**Homework 2**

**Student: Manzoni Antonio P38000234**

**Exercise 1**

The exercise requires to state if the following distributions are involutive or not and, if it is possible, to find the annihilator.

A. , 𝑈 ∈ ℝ𝟑

This distribution is one dimensional and so is **involutive** since = 0 and the zero vector belongs to any distribution by default.

The next step is to find the annihilator for this distribution. We know that , where n is equal to 3 and is equal to 1, so is equal to 2.

Now we need to define a set of co-vectors such that .

**= =**

There is one equation, and we must find two linear independent co-vectors field. I can impose for instance these equalities and , where a1 could be any value. The resultant co-vector will be .

Another solution is choosing and , leaving freely chosen and obtain .

In the end the co-distribution is:

**B.**  , 𝑈 ∈ ℝ𝟑

First thing to do is to find the dimension of the distribution, so I construct the matrix and evaluate the rank.

= If I select the 2x2 matrix the determinant is equal to -4 and so:

, and the dimension of the distribution is equal to 2, it means that the distribution is **not singular**.

The next step is to calculate the lie bracket: :

=

If the lie bracket belongs to the span generated by the vectors it is possible to say that the distribution is involutive. To prove this, we build the following 3x3 matrix which has as columns the vectors :

The determinant of this matrix can be evaluated using the Sarrus’s rule and it is:

It means that the rank of F1(x) is equal to 3, so and so we can state that the distribution is **not involutive**.

The last step is to find the annihilator of the distribution. Firstly, we compute the dimension of the annihilator:

= n – dim () = 3 – 2 = 1

Now we need a set of co-vectors such that , so:

**= =**

If we chose we have:

And the co-distribution is:

**C.**  , 𝑈 ∈ ℝ𝟑

First thing to do is to find the dimension of the distribution, so I construct the matrix and evaluate the rank.

= If I select the 2x2 matrix the determinant is equal to -1 and so:

, and the dimension of the distribution is equal to 2, it means that the distribution is not singular.

The next step is to calculate the lie bracket: :

=

If the lie bracket belongs to the span generated by the vectors it is possible to say that the distribution is involutive. To prove this, we build the following 3x3 matrix which has as columns the vectors :

The determinant of this matrix can be evaluated using the Sarrus’s rule and it is:

If the matrix F1(x) is full rank (rank(F1(x)) = 3), so and the distribution is **not involutive**. Otherwhise, if the rank of F1 will be equal to 2 and so the distribution is **involutive**.

The last step is to find the annihilator of the distribution. Firstly, we compute the dimension of the annihilator:

*= n – dim () = 3 – 2 = 1*

Now we need a set of co-vectors such that , so:

**= =**

If we chose we have:

And the co-distribution is:

**Exercise 2**

In this exercise we have an omnidirectional mobile robot, provided with 3 Mecanum wheels, which has as vector of generalized coordinates the following one:

Where:

* and are the Cartesian coordinates of the center of the robot.
* is the vehicle orientation.
* represent the angle of rotation of each wheel around its axis.

And it is subject to the following Pfaffian constraints:

Immagine che contiene testo, Carattere, bianco, linea

Descrizione generata automaticamente

Where is the distance between the wheels and the center of the robot, which I’ve set equal to 1.2, and is the radius of the wheels, which I’ve chosen equal to 0.5.

Now we need to compute the kinematic model of this robot and show if the system is holonomic or not. To accomplish this request is necessary to calculate the G matrix which is a matrix that enters in the Null of , so and has six rows and three columns.

After that we can compute the kinematic model of the robot:

Where is the vector of the inputs and 𝑢 ∈ .

The next step is to build the accessibility distribution which is the distribution generated by the vector fields (which are the three columns of the G matrix) and all the Lie brackets that can be generated by these vector fields. So, we compute the partial derivatives of these vectors using the command jacobian, and then the Lie brackets.

We can notice that the accessibility distribution is a 6x6 matrix but if we compute the rank of this matrix we obtain 5, so there is a vector that is linearly dependent and so 𝑣 = dim(∆𝑎) = 5.

At the end we have: , and

So, we are in the case of and finally we can affirm that the system is **nonholonomic** and **has only partially integrable constraints**.

**Exercise 3**

The exercise requires to implement via software a path planning algorithm based on a cubic Cartesian polynomial, which takes the unicycle from an initial configuration to a random final configuration such that . The trajectory must satisfy the following constraints: and . I’ll report here the plots of one of the simulations that has been carried out.

It is possible to observe that in the simulation the robot reaches the random final configuration starting from the initial one with a final orientation almost equal to 2π.

Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

As we said to define the trajectory we chose a cubic polynomial: where the four coefficients are computed as follows:

= 1

Solving from the third equation and substituting inside the fourth one we obtain: and . Where is a parameter that is set as input in the program. Initially I’ve set and as we can see the bound on the angular velocity is not satisfied.

Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamente

Then, if the previous case occurs, the program asks to increase the and, as we can see from the following picture, if I set the value equal to 100, for instance, the angular velocity overcomes the previous issue and remains confined in its bound.

Immagine che contiene diagramma, testo, Diagramma, linea

Descrizione generata automaticamente

**Exercise 4**

The exercise requires to implement an input/output linearization control approach to control the unicycle’s position, given the trajectory of the previous exercise. As first thing we define a point B which is located along the sagittal axis of the unicycle at a distance which I assume equal to 0.6 (located ahead).

We choose outputs that represent the coordinates of the point B and (x,y) which are the coordinates of the center of the wheel in such way:

The controller is implemented in Simulink. In one block I calculate the reference  and which is the trajectory from the previous exercise. In another block I evaluate and . The next block calculates 𝑢1 and 𝑢2 using formulas present on the slides, where I choose the gains 𝑘1 = 𝑘2 = 5. The last block calculates heading and angular velocity. As we can see the trajectory is followed correctly by the robot.

Immagine che contiene linea, Diagramma, schermata, diagramma

Descrizione generata automaticamente

And also the orientation coincides with the one of the previous exercise.

Immagine che contiene testo, linea, Diagramma, diagramma

Descrizione generata automaticamente

To make sure that the controller works as it should, I plot the errors with respect to the desired values.

Immagine che contiene testo, Diagramma, linea, diagramma

Descrizione generata automaticamente

Immagine che contiene linea, Diagramma, diagramma, testo

Descrizione generata automaticamente

Immagine che contiene Diagramma, diagramma, linea, testo

Descrizione generata automaticamente

Immagine che contiene testo, Diagramma, linea, diagramma

Descrizione generata automaticamente

Immagine che contiene Diagramma, linea, diagramma, testo

Descrizione generata automaticamente

As we can seethe errors are of the order of so we can affirm that the controller works well.

An interesting feature is that the controller left the orientation uncontrolled, but despite that the error on theta is of the order of which we can assume to be small.

At last, I plot the heading and the angular velocity and as we can see they still maintain their bounds.

Immagine che contiene testo, Diagramma, linea, diagramma

Descrizione generata automaticamente

**Exercise 5**

In this exercise our goal is to implement a posture regulator for the unicycle based on polar coordinates, with the state feedback computed through the Runge-Kutta odometric localization method.

The initial configuration is and the final configuration is . As first thing I discretize . Then I instantiate a block function which implement 2𝑛𝑑 Order Runge-Kutta Approximation where I choose a simple time equal to 𝑇𝑠 = 0.001 𝑠. The controller works with polar coordinates, so we need a block to convert the cartesian coordinates into the new ones, that will be the input of our regulator:

Immagine che contiene testo, Carattere, calligrafia, bianco

Descrizione generata automaticamente

Where I’ve chosen as gains the following ones: k1 = 5, k2=6, k3= 0.1.

As last thing I plot the graphs for 𝑥, 𝑦, 𝜃 and we can observe that all the variables reaching 0 in a time of 15 seconds, so we can conclude that we have achieved the desired posture.

Immagine che contiene schermata, testo, linea, Diagramma

Descrizione generata automaticamente

Immagine che contiene linea, schermata, testo, Diagramma

Descrizione generata automaticamente

Immagine che contiene linea, testo, schermata, Diagramma

Descrizione generata automaticamente

To complete the analysis and to ensure that the controller adjustment is reliable, I show the errors on 𝑥, 𝑦,𝜃,𝜌,𝛾,

Immagine che contiene testo, schermata, linea, Diagramma

Descrizione generata automaticamente

Immagine che contiene linea, testo, schermata, Diagramma

Descrizione generata automaticamente

Immagine che contiene testo, linea, Diagramma, schermata

Descrizione generata automaticamente

Immagine che contiene testo, schermata, linea, Diagramma

Descrizione generata automaticamente

Immagine che contiene testo, linea, schermata, Diagramma

Descrizione generata automaticamente

Immagine che contiene testo, linea, Diagramma, diagramma

Descrizione generata automaticamente

As we can notice all the errors converge to 0, so in the end we can affirm that everything works correctly.