UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria dell'Automazione e Robotica

Field and Service Robotics

HOMEWORK 2

Academic Year 2024/2025

Relatore
Ch.mo prof. Fabio Ruggiero
Candidato
Emmanuel Patellaro
P38000239

Exercise n°1

Let us implement via MATLAB the *Path Planning Algorithm for a Unicycle* based on a *Cubic Cartesian Polynomial* from the initial configuration $q_i = [x_i \quad y_i \quad \theta_i]^T = [0 \quad 0 \quad 0]^T$ to the final random configuration $q_f = [x_f \quad y_f \quad \theta_f]^T$ generated automatically by the code such that $\|q_f - q_i\| = 1$. If necessary, let us implement a *Time Scaling* in such a way to satisfy the following velocity bounds $|v(t)| \leq 0.5m/s$ and $|\omega(t)| \leq 2rad/s$. Remember that the Unicycle is a *differentially flat system* with x and y as *flat outputs* [1]. Now, let us consider the cubic polynomials for the flat outputs

$$x(s) = s^3 x_f - (s-1)^3 x_i + \alpha_x s^2 (s-1) + \beta_x s (s-1)^2$$

$$y(s) = s^3 y_f - (s-1)^3 y_i + \alpha_y s^2 (s-1) + \beta_y s (s-1)^2$$

with $\alpha_x = kcos\theta_f - 3x_f$, $\alpha_y = ksin\theta_f - 3y_f$, $\beta_x = kcos\theta_i + 3x_i$ and $\beta_y = ksin\theta_i + 3y_i$. Then, the time derivatives $x'(s)$, $x''(s)$, $y'(s)$ and $y''(s)$ have been computed. The *Timing Law* chosen is the one obtained, also in this case, with the *Cubic Polynomial*, in such a way to have

$$s(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0 \qquad \dot{s}(t) = 3a_3 t^2 + 2a_2 t + a_1$$

$$\ddot{s}(t) = 6a_3 t + 2a_2$$

with $a_0 = 0$, $a_1 = 0$, $a_2 = \frac{3}{T^2}$ and $a_3 = -\frac{2}{T^3}$. Furthermore

$$\theta(s) = atan2(y'(s), x'(s)) \qquad \tilde{v}(s) = \sqrt{x'(s)^2 + y'(s)^2}$$

$$\tilde{\omega}(s) = \frac{y''(s)x'(s) - x''(s)y'(s)}{x'(s)^2 + y'(s)^2}$$

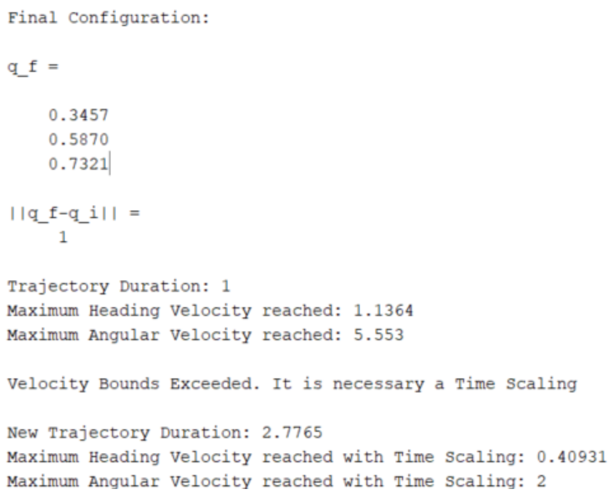
and

$$v(t) = \tilde{v}(s)\dot{s} \qquad \omega(t) = \tilde{\omega}(s)\dot{s}$$

By default, a trajectory duration $T = 1s$ and a gain $k = 1$ are considered. In most cases, since the default T is quite short, a *Time Scaling* is necessary. For

[1] A system is differentially flat if and only if there exists a flat output $y = h(x) \in C^n$ such that it is possible to express the state and the input as a function of y and its time derivative

this purpose, the highest percentage of limit violation by the two velocities is calculated, and then T is *increased accordingly*. Let us show some results through several Plots.



```
Final Configuration:

q_f =

    0.3457
    0.5870
    0.7321

||q_f-q_i|| =
     1

Trajectory Duration: 1
Maximum Heading Velocity reached: 1.1364
Maximum Angular Velocity reached: 5.553

Velocity Bounds Exceeded. It is necessary a Time Scaling

New Trajectory Duration: 2.7765
Maximum Heading Velocity reached with Time Scaling: 0.40931
Maximum Angular Velocity reached with Time Scaling: 2
```

Figure 1: MATLAB Overview

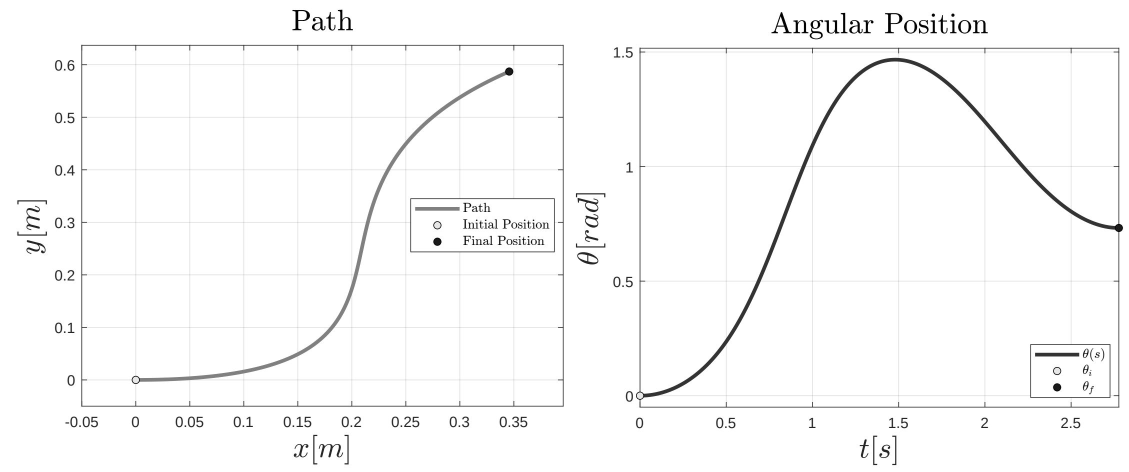


Figure 2: Path and Angular Velocity

As can be seen from Figure 1, the new time T is equal to 2.7765.

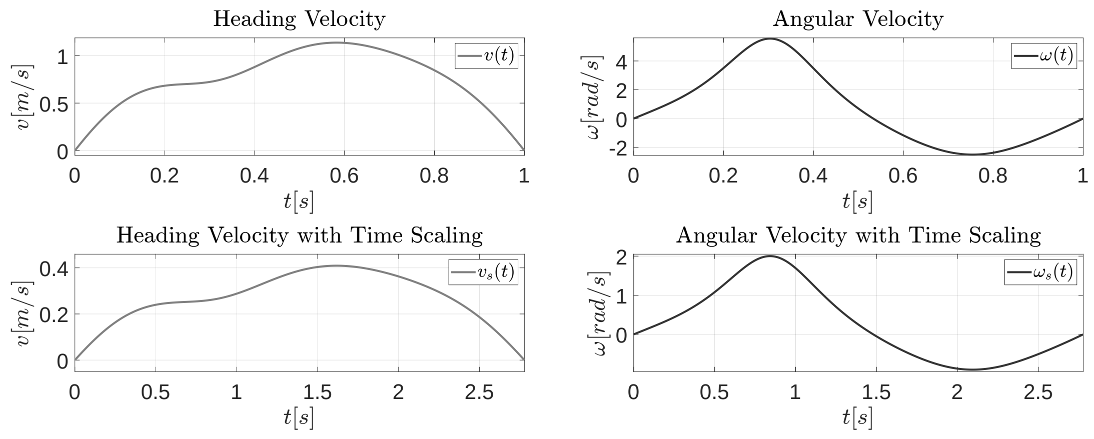


Figure 3: Scaled and Not-Scaled Velocities

Thus, **the final configuration is successfully reached**, and **the velocity limits are also respected**.
The path can take *different forms* depending on the position of the random final point, but the one shown in Figure 2 is one of the most common (other paths are included in the zip file as PDF plots).
Obviously, the "shape" of the velocities over time is the same, but simply *stretched over a longer time span.*

Exercise n°2

Taking into account the *trajectory* in the previous exercise shown in Figure 2, let us implement via *Simulink* an *Input/Output Linearization Control* approach to control the Unicycle's position [2].
It is necessary to adjust the trajectory accordingly to fit the desired coordinates (y_1, y_2) of the *reference point B* along the *Sagittal Axis* with a certain distance b to the wheel's center (x, y). From Figure 4, it is possible to notice that

$$y_1 = x + bcos(\theta) \qquad y_2 = y + bsin(\theta)$$

[2]Remember that the Unicycle Kinematic Model is $\begin{cases} \dot{x} = vcos(\theta) \\ \dot{y} = vsin(\theta) \\ \dot{\theta} = \omega \end{cases}$

Considering the model in note 2, it is possible to write

$$\begin{bmatrix}\dot{y_1}\\ \dot{y_2}\end{bmatrix} = \begin{bmatrix}cos(\theta) & -bsin(\theta)\\ sin(\theta) & bcos(\theta)\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} = T(\theta)\begin{bmatrix}v\\ \omega\end{bmatrix}$$

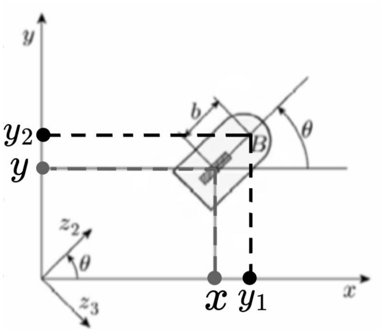


Figure 4: Unicycle

Notice that $det(T(\theta)) \neq 0 \Leftrightarrow b \neq 0$ and so T is *invertible*. In this way, it is possible to say that

$$\begin{bmatrix}v\\ \omega\end{bmatrix} = T(\theta)^{-1}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \Rightarrow \begin{cases}\dot{y_1} = u_1\\ \dot{y_2} = u_2\end{cases}$$

with u_1 and u_2 two *virtual control input*, which can be designed as

$$u_1 = \dot{y}_{1,d} + k_1(y_{1,d} - y_1) \qquad u_2 = \dot{y}_{2,d} + k_2(y_{2,d} - y_2)$$

and with $k_1, k_2 > 0$. Doing this, it is possible to show the *exponential convergence* to the desired $y_{1,d}$ and $y_{2,d}$.
We know that with an *Underactuated System* (like the Unicycle) it is not possible to apply a *Feedback Linearization*. But this has been done here. The '*Trick*' is that you are considering a model with *2 DoFs* and *2 Inputs*. So, neglecting the orientation, the system can be considered as a *Fully Acutated System*. In fact, *the orientation is completely uncontrolled*. The strength of this control is that, unlike the controller based on approximate linearization and the (almost) nonlinear control (showed in the next exercise), it does not require a persistent trajectory[3]. Now, let us show some example with different choices of b considering $k_1 = k_2 = 60$.

[3]The problem of tracking full pose of non-persistent trajectories is structural

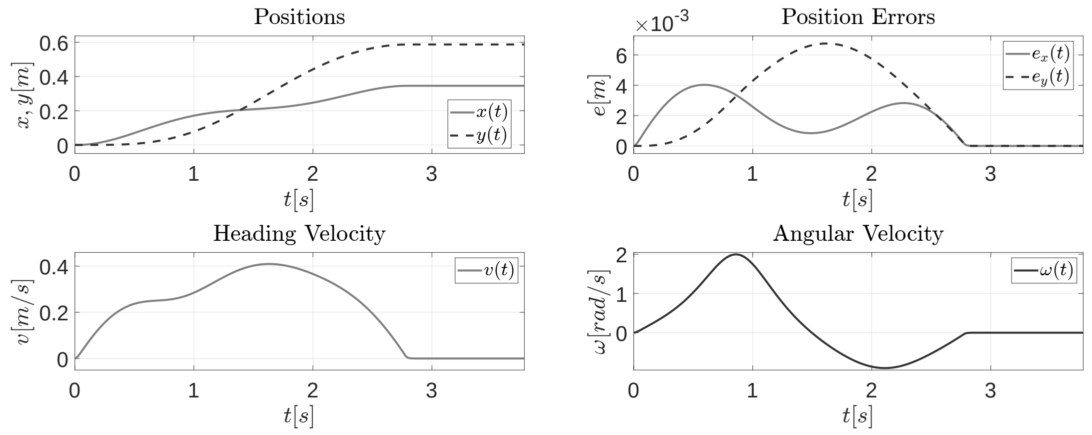


Figure 5: $b = 0.6$

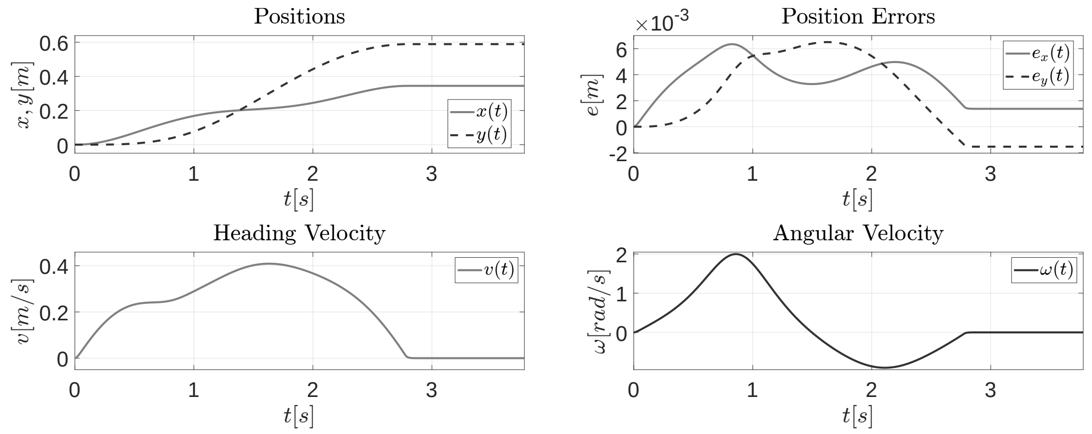


Figure 6: $b = 6$

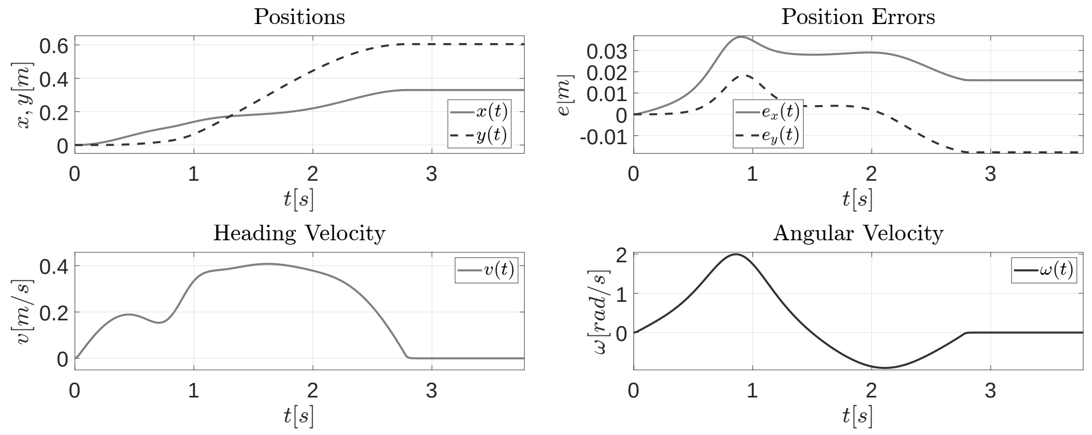


Figure 7: $b = 60$

As can be seen from the figures above, the *Heading and Angular Velocities* remain very similar to each other. As b increases, both the position error and the residual one at the end of the simulation increase. For $b = 0.6$ and $b = 6$, the results remain optimal, with errors on the order of 10^{-3}. In the last example, instead, the error is greater. While in the previous cases the path remains very similar to the desired one, the same cannot be said for the last case, as shown in Figure 8. Furthermore, if the value of b is increased but the gains are kept low, one might encounter the error of a complete trajectory deviation and a total mismatch in the final result.

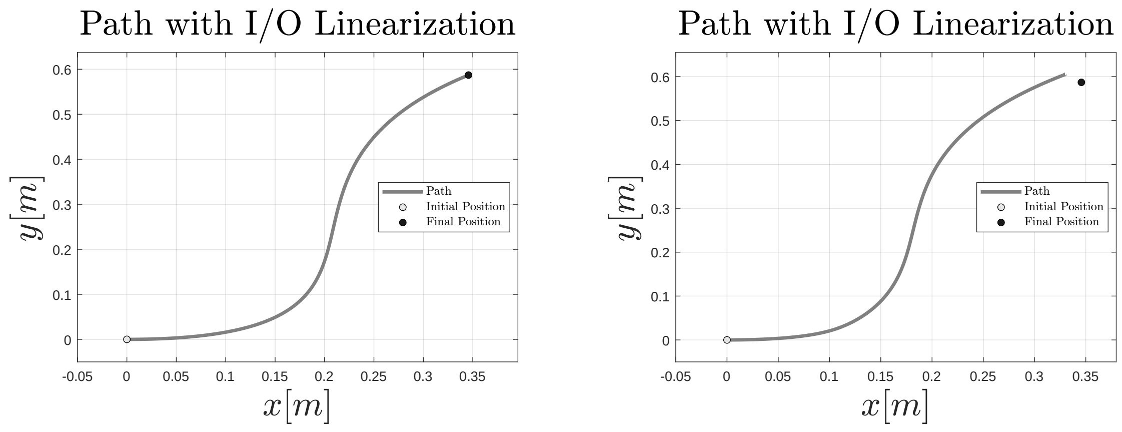


Figure 8: On the left the path in the first two cases, on the right the path with $b = 60$

To avoid this problem, it is necessary to increase the gains k_1 and k_2. In fact, with the choice $k_1 = k_2 = 200$ the final point is successfully reached also for bigger value of b.

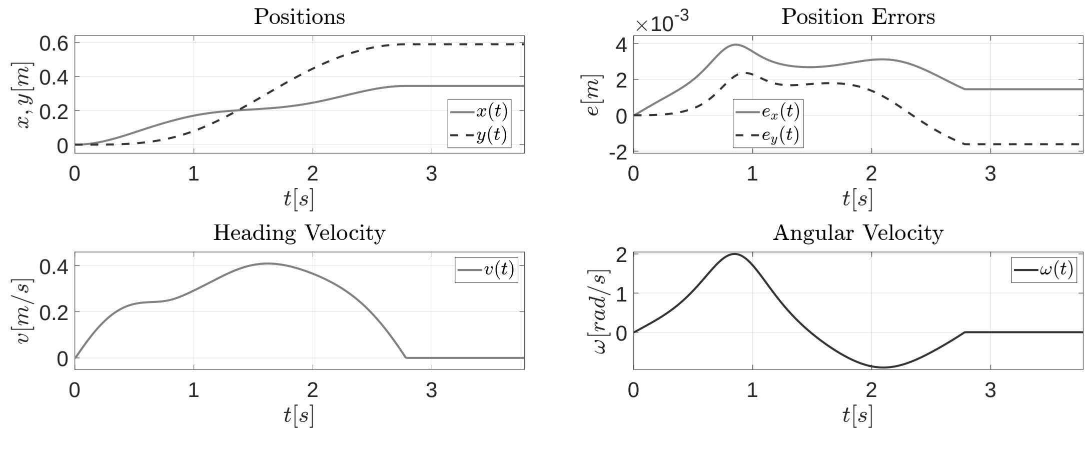


Figure 9: $b = 60$ and $k_1 = k_2 = 200$

It is possible to achieve **excellent results** even for much higher values of b. However, it is obviously necessary to appropriately adjust the gains.
As previously mentioned, the different values of b have no effect on the orientation

and its corresponding error. For this reason, it has not been shown for each case. Obviously, in simulation, the results obtained for the orientation are not entirely bad. However, this will most likely not correspond to the real behavior of the unicycle, since there is no feedback on the orientation and it is not being controlled. Greater are k_1 and k_2, less in simulation is the error.

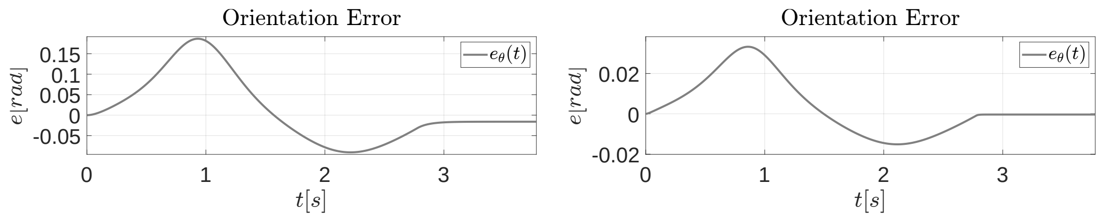


Figure 10: On the left Orientation Error with $k_1 = k_2 = 10$, on the right with $k_1 = k_2 = 60$

If, instead, the value of b is lowered to a value such as 0.00006 or less, we observe a certain loss of control in the angular velocity, precisely due to the reasons highlighted earlier. In the following example, only the plot of the two velocities will be shown to highlight the uncontrolled behavior of the orientation. The smaller b is, the more pronounced this behavior will be.

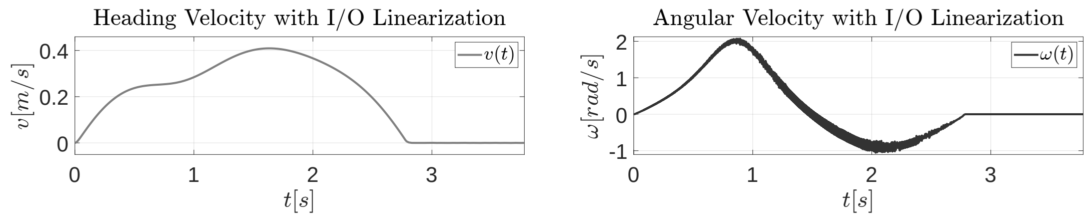


Figure 11: $b = 0.00006$

The zip file contains other plots examples with different choices of gains.

Exercise n°3

Let us consider the *Bernoulli's Leminscate Trajectory*

$$x_d(t) = \frac{rcos(10t)}{1 + sin^2(10t)} \qquad y_d(t) = \frac{rcos(10t)sin(10t)}{1 + sin^2(10t)}$$

with $t \in [0, \frac{4\pi}{10}]$.

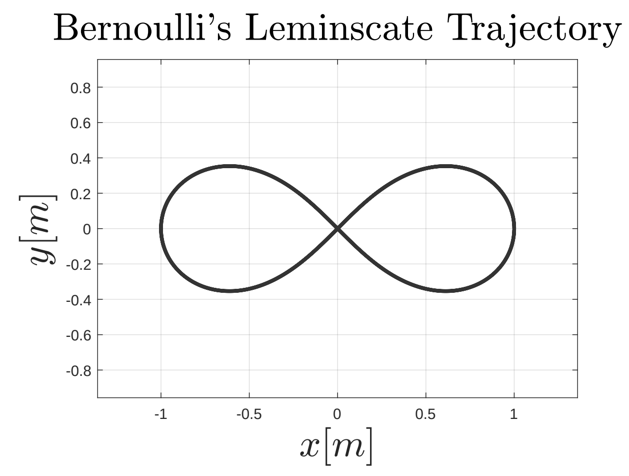


Figure 12: Bernoulli's Leminscate Trajectory

Considering two arbitrary values for $r > 0$ (in this example $r = 1$ and $r = 10$), let us design via *Simulink* the *Linear* and *(Almost) NonLinear Controllers* for a Unicycle, supposing that, at the time $t = 0$, it starts from a random position within $0.5m$ from the desired initial one.

Linear Control

Considering the same desired values as in Exercise n°1 (but in the time domain), and considering the error rotated in the current heading direction [4] , it is possible to make the approximation $e(t) = 0$ and to consider the two control inputs

$$u_1(t) = -k_1 e_1(t) \qquad u_2(t) = -k_2(t)e_2(t) - k_3 e_3(t)$$

in such a way to obtain a *Closed-Loop System* of the form $\dot{e}(t) = A(t)e(t)$ with $k_1 = k_3 = 2\zeta a, \quad k_2(t) = \frac{a^2 - \omega_d(t)}{v_d(t)}, \quad 0 < \zeta < 1, a > 0$. It is possible to notice that the condition $v_d(t) \neq 0$ is necessary. So, as mentioned in the previous exercise, here *the trajectory should be persistent.*

[4] $e(t) = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d(t) - x(t) \\ y_d(t) - y(t) \\ \theta_d(t) - \theta(t) \end{bmatrix}$

(Almost) NonLinear Control

Instead, doing the same assumption $e(t) = 0$, it is possible to consider the control laws

$$u_1(t) = -k_1(v_d(t), \omega_d(t))e_1(t)$$

$$u_2(t) = -k_2 v_d(t)\frac{sin e_3(t)}{e_3(t)}e_2(t) - k_3(v_d(t), \omega_d(t))e_3(t)$$

In this particular case, the following gains have been considered $k_1(v_d(t), \omega_d(t)) = k_3(v_d(t), \omega_d(t)) = \frac{r^2c^2\sqrt{v_d(t)^2+\omega_d(t)^2}}{v_d(t)^2}$, $k_2 = \frac{3c^2}{r}$, $c = \frac{36}{r^{0.3}}$. The Simulink model is practically identical to the linear case, with the only difference being in the formulas inside the MATLAB Functions and in the computation of the gains $k_1(v_d(t), \omega_d(t))$ and $k_3(v_d(t), \omega_d(t))$ instead of $k_2(t)$.

Let us see the results for both controls, considering $a = 600$ and $\zeta = 0.5$.

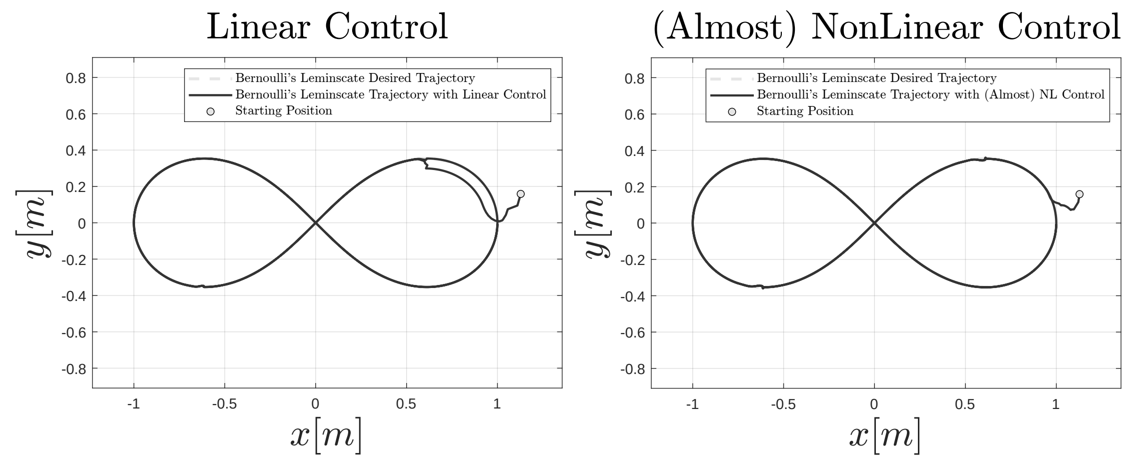


Figure 13: Bernoulli's Leminscate Trajectory with $r = 1$

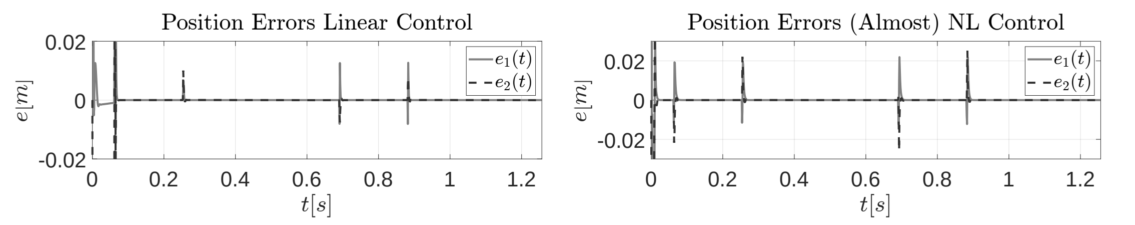


Figure 14: Position Errors with $r = 1$ (without considering the initial error due to the random position)

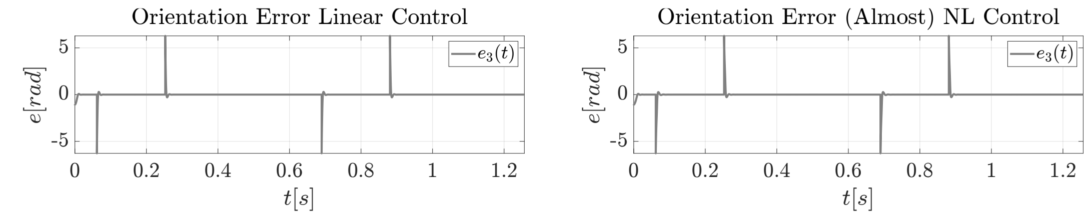


Figure 15: Orientation Error with $r = 1$

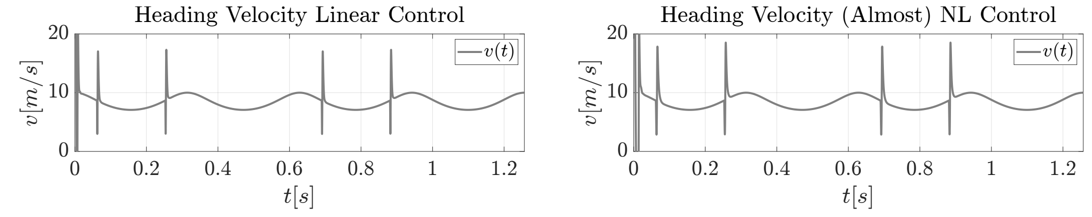


Figure 16: Heading velocities with $r = 1$ (without considering the first large peaks due to the random position)

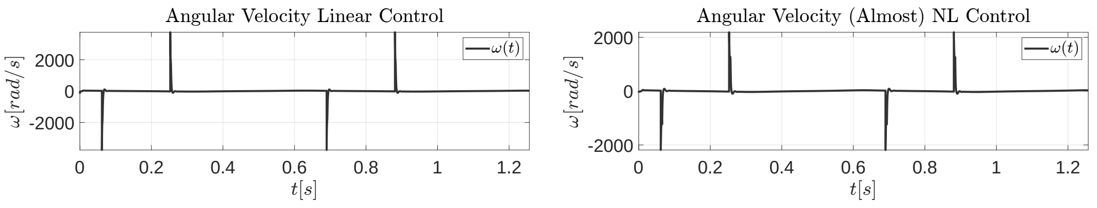


Figure 17: Angular velocities with $r = 1$

As we can see from the previous figures, **the unicycle follows the desired trajectory almost perfectly**, as the executed trajectory nearly completely overlaps with the desired one (depicted in gray).
Obviously, since the time considered is just over one second, the velocities are very high due to the need to execute the trajectory in a very short amount of time. In fact, very *high gains* are necessary to execute the desired trajectory in the best possible way. The lower these values are, the worse the trajectory tracking will be. However, no limits on the velocity values were imposed in the assignment.
From Figure 13, it is possible to notice that, in the case of the *(Almost) NonLinear Control*, the Unicycle tends to 'enter' the desired trajectory earlier with respect the *Linear* case, but this is only due to this particular case and initial position. In other cases they could enter exactly in the same way.
Moreover, from Figure 14 we can observe slight error peaks during the changes of

direction at the upper and lower parts of the curve.

Let us see now the example with $r = 10$ (an order of magnitude larger with respect the previous case).

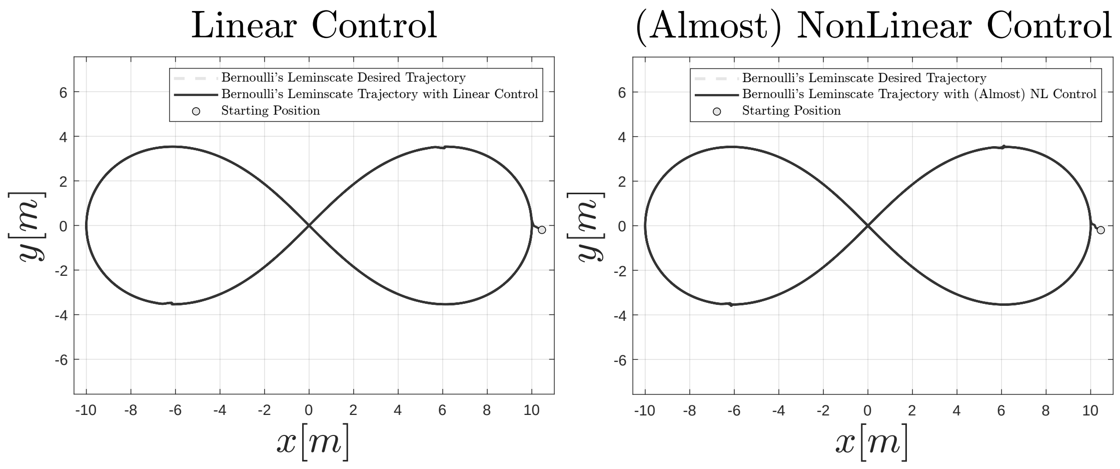


Figure 18: Bernoulli's Leminscate Trajectory with $r = 10$

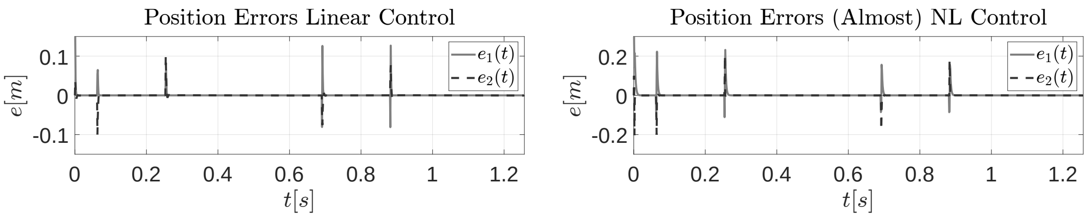


Figure 19: Position Errors with $r = 10$ (without considering the initial error due to the random position)

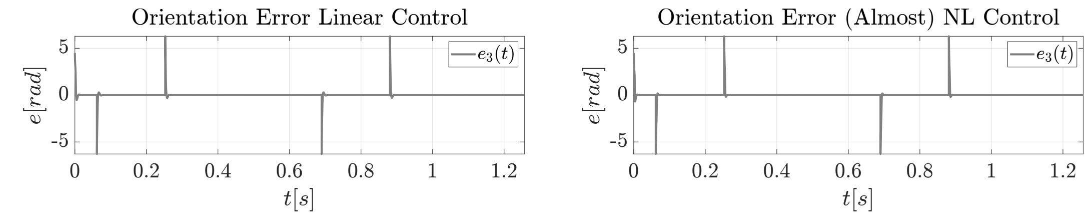


Figure 20: Orientation Error with $r = 10$

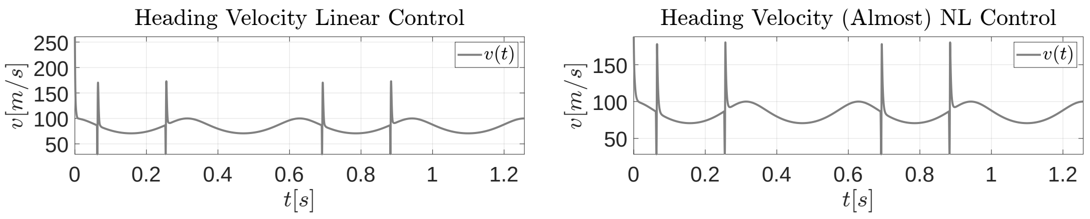


Figure 21: Heading velocities with $r = 10$

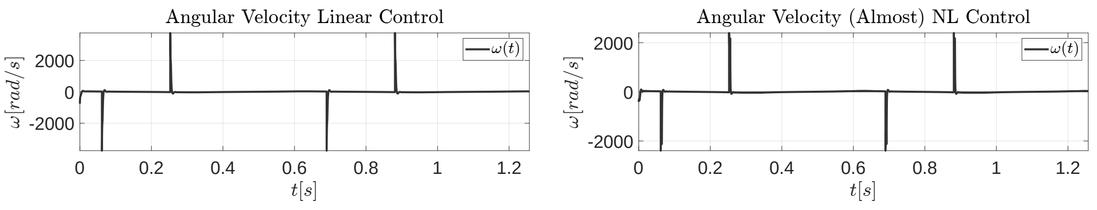


Figure 22: Angular velocities with $r = 10$

Comparing the two cases, it is possible to notice some substantial differences. First of all, with $r = 10$, since a longer trajectory must be covered in the same amount of time, the velocities will be significantly higher.
The position errors are also larger in this case but, clearly, given the overall scale of the trajectory, they are not very relevant.
In particular, in the *(Almost) NonLinear Control*, gains dependent on r were implemented in such a way as to optimally execute both trajectories with lower values than in the first case (such as $r = 0.1$), and higher values than in the second (such as $r = 100$, but also several orders of magnitude larger).
Regarding velocities and gain values, the same considerations as in the previous case can be made. Lower are $v_d(t)$ and $\omega_d(t)$, lower will be the gains depending on them.
Instead, for the orientation error, for both values of r, there are peaks in correspondence of the peaks of the angular velocity, that is in those points of the trajectory where there are abrupt changes in direction.
The critical areas of the trajectory, as can be seen from the trajectories plots, are particularly the one at the top right and the one at the bottom left, where, depending on the choice of the gains, for both controllers, the criticality may be more or less pronounced, as can be seen from Figure 23.

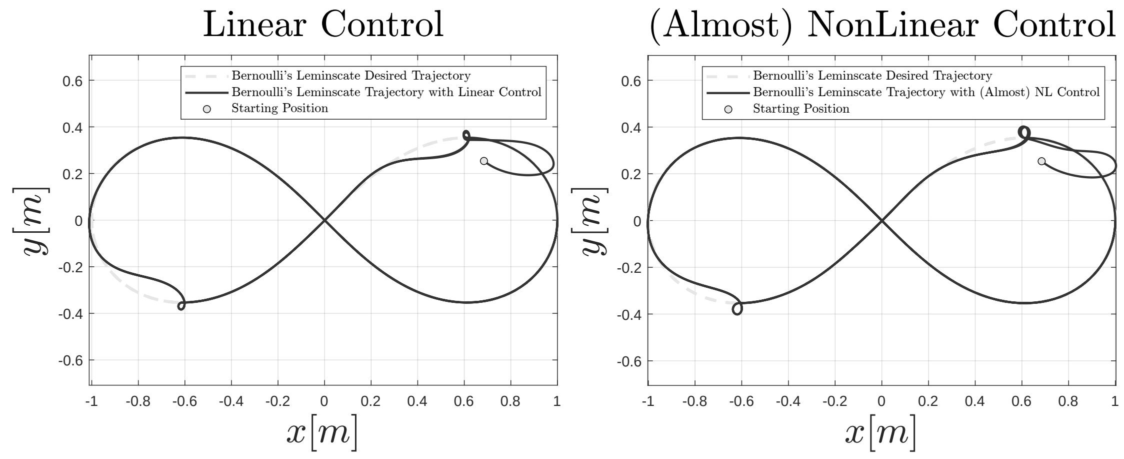


Figure 23: Bernoulli's Leminscate Trajectory with lower gains and $r = 1$

To conclude, the two controls, thanks to an *appropriate choice of parameters*, yield **perfectly overlapping results**, except for some subtle differences.
In particular, the *(Almost) NonLinear Control*, with the choice of gains made for this specific case, proves to be more optimal in terms of handling *abrupt variations* in such a way to attenuate them.
It can also be noted that the *(Almost) NonLinear Control* does not require gains as high as those needed by the *Linear* one. Therefore, from a purely practical viewpoint, it can be stated that, under the same conditions (gains more or less of the same values), the first one *performs better* also in this regard.
Obviously, depending on the application, such as the presence of velocity limits or other constraints, *the results may vary.*

Exercise n°4

Let us implement via *Simulink* the *Unicycle Posture Regulator* based on *Polar Coordinates* with the *State Feedback* computed through the *Runge-Kutta Odometric Localization Method*, starting from the configuration $q_i = [x_i \quad y_i \quad \theta_i]^T = [10 \quad 1 \quad \frac{\pi}{4}]^T$ to the final configuration $q_f = [x_f \quad y_f \quad \theta_f]^T = [0 \quad 0 \quad 0]$.

The new Polar Coordinates [5] are, as it is possible to see from Figure 24

$$\begin{cases} \rho = \sqrt{x^2 + y^2} \\ \gamma = atan2(y, x) - \theta + \pi \\ \delta = \gamma + \theta \end{cases}$$

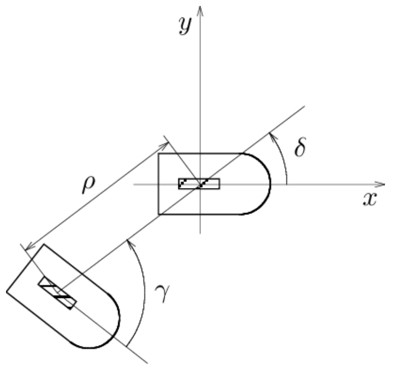


Figure 24: Polar Coordinates for the Unicycle

It is possible to design the feedback controller

$$v = k_1 \rho cos\gamma \qquad \omega = k_2\gamma + k_1 sin\gamma cos\gamma(1 + k_3\frac{\delta}{\gamma})$$

with $k_1, k_2, k_3 > 0$ in such a way to make the system converging to $[\rho \quad \gamma \quad \delta]^T = [0 \quad 0 \quad 0]^T$. Thanks to this, in the new Closed Loop System there are anymore *no singularities* and it is possible to bring the robot in the final desired configuration. However it is necessary a *Localization* procedure. In this case the 2^{nd} *order Runge-Kutta Approximation* has been used:

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

where v_k and ω_k are reconstructed from the proprioceptive sensors (not implemented in Simulink and, for this reason, they are not taken from the output of

[5]The Unicycle Kinematic Model in Polar Coordinates is $\begin{cases} \dot{\rho} = -vcos(\gamma) \\ \dot{\gamma} = \frac{sin\gamma}{\rho}v - \omega \\ \dot{\delta} = \frac{sin\gamma}{\rho}v \end{cases}$

the Unicycle 'stateDot' but the velocities used were those that provide the best possible approximation of them). Let us see now some relative plots.

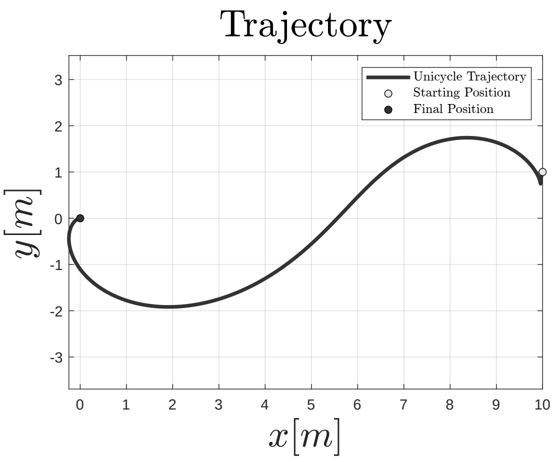


Figure 25: Trajectory

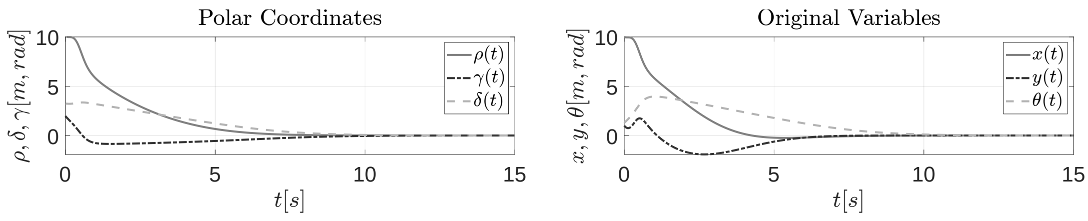


Figure 26: On the left, Polar coordinates trend and, on the right, Original Variables trend

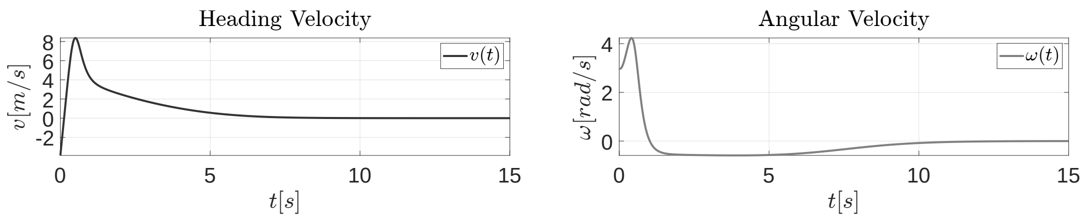


Figure 27: Heading and Angular Velocities

As can be seen from the graphs above, **the variables are perfectly regulated to zero** with a settling time of about $10s$. In fact, *zero regulation of the polar coordinates*, as can easily be verified also from the formula, *also leads to zero regulation of the initial variables.* The choice of gains is of crucial importance, as even a small variation in them results in a significant change in the system's behavior in terms of settling time, velocity peaks, and oscillations. In this case $k_1 = 1$, $k_2 = 2$ and $k_3 = 1$ have been used.