FDS 2024: Field and service robotics
Homework 2

Andrea Morghen

Question 1: State whether each of following distributions are involutive or not, and briefly justify your answer. If possible, find the annihilator for each distribution.

To find the annihilator of the distributions, it is possible to use matlab or other mathematical problem-solving software by solving the problem of finding the null of the transposed distributions, in fact the following applies:

$$w * D = 0 => D^T * w^T = 0$$

This operation was performed to refute the calculations. However, full mathematical calculations are given below.

a. $\left\{ \begin{array}{c} 3 * x1 \\ 0 \\ -1 \end{array} \right\}$

As reported in the courses Slides[2]: "by definition, **any 1-dimensional distribution is involutive** since $[f(x), f(x)] = 0$, and the zero vector belongs to any distribution by default."
To compute the annihilator of the distribution we must find the vectors whose product to the vector of the distribution produce the null vector:

$$\begin{pmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \end{pmatrix} \begin{pmatrix} 3x_1 \\ 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} => \begin{cases} 3w_1x_1 - w_3 = 0 \\ 3w_4x_1 - w_6 = 0 \end{cases}$$

The solution is in the form:

$$\begin{bmatrix} \alpha & \beta & 3\alpha x_1 \\ \gamma & \delta & 3\gamma x_1 \end{bmatrix}$$

The possible values of $\alpha, \beta, \gamma, \delta$ discriminate different possibilities, the only constrain is that the two rows must be linearly independent. In order to respect the constrain it should be $\alpha \neq \gamma$ or $\beta \neq \delta$. The annihilator is the span of this two possible solution (they can be particolarized in the case $\alpha = 1$, $\beta = 0$,$\gamma = 0$, $\delta = 1$ without loss of generality):

$$\Delta_A = \text{span} \begin{bmatrix} 1 & 0 & 3x_1 \\ 0 & 1 & 0 \end{bmatrix}$$

Therefore, **this distribution is involutive and its annihilator is the Δ_A defined before.**

$$\textbf{b.}\begin{Bmatrix} 1 & 0 \\ 0 & 0 \\ x_2 & x_1 \end{Bmatrix}$$

The distribution decreases in size where $x_1 = 0$. The two cases are analyzed separately.

In the case $x_1 = 0$: in the same way as in the first exercise, it is involutive because any 1-dimensional distribution is involutive.
In the case $x_1 \neq 0$: let's compute the Lie brackets between the two vectors

$$[v_1, v_2] = \frac{\partial v_2}{\partial x} v_1 - \frac{\partial v_1}{\partial x} v_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ x_2 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ x_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

So the matrix F is:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ x_2 & x_1 & 1 \end{pmatrix}$$

Its rank is 2 so and the Lie bracket of the two vector is a vector field belonging to the distribution (combination of the other two), so **the distribution is involutive**.

To compute the annihilator of the distribution we must find the vectors whose product to the vector of the distribution produce the null vector:

In the case $x_1 = 0$: $\begin{pmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow \begin{cases} w_1 + w_3 x_2 = 0 \\ w_1 + w_3 x_2 = 0 \end{cases}$

The solution is in the form: $\begin{bmatrix} -\alpha x_2 & \beta & \alpha \\ -\gamma x_2 & \delta & \gamma \end{bmatrix}$

The possible values of $\alpha, \beta, \gamma, \delta$ discriminate different possibilities, the only constrain is that the two rows must be linearly independent. In order to respect the constrain it should be $\alpha \neq \gamma$ or $\beta \neq \delta$. The annihilator is the span of this two possible solution (they can be particolarized in the case $\alpha = 1$, $\beta = 0$,$\gamma = 0$, $\delta = 1$ without loss of generality):

$\Delta_{A1} = \text{span} \begin{bmatrix} -x_2 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

In the case $x_1 \neq 0$: $\begin{pmatrix} w_1 & w_2 & w_3 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ x_2 & x_1 \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix} \Rightarrow \begin{cases} w_1 + w_3 x_2 = 0 \\ w_3 x_1 = 0 \end{cases}$

The annihilator is in the form: $\begin{bmatrix} 0 & \alpha & 0 \end{bmatrix}$

It can be particolarized by the choice $\alpha = 1$.

$\Delta_{A2} = \text{span} \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$

Therefore, **this distribution is involutive (for any value of x_1) and its annihilator is the span of Δ_{A1} if $x_1 = 0$ or the span of Δ_{A2} if $x_1 \neq 0$**

$$\textbf{c.} \begin{Bmatrix} 2 * x_3 & x_2 \\ -1 & x_1 \\ 0 & 1 \end{Bmatrix}$$

Let's compute the Lie brackets between the two vectors

$$[v_1, v_2] = \frac{\partial v_2}{\partial x} v_1 - \frac{\partial v_1}{\partial x} v_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 2x_3 \\ -1 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_2 \\ x_1 \\ 1 \end{pmatrix} = \begin{pmatrix} -3 \\ 2x_3 \\ 0 \end{pmatrix}$$

So the matrix F is:

$$\begin{pmatrix} 2x_3 & x_2 & -3 \\ -1 & x_1 & 2x_3 \\ 0 & 1 & 0 \end{pmatrix}$$

Its determinant is $3 - 4x_3 2$, its determinant is equal to 0 when $x_3 = \sqrt{(\frac{3}{4})}$ so its rank is always 3 besides from this value of x_3. The Lie bracket of the two vector is not a vector field belonging to the distribution besides that value of x_3, so **the distribution is not involutive besides where** $x_3 = \sqrt{(\frac{3}{4})}$.

To compute the annihilator of the distribution we must compute the base of the vectors whose product to the vector of the distribution produce the null vector:

$$\begin{pmatrix} w_1 & w_2 & w_3 \end{pmatrix} \begin{pmatrix} 2x_3 & x_2 \\ -1 & x_1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \Rightarrow \begin{cases} 2w_1 x_3 - w_2 = 0 \\ w_1 x_2 + w_2 x_1 + w_3 = 0 \end{cases}$$

The annihilator is in the form: $\begin{bmatrix} \alpha & 2x_3\alpha & -\alpha(2x_1x_3 + x_2) \end{bmatrix}$

It can be particolarized by the choice $\alpha = 1$.

$\Delta_A = \text{span} \begin{bmatrix} 1 & 2x_3 & -2x_1x_3 - x_2 \end{bmatrix}$

Therefore, **this distribution is not involutive besides where** $x_3 = \sqrt{(\frac{3}{4})}$ **and its annihilator is the span of** $[1, 2x_3, -2x_1x_3 - x_2]$

Question 2: . Consider an omnidirectional mobile robot having 3 Mecanum wheels placed at the vertices of an equilateral triangle, each oriented in the direction orthogonal to the bisectrix of its angle. Let x and y be the Cartesian coordinates of the center of the robot, θ the vehicle orientation and α, β, γ represent the angle of rotation of each wheel around its axis. Also, denote by r the radius of the wheels and by l the distance between the centre of the robot and the centre of each wheel. This mechanical system is subject to the following Pfaffian constraints, where q = [$x, y, \theta, \alpha, \beta, \gamma$]

$$\begin{pmatrix} \frac{\sqrt{3}\cos(\overline{\theta})}{2} - \frac{\sin(\overline{\theta})}{2} & \sin(\overline{\theta}) & -\frac{\sin(\overline{\theta})}{2} - \frac{\sqrt{3}\cos(\overline{\theta})}{2} \\ \frac{\cos(\overline{\theta})}{2} + \frac{\sqrt{3}\sin(\overline{\theta})}{2} & -\cos(\overline{\theta}) & \frac{\cos(\overline{\theta})}{2} - \frac{\sqrt{3}\sin(\overline{\theta})}{2} \\ \overline{l} & \overline{l} & \overline{l} \\ \overline{r} & 0 & 0 \\ 0 & \overline{r} & 0 \\ 0 & 0 & \overline{r} \end{pmatrix}$$

Compute a kinematic model of such an omnidirectional robot and show whether this system is holonomic or not. [Hint: Use Matlab symbolic toolbox and the null command to ease your work. Do not care about the physical meaning of the kinematic inputs.]

In order to compute a kinematic model of the omnidirectional robot we have to construct a distribution of the null of Pfaffian matrix ($\Delta = span\{g_j(q) : A^T * g_j = 0, j = 1, ...m\}$). So, in matlab, the distribution have been computed:

$$G = \begin{pmatrix} -\frac{r\left(3\cos(\theta)-\sqrt{3}\sin(\theta)\right)}{3\sqrt{3}} & -\frac{2r\sin(\theta)}{3} & \frac{r\left(3\cos(\theta)+\sqrt{3}\sin(\theta)\right)}{3\sqrt{3}} \\ -\frac{r\left(3\sin(\theta)+\sqrt{3}\cos(\theta)\right)}{3\sqrt{3}} & \frac{2r\cos(\theta)}{3} & \frac{r\left(3\sin(\theta)-\sqrt{3}\cos(\theta)\right)}{3\sqrt{3}} \\ -\frac{r}{3l} & -\frac{r}{3l} & -\frac{r}{3l} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

A generic vector belonging to the distribution can be expressed as linear combination of the vector fields spanning the distribution:

$$\dot{q}(x) = \sum_{j=1}^{n} g_i(q) * u_j = G * u$$

This is the kinematic model of a mechanical system subject to $A^T(q)\dot{q} = 0$ and expresses all the admissible trajectories:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{pmatrix} = \begin{pmatrix} -\frac{r\left(3\,\cos(\theta)-\sqrt{3}\,\sin(\theta)\right)}{3\sqrt{3}} \\ -\frac{r\left(3\,\sin(\theta)+\sqrt{3}\,\cos(\theta)\right)}{3\sqrt{3}} \\ -\frac{r}{3\,l} \\ 1 \\ 0 \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} -\frac{2\,r\,\sin(\theta)}{3} \\ \frac{2\,r\,\cos(\theta)}{3} \\ -\frac{r}{3\,l} \\ 0 \\ 1 \\ 0 \end{pmatrix} u_2 + \begin{pmatrix} \frac{r\left(3\,\cos(\theta)+\sqrt{3}\,\sin(\theta)\right)}{3\sqrt{3}} \\ \frac{r\left(3\,\sin(\theta)-\sqrt{3}\,\cos(\theta)\right)}{3\sqrt{3}} \\ -\frac{r}{3\,l} \\ 0 \\ 0 \\ 1 \end{pmatrix} u_3$$

The holonomy or nonholonomy of $A^T(q)\dot{q} = 0$ can be discriminated through the associated kinematic model $\dot{q} = G(q)u$. Lets compute the accessibility distribution Δ_A of the matrix G:

$$\Delta_A = [g_1, g_2, g_3, [g_1, g_2], [g_2, g_3], [g_1, g_3], ...]$$

The lie brackets to be made were calculated in Matlab, and after selecting a set of linearly independent vectors the accessibility distribution is:

$$\Delta_A = \begin{pmatrix} -\frac{2\,r\,\cos\left(\theta+\frac{\pi}{6}\right)}{3} & -\frac{2\,r\,\sin(\theta)}{3} & \frac{2\,r\,\sin\left(\theta+\frac{\pi}{3}\right)}{3} & \frac{2\,\sqrt{3}\,r^2\,\sin\left(\theta+\frac{\pi}{3}\right)}{9\,l} & -\frac{2\,\sqrt{3}\,r^2\,\cos\left(\theta+\frac{\pi}{6}\right)}{9\,l} \\ -\frac{2\,r\,\cos\left(\theta-\frac{\pi}{3}\right)}{3} & \frac{2\,r\,\cos(\theta)}{3} & -\frac{2\,r\,\cos\left(\theta+\frac{\pi}{3}\right)}{3} & -\frac{2\,\sqrt{3}\,r^2\,\cos\left(\theta+\frac{\pi}{3}\right)}{9\,l} & -\frac{2\,\sqrt{3}\,r^2\,\sin\left(\theta+\frac{\pi}{6}\right)}{9\,l} \\ -\frac{r}{3\,l} & -\frac{r}{3\,l} & -\frac{r}{3\,l} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

By computing the rank of the accessibility distribution is possible to discriminate the type of system we are dealing with. In this case $rank = 5$ (it was not possible to find a sixth linearly independent vector between the caclulated lie brackets), so the system is not controllable, the system has one holonomic constraints and two non-holonomic constraints and so it is **non-holonomic** (if $m < v < n$, with $v = dim(\Delta_A(q))$), $A^T(q)\dot{q} = 0$ have $6-5 = 1$ holonomic/integrable constraint).

Therefore, the system is nonholonomic

Question 3: Implement via software the path planning algorithm for a unicycle based on a cubic Cartesian polynomial. Plan a path leading the robot from the configuration $q_i = [x_i, y_i, \theta_i]^T = [0, 0, 0]^T$, to a random configuration $qf = [x_f, y_f, \theta_f]^T$ generated automatically by the code such that $||q_f - q_i|| = 1$. Then, determine a timing law over the path to satisfy the following velocity bounds $|v(t)| \leq 2m/s$ and $|w(t)| \leq 1rad/s$. [Hint: For the final configuration, use the command rand(1,3): save the result in a vector and divide it by its norm.]

The matlab script contains the solution with a brief explanation of the code. The file is on the github repository:
https://github.com/Andremorgh/FSR2024.git [1]
The file is in the folder HW2 and it is named **hw2_eserc_3.mlx.**
A high-level abstraction explanation of the implemented programming logic is given here. Run the matlab program directly to view the full operation of the code.

Using the cubic Cartesian polynomial formulas, it is possible to derive the complete path from q_i to q_f.
A **trapezoidal velocity profile** function with a fixed travel time of 2 seconds was created as function s.

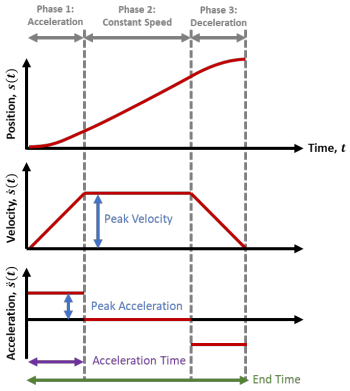

Figure 1: Trapezoidal velocity profile

As required by the trace, the point qf is extracted randomly at each iteration, so **the following is an execution of the programme with** $q_f = [0.563, 0.682, 0.466]$
The formulas used to obtain the trajectory are as follows:

$$x = s^3 * qf(1) - (s-1)^3 * qi(1) + \alpha_x * s^2 * (s-1) + \beta_x * s * (s-1)^2; \quad (1)$$

$$y = s^3 * qf(2) - (s-1)^3 * qi(2) + \alpha_y * s^2 * (s-1) + \beta_y * s * (s-1)^2; \quad (2)$$

$$x' = 3 * s^2 * qf(1) - 3 * (s-1)^2 * qi(1) + \alpha_x * s * (3 * s - 2) + \beta_x * (3 * s - 1) * (s-1); \quad (3)$$

$$y' = 3 * s^2 * qf(2) - 3 * (s-1)^2 * qi(2) + \alpha_y * s * (3 * s - 2) + \beta_y * (3 * s - 1) * (s-1); \quad (4)$$

$$x'' = 6 * s * qf(1) - 6 * (s-1) * qi(1) + \alpha_x * (6 * s - 2) + \beta_x * (6 * s - 4); \quad (5)$$

$$y'' = 6 * s * qf(2) - 6 * (s-1) * qi(2) + \alpha_y * (6 * s - 2) + \beta_y * (6 * s - 4); \quad (6)$$

$$\theta = atan2(y', x'); \tag{7}$$
$$v = sqrt(x'^2 + y'^2) * \dot{s}; \tag{8}$$
$$\omega = \frac{(y'' * x' - x'' * y')}{(x'^2 + y'^2)} * \dot{s}; \tag{9}$$

With:

$$\alpha_x = k * \cos qf(3) - 3 * qf(1) \tag{10}$$
$$\alpha_y = k * \sin qf(3) - 3 * qf(2) \tag{11}$$
$$\beta_x = k * \cos qi(3) + 3 * qi(1) \tag{12}$$
$$\beta_y = k * \sin qi(3) + 3 * qi(2) \tag{13}$$
$$k = 1; \tag{14}$$

These are the results obtained by imposing initial and final conditions and using the trapezoidal velocity profile with a duration of 2 seconds:

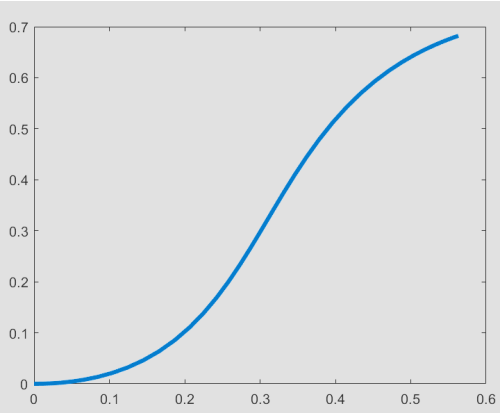

Figure 2: Path in the x y space

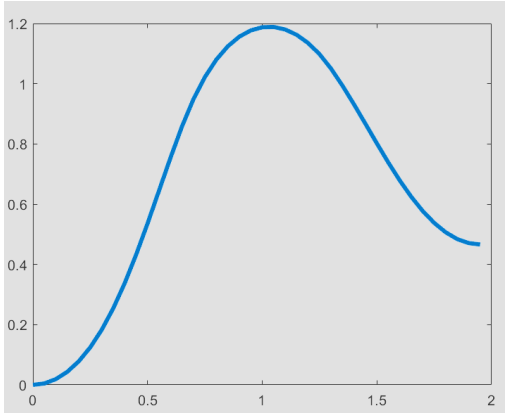

Figure 3: Trajectory of theta

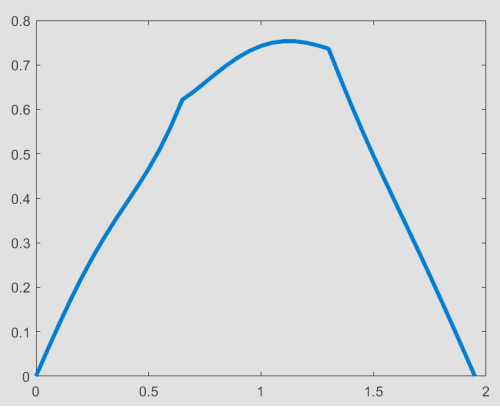

Figure 4: Value of v along the path

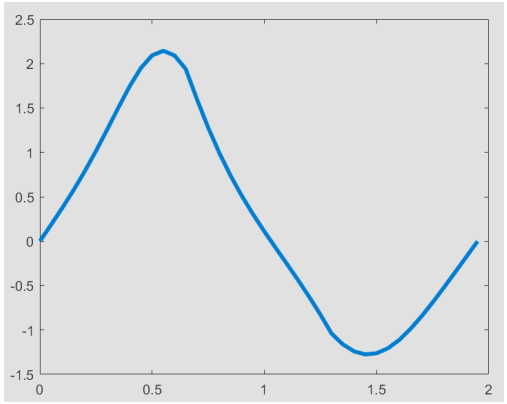

Figure 5: Value of w along the path

In order to determine a timing law on the route that satisfies the speed limits on |v(t)| and |w(t)|, we first check whether these limits are exceeded with the basic time law (2 seconds), if these limits are exceeded, it is possible to scale the time and speeds inversely.

By multiplying the trajectory execution time by τ the speeds |v(t)| and |w(t)| will be reduced by a factor of $\frac{1}{\tau}$.
The τ factor is then derived by taking the maximum between max(v)/2 and max(w)/1, this value could be rational, to conserve the entirety of the trajectory time the first greater integer is taken.
In this case τ = 2.1449 so the final time is $t = next - int(2 * 2.1449) = 5$. Once the trajectory is scaled, these are the values of v and w:

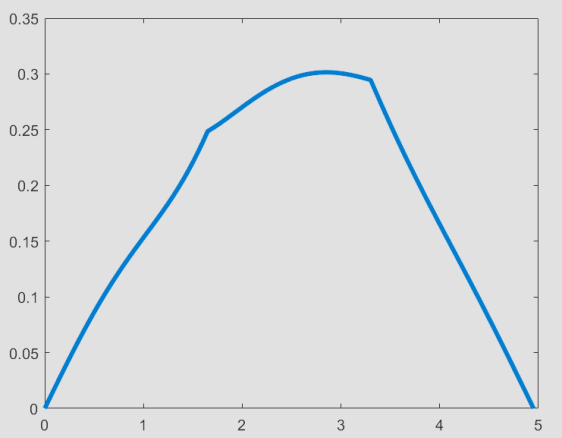

Figure 6: Value of v along the path after the time scaling

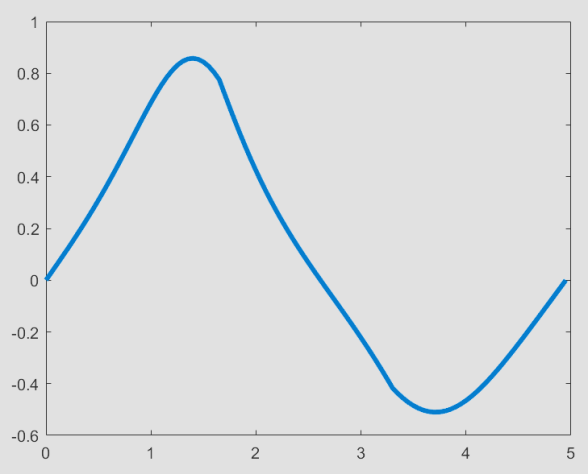

Figure 7: Value of w along the path after the time scaling

It can be observed that both speed limits are now respected throughout the entire trajectory.

Question 4: . Given the trajectory in the previous point, implement via software an input/output linearization control approach to control the unicycle's position. Adjust the trajectory accordingly to fit the desired coordinates of the reference point B along the sagittal axis, whose distance to the wheel's center it is up to you

The matlab and simulink scripts contain the solution with a brief explanation of the code. The files are on the github repository:
https://github.com/Andremorgh/FSR2024.git [1]
The files are in the folder HW2 and they are named **HW2_es3.mlx** and **HW2_es4_simulink.slx.**

A high-level abstraction explanation of the implemented programming logic is given here. Run the matlab program and the simulink model directly to view the full operation of the code.

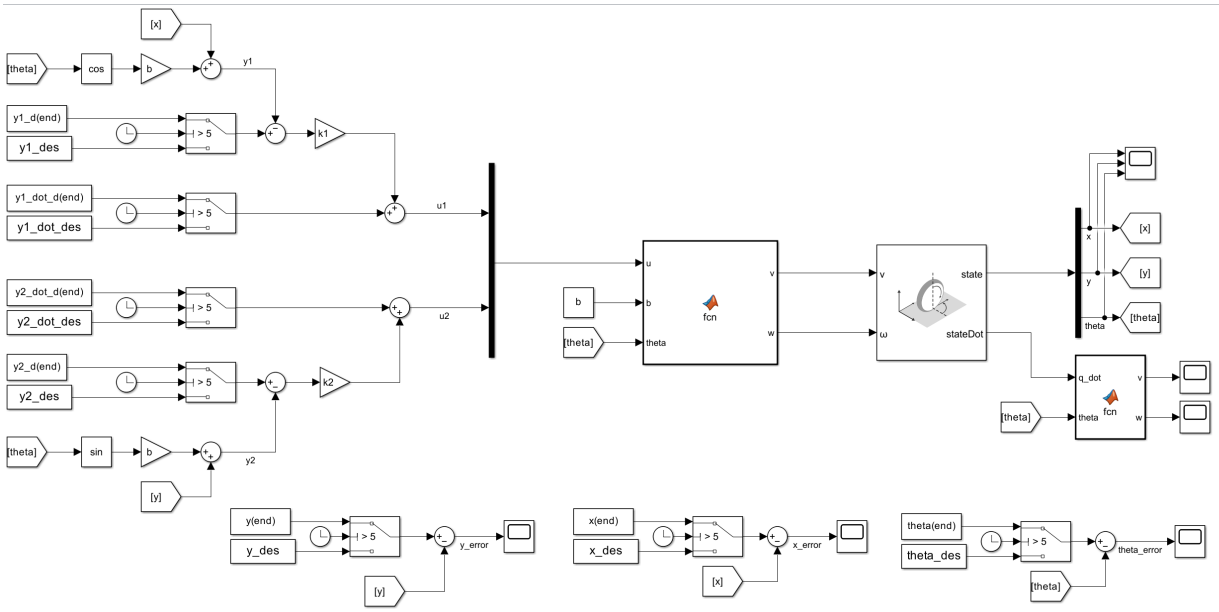

Figure 8: Simulink model

Some parameters and variables are defined in Matlab and then imported. The chosen trajectory to follow is the one obtained in step 3 with the choice $qf = [0.563, 0.682, 0.466]$. The variables y_1 and y_2 are the result of a change of variables, they are defined as follows:

$$\begin{cases} y_1 = x + b\cos\theta \\ y_2 = y + b\sin\theta \end{cases}$$

Where b = 0.05.
The prime derivatives of y_1 and y_2 are also computed according to the following formulas:

$$\begin{cases} \dot{y}_1 = \cos\theta * v - b\sin\theta * w \\ \dot{y}_2 = \sin\theta * v + b\cos\theta * w \end{cases}$$

Where v and w are the desired velocities obtained from exercise 3. The simulink scheme presents a series of sums and

differences to obtain the desired variables u_1 and u_2:

$$\begin{cases} u_1 = \dot{y}_1 + k_1 \cos\theta \\ u_2 = \dot{y}_2 + k_2 \sin\theta \end{cases}$$

Where $k_1 = 10$ and $k_2 = 10$.

Switches in the simulink model make it possible to switch from a variable trajectory to a constant value (final value of the trajectory), allowing the system to remain stable, which in the absence of a control signal would begin to diverge. The switch time is selected from the matlab file as the trajectory time minus a sample (in the current case $t_{switch} = t - \frac{1}{fs} = 5 - 0.05 = 4.95$).

The v and w inputs to the unicycle model are as follows (virtually identical to those obtained in exercise 3):

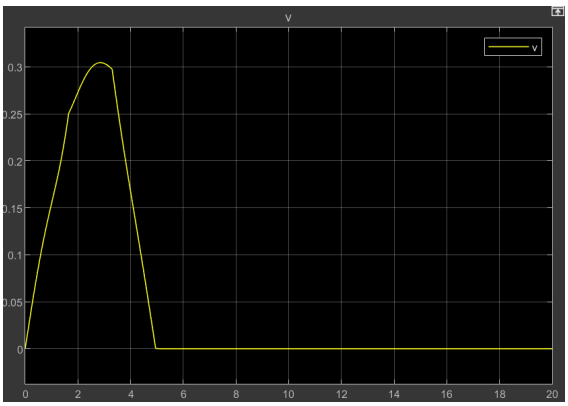

Figure 9: Unicycle v

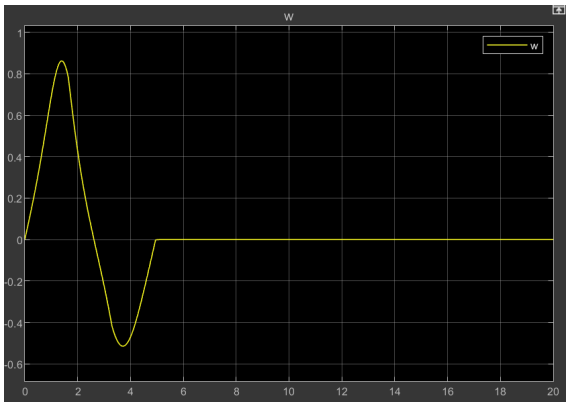

Figure 10: Unicycle w

From the unicycle model we obtain the following values for the variables x, y, θ:

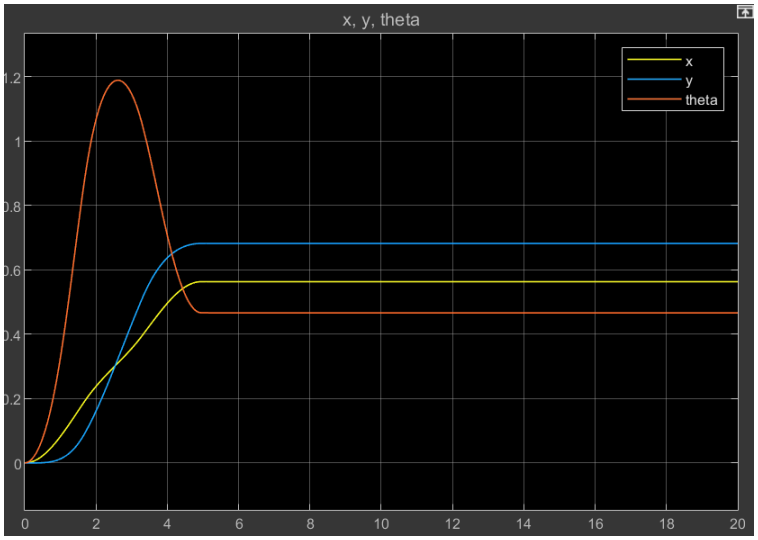

Figure 11: Value of x y and theta

Comparing the values of x, y, θ with the desired values, it can be observed that the error obtained is for x and y of the order of 10^{-4} and for theta is of the order of 10^{-3}.

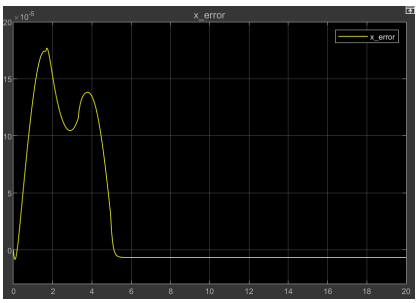

Figure 12: x error

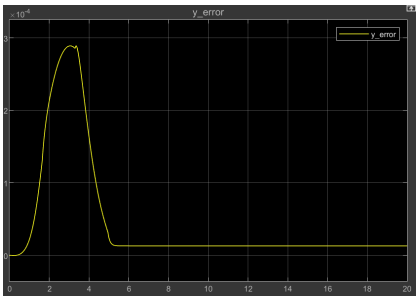

Figure 13: y error

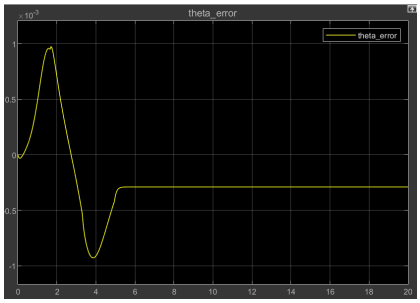

Figure 14: θ error

It is important to note that with a different choice of qf, it is possible to generate more complicated trajectories that bring to light the weakness of this type of control. In fact, as the trajectory becomes more complicated, the error on theta increases while the error on x and y remains approximately 10^{-4} to 10^{-4}, this is due to the fact that "**this approach controls the position of the point b only, leaving the orientation uncontrolled**".

+ Shown below is the path in xy space, θ and the error on θ in the case of a more difficult trajectory:

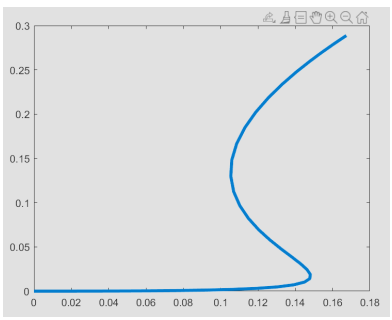

Figure 15: path in xy space

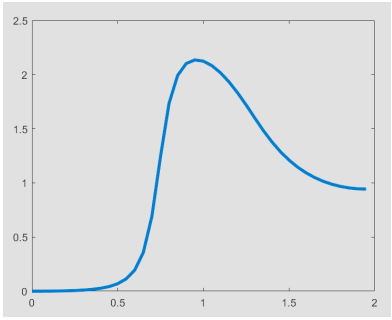

Figure 16: θ trajectory

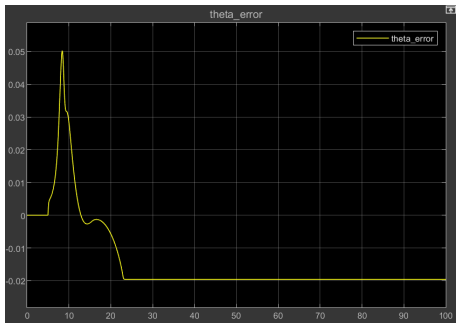

Figure 17: θ error

The error on theta is increased by an order of magnitude compared to the previous case, with longer and more complicated trajectories this effect can only be amplified.

Question 5: Implement via software the unicycle posture regulator based on polar coordinates, with the state feedback computed through the Runge-Kutta odometric localization method. Starting and final configurations are $q_i = [x_i, y_i, \theta_i]^T = [\alpha + 1, 1, \pi/4]^T$, with α the last digit of your matriculation number, and $q_f = [x_f, y_f, \theta_f]^T = [0, 0, 0]^T$

The matlab and simulink scripts contain the solution with a brief explanation of the code. The files are on the github repository:
https://github.com/Andremorgh/FSR2024.git [1]
The files are in the folder HW2 and they are named **HW2_eserc_5.mlx** and **HW2_es5_simulink.slx.**

A high-level abstraction explanation of the implemented programming logic is given here. Run the matlab program and the simulink model directly to view the full operation of the code.

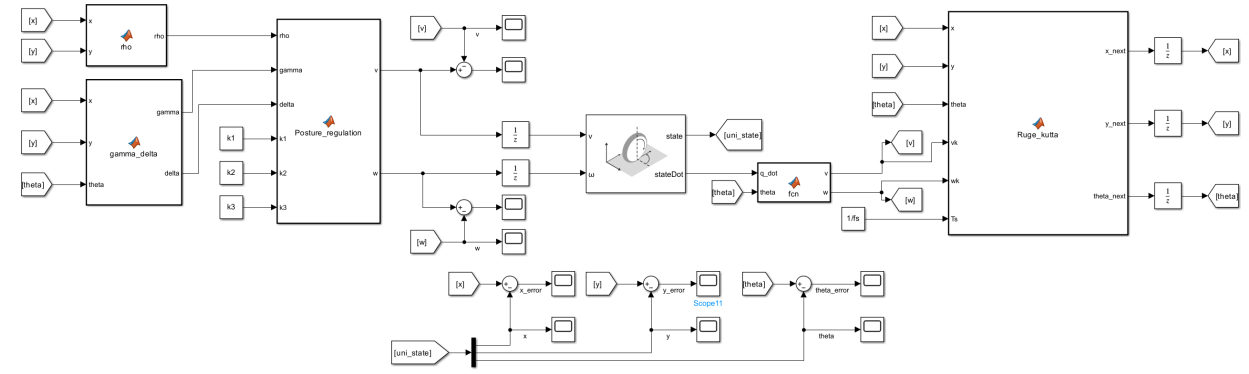

Figure 18: Simulink model

The posture controller of a unicycle based on polar coordinates was implemented, the formulas used, which can be found on the course slides [2] (pg 88-97), are not shown, k values were chosen as follows:

$$k_1 = 3; k_2 = 3; k_3 = 0.2;$$

These are the results obtained with these values of k:

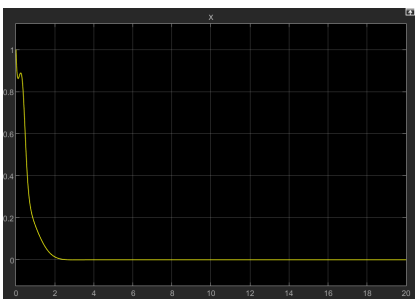

Figure 19: x from unicycle model

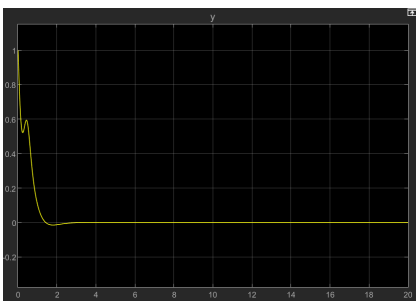

Figure 20: y from unicycle mode

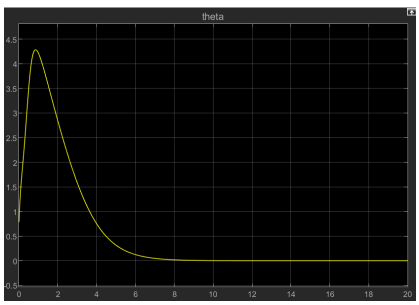

Figure 21: θ from unicycle mode

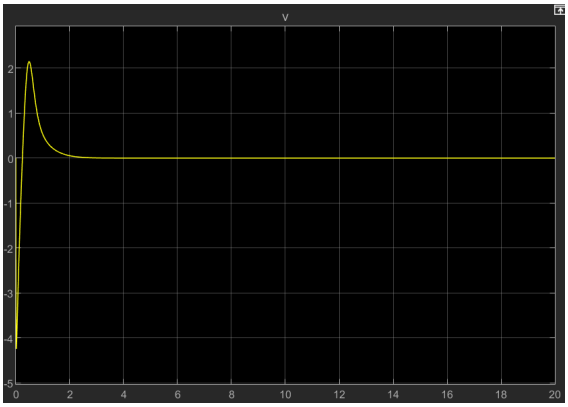

Figure 22: v from unicycle model

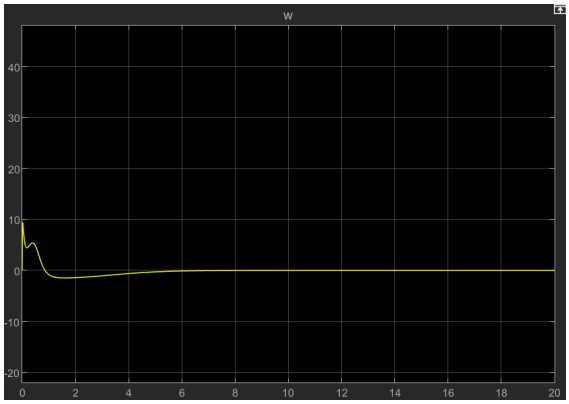

Figure 23: ω from unicycle mode

Such values make it possible to converge to the final state qf=(0 0 0) in a time of about 8 seconds (3 seconds for x and y alone), while maintaining moderate values of v and w and asymptotic behaviour. It is possible to try increasing these values of k, what is obtained is both a faster convergence free of undershoots and overshoots of x and y but also an unstable behaviour of theta. **It turns out that the determining term in the convergence or non-convergence of theta is the factor k3**. This variable must have a lower value than the other two to maintain stability and exceeding a certain value independent of the value of k1 and k2 leads to the instability of theta regardless. These are the trends of theta for $k_1 = k_2 = k_3 = 1$ and $k_1 = k_2 = k_3 = 5$:

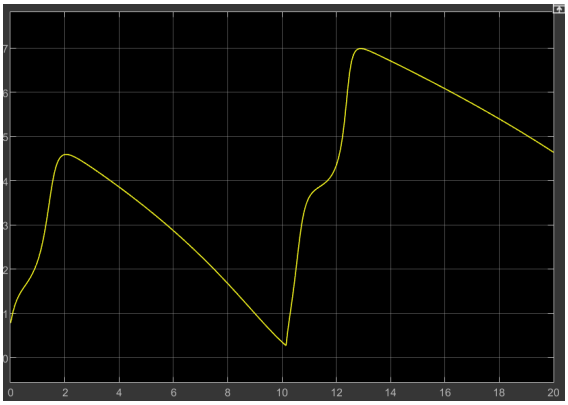

Figure 24: θ for $k_1 = k_2 = k_3 = 1$

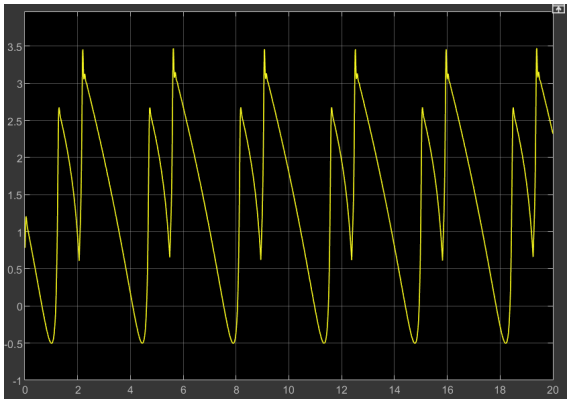

Figure 25: θ for $k_1 = k_2 = k_3 = 5$

The state feedback is calculated using the Runge-Kutta method for odometric localisation with:

$$T_s = \frac{1}{f_s} = \frac{1}{100} = 0.01$$

. It is worth noting that the unicycle represents a simplification of real two-wheel models (differential drive robots), the simulink model of the unicycle returns v and w already calculated, instead for a more accurate simulation, closer to reality, it would be better to have the measure the displacement of each unicycle's wheel (in reality obtained through the encoders) and reconstruct from them v and w as follows:

$$v_k = \frac{\rho}{2T_s}(\Delta\phi_r + \Delta\phi_l); \tag{15}$$

$$\omega_k = \frac{\rho}{2dT_s}(\Delta\phi_r - \Delta\phi_l); \tag{16}$$

Where ρ is the radius of the wheel and d is the distance between the two wheels.

References

[1] Andrea Morghen. FSR2024. 2024. URL: https://github.com/Andremorgh/FSR2024.git.

[2] PRISMA UNINA. FSR_Slides_2024. 2024.