma, diagramma, testo

Descrizione generata automaticamenteImmagine che contiene linea, Diagramma, testo, Parallelo

Descrizione generata automaticamenteImmagine che contiene linea, Diagramma, diagramma

Descrizione generata automaticamente

Obviously being each simulation on its own, because each of them is generated a different end point; the final point of exercise 4 will be once again different from the exercise 3\_2 which will be once again different from the exercise 3\_1.

Before proceeding with the errors, the graphs concerning heading and angular velocity are also reported, to underline that the constraints imposed in the previous exercise are respected. As before, on the left are the results obtained by simulink, while on the right are the matlab file from which the Immagine che contiene Diagramma, linea, diagramma, pendio

Descrizione generata automaticamentetrajectory reference is taken:

Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

At the end the errors , shown in the following pictures, are of the order , therefore we can define them acceptable.

Immagine che contiene linea, Diagramma, diagramma, Parallelo

Descrizione generata automaticamenteImmagine che contiene linea, Diagramma, diagramma, testo

Descrizione generata automaticamenteImmagine che contiene linea, Diagramma, diagramma, testo

Descrizione generata automaticamenteImmagine che contiene linea, Diagramma, diagramma, Parallelo

Descrizione generata automaticamenteImmagine che contiene linea, Diagramma, diagramma, testo

Descrizione generata automaticamente

Analyzing the whole, in theory, this technique controls only the position but not the orientation of the unicycle. However, in practice, to accurately follow the desired trajectory, the unicycle must orient itself in a manner similar to ​. Despite this observation, if we examine the error on , we find that it consistently remains on the order of , similar to the one for variables and, but slightly larger. Specifically, the maximum errors on and are and respectively, while the maximum error in is .

**Exercise 5:**

Our goal is to maneuver our unicycle robot from the to in order to implement the unicycle posture regulator based on polar coordinates (using Simulink software). The initial step involves initializing the Unicycle Kinematic Model block with ​. Subsequently, I have implemented a MATLAB function to execute the 2nd Order Runge-Kutta Approximation, with a chosen simple time of . In order to design the following feedback controller:

Immagine che contiene linea, diagramma, schermata, Diagramma

Descrizione generata automaticamenteImmagine che contiene linea, schermata, Diagramma, diagramma

Descrizione generata automaticamenteI have also deployed a MATLAB function to implement Cartesian/Polar Conversion. Subsequently, by defining the gains , we can observe the following plots for . As evident from the plots, all variables reach 0 in a simultation of 15 seconds, indicating the attainment of the desired posture.

Immagine che contiene linea, Diagramma, schermata, diagramma

Descrizione generata automaticamenteTo confirm the success of the adjustments, we assess the errors on . At the end, for accurate error evaluation, the norm of the error was computed for each variable (using the MATLAB function ‘norm()’). As we can see from the following graphs, all errors converge to zero.

Immagine che contiene schermata, linea, testo, Diagramma

Descrizione generata automaticamente

Immagine che contiene linea, testo, Diagramma, schermata

Descrizione generata automaticamente

Immagine che contiene linea, Diagramma, schermata, diagramma

Descrizione generata automaticamente

Immagine che contiene testo, schermata, linea, Rettangolo

Descrizione generata automaticamente

Immagine che contiene linea, Diagramma, testo, schermata

Descrizione generata automaticamente

Immagine che contiene linea, Diagramma, diagramma, testo

Descrizione generata automaticamente