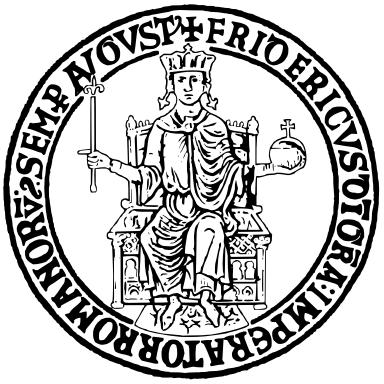
Università degli Studi di Napoli Federico II



Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione

Corso di Laurea in Ingegneria dell'Automazione e Robotica

FIELD AND SERVICE ROBOTICS

Homework 2 Report

Student:
Emanuela Varone

Matricola:
P38000284

Contents

1 Path planning algorithm for a unicycle

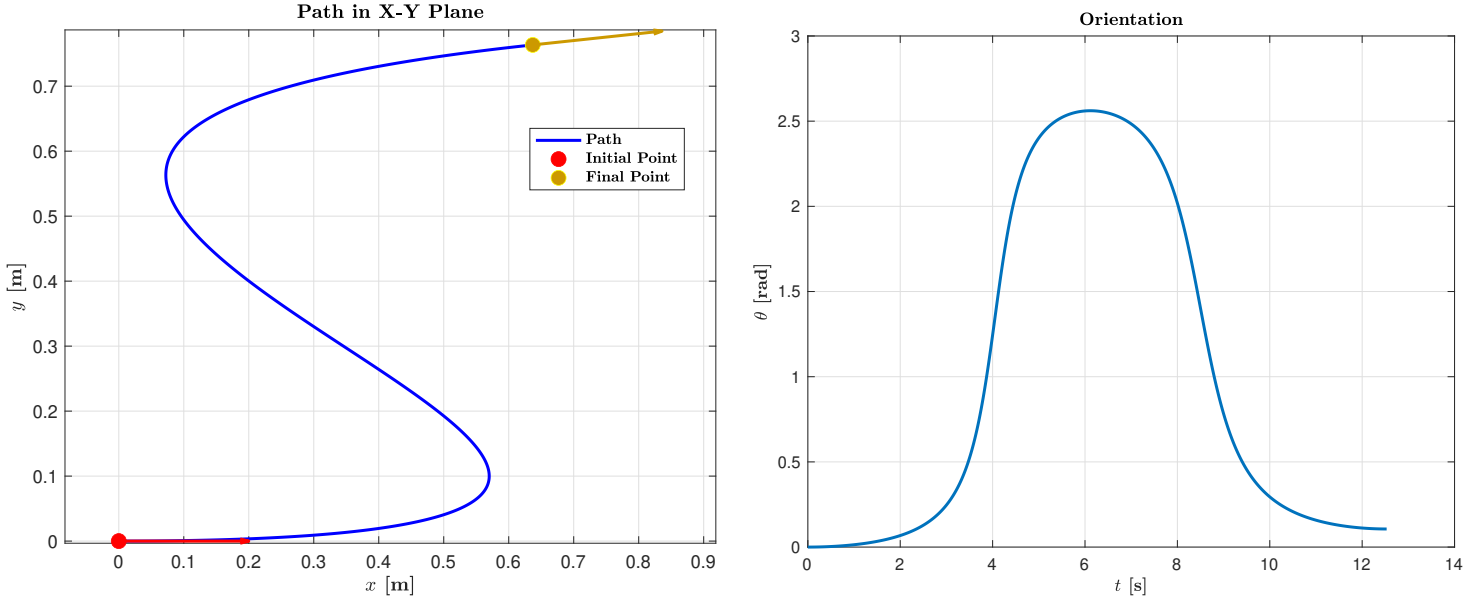
The goal of this section is to generate a smooth trajectory that leads the unicycle from an initial configuration $\mathbf{q}_i$ to a random final configuration $\mathbf{q}_f$ while respecting velocity constraints:

- **Linear velocity:** $|v(t)| \leq 0.5\,\mathrm{m/s}$
- **Angular velocity:** $|\omega(t)| \leq 2\,\mathrm{rad/s}$

MATLAB code (see Point_1_HW2.mlx file in HW2_FSR_Emanuela_Varone/Point_1 folder) is structured in sections:

- **Initialization** of q_i, generation of q_f and definition of polynomial coefficients.
- **Loop scaling**: computation of derivatives with respect to s and time, determination of $v(t)$ and $\omega(t)$, checking constraints and updating T.
- **Plot**: six plots are produced, including velocities without scaling, with scaling, path and orientation .

1.1 Path and orientation



(a) Path in X-Y Plane

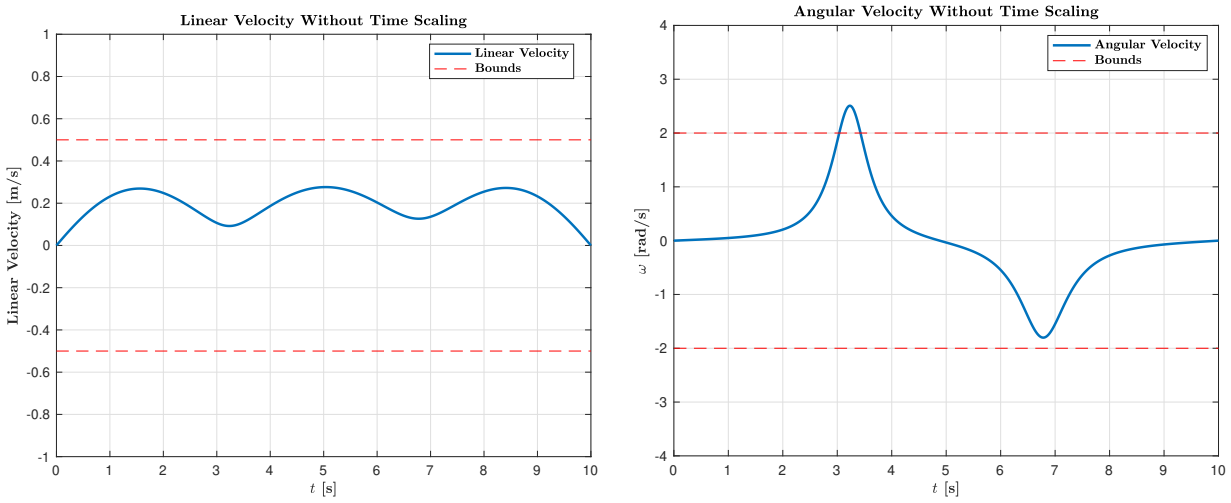
(b) Orientation $\theta(t)$ over time

Figure 1: Path and orientation visualization

The trajectory shown in 1a takes on an S shape, typical of paths that connect different initial and final orientations in a continuous and fluid manner. The pattern is smooth, without abrupt changes in direction and suggests a good compromise between shortness and smoothness of motion. Regarding the orientation $\theta(t)$, represented in 1b, a bell-shaped curve is observed to evolve monotonically and symmetrically, confirming the effectiveness of the parameterization adopted and the consistency with the geometry of the trajectory. **NOTE:** The attached video ("robot animation.mp4") in the HW2_FSR_Emanuela_Varone/Point_1/VIDEO folder shows the trajectory performed by the robot.

1.2 Velocity without scaling

As it is possible to notice by observing Figs.2a-2b, while v satisfies the imposed bounds, $|\omega|$ exceeds 2.rad/s, particularly at the beginning of the middle phase where the trajectory is more curved, making it necessary to lengthen the time.



(a) Linear Velocity

(b) Angular velocity

Figure 2: Velocities without scaling

1.3 Velocity with scaling

Since, as noticed in above subsection, the angular velocity does not satisfy the upper bound, it was necessary to slow down the timing law via uniform scaling. The scaling loop is executed until speed constraints are met. In each iteration are performed:

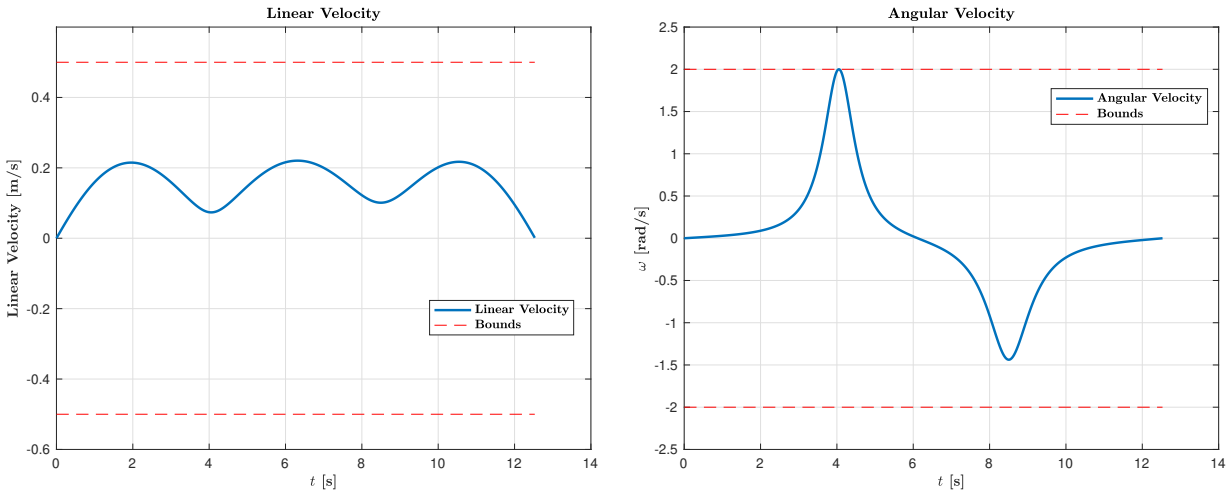
1. Computation of the velocities $v(t)$ and $\omega(t)$ along the trajectory, using the derivatives with respect to s and the temporal law $s(t)$;
2. Evaluation of the kinematic peaks: $\max_{\text{v}} = \max|v(t)|$ and $\max_{\omega} = \max|\omega(t)|$;
3. Calculation of the scaling factor:
$$\alpha = \max\left(\frac{\max_{\text{v}}}{0.5}, \frac{\max_{\omega}}{2}\right);$$
4. Updating the duration: $T \leftarrow \alpha\, T$ and recalculating the coefficients of the cubic time law $s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$:
$$a_0 = 0, \quad a_1 = 0, \quad a_2 = \frac{3}{T^2}, \quad a_3 = -\frac{2}{T^3};$$
5. Repeat the cycle until the constraints are satisfied.

This approach guarantees a dynamic adaptation of the path duration to the imposed kinematic constraints, keeping the shape of the trajectory unchanged and acting only on the travel time. As result, by increasing T, the velocities were reduced uniformly. As shown in Table 1, the time scaling ensured:

Parameter	**Value**
Initial configuration (x_i, y_i, θ_i)	$(0.0000, 0.0000, 0.0000)$
Final configuration (x_f, y_f, θ_f)	$(0.6370, 0.7635, 0.1062)$
Initial time duration T	10.0000 s
Maximum linear velocity $v_{\max}$	0.27631 m/s
Maximum angular velocity $\omega_{\max}$	2.5073 rad/s
Scaling factor	1.2537
New time duration T_{scaled}	12.5366 s
$v_{\max}$ after scaling	0.2204 m/s
$\omega_{\max}$ after scaling	2.0000 rad/s

Table 1: Simulation Results

The total time was scaled from $T_0 = 10$s to $T_f = 12.5366$s : the result is a velocity profile that stays within the imposed limits. In particular, the profile of ω (Figure 3b) shows a slight flattening in the peaks, while maintaining the overall shape.



(a) Linear Velocity

(b) Angular velocity

Figure 3: Velocities with scaling

2 Input/Output linearization control

Given the trajectory in the previous section, the goal of this section is to implement an input/output linearization control approach to control the unicycle's position. The results discussed in the following paragraph refer to Point_2_HW2.mlx and Point2_IO_Linearization_HW2.slx in HW2_FSR_Emanuela_Varone/Point_2 folder.
The unicycle model is controlled using an input/output linearization strategy applied to a virtual output point B located along the sagittal axis of the robot, at a distance b from the center of the wheel. Through differential flatness and feedback linearization, it is possible to design for the output variables y_1 and y_2, corresponding to the Cartesian coordinates of point B, a linear controller of the form:

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

Have been tested and compared the results for three different values:

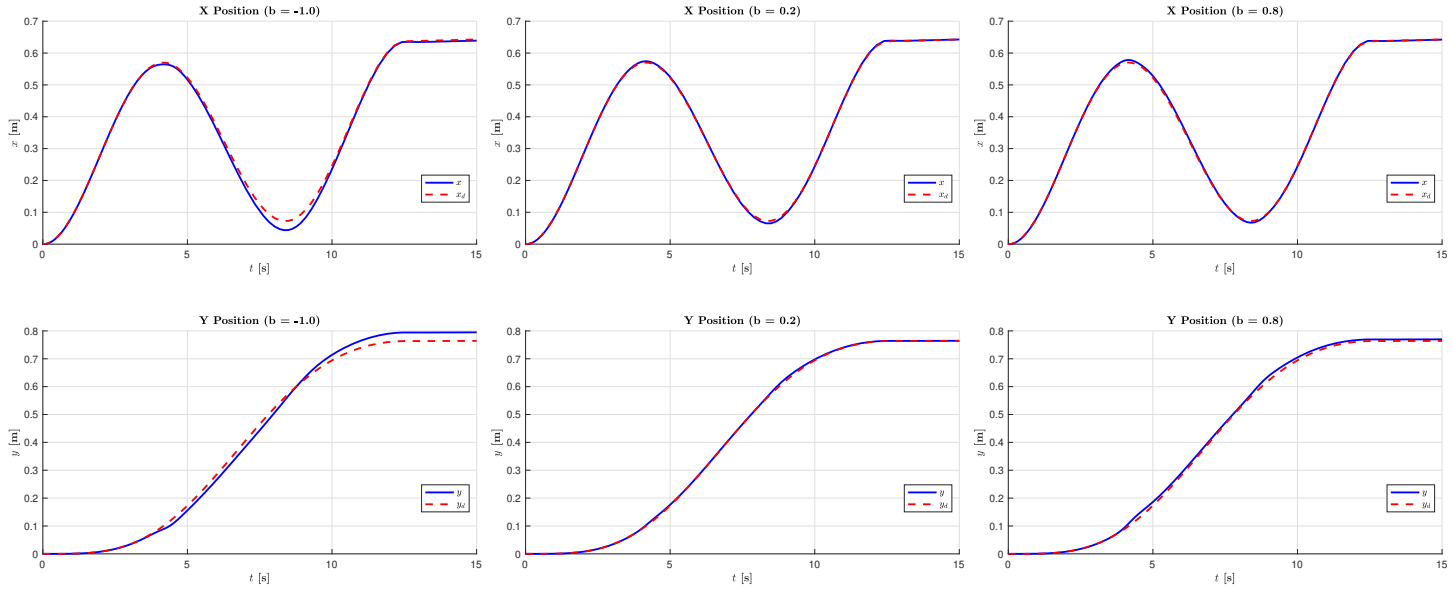
1. b = -1.0
2. b = 0.2
3. b = 0.8

This choice allowed to observe how shifting the control point behind, near or ahead of the robot's center influences tracking accuracy and control effort. To select the three different values of b and compare the relative results produced, a Multiport Switch block was used in Simulink.

2.1 Tracking Accuracy

Fig.4 shows x and y trajectories compared with reference ones for the three values of b chosen.

- **For** $b = -1$ (plot at left): A delay in convergence is observed for both x and y. The actual trajectory follows the reference with some initial tracking error, more visible in the y component. The system appears more "slow" to react and less accurate.
- **Per** $b = 0.2$ (central plot): The trend is very close to the reference trajectory, with very little tracking error. The behavior is well balanced, an indication that this value of b represents a good compromise between speed and accuracy. It could be considered an "optimal" or close to the best value for this scenario.
- **For** $b = 0.8$ (right plot): Again, the tracking is good, but a very slight advance of the actual trajectory with respect to the reference can be seen, especially in the y component. This suggests a more aggressive or reactive system, which, however, could become unstable for even higher values of b.



(a) b = -1 (b) b = 0.2 (c) b = 0.8

Figure 4: Comparison of trajectories obtained for different values of b

As b increases, the system becomes more responsive, but it also risks anticipating the reference too much. For negative values of b, the system is slower and less accurate. An intermediate value such as b=0.2 is a good compromise between speed and accuracy. This can also be deduced by analyzing Fig.5

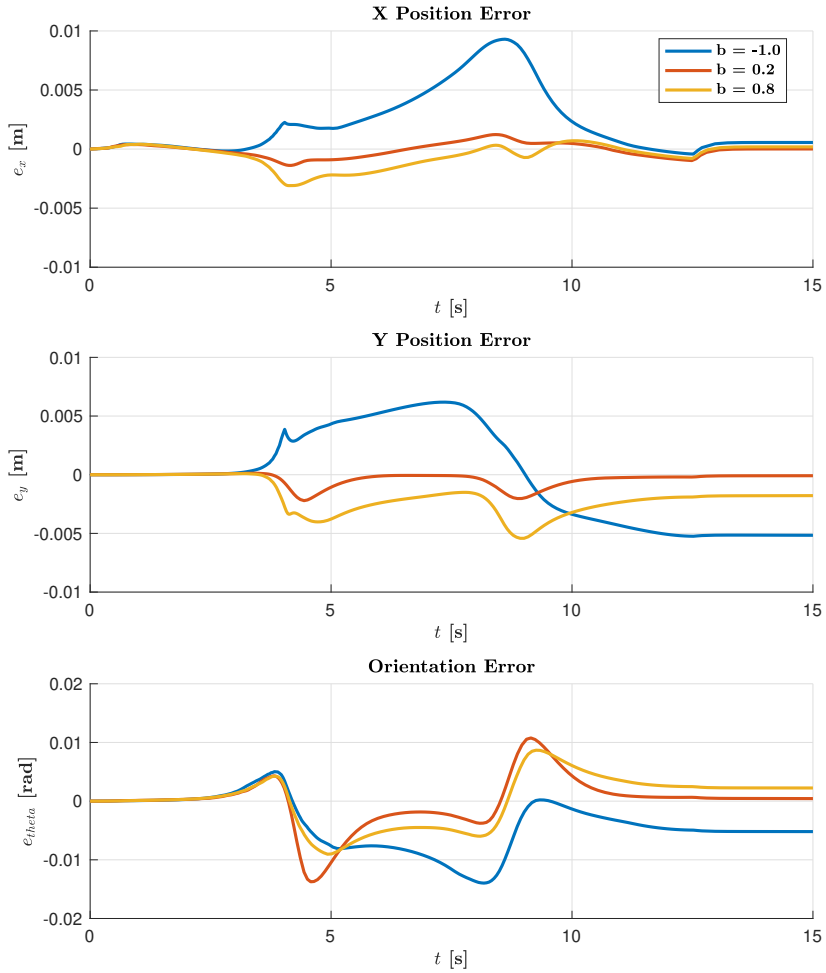


Figure 5: Comparison of errors obtained for different values of b

Looking at the third plot in Fig.5 it can be seen that total zero convergence of the orientation error, although this is very small

(in particular in the case $b = 0.2$), is not obtained. This is due to the fact that the control under consideration does not act on the orientation.

2.2 Velocities

Fig.6 shows the trends of linear and angular velocities for the three values of b chosen.

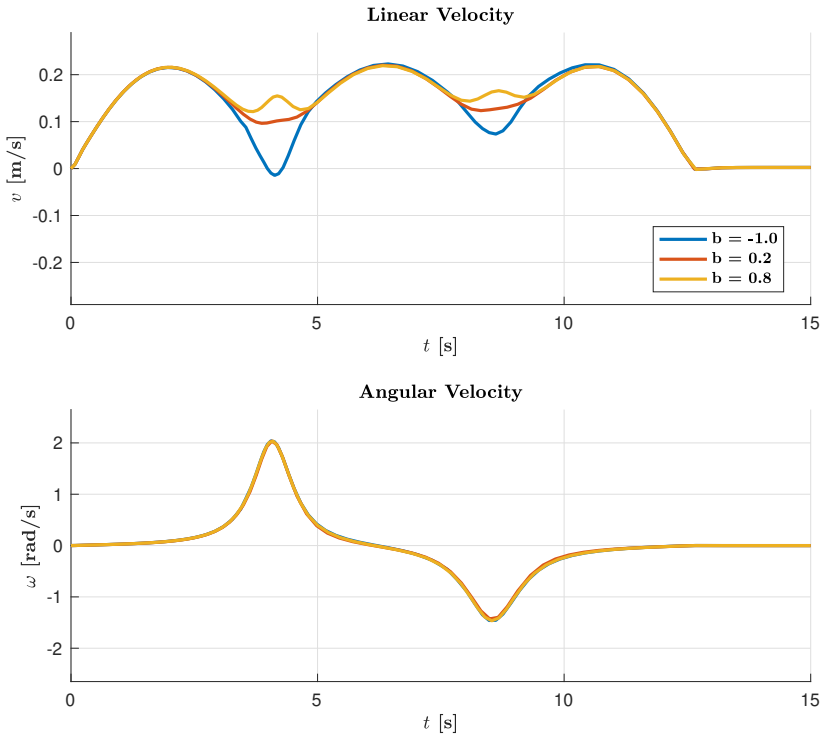


Figure 6: Comparison of velocities obtained for different values of b

Regarding the linear velocity plot (top): When $b = -1.0$, it is possible to see significant oscillations with deeper dips, reaching negative values around $t = 3s$. As b increases to 0.2, the oscillations become more moderate. At $b = 0.8$, the linear velocity profile shows a behavior with less pronounced fluctuations. After about $t = 10s$, all three cases converge to the same steady-state value (zero). Regarding the angular velocity plot (bottom): interestingly, the angular velocity appears almost identical for all three values of b. All curves show a positive peak around $t = 3s$ and a negative peak around $t = 6-7s$. The angular velocity also converges to zero after about $t = 10s$. These observations suggest: when b is negative (point B is behind the wheel center), the system exhibits more aggressive control actions with higher amplitude oscillations in linear velocity. As b becomes more positive (point B moves further ahead of the wheel center), the control becomes smoother with reduced oscillations. There appears to be a stability trade-off: larger positive values of b lead to smoother linear velocity profiles, suggesting better stability and less aggressive control actions. The convergence time seems relatively unaffected by the value of b, as all cases stabilize around $t = 10s$.

2.3 Tracking Error Analysis for Point B

By observing the position error plots of point B Fig.7, several important considerations can be made regarding the system's behavior as the parameter b varies:

y_1 Position Error (top plot):

- With $b = -1.0$, oscillations are more pronounced, alternating between positive and negative values, with peaks reaching approximately ± 0.01 m.
- With $b = 0.2$, the oscillations are significantly smaller, with errors remaining within approximately ± 0.002 m.
- With $b = 0.8$, the amplitude of the error increases again, with marked negative peaks (down to around -0.007 m).

y_2 Position Error (bottom plot):

- With $b = -1.0$, an oscillatory behavior is observed with a significant positive peak around $t = 7\,\text{s}$.
- With $b = 0.2$, the error is minimized compared to other values of b, staying within the range of $\pm 0.002 \times 10^{-3}\,\text{m}$.
- With $b = 0.8$, more pronounced negative peaks are observed (down to approximately $-5 \times 10^{-3}\,\text{m}$).

From the comparison of the results obtained it emerges that:

- The intermediate value $b = 0.2$ appears to offer the best compromise in terms of control performance, producing the smallest position errors along both axes.
- Both ($b = -1.0$) and ($b = 0.8$) values result in larger tracking errors, although with different characteristics.
- A clear relationship exists between the parameter b and system stability: placing the controlled point B too far from the wheel's center (in either direction) tends to reduce tracking precision.
- For all values of b, the system eventually converges to zero error around $t = 10\,\text{s}$, demonstrating overall stability of the control approach.
- The behavior of the tracking errors is consistent with the observed velocity profiles: values of b that lead to more oscillatory linear velocities also correspond to higher position tracking errors.

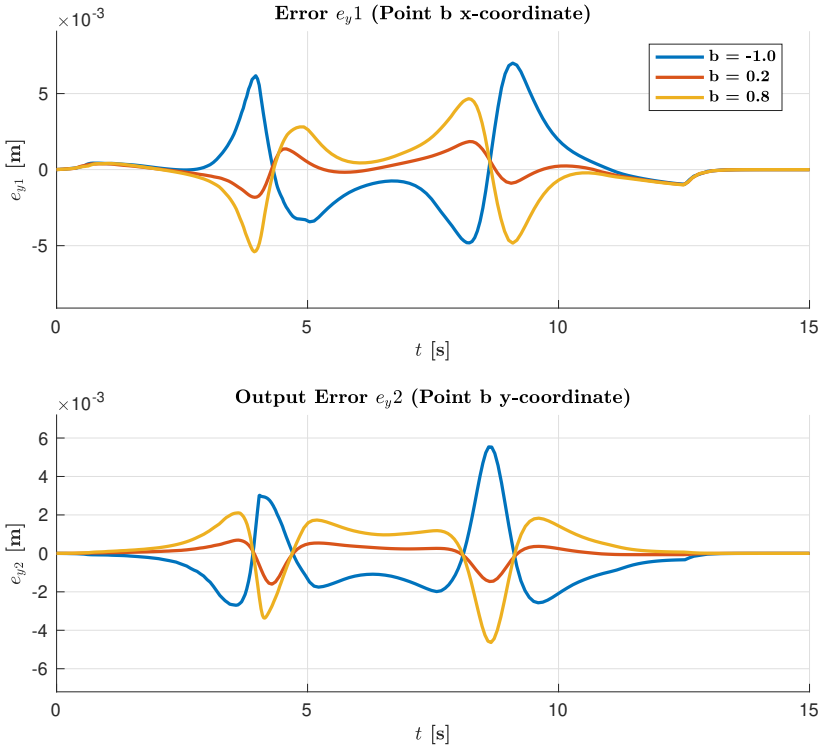


Figure 7: Comparison of tracking error for Point B obtained for different values of b

These results confirm that in input/output linearization for unicycle control, the choice of the controlled point B along the sagittal axis is crucial for optimizing performance. In this specific case, a moderately positive value ($b = 0.2$) appears to represent the optimal point for trajectory tracking accuracy.

3 Linear and Almost Nonlinear Controllers for a Unicycle Robot Following a Bernoulli Lemniscate Trajectory

This section presents the design and performance analysis of two different control approaches for a unicycle robot following a Bernoulli lemniscate trajectory. The two controllers implemented are:

- A linear controller based on approximate linearization;
- An (almost) nonlinear controller.

The performance of both controllers is evaluated across two distinct scenarios with different scale parameters for the reference trajectory, allowing to assess their behavior under varying conditions. The implementation was carried out in MATLAB/Simulink. The workflow consists of the following steps:

1. Generation of the reference trajectory (Bernoulli lemniscate), computation of the desired velocities and randomly calculation of the initial position of the unicycle within 0.5 meters of the desired initial one;
2. Implementation of the controllers in Simulink;
3. Analysis of the results.

The MATLAB script generates the reference trajectory, calculates the desired velocities, sets up the simulation parameters and performs the visualization of the results. The Simulink model contains the implementation of both controllers and the unicycle dynamics. The results discussed in the following paragraph refer to Point_3_HW2.mlx and Point_3.slx in HW2_FSR_Emanuela_Varone/Point_3 folder.

3.1 Reference Trajectory

The reference trajectory is defined by a Bernoulli lemniscate with the following parametric equations:

$$x_d(t) = \frac{r\cos((\alpha+1)t)}{1+\sin^2((\alpha+1)t)}$$
$$y_d(t) = \frac{r\cos((\alpha+1)t)\sin((\alpha+1)t)}{1+\sin^2((\alpha+1)t)}$$

where:

- $\alpha = 4$ (derived from the last matriculation number)
- $t \in \left[0, \frac{4\pi}{\alpha+1}\right]$
- r is a scale parameter that takes two different values:
  - $r_1 = 2.0$ (larger scale)
  - $r_2 = 0.2$ (smaller scale, one order of magnitude difference)

With $r_1 = 2.0$, the system must handle high linear velocities and follow a longer trajectory, testing the controllers' ability to maintain stable performance over longer distances. With $r_2 = 0.2$, the challenges shift to motion precision and the ability to execute tighter turns in a smaller space, where relative errors become more significant.

3.2 Linear Controller Implementation

The linear controller based on approximate linearization was implemented as follows:

```
function [v,omega] = control_inputs_app_lin(x_d,y_d,theta_d,v_d,omega_d,x,y,theta)
    zita = 0.6;  % Damping ratio
    a = 1;       % Natural frequency parameter
    k1 = 2*zita*a;
    k3 = k1;
    k2 = (a^2-0.01*abs(omega_d^2))/abs(v_d);  % Gain adjusted based on reference velocities

    M = [cos(theta) sin(theta) 0; -sin(theta) cos(theta) 0; 0 0 1];
    displacement = [x_d-x; y_d-y; theta_d-theta];
    e = M*displacement;
```

```
    % Control inputs calculation
    u1 = -k1*e(1);
    u2 = -k2*e(2)-k3*e(3);
    v = v_d*cos(e(3))-u1;
    omega = omega_d-u2;
end
```

In this implementation, the control gains are selected based on a second-order system approach, where ζ represents the damping ratio and a is related to the natural frequency. The gain k_2 is dynamically adjusted based on the reference angular velocity to maintain stability during sharp turns.

3.3 (Almost) Nonlinear Controller Implementation

The almost nonlinear controller was implemented with adaptive gains that scale with the reference velocities:

```
function [v,omega] = control_inputs_almost_nl(x_d,y_d,theta_d,x,y,theta,v_d,omega_d)
    % Adaptive gains that scale with reference velocities
    k1 = 3+0.2*abs(v_d)+0.1*abs(omega_d);
    k2 = 3.5;
    k3 = 5+0.2*abs(omega_d)+0.3*abs(v_d);

    M = [cos(theta) sin(theta) 0; -sin(theta) cos(theta) 0; 0 0 1];
    displacement = [x_d-x; y_d-y; theta_d-theta];
    e = M*displacement;
    sinc_e3 = sin(e(3))/e(3);

    % Control inputs calculation
    u1 = -k1*e(1);
    u2 = -k2*v_d*(sinc_e3)*e(2)-k3*e(3);

    v = v_d*cos(e(3))-u1;
    omega = omega_d-u2;
end
```

The control gains (k_1, k_3) adaptively increase with the reference velocities, allowing for more aggressive corrections during high-speed maneuvers while maintaining stability at lower speeds.

3.4 Results and Analysis

The simulation results are presented for both values of the scale parameter r. For each case, the position tracking, position errors and control inputs (linear and angular velocities) are analyzed. NOTE: As required by specification, the results produced in the interval $t \in \left[0, \frac{4\pi}{5}\right]$ were examined.

3.4.1 Position Tracking and Errors

Fig.8 shows x and y trajectories compared with reference ones for $r = 0.2$.

- **X-axis Tracking**: In the case of the **linear controller**, a significant phase delay is observed between the desired trajectory (x_d) and the actual trajectory (x_l). The amplitude of the actual oscillation is also greater than the desired one, indicating overshoot. With the **nonlinear controller**, the tracking performance is noticeably better. The phase delay is reduced, and the amplitude of the tracked signal more accurately follows the desired one, especially at peak points.
- **Y-axis Tracking**: For the **linear controller**, a considerable deviation between the desired trajectory (y_d) and the actual trajectory (y_l) is observed. The actual trajectory shows a much larger amplitude and does not adequately follow the

reference. The **nonlinear controller** demonstrates a significant improvement in y-axis tracking. The trajectory gradually converges towards the desired one, with a noticeable reduction in tracking error towards the end of the observed time interval.

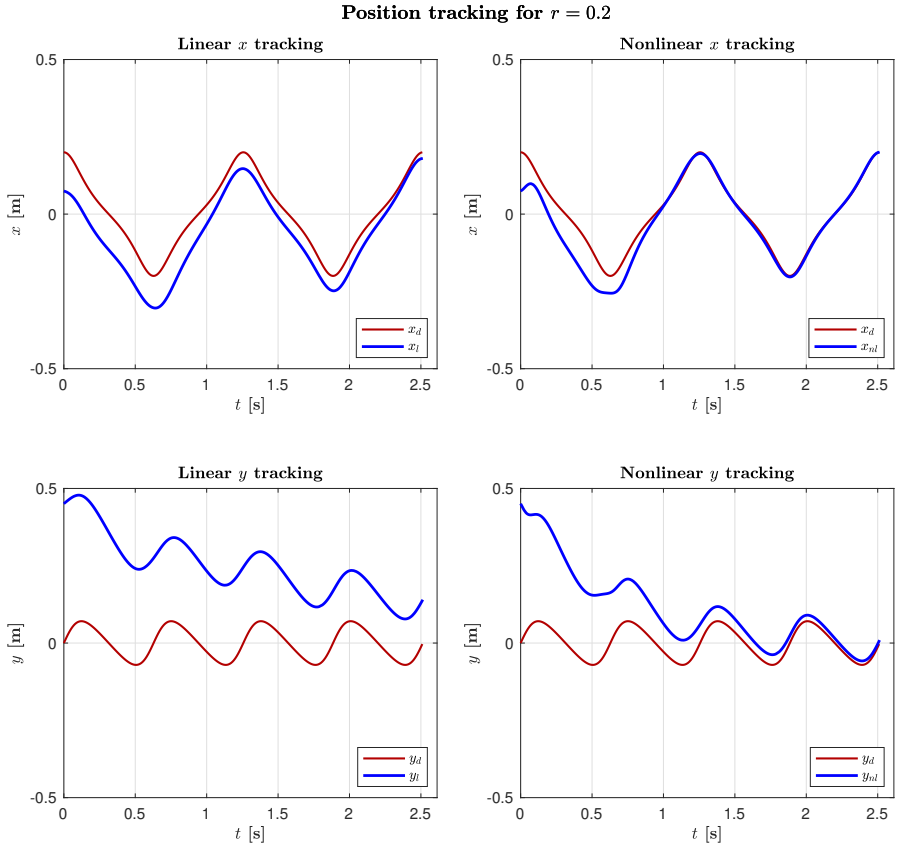


Figure 8: Tracking performance with $r = 0.2$

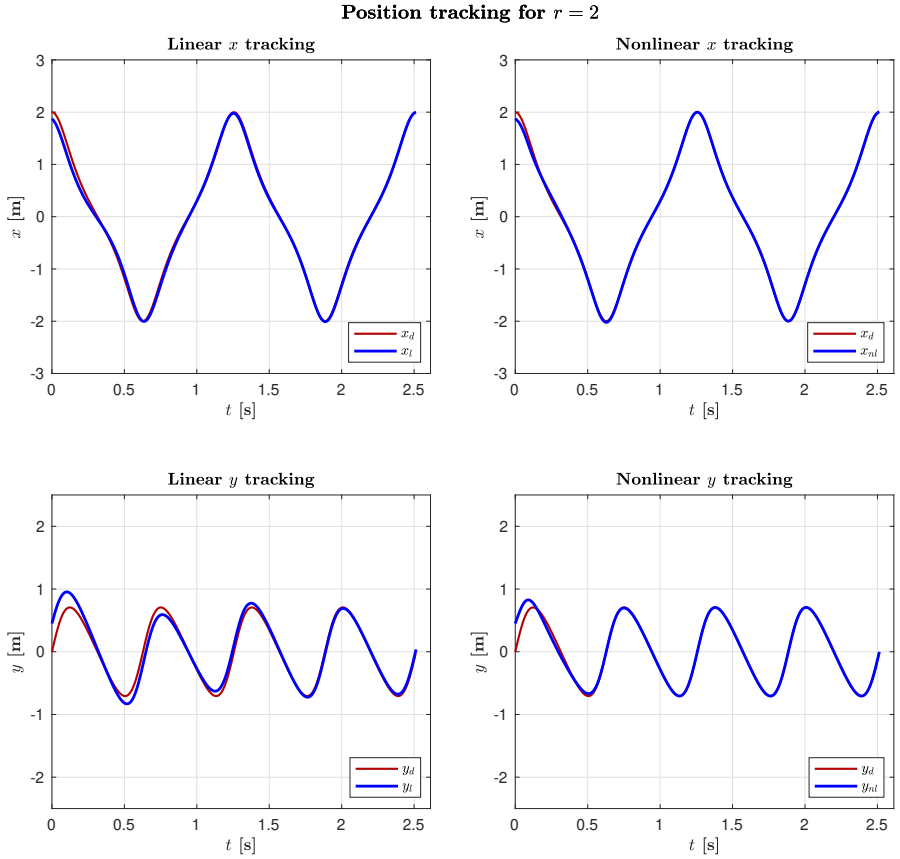


Figure 9: Tracking performance with $r = 2$

Fig.9 shows x and y trajectories compared with reference ones for $r = 2$.

- **X-axis Tracking**: With a larger radius, both controllers exhibit similar and substantially improved performance in x-axis tracking. The desired and actual trajectories almost coincide, with minimal and hardly noticeable differences, between the linear and nonlinear controllers in this case.
- **Y-axis Tracking**: For the **linear controller**, y-axis tracking is significantly improved compared to the $r = 0.2$ case, with minimal and well-contained tracking error.The **nonlinear controller** shows optimal performance, with near-perfect tracking of the desired trajectory along the y-axis.

From the analysis of the plots obtained it is possible to deduce the following observations. The performance of both controllers improves significantly when the reference radius increases from 0.2 to 2. This suggests that both controllers are more effective for trajectories with larger rays, likely because the system dynamics becomes more linear or easier to handle. The nonlinear controller shows a clear advantage over the linear one, especially in the case of $r = 0.2$ and particularly in y-axis tracking. This aligns with theoretical expectations, as the nonlinear controller better handles the intrinsic nonlinearities of the unicycle model. In both cases, y-axis tracking presents more challenges compared to the x-axis, especially for the linear controller at $r = 0.2$. The nonlinear controller demonstrates greater robustness to changes in the reference radius, maintaining good performance with both $r = 0.2$ and $r = 2$. These considerations can also be deduced by analyzing Figs.10-11:

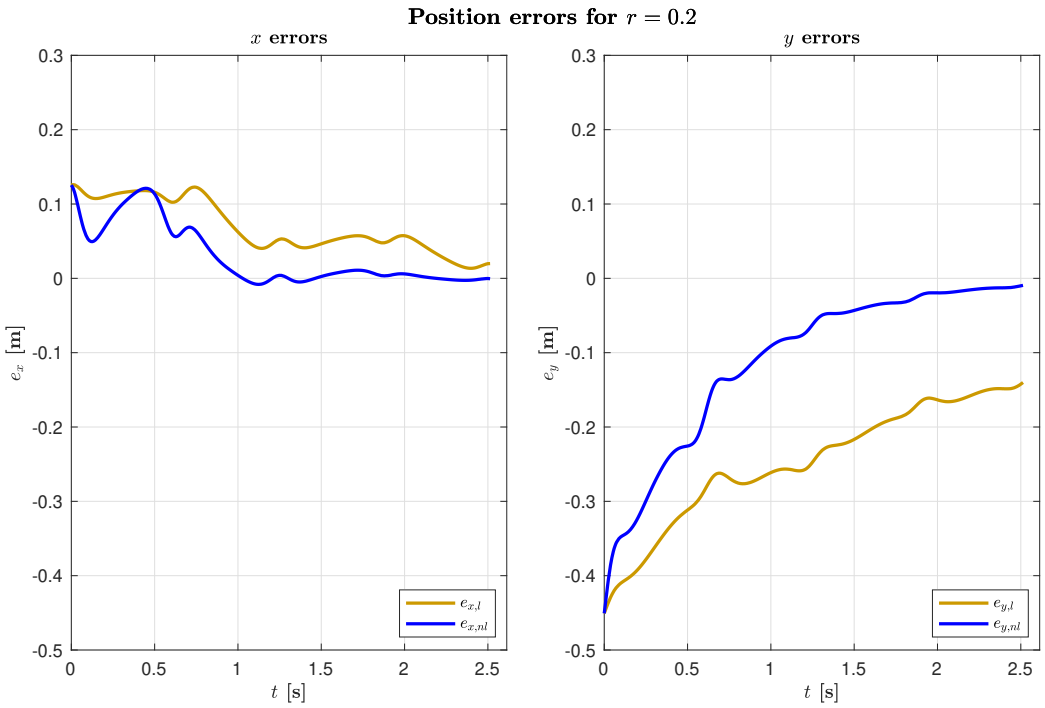


Figure 10: Position error for $r = 0.2$

In particular, with reference to Fig.10, it can be noted that:

- on the x-axis the nonlinear controller shows a significantly better behavior than the linear controller. The error of the nonlinear controller converges more quickly towards values close to zero after about 1 second. The linear controller maintains a positive residual error (about 0.05 m) even towards the end of the observation interval, indicating a slower and less effective convergence. Both controllers start with a positive error of about 0.12 m, but follow different convergence dynamics.
- on the y-axis both controllers start with a negative error (about -0.45 m), highlighting a significant initial difficulty in following the reference trajectory. The nonlinear controller shows a faster convergence, managing to bring the error almost to zero by the end of the interval. The linear controller shows a much slower and incomplete convergence, remaining at about -0.15 m at the end of the observation interval. The performance difference between the two controllers is particularly marked on the y-axis with small r.

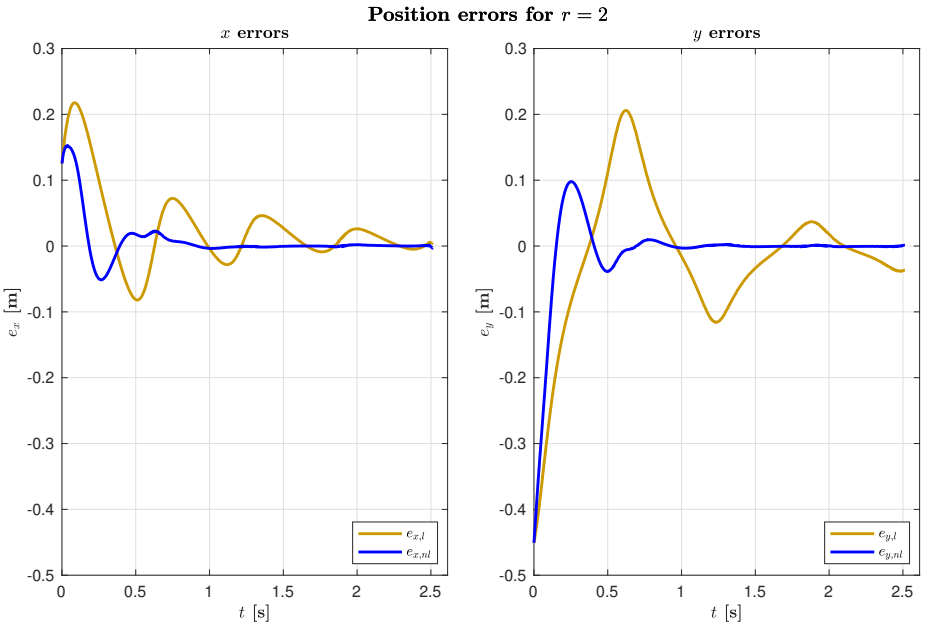


Figure 11: Position error for $r = 2$

With reference to Fig.11, it can be noted that:

- on the x-axis, with a larger radius, both controllers show an initial oscillatory behavior, but with different characteristics. The nonlinear controller has smaller amplitude oscillations and converges more quickly (around 0.8 seconds). The linear controller shows more marked oscillations with a larger positive peak (around 0.21 m) followed by a negative oscillation. After about 1.5s, both controllers manage to stabilize with errors close to zero, but the nonlinear controller maintains smaller errors.

- on the y-axis, both controllers start with a significant negative error (around -0.45 m), similar to the case with r = 0.2. The nonlinear controller converges quickly, reaching a stable behavior within 0.5-0.7 seconds with minimal errors. The linear controller has larger amplitude oscillations, with a large positive peak (around 0.2 m) followed by negative oscillations. Towards the end of the observation interval, the linear controller continues to show small oscillations, while the nonlinear controller maintains a near-zero error.

3.4.2 Velocities

The analysis of the velocity plots Figs.12-13 produced for the two different values of r and for the two controllers is provided below. Fig.12 shows the trends of linear and angular velocities over time for $r = 0.2$. It can be noticed that:

- **Linear velocity**: the velocity profile of the linear controller shows relatively small oscillations around a mean value of about 0.9 m/s, with a range between 0.7 and 1.0 m/s. A fairly regular behavior with gradual variations is noted. For the nonlinear controller, the velocity profile shows much more marked oscillations, with an initial negative value (-1.0 m/s) followed by a significant positive peak (about 1.4 m/s) and a subsequent drastic decrease to almost 0 m/s around t = 0.5s. This behavior highlights a more aggressive control strategy in the initial phases.

- **Angular velocity**: Surprisingly, both controllers show almost identical angular velocity profiles, oscillating between about -15 rad/s and +15 rad/s with a regular sinusoidal behavior. This similarity suggests that the main difference between the two control approaches arises primarily in the management of linear rather than angular velocity for tracking with small radius.

Fig.13 shows the trends of linear and angular velocities over time for $r = 2$. It can be seen that:

- **Linear velocity**: the velocity profile of the linear controller shows a regular oscillation between about 7.2 m/s and 10.1 m/s, with an almost sinusoidal trend around an average value of about 8.7 m/s. For the nonlinear controller, a distinctive initial behavior is observed, with a lower starting value (about 7.8 m/s) followed by a transient in the first 0.5 seconds. Subsequently, the profile converges to a trend very similar to that of the linear controller. The difference in scale with

respect to the r = 0.2 case is remarkable: here the linear velocities are about 10 times greater, consistently with the increased reference radius.

- **Angular velocity**: Also in this case, the angular velocity profiles of the two controllers are very similar to each other, oscillating between about -15 rad/s and +15 rad/s. A slight difference can be noticed in the initial phase, where the nonlinear controller exhibits a slightly different peak and a more abrupt transition towards negative values.

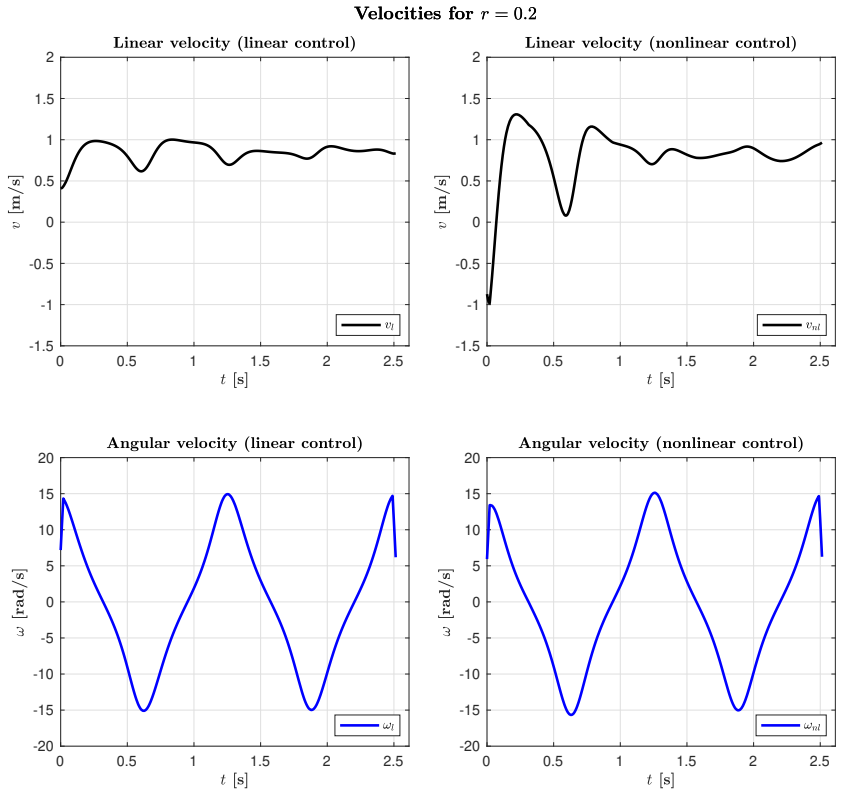


Figure 12

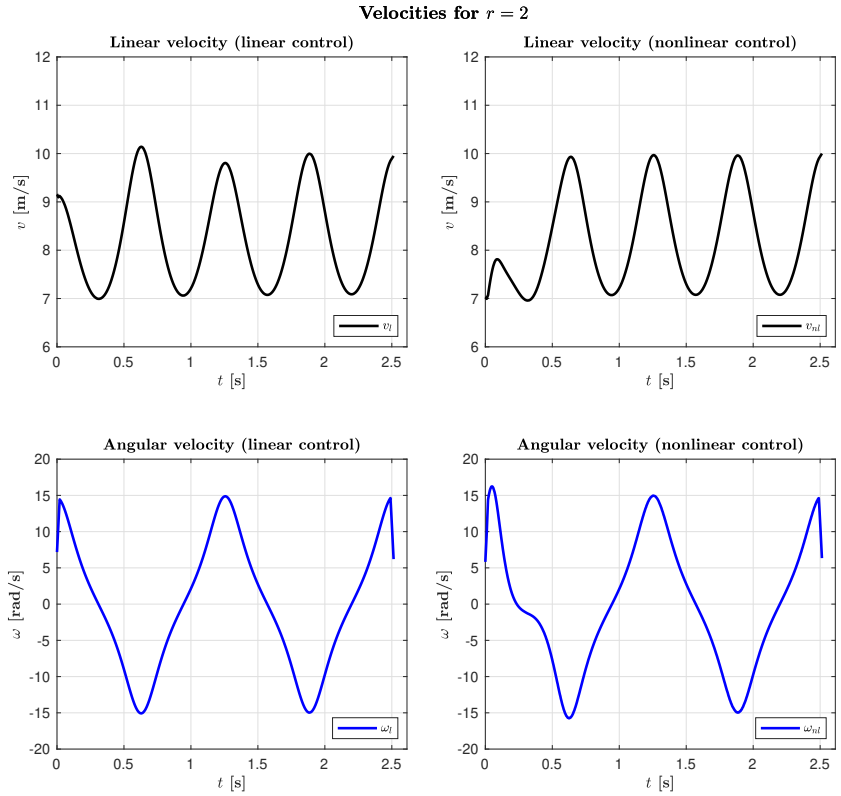


Figure 13

4 Unicycle posture regulator based on polar coordinates and Runge-Kutta odometric localization method

This section presents a detailed analysis of the implementation and performance of a unicycle posture regulator based on polar coordinates. The control system uses a state feedback approach computed through the Runge-Kutta odometric localization method. The primary objective is to drive a unicycle robot from an initial configuration ($q_i = [x_i, y_i, \theta_i]^T = [5, 1, \pi/4]^T$, with $\alpha = 4$ being the last digit of the matriculation number) to a target configuration ($q_f = [x_f, y_f, \theta_f]^T = [0, 0, 0]^T$) with high precision in both position and orientation.
The control system was implemented using MATLAB/Simulink (the results discussed in the following paragraph refer to Point_4_HW2.mlx and Point_4.slx in HW2_FSR_Emanuela_Varone/Point_4 folder). The implementation consists of several interconnected blocks:

- **Polar Coordinate Transformation**: converts Cartesian coordinates to polar coordinates;
- **Control Law**: generates control inputs based on the polar representation;
- **Unicycle Model**: implements the kinematic equations of the unicycle;
- **Runge-Kutta Integrator**: performs numerical integration for odometric localization.

The following subsections present and discuss the key results.

4.1 Unicycle State

The unicycle state variables (x, y, θ) are plotted against time (Fig.14), showing the evolution of the system. The x-coordinate monotonically decreases from 5m to 0m, showing a regular and stable behavior. The y-coordinate shows a more complex behavior with an initial oscillation followed by a negative excursion to about -0.7 m, then rising and stabilizing at 0m. The orientation θ starts from $\pi/4$, initially increases to about 4 radians and then gradually decreases towards zero. This behavior reveals a rotation strategy of the unicycle to progressively align itself with the target direction. All variables stabilize correctly at the desired values within about 10 seconds.

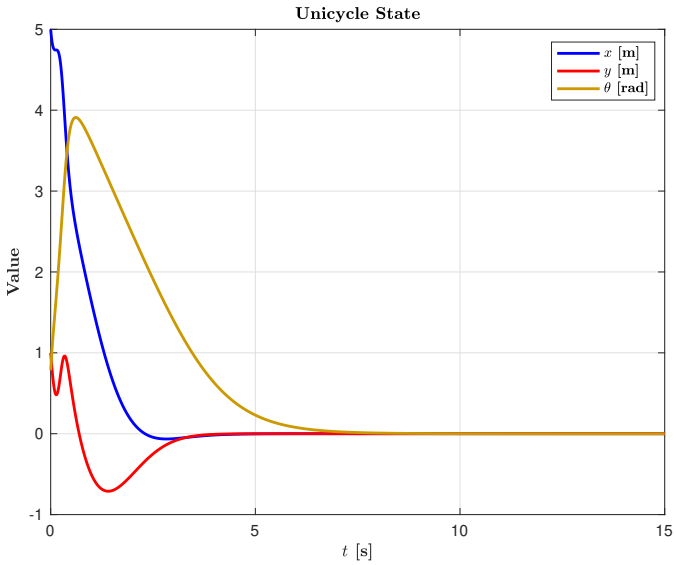


Figure 14: Unicycle state

4.2 Runge-Kutta Estimation

The comparison between the Runge-Kutta estimated state and the actual state is crucial to evaluate the accuracy of the odometric localization method. The estimated trajectories x_k, y_k and θ_k (Fig.15) are practically indistinguishable from the real ones shown in Fig.14 , demonstrating the excellent accuracy of the R-K method for this application. This almost perfect overlap confirms that the numerical integration error is negligible in the adopted simulation conditions.

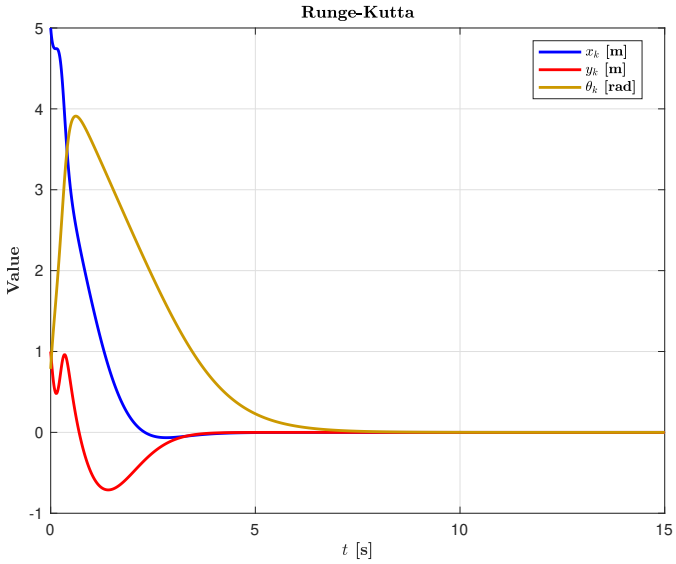


Figure 15: Runge-Kutta Estimation

4.3 Total Posture Error

The plot in Fig.16 is particularly interesting. The error scale is of the order of 10^{-4}, indicating excellent final accuracy of the regulator. Contrary to what one might expect, the error does not monotonically decrease towards zero but increases rapidly at the beginning and then stabilizes around $3.5 * 10^{-4}$. The initial rapid growth followed by stabilization suggests that the system quickly reaches a configuration close to the target, but maintains a small residual error at a steady state.

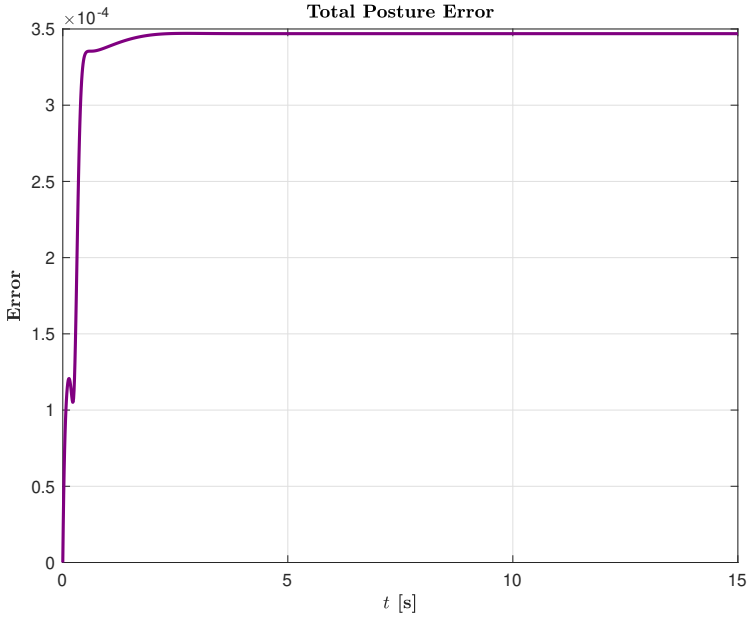


Figure 16: Total posture error

4.4 Velocities

By considering Fig.17, the analysis of control inputs is provided below. The linear velocity v shows an initial rapid oscillation between negative (about -8 m/s) and positive (about 8 m/s) values before settling on a positive value that gradually decreases towards zero. The angular velocity ω shows an initial peak at about 7.5 rad/s, followed by a negative phase around -1 rad/s before gradually rising towards zero. Both control signals settle to zero when the system reaches the target, as expected for a regulation problem. The shape of the control signals suggests that the system is overdamped in the final phase of convergence, ensuring a stable approach to the target without oscillations.

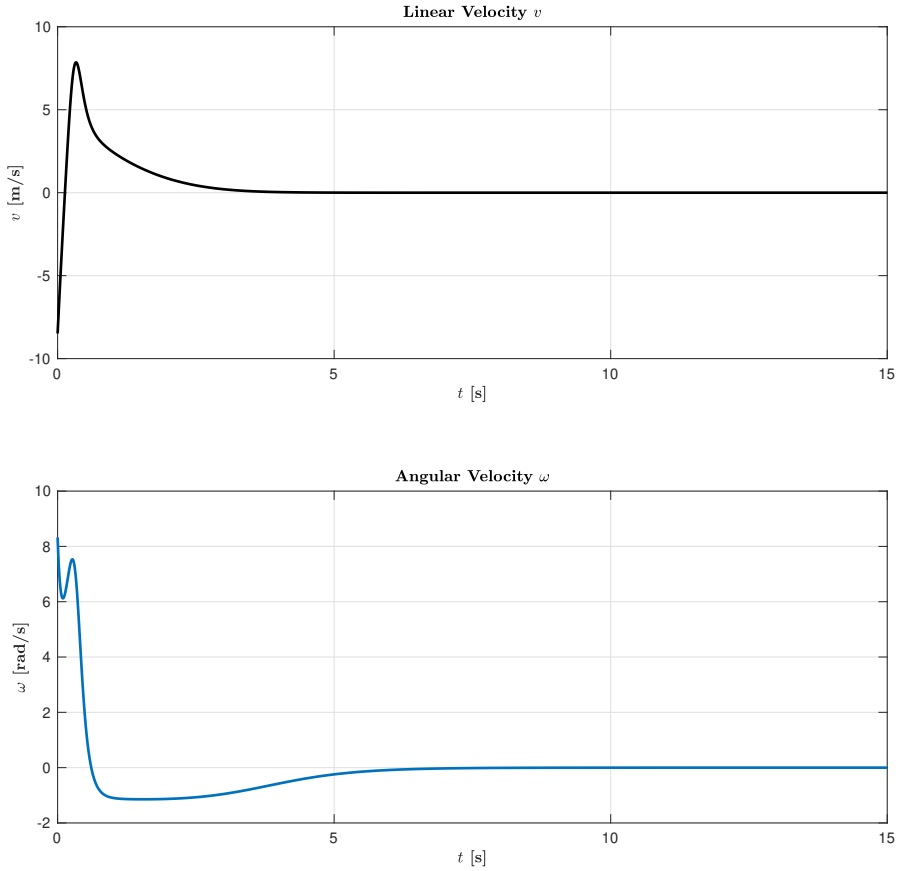


Figure 17: Velocities

4.5 Trajectory in the plane

The attached video ("unicycle_motion.mp4") in the HW2_FSR_Emanuela_Varone/Point_4/VIDEO folder shows the trajectory performed by the robot. The trajectory of the unicycle in the Cartesian plane reveals an interesting path from the initial configuration [5, 1, $\pi/4$] to the target position [0, 0, 0]. It is possible to observe that:
the unicycle does not follow a straight line to the target, but executes a curved path with a significant negative excursion on the y-axis up to about -0.7 meters before climbing back up to the target. The final orientation (blue arrow) aligns correctly with the target orientation (red arrow), confirming that the controller has effectively managed both positioning and orientation. The message "TARGET REACHED!" indicates that the system has met the convergence criteria, with a distance to the target of 0.000 m and a residual orientation error of 0.049 rad. This last value, although small, may merit further analysis to determine if it is acceptable for the specific application. The convergence time is 6.41 seconds, which seems reasonable considering the initial distance from the target.