Università degli studi di
Napoli Federico II

Laurea magistrale in ingegneria dell'automazione e robotica

Field and service robotics

Homework 2

a.y. 2023/24

Professor: Fabio Ruggiero

Salvatore Del Peschio

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

a) $\Delta_1 = \left\{ \begin{bmatrix} 3x_1 \\ 0 \\ -1 \end{bmatrix} \right\}, U \in \mathbb{R}^3$

b) $\Delta_2 = \left\{ \begin{bmatrix} 1 \\ \mathbf{0} \\ \boldsymbol{x}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{0} \\ -\alpha \\ \boldsymbol{x}_1 \end{bmatrix} \right\}, U \in \mathbb{R}^3$ **with α the last digit of your matriculation number.**

c) $\Delta_3 = \left\{ \begin{bmatrix} 2x_3 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} x_2 \\ x_1 \\ 1 \end{bmatrix} \right\}, U \in \mathbb{R}^3$

a) The distribution is certainly involutive since any 1-dimensional distribution is involutive. Using the definition of annihilator

$$\Delta^{\perp}(x) = \{w^* \in \mathbb{R}^{n*} :< w^*, v >= 0, \forall v \in \Delta(x)\}$$

We obtain

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} 3x_1 \\ 0 \\ -1 \end{bmatrix} = 0 \Rightarrow 3w_1x_1 - w_3 = 0$$

The generic covector in $\Delta^{\perp}(x)$ will be of the type $\begin{bmatrix} \beta & \alpha & 3\beta x_1 \end{bmatrix}$. We know that $dim(\Delta^{\perp}) = 2$, setting $\alpha = 0, \beta = 1$ and $\alpha = 1, \beta = 0$ we get

$$\Delta^{\perp} = span\{\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 3x_1 \end{bmatrix}\}$$

b) Using $\alpha = 0$

$$\Delta_2 = \left\{ \begin{bmatrix} 1 \\ 0 \\ x_2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ x_1 \end{bmatrix} \right\}$$

When $x_1 = 0$ our distribution has $dim(\Delta_2) = 1$ so we will study the two cases.

i. If $x_1 = 0$ we have an involutive distribution since any 1-dimensional distribution is involutive.

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ x_2 \end{bmatrix} = 0 \Rightarrow w_1 + w_3x_2 = 0$$

The covector will be of the type $\begin{bmatrix} -\alpha x_2 & \beta & \alpha \end{bmatrix}$ so we get

$$\Delta^{\perp} = span\{\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} -x_2 & 0 & 1 \end{bmatrix}\}$$

ii. If $x_1 \neq 0$ we want to know if the distribution is involutive, using the definition we must check if

$$\begin{bmatrix} \tau_1, & \tau_2 \end{bmatrix} \in \Delta_2$$

We obtain the F matrix that has $rank(F) = 2$ so the distribution is involutive

$$F = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ x_2 & x_1 & 1 \end{bmatrix}$$

Then, we proceed to find the annihilator

$$\begin{cases} \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ x_2 \end{bmatrix} = 0 \\ \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ x_1 \end{bmatrix} = 0 \end{cases} \Rightarrow \begin{cases} w_1 + w_3 x_2 = 0 \\ w_3 x_1 = 0 \end{cases}$$

The covector will be of the type $\begin{bmatrix} 0 & \alpha & 0 \end{bmatrix}$ so we get

$$\Delta^{\perp} = span\{\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}\}$$

c) Computing the Lie brackets we obtain the F matrix

$$F = \begin{bmatrix} 2x_3 & x_2 & -3 \\ -1 & x_1 & 2x_3 \\ 0 & 1 & 0 \end{bmatrix}$$

Its determinant $\Delta(F) = -4x_3^2 + 3$ is equal to zero if $x_3 = \sqrt{\frac{3}{4}}$ so the distribution is not involutive in every other point except for $x_3 = \sqrt{\frac{3}{4}}$. We proceed to compute the annihilator

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

The covector will be of the type $\begin{bmatrix} \alpha & 2\alpha x_3 & -\alpha x_2 - 2\alpha x_1 x_3 \end{bmatrix}$ so we get

$$\Delta^{\perp} = span\{\begin{bmatrix} 1 & 2x_3 & -x_2 - 2x_1 x_3 \end{bmatrix}\}$$

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 $q = \begin{bmatrix} x & y & \theta & \alpha & \beta & \gamma \end{bmatrix}^T$**

$$A^T(q)\dot{q} = \begin{bmatrix} \frac{\sqrt{3}}{2}\cos\theta - \frac{1}{2}\sin\theta & \frac{1}{2}\cos\theta + \frac{\sqrt{3}}{2}\sin\theta & l & r & 0 & 0 \\ \sin\theta & -\cos\theta & l & 0 & r & 0 \\ -\frac{\sqrt{3}}{2}\cos\theta - \frac{1}{2}\sin\theta & \frac{1}{2}\cos\theta - \frac{\sqrt{3}}{2}\sin\theta & l & 0 & 0 & r \end{bmatrix} \dot{q} = 0_6$$

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.]

To obtain the kinematic model of the robot, we must compute a distribution from the Pfaffian matrix. The distribution is spanned by the vectors $g_j(q)$, $j = 1, \ldots, m$ such that

$$A^T(q)g_j(q) = 0$$

It has been obtained in MATLAB using the *null*() method.

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

The expression $\dot{q} = G(q)u$ represents the kinematic model of the omnidirectional mobile robot and expresses all the admissible trajectories:

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

The holonomy of our system can be confirmed by obtaining the accessibility distribution Δ_A of the matrix computing all the Lie brackets between the columns. In MATLAB, a *lie_bracket*() method has been developed, and a while loop will be executed until it is not possible to find an independent column in an entire cycle.

We obtain the following Δ_A with $rank(\Delta_A) = 5$

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

We deduce that the system is not controllable, and some model-configuration are prevented. It has two nonholonomic and one holonomic constraints. In conclusion, the system is non-holonomic.

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 $q_i = \begin{bmatrix} x_i & y_i & \theta_i \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, to a random configuration $q_f = \begin{bmatrix} x_f & y_f & \theta_f \end{bmatrix}^T$ 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 $|v(t)| \leq 2$ m/s and $|\omega(t)| \leq 1$rad/s. [Hint: For the final configuration, use the command rand $(1, 3)$: save the result in a vector and divide it by its norm.]**

```
MATLAB files: "ex_3_&_4.mlx"
```

To compute the path based on a cubic Cartesian polynomial, once the variables are initialized, the values of the coefficients $\alpha_x, \alpha_y, \beta_x, \beta_y$, and the x_s, y_s of the polynomial path have been obtained with the following formulas:

$$\beta_x = k\cos\theta_i + 3x_i$$
$$\beta_y = k\sin\theta_i + 3y_i$$
$$\alpha_x = k\cos\theta_f - 3x_f$$
$$\alpha_y = k\sin\theta_f - 3y_f$$
$$x(s) = s^3x_f - (s-1)^3x_i + \alpha_x s^2(s-1) + \beta_x s(s-1)^2$$
$$y(s) = s^3y_f - (s-1)^3y_i + \alpha_y s^2(s-1) + \beta_y s(s-1)^2$$

Then, the velocity profile has been created using the *cubicpolytraj*() method provided by MATLAB with $t_f = 3$. Once the velocity profile is obtained, we can substitute the values of s into the symbolic representation of the path and plot the result.

By differentiating x_s and y_s, we can compute the values of v and ω as:

$$v = \sqrt{(x'(s)^2 + y'(s)^2)}\dot{s}(t)$$
$$\omega = \frac{(y''(s)x'(s) - x''(s)y'(s))}{(x'(s)^2 + y'(s)^2)}\dot{s}(t)$$

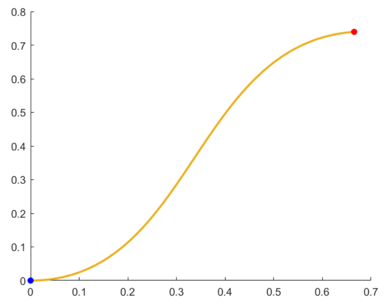
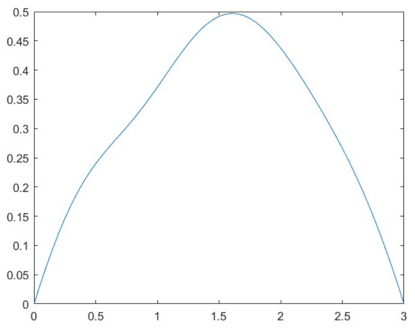
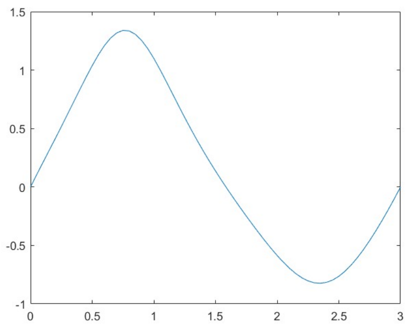


Figure 1: Resulting path with $q_f = [0.6651, 0.7395, 0.1037]$



(a) Plot of v before scaling

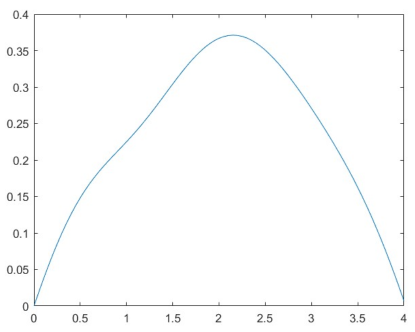


(b) Plot of ω before scaling

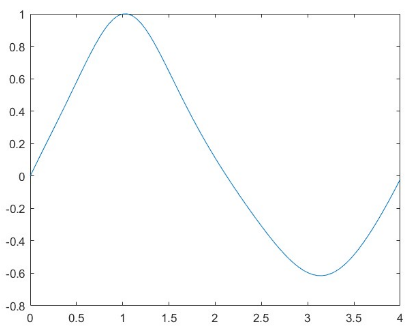
Plotting the velocities, we can observe that ω exceeds the limit value. We then compute the scaling factor k to be applied to t_f in order to respect the limit values. This is done by calculating the maximum of the following ratios:

$$\frac{max(|\omega|)}{\omega_{max}} \qquad \frac{max(|v|)}{v_{max}}$$

where $max(|\omega|)$ is the absolute value of the maximum ω along the path, and ω_{max} is its limit value. The resulting k is applied to t_f, and the new velocity profile, v and ω, are computed. Plotting we can observe that they are uniformly scaled, and the limits are now satisfied.



(a) Plot of v after scaling



(b) Plot of ω after scaling

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.**

```
MATLAB files: "ex_3_&_4.mlx", "ex_4.slx"
```

Utilizing the same trajectory previously generated an input/output linearization control has been developed using the Simulink environment. Setting $b = 0.01$ the value of $y_1, y_2, \dot{y}_i, \dot{y}_2$ have been computed with the following formulas

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

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

Setting $K_1 = 7$ and $K_2 = 7$ the values of u_1 and u_2 are obtained with the following

$$\begin{cases} u_1 = \dot{y}_1 + K_1\cos\theta \\ u_2 = \dot{y}_2 + K_2\sin\theta \end{cases}$$

Implemented in the subsequent Simulink scheme

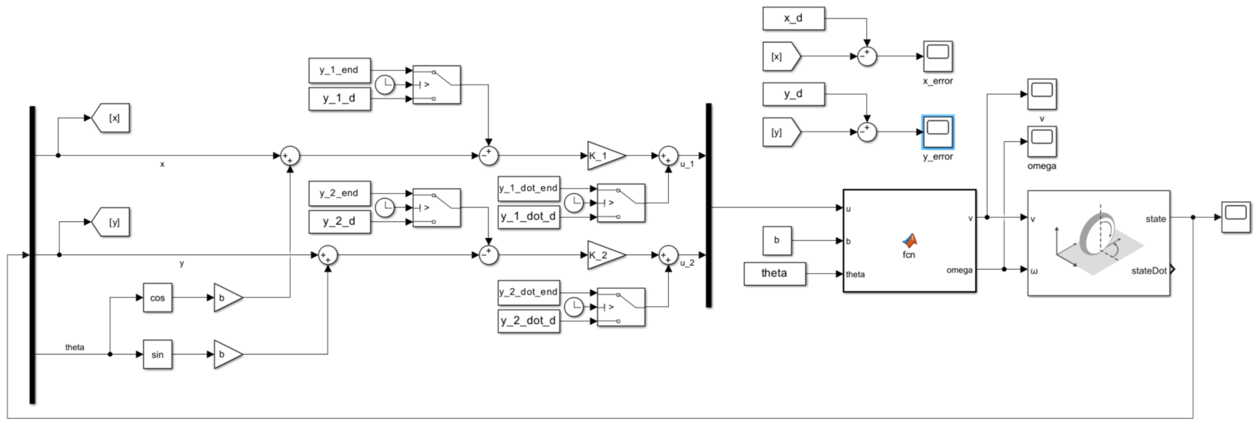


Figure 4: Simulink scheme for the input/output linearization control

We obtain the following behavior of the unicycle that successfully reaches the final position.

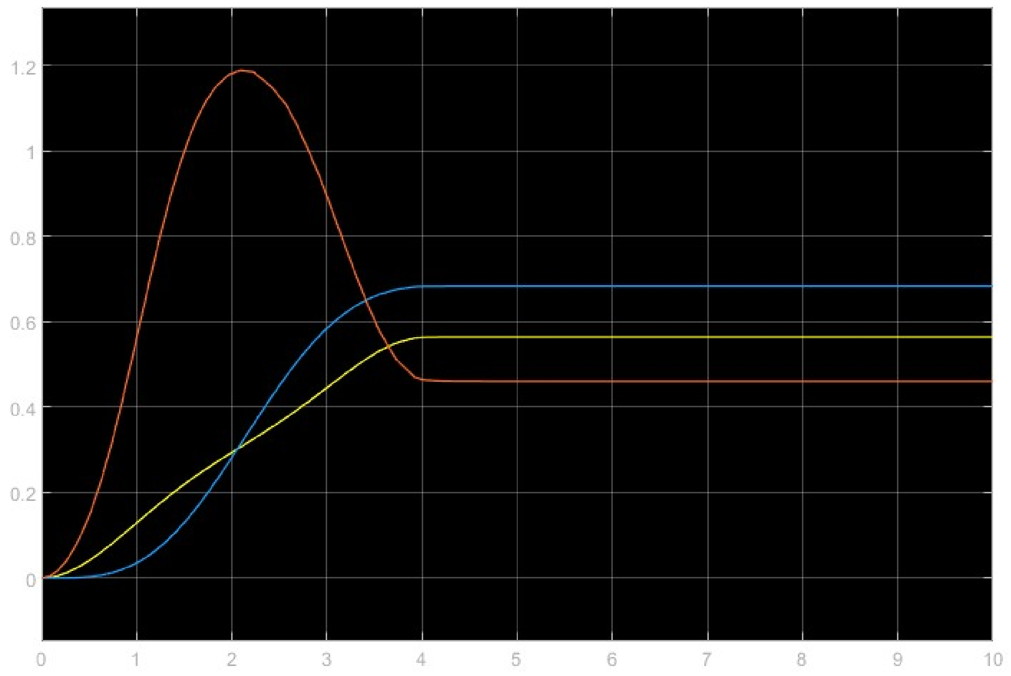
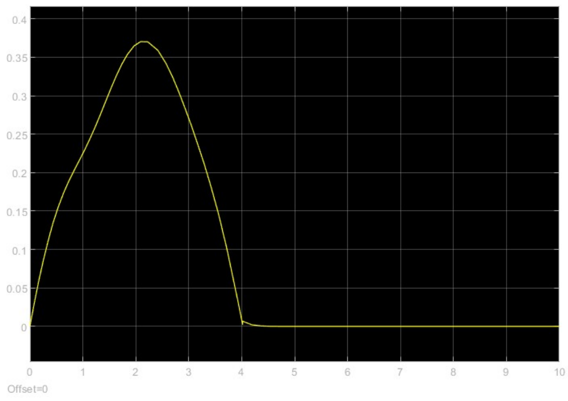


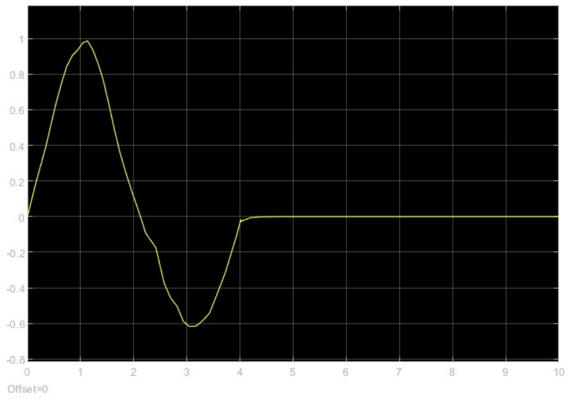
Figure 5: Plot of the state of the model: x (yellow), y (blue) and θ (red)

Through the scopes inserted we can observe that the input of our unicycle model are almost equal to the plot of the previous exercise. They have been computed starting from the values of u_1 and u_2.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = T(\theta)^{-1} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad T = \begin{bmatrix} \cos\theta & -b\sin\theta \\ \sin\theta & b\cos\theta \end{bmatrix}$$

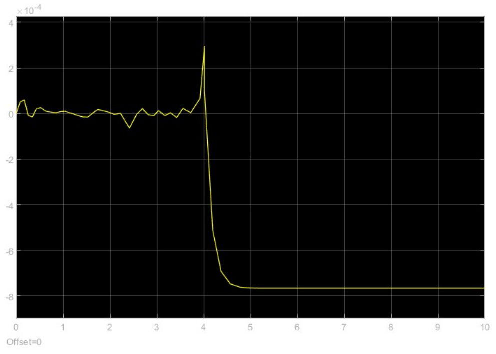


(a) v

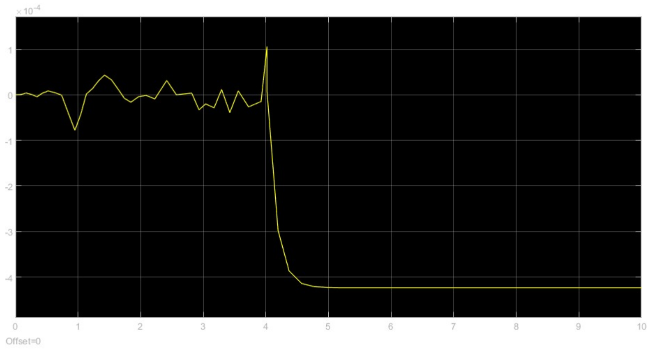


(b) ω

We can observe that the presence of multiple switches at the end of the trajectory make sure to retain the final position avoiding the divergence caused by the absence of control. Analyzing the error for the x and y values we obtained a satisfying behavior with an error in the order of 10^{-5} during the motion and a constant error in the order of 10^{-4} after the switch of the control.



(a) Plot of x error



(b) Plot of y error

It is important to notice that the values of θ have not been reported because we know that this approach controls the position of the point B only, leaving the orientation uncontrolled.

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 $q_i = \begin{bmatrix} x_i & y_i & \theta_i \end{bmatrix}^T = \begin{bmatrix} \alpha+1 & 1 & \pi/4 \end{bmatrix}^T$, with α the last digit of your matriculation number, and $q_f = \begin{bmatrix} x_f & y_f & \theta_f \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$.**

```
MATLAB files: "ex_5.m", "ex_5_sim.slx"
```

The following Simulink scheme has been developed to implement the unicycle posture regulator based on polar coordinates

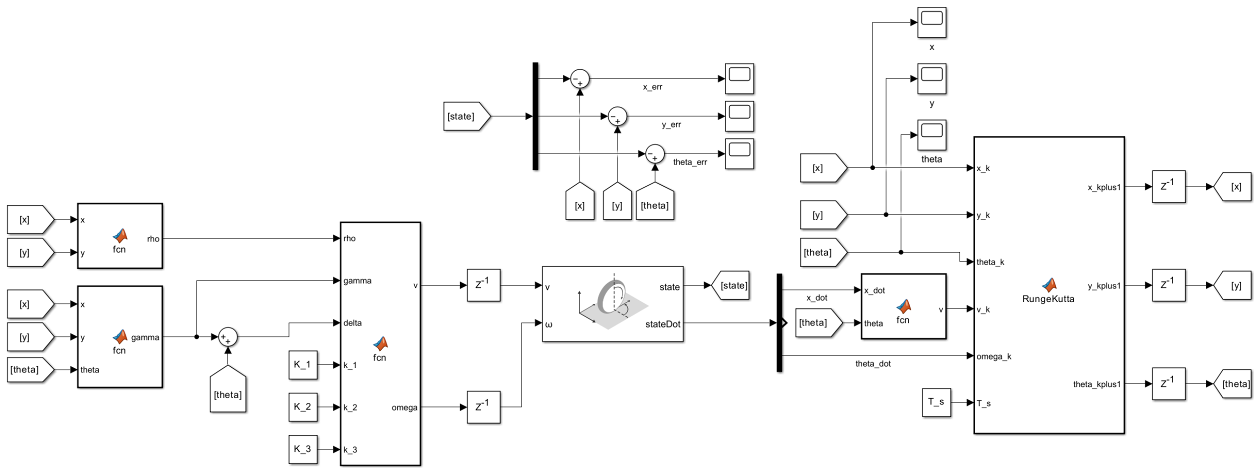


Figure 8: Simulink scheme

In the left side of the scheme ρ, γ and δ are obtained

$$\rho = \sqrt{x^2 + y^2}$$

$$\gamma = atan2(y, x) - \theta + \pi$$

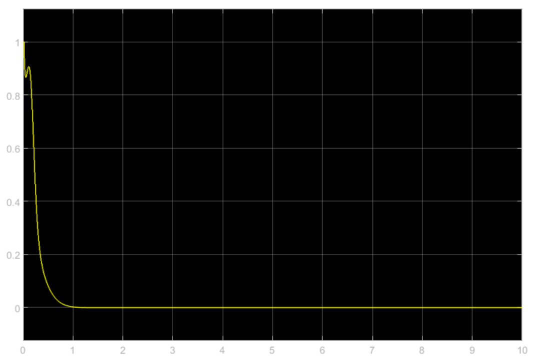
$$\delta = \gamma + \theta$$

and used as input in the MATLAB function block to compute the control signals v and ω. Setting $K_1 = 7, K_2 = 7$ and $K_3 = 0.1$ the design of the controller to reach the position $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ is

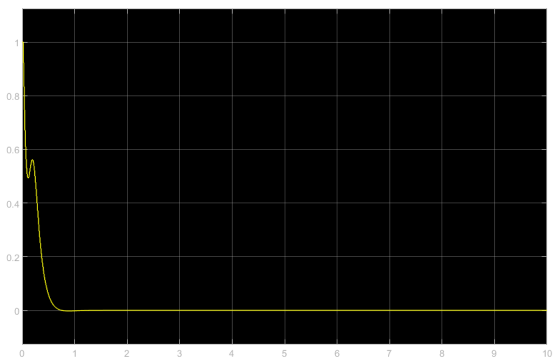
$$v = K_1 \rho \cos\gamma$$

$$\omega = K_2\gamma + K_1 \sin\gamma\cos\gamma(1 + K_3\frac{\delta}{\gamma})$$

Carrying out the simulation we obtain the subsequent values of x, y and $theta$ successfully converging to the desired position.



(a) x



(b) y

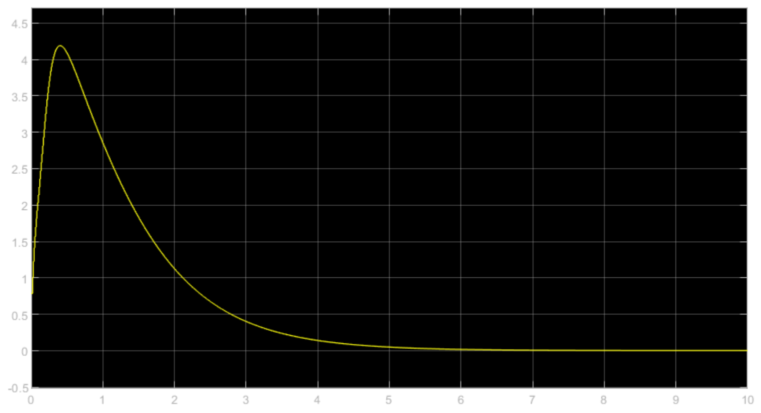
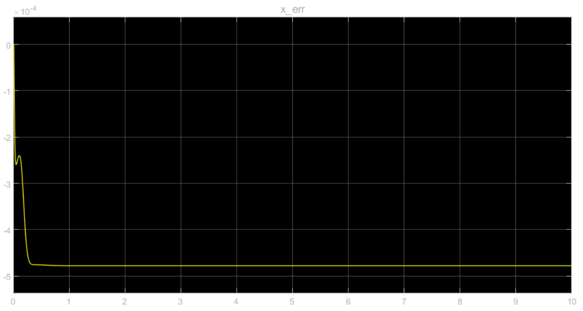


Figure 10: θ

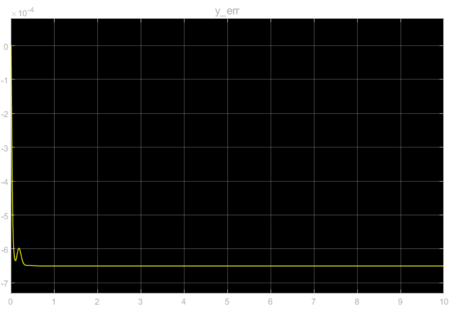
The state feedback has been computed using the Runge-Kutta odometric localization method retrieving the value of the unicycle state from the Simulink block

$$\begin{cases} x_{k+1} = x_k + v_k T_s \cos\left(\theta_k + \frac{1}{2}\omega_k T_s\right) \\ y_{k+1} = y_k + v_k T_s \sin\left(\theta_k + \frac{1}{2}\omega_k T_s\right) \\ \theta_{k+1} = \theta_k + \omega_k T_s \end{cases}$$

Following are shown the errors in the estimation of the unicycle posture



(a) x error



(b) $y$error

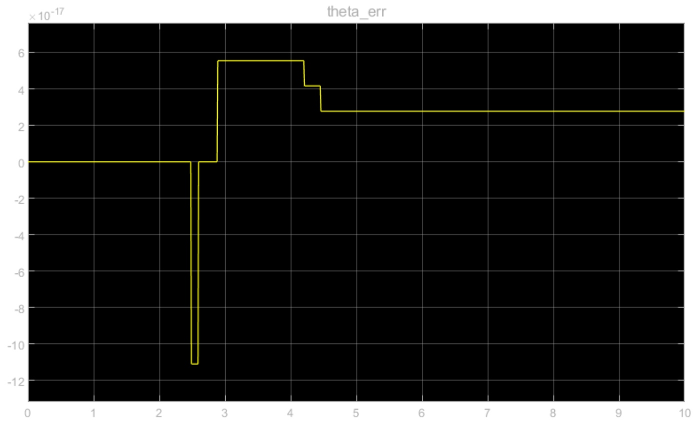


Figure 12: θ error

It is worth noticing that the state of the unicycle given to the *RungeKutta* MATLAB function block have been obtained directly, in a practical scenario they can't been measured directly but computed through the reading of encoder sensors attached to the wheels of our differential drive robot. Once obtained the displacement of each unicycle's wheel $\Delta\Phi_r, \Delta\Phi_s$ we can compute v_k and ω_k as

$$v_k = \frac{\rho}{2T_s}(\Delta\Phi_r + \Delta\Phi_s)$$

$$\omega_k = \frac{\rho}{dT_s}(\Delta\Phi_r - \Delta\Phi_s)$$

with d the wheel track of the differential drive and ρ the wheels radius.