Journal of Mechatronics Engineering

Simulation-Based Learning of Feedback Control Through Planar Robot Motion

DENIS MOSCONI[1], JOÃO DOMINGOS PEREIRA[1], MARCELO BECKER[2]

[1]Federal Institute of Education, Science and Technology of São Paulo - IFSP
[2]University of São Paulo - USP

<denis.mosconi@ifsp.edu.br> <joao.pereira@ifsp.edu.br> <becker@sc.usp.br>

DOI: 10.21439/jme.v9i1.136

Abstract. The understanding of feedback control concepts can be challenging for students when theoretical analysis is not directly connected to observable system behavior. Simulation environments can help bridge this gap by allowing learners to visualize the effects of control actions on system dynamics. This work presents a didactic framework for teaching classical proportional feedback control using the ROS 2 Turtlesim simulator. The proposed approach combines mathematical modeling, controller design, and robotic simulation to illustrate how proportional gain influences system response, convergence speed, and trajectory behavior. Simulation results demonstrate that the framework clearly exposes both theoretical properties and practical implementation effects, including actuator saturation and discretization phenomena. Because the system is simple to implement and relies on widely accessible open-source tools, it provides an effective platform for introducing students to feedback control concepts while simultaneously connecting classical control theory with mobile robotics applications.

Keywords: Engineering education. Feedback control. Mobile robotics. Proportional control. ROS 2.

1 Introduction

Classical control theory remains a foundational component in the education of engineers involved in the modeling, analysis, and control of dynamic systems, especially in introductory courses on control, automation, and robotics. Despite the emergence of modern, robust, and intelligent control techniques, classical approaches based on linear system theory, transfer functions, and frequency-domain analysis continue to play a central pedagogical role, as they provide intuitive insight into stability, transient response, and steady-state behavior (NISE, 2013; DORF; BISHOP, 2018). Mastery of these concepts is widely recognized as a prerequisite for understanding more advanced control strategies (KLUEVER, 2018; KIRK, 1970).

Nevertheless, teaching classical control poses persistent challenges. Students are often required to integrate mathematical tools such as differential equations, Laplace transforms, and linear algebra while simultaneously reasoning about abstract dynamic behavior of controlled systems, such as transient responses, stability, and steady-state characteristics. When instruction is predominantly theoretical, learners may struggle to relate mathematical models to physical phenomena, which can hinder conceptual understanding and motivation (BELHOT; FIGUEIREDO; MALAVÉ, 2001; COELHO et al., 2001). Although textbook examples help illustrate analytical procedures, they typically lack direct visual correspondence with the system behavior, limiting their effectiveness as didactic instruments.

Among the various pedagogical alternatives for teaching classical control systems, simulation-based resources have been widely considered as a means of supporting learning. Computational environments allow students to visualize system responses, experiment with parameter variations, and explore the effects of feedback in a controlled setting (SILVA; QUIRINO; GASOTO, 2012). Virtual and remote laboratories further expand access to experimental activities without the financial and logistical constraints associated with physical laboratories (SESSHINI; GALVEZ, 2007; ASSIS; COELHO; LIMA, 2008; ARAÚJO et al., 2022). Howe-

ver, many of these solutions either rely on proprietary platforms or introduce implementation-level complexity that shifts the learner's focus away from core classical control principles, which can be counterproductive in introductory courses.

From an educational perspective, numerous studies have investigated simulation-based strategies to reduce the gap between mathematical formalism and practical understanding in classical control education. The literature consistently indicates that computational simulations, virtual laboratories, and interactive environments can enhance student engagement and conceptual comprehension by providing visual feedback and opportunities for experimentation (BELHOT; FIGUEIREDO; MALAVÉ, 2001; COELHO et al., 2001; SILVA; QUIRINO; GASOTO, 2012). Remote and virtual laboratory initiatives further broaden access to experimental activities while alleviating the financial and logistical constraints associated with physical laboratories (SESSHINI; GALVEZ, 2007; ASSIS; COELHO; LIMA, 2008; ARAÚJO et al., 2022). Despite these advances, many simulation-based educational solutions either depend on proprietary platforms, require specialized infrastructure, or emphasize operational experimentation without a clear and progressive integration of analytical modeling, classical control design, and implementation, particularly in introductory control courses.

Within this context, robotic simulators can be regarded as promising and conceptually attractive resources for educational purposes. By integrating sensing, actuation, and control within a visually intuitive framework, robotics-based environments offer the potential to connect mathematical formulations with observable system behavior. Simulators such as V-REP[1], Webots[2], and Gazebo[3] have been explored in educational and research settings; however, they typically rely on three-dimensional environments, detailed physical modeling, and multiple layers of abstraction, which may increase computational demands and cognitive load for beginners (KERSCHBAUMER; LIMA; SIMÃO, 2014; BREGANON et al., 2021). In contrast, Turtlesim[4], a lightweight two-dimensional simulator integrated with the Robot Operating System (ROS), provides a geometrically simplified yet conceptually expressive environment in which control actions and trajectories can be directly observed, making it particularly suitable for introductory discussions of classical control principles and planar motion (RENARD, 2024).

Recent contributions in robotics and engineering education further reinforce the relevance of simulation-based and ROS-centered approaches as effective pedagogical tools. Several studies have explored the use of ROS in educational contexts, highlighting its potential to support hands-on, modular, and industry-aligned learning experiences (SANTOS et al., 2023; CAñAS et al., 2020; CHEN, 2022; COONEY et al., 2018; FARZAN, 2023; RAUDMäE et al., 2023; ROLDáN-ÁLVAREZ; MAHNA; CAñAS, 2021; ROLDáN-ÁLVAREZ et al., 2023; RYALAT et al., 2025; RYALAT, 2025). These works collectively emphasize complementary aspects of contemporary robotics education. Project-based learning strategies supported by ROS have been shown to improve student engagement and practical understanding of robotic systems (SANTOS et al., 2023; FARZAN, 2023), while web-based and open educational platforms enable scalable and accessible learning environments (CAñAS et al., 2020; ROLDáN-ÁLVAREZ; MAHNA; CAñAS, 2021; ROLDáN-ÁLVAREZ et al., 2023). Additional efforts have focused on developing structured teaching materials and introductory frameworks to facilitate ROS adoption by beginners (CHEN, 2022; COONEY et al., 2018), as well as on integrating physical robotic platforms and digital twins to bridge educational and professional training (RAUDMäE et al., 2023). More broadly, recent studies highlight the strategic importance of ROS and robotics-based environments in modern engineering education, particularly in aligning academic training with real-world technological ecosystems (RYALAT et al., 2025; RYALAT, 2025).

Despite these advances, a critical gap remains in the development of didactic methodologies that simultaneously preserve analytical rigor, visual clarity, and conceptual simplicity. In particular, there is a lack of simulation-based approaches capable of integrating fundamental concepts of classical control—such as transfer functions, stability analysis, and proportional control—with essential notions from robotics, including planar motion, coordinate systems, and reference frame transformations, in a coherent and accessible manner. As a result, students often struggle to relate mathematical formulations studied in control theory to observable motion and spatial representations, limiting a deeper conceptual understanding of control principles and their implementation within physically meaningful contexts.

[1] <https://www.coppeliarobotics.com/>
[2] <https://cyberbotics.com/>
[3] <https://gazebosim.org/>
[4] <http://ros.org/wiki/turtlesim>

The objective of this work is to propose a didactic control strategy for planar point-to-point motion using the Turtlesim simulator integrated with the Robot Operating System (ROS 2). The proposed methodology aims to integrate classical control theory, kinematic modeling, and coordinate frame transformations within a single, accessible simulation environment. Specifically, the work seeks to demonstrate how a proportional control law can be formulated in a global reference frame, mapped into the robot's local frame, and implemented in a manner that allows direct visualization and analytical interpretation of the resulting motion.

The relevance of this contribution lies in its pedagogical coherence and practical accessibility. By employing a lightweight two-dimensional robotic simulator and open-source tools, the proposed approach minimizes computational and conceptual overhead while maintaining fidelity to fundamental control principles. This combination enables students to simultaneously observe system behavior, interpret mathematical models, and understand the role of coordinate transformations in control implementation. Consequently, the methodology addresses a concrete educational need, offering a reproducible and conceptually grounded framework that strengthens the connection between theory and practice in introductory control and robotics education.

The main contributions of this work can be summarized as follows. First, a didactic framework is proposed that integrates classical proportional control, kinematic modeling, and robotic simulation within a single coherent learning environment. Second, the approach explicitly connects analytical control concepts—such as transfer functions, stability, and transient response—with observable motion in a planar robotic system, facilitating conceptual understanding for students. Third, the methodology emphasizes simplicity and accessibility by relying exclusively on lightweight and open-source tools, making it easily reproducible in educational settings. Finally, the work highlights the pedagogical value of discrepancies between ideal theoretical models and practical implementations, allowing learners to explore the effects of discretization, saturation, and communication delays in a controlled and intuitive manner. It is important to emphasize that the purpose of this work is not to propose a novel control strategy, but rather to provide a structured and effective educational framework for teaching classical control concepts through robotic simulation.

2 Methodology

The methodology proposed in this work follows a structured and sequential approach that integrates mathematical modeling, computational simulation, classical control design, and experimental validation within a robotic software framework. The procedure is designed to be conceptually accessible, analytically transparent, and easily reproducible, enabling students to relate theoretical control concepts to observable system behavior in a simulated environment.

2.1 Computational Environment and Simulation Framework

The experimental platform adopted in this study is built upon the Robot Operating System 2 (ROS 2), specifically the Humble Hawksbill distribution, a long-term support (LTS) release widely adopted in academic and industrial robotics. ROS 2 provides a standardized middleware infrastructure for communication between distributed software components, allowing modular implementation of sensing, control, and actuation through message-based interfaces.

Within this framework, the *Turtlesim* simulator—distributed as a native ROS 2 package—is employed as a two-dimensional robotic simulation environment. Turtlesim offers a graphical workspace in which a simplified mobile agent, represented by a turtle, moves on the Cartesian plane according to commanded linear and angular velocities. The simulator publishes the agent pose in real time and accepts velocity commands generated by external control nodes, enabling closed-loop interaction between the controller and the simulated plant.

This architecture reflects the structure of real robotic control systems while remaining sufficiently simple for introductory educational purposes. Figure1 illustrates the Turtlesim environment and the planar motion of the agent during the execution of the control algorithm.

2.2 Control Problem Definition

The control problem addressed in this work consists of driving the simulated agent from an arbitrary initial position (x_0, y_0) to a desired target position (x^d, y^d) within the planar workspace. The objective is to achieve stable and convergent point-to-point motion using a closed-loop proportional controller acting on velocity commands.

This problem was selected due to its pedagogical relevance: it is sufficiently simple to allow analytical

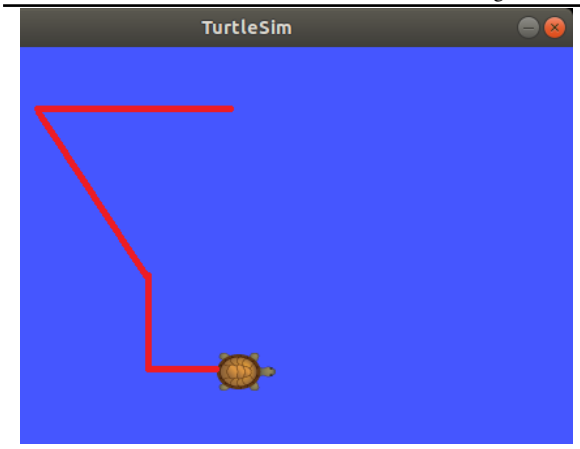

Figure 1: Turtlesim simulation environment. The mobile agent is represented by a turtle moving on a two-dimensional Cartesian plane. The red trace corresponds to the trajectory executed by the agent during a simulation.

treatment using classical control tools, while still enabling clear visualization of feedback effects, transient behavior, and the influence of controller gain on system performance.

2.3 Analysis of the Results

The analysis of the results is conducted by comparing the behavior predicted by the mathematical model with the response observed in the Turtlesim simulation during controller execution. Particular attention is given to the trajectory followed by the agent, the applied control signals, and the convergence of the position error over time.

This comparison allows verification of model consistency, assessment of closed-loop stability, and evaluation of the influence of controller parameters. The results are interpreted from a pedagogical perspective, emphasizing the relationship between analytical predictions and observed system behavior.

2.4 Hardware Utilized

All simulations were performed on a personal computer equipped with an Intel Core i7-10510U processor running at 2.30 GHz, 8 GB of RAM, a dedicated graphics card with 2 GB of memory, and a 512 GB NVMe SSD. The operating system used was Ubuntu Linux 22.04, which is fully compatible with ROS 2 Humble Hawksbill and ensures stable execution of the simulation and control software.

2.5 Online Material Available

To support reproducibility and facilitate educational use, the source codes developed in this work are publicly available. The MATLAB scripts used for modeling and analysis, as well as the ROS 2 Python nodes implementing the controller, can be accessed at: <https://github.com/Denismosconi/turtle_control>.

3 Mathematical Modeling and System Analysis

The plant considered in this study corresponds to the mobile agent provided by the *Turtlesim* simulator, which can be employed in educational contexts for introductory studies in control and mobile robotics. This agent represents a purely kinematic abstraction rather than a full dynamic system, since it does not account for physical properties such as mass, inertia, friction, or actuator dynamics. Its state is fully described by the pose vector $\mathbf{q}(t) = [x(t),\ y(t),\ \theta(t)]^T$, which specifies the planar position and orientation of the agent with respect to a fixed global reference frame defined in the simulation environment.

The coordinates $x(t)$ and $y(t)$ denote the Cartesian position of the turtle in the inertial frame attached to the simulator window, while $\theta(t)$ represents its orientation relative to this same global frame. This distinction between global pose representation and locally applied velocity commands is fundamental for the proper formulation of the control problem and will become particularly relevant when discussing coordinate transformations.

The system inputs are the commanded linear and angular velocities, while the outputs correspond directly to the temporal evolution of the pose variables. Because the velocity commands are instantaneously integrated by the simulator to generate position and orientation, the Turtlesim agent can be interpreted as a set of pure integrators. This characteristic makes it particularly suitable for didactic purposes, as it allows students to directly associate mathematical models with observable motion in the simulation.

Given the absence of dynamic coupling terms, the kinematic equations governing the motion of the agent are fully decoupled along each coordinate. As a result, the evolution of the position along the x-axis, the y-axis, and the orientation angle θ can be modeled independently as integrals of their respective velocity inputs. These relationships may be expressed in the time

domain as

$$x(t) = \int_0^t v_x(\tau)\, d\tau, \tag{1}$$

$$y(t) = \int_0^t v_y(\tau)\, d\tau, \tag{2}$$

$$\theta(t) = \int_0^t \omega(\tau)\, d\tau, \tag{3}$$

where v_x and v_y denote the linear velocity components and ω represents the angular velocity.

Since all three equations share the same mathematical structure, the subsequent analysis may be carried out without loss of generality by considering a single translational axis. Applying the Laplace transform to the integral relation yields the transfer function that relates the velocity input $V(s)$ to the position output $X(s)$,

$$G(s) = \frac{X(s)}{V(s)} = \frac{1}{s}. \tag{4}$$

This expression confirms that the plant behaves as a pure integrator, characterized by a single pole at the origin of the complex plane.

The steady-state behavior of the system for standard test inputs can be examined using the Final Value Theorem,

$$\lim_{t\to\infty} x(t) = \lim_{s\to 0} sX(s), \tag{5}$$

provided that the limits exist. For an impulsive velocity input, the resulting position converges to a finite value, corresponding to a transient displacement followed by rest. In contrast, a step input, which represents a constant velocity command, produces an unbounded position response that increases linearly with time. Similarly, a ramp input in velocity leads to an accelerated motion, causing the position to grow quadratically in time.

The responses to standard test inputs further illustrate the integrative nature of the plant. An impulsive velocity input produces a finite displacement, whereas step and ramp inputs lead to unbounded position responses, corresponding respectively to uniform and accelerated motion. These behaviors are consistent with the presence of a pure integrator and indicate that the system does not exhibit satisfactory performance when operated without feedback control.

Figure 2 illustrates the time responses of the system to impulse, step, and ramp inputs plotted together in a single graph. These responses clearly reflect the integrative nature of the plant and provide an intuitive visualization of how different classes of velocity commands translate into motion in the simulation environment.

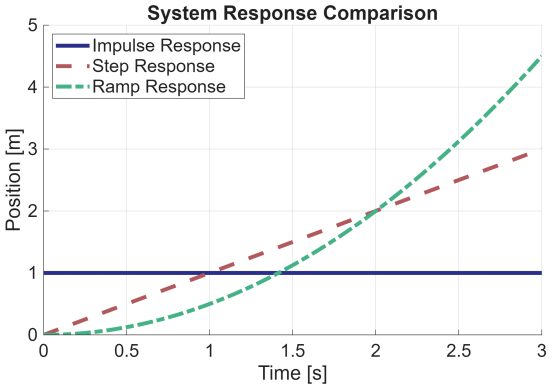

Figure 2: Open-loop time responses of the Turtlesim kinematic model to impulse, step, and ramp velocity inputs.

From a stability standpoint, the system may be assessed using the Bounded-Input Bounded-Output (BIBO) criterion. A linear time-invariant system is BIBO stable if every bounded input produces a bounded output. In the present case, although the impulse response is bounded, a bounded step input yields an unbounded position output. Consequently, the plant does not satisfy the BIBO stability condition. This result is consistent with the presence of a pole at the origin and indicates that the open-loop system is marginally stable in the internal sense but unstable under the BIBO definition.

These observations reaffirm that open-loop control is inadequate for positioning tasks, as the system is incapable of converging to a desired location under bounded velocity commands. Therefore, closed-loop control with feedback is required to ensure stability and convergence, motivating the controller design presented in the subsequent section.

4 Feedback Controller Design

Given that the plant is characterized by a transfer function equivalent to a pure integrator, driving the turtle from an initial condition to a desired reference position cannot be satisfactorily achieved in open loop, thus requiring the introduction of feedback. From this standpoint, the use of a proportional controller in a closed-loop configuration is sufficient to ensure asymptotic stability and convergence to the target point. Figure 3a depicts the block diagram associated with this control structure, whereas Figure 3b presents its reduced representation, in which the closed-loop dynamics are con-

densed into a single equivalent block that incorporates the effect of the proportional gain on the plant.

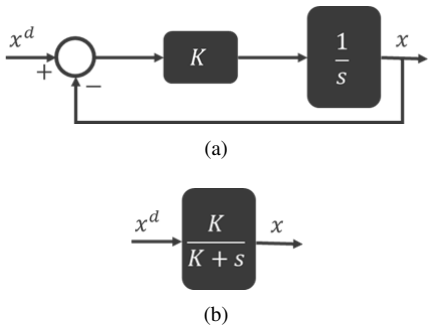

Figure 3: Block diagram for the feedback proportional control

The steady-state performance of the closed-loop system can be formally assessed through the Final Value Theorem. For a unit-step reference input $R(s) = 1/s$, the closed-loop transfer function $G(s) = \frac{K_p}{s+K_p}$ yields

$$\lim_{t\to\infty} x(t) = \lim_{s\to 0} sG(s)\frac{1}{s} = \lim_{s\to 0} \frac{K_p}{s+K_p} = 1, \quad (6)$$

provided that $K_p > 0$. This result demonstrates that the steady-state error to a constant position reference is null, regardless of the specific positive value assigned to the proportional gain. Consequently, the controlled position asymptotically coincides with the imposed reference, fulfilling the primary objective of the positioning task.

The transient responses for different values of K_p are depicted in Figure 4. As expected for a first-order system, increasing the proportional gain reduces the time constant $\tau = 1/K_p$, leading to faster convergence without modifying the final value of the response. The gain therefore influences exclusively the speed of convergence, while preserving the steady-state accuracy. In the present study, only step inputs are considered at this stage, since they appropriately represent fixed position references in point-to-point motion problems.

Additionally, the root locus shown in Figure 5 confirms that, for the adopted ideal continuous-time linear model, the closed-loop pole is located at $s = -K_p$, remaining strictly in the left half of the complex plane for all $K_p > 0$. Under this modeling assumption, the closed-loop response is asymptotically stable and non-oscillatory, which is consistent with the behavior expected from a first-order system. Furthermore, since a bounded step input produces a bounded output in this closed-loop model, the system satisfies the BIBO stability condition.

It should be emphasized, however, that these conclusions are valid within the scope of the idealized analytical model adopted in this section. The practical implementation in ROS 2/Turtlesim is executed in discrete time and is subject to communication timing, numerical truncation, switching between control stages, and command saturation. Consequently, for sufficiently high gains, the simulated response may deviate from the ideal first-order prediction, exhibiting non-smooth transients or oscillatory effects that are not represented in the simplified continuous-time model. Rather than invalidating the analytical treatment, these discrepancies delimit its range of validity and constitute an important pedagogical element of the proposed framework.

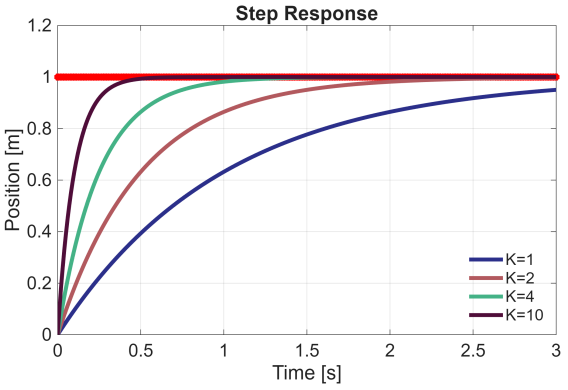

Figure 4: Step response for different K_p values.

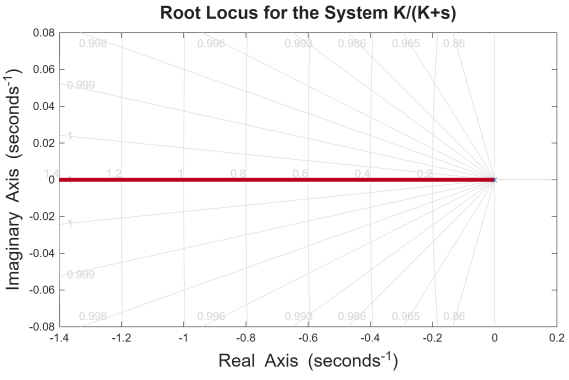

Figure 5: Root locus for the closed loop system.

Two control loops were used in this work: one to

adjust the turtle orientation, controlling its angular position, and another to adjust its displacement, controlling its linear position in the plane.

The angular position control can be approached in a scalar manner, since in this case the agent rotates only around the z-axis that passes through its geometric center. Therefore, the block diagram presented in the Figure 3 represents perfectly this control loop.

However, the linear position control can not be treated as a scalar one, instead, it should be approached from a vector-based perspective. This is because the turtle linear position is measured relative to a global reference frame, while the speed applied to it (and determined by the controller) is given relative to a reference frame fixed on the turtle itself.

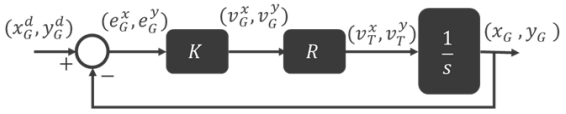

Figure 6: Block diagram for the linear position control from a vectorial approach.x_G^d and y_G^d, are the desired position in global frame, e_G^x and e_G^y are the x and y position errors in global frame, v_G^x and v_G^y are the x and y linear velocities in global frame, v_T^x and v_T^y are the x and y linear velocities in turtle frame and x_G and y_G are the turtle position in global frame.

The Figure 6 presents the block diagram for the control of the turtle linear position through a vectorial approach: the block R express the rotation matrix that converts the velocity vector from the global frame to the turtle frame. Such transformation matrix is related to the z-axis rotation of the agent for an angle θ and is given as:

$$^{T}R_G(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Therefore, the project can be performed as follows. First the linear position error is determined, referenced to the global frame:

$$\begin{bmatrix} e_G^x \\ e_G^y \\ e_G^z \end{bmatrix} = \begin{bmatrix} x_G^d \\ y_G^d \\ z_G^d = 0 \end{bmatrix} - \begin{bmatrix} x_G \\ y_G \\ z_G = 0 \end{bmatrix} \tag{8}$$

The proportional feedback control law determines the linear velocity in the global frame as:

$$v_G = \begin{bmatrix} v_G^x \\ v_G^y \\ v_G^z \end{bmatrix} = \begin{bmatrix} e_G^x \\ e_G^y \\ 0 \end{bmatrix} K_p \tag{9}$$

Then, the velocity is rewritten in the turtle frame:

$$v_T = \begin{bmatrix} v_T^x \\ v_T^y \\ v_T^z \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_G^x \\ v_G^y \\ v_G^z = 0 \end{bmatrix} \tag{10}$$

In equations above, the components of z-axis were considered but are always zero (the choice to present the three axes is to allow the work to be reproducible and expandable to more dimensions). So, the turtle velocity in its own frame, for each axis can be obtained explicitly.

The z-axis turtle velocity (v_T^z) is always null, as mentioned above, since the agent moves only in the xy plane.

The y-axis turtle velocity v_T^y is given as:

$$v_T^y = v_G^x(-sin\theta) + v_G^y(cos\theta) = 0 \tag{11}$$

The null value of v_T^y can be explained due the fact that during the translation stage, the turtle is assumed to be previously oriented such that its body-fixed x_T axis is aligned with the direction from the current position to the desired target. Since the proportional controller is defined in the global frame as $\mathbf{v}_G = K_p\,\mathbf{e}_G$, with $\mathbf{e}_G = [e_x^G \quad e_y^G]^T$, the commanded velocity vector $\mathbf{v}_G = [v_x^G \quad v_y^G]^T$ is colinear with $\mathbf{e}_G$. Therefore, the turtle heading angle can be defined consistently with the direction of $\mathbf{v}_G$ as

$$\theta = \operatorname{atan2}(v_y^G,\, v_x^G), \tag{12}$$

so that the trigonometric terms correspond to the normalized components of $\mathbf{v}_G$. Defining

$$h = \|\mathbf{v}_G\| = \sqrt{(v_x^G)^2 + (v_y^G)^2}, \tag{13}$$

one obtains

$$\cos\theta = \frac{v_x^G}{h}, \qquad \sin\theta = \frac{v_y^G}{h}. \tag{14}$$

Substituting these expressions into the lateral velocity component in the turtle frame, given by (11), yields

$$v_T^y = -v_G^x\left(\frac{v_G^y}{h}\right) + v_G^y\left(\frac{v_G^x}{h}\right) = \frac{-v_G^x v_G^y + v_G^y v_G^x}{h} = 0, \tag{15}$$

for $h \neq 0$. Hence, once the turtle is aligned with the target direction, the transformed velocity vector has no lateral component in the body-fixed frame, i.e., the commanded motion occurs exclusively along x_T

(forward/backward), as expected for a nonholonomic, car-like translation stage.

The x-axis turtle velocity (v_T^x) is obtained as the projection of the commanded global velocity onto the body-fixed longitudinal direction of the agent. Using the rotation relationship, one obtains:

$$v_T^x = v_G^x \cos\theta + v_G^y \sin\theta \tag{16}$$

which can be interpreted geometrically as the dot product between the global velocity vector and the unit vector aligned with the turtle heading, i.e., $v_T^x = \mathbf{v}_G \cdot \hat{x}_T$, where $\hat{x}_T = [\cos\theta \quad \sin\theta]^T$. Since $\mathbf{v}_G = K_p \mathbf{e}_G$, the longitudinal velocity may also be written as

$$v_T^x = K_p \|\mathbf{e}_G\| \cos(\alpha), \tag{17}$$

where α denotes the angle between the position error vector $\mathbf{e}_G$ and the current turtle heading. Consequently, the sign of v_T^x is determined by the relative orientation between these vectors. If $|\alpha| < \pi/2$, the projection is positive and the turtle moves forward. However, if overshoot[5] occurs and the position error reverses direction while the turtle maintains its orientation, the angle α becomes greater than $\pi/2$, yielding $\cos(\alpha) < 0$ and therefore $v_T^x < 0$. In this case, reverse motion naturally emerges without requiring any structural modification of the controller, arising solely from the geometric relationship between the error vector and the body-fixed longitudinal axis.

Figure 7 presents the flowchart of the algorithm developed for position control of the Turtlesim agent. First, the user specifies the desired position (x^d, y^d). Based on the current pose of the turtle and the target coordinates, the desired orientation θ^d is then computed. This angle is defined with respect to the global x-axis and corresponds to the direction of the vector connecting the current position (x, y) to the target point (x^d, y^d).

Once θ^d is determined, a proportional feedback controller is activated to regulate the angular position of the turtle. The objective of this stage is to minimize the angular error $e_\theta = \theta^d - \theta$ and align the agent with the direction of the target. The angular control loop remains active until the orientation error falls within a predefined tolerance, ensuring that the turtle is properly oriented toward the desired destination.

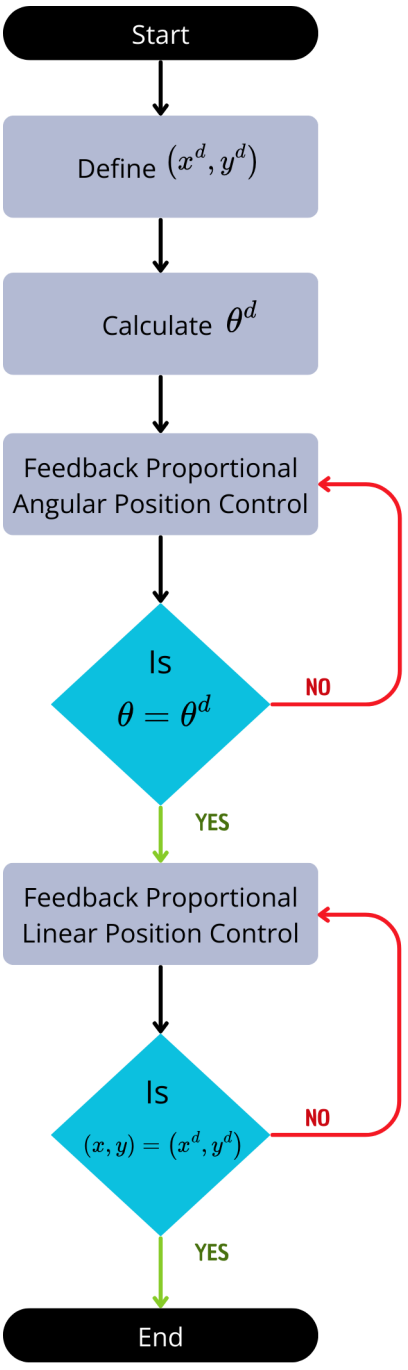

Figure 7: Flowchart of the control algorithm developed.

[5] Although the ideal continuous-time closed-loop model is first-order and therefore does not predict overshoot, the implemented simulation includes discrete-time execution, communication timing effects, numerical approximations, and signal limitations. Under these non-ideal conditions, overshoot or oscillatory behavior may arise, especially for high proportional gains.

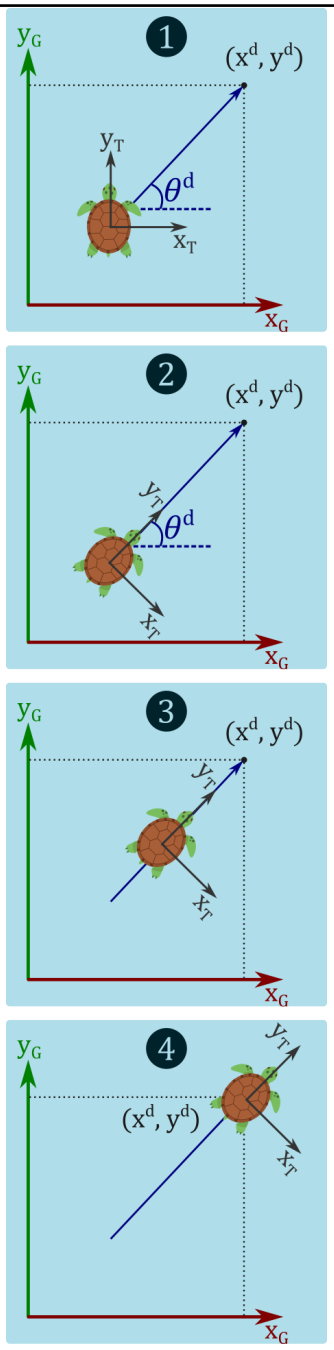

Figure 8: Illustration of the turtle movement from its starting point to its desired end point.

After the orientation phase is completed, the angular control action is disabled and the linear displacement phase begins. A proportional feedback controller is then applied, generating a linear velocity command that drives the agent toward the target coordinates. The linear motion is continuously adjusted based on the remaining position error until the turtle reaches the desired location within an admissible tolerance band. The introduction of this tolerance prevents persistent oscillations or infinite control looping due to numerical precision limits or discrete-time effects.

This sequential control structure—first orientation, then translation—simplifies the control design, enhances clarity for educational purposes, and allows students to clearly distinguish the effects of angular and linear feedback actions in the simulated environment.

While Figure 7 emphasizes the logical sequence of control decisions implemented in the algorithm, Figure 8 illustrates the geometric consequences of these control actions in the Cartesian plane, by conceptually dividing the process into four distinct stages.

First, the user specifies the desired Cartesian coordinates (x^d, y^d). Based on the current pose (x, y) and the target position, the desired orientation θ^d is computed as the angle of the vector connecting the current position to the target point, measured with respect to the global x-axis.

In the second stage, the turtle performs a pure rotational motion about its own geometric center in order to align its body-fixed longitudinal axis with the direction of the target. During this phase, the proportional feedback controller acting on angular position is active, regulating the angular error $e_\theta = \theta^d - \theta$ until it falls within a predefined tolerance band.

Once the desired orientation is achieved, the angular control loop is deactivated and the third stage begins. The turtle, now properly aligned, initiates translational motion toward the target. In this phase, a proportional feedback controller acting on linear position generates the commanded velocity, driving the agent toward (x^d, y^d) while continuously reducing the position error.

Finally, when the turtle reaches the desired position within the specified tolerance, both angular and linear velocities are set to zero, ensuring that the agent remains stationary at the target point.

It is important to emphasize that the proposed algorithm does not include a final reorientation stage after the target position is reached. Nevertheless, such functionality can be straightforwardly incorporated by adding an additional angular position control loop activated upon arrival at the target. In this extended configuration, the user would specify not only the desired position (x^d, y^d) but also a final desired orientation θ_f^d, thus defining a complete pose reference (x^d, y^d, θ_f^d).

The developments presented in this section establish a coherent bridge between classical control theory and robotic motion implementation. By combining the analytical properties of a first-order closed-loop system with a geometrically consistent transformation between global and body-fixed reference frames, the proposed strategy preserves mathematical rigor while remaining computationally simple. The decomposition into angular and linear control loops enhances conceptual clarity and facilitates didactic exploration of stability, convergence, and gain influence. Having defined the control architecture and its theoretical foundations, the subsequent section evaluates the practical behavior of the implemented algorithm within the Turtlesim environment, comparing analytical predictions with simulated motion and control signals.

5 Results and Discussions

To evaluate the practical performance of the proposed control strategy, four independent simulations were conducted by varying exclusively the proportional gain $K_p = \{1, 2, 4, 10\}$. Two feedback control loops were employed: one for angular position regulation and another for linear position control. Although distinct control actions are associated with orientation and translation, the same gain value was adopted for both loops in each simulation, allowing a direct assessment of the influence of K_p on the overall closed-loop behavior. In all experiments, the initial pose of the turtle was maintained at the default Turtlesim configuration, $\{x_0, y_0, \theta_0\} = \{5.5,\ 5.5,\ 0\}$, while the desired target position was fixed at $\{x^d, y^d\} = \{2.0,\ 10.0\}$. No final orientation constraint was imposed upon arrival at the target; therefore, the terminal heading corresponds to the orientation naturally achieved during convergence. The results are presented by analyzing sequentially the two main phases of motion: first, the rotational alignment required to orient the agent toward the target, and subsequently, the translational displacement that drives the turtle to the desired position. The analysis focuses on transient behavior, convergence characteristics, control signal profiles, and the relationship between analytical predictions and simulated responses, with particular attention to deviations observed under high-gain conditions.

Before discussing the simulation results, it is important to clarify the relationship between the analytical model of Section 4 and the implemented control loop. The theoretical analysis was intentionally developed from an ideal continuous-time first-order approximation, with the purpose of providing a clear and didactically tractable interpretation of proportional feedback behavior. By contrast, the practical implementation in ROS 2/Turtlesim involves discrete-time execution, message-based communication, stage switching between angular and linear control, and saturation of the commanded velocities. Therefore, the results presented in this section should be interpreted as the behavior of the implemented educational framework under non-ideal conditions, rather than as an exact realization of the simplified analytical model. In this sense, deviations observed for large K_p values do not contradict the theoretical development, but rather indicate the limits of validity of the adopted approximation and help expose implementation effects that are pedagogically relevant.

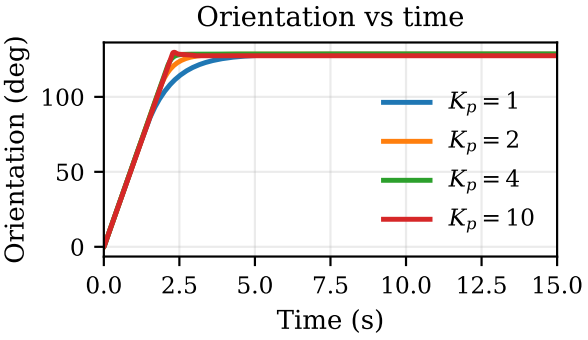

Figure 9: Angular position θ of the turtle (degrees) for different proportional gains. Increasing K_p reduces the convergence time, consistent with the ideal first-order closed-loop model. For higher gains, slight deviations from the ideal non-oscillatory response can be observed due to non-ideal implementation effects.

Angular motion analysis. The first stage of the motion consists of aligning the turtle with the direction of the target point. Figure 9 presents the angular position of the agent as a function of time for the four proportional gains considered in the experiments ($K_p = 1, 2, 4,$ and 10). As expected, the controller drives the turtle toward the desired orientation $\theta^d = 128°$, corresponding to the direction of the vector connecting the initial position to the target location.

From Figure 9, it can be observed that increasing the proportional gain accelerates the angular response, reducing the time required for the turtle to reach the desired orientation. This behavior is consistent with the theoretical analysis of the closed-loop system presented in Section 4, where the time constant of the first-order

dynamics is given by $\tau = 1/K_p$. Consequently, larger gains lead to smaller time constants and faster transient responses.

However, when comparing the observed responses with the theoretical step responses illustrated previously in Figure 4, it becomes evident that the settling times obtained in simulation are not as small as predicted by the analytical model, although they still decrease as K_p increases. This discrepancy is primarily associated with limitations imposed on the control signal. Specifically, the angular velocity generated by the proportional controller saturates when its magnitude exceeds the maximum allowable velocity of the simulator. This phenomenon can be clearly observed in Figure 10, where the control signal is clipped during the initial phase of the motion.

As the gain increases, the commanded velocity $\omega = K_p e_\theta$ becomes larger, causing the control signal to remain saturated for a longer period. Once the angular error decreases sufficiently, the velocity re-enters the admissible range, producing a visible discontinuity in the control signal. This transition from the saturated region to the permissible region introduces a non-smooth behavior that slightly alters the expected first-order transient response.

A second discontinuity can also be observed at the very beginning of the motion. Immediately after the initial instant t_0, the angular velocity abruptly changes from zero to the value imposed by the proportional law, $\omega = K_p e_\theta$. This abrupt transition corresponds to a step in the control signal, which implies an impulsive acceleration at the start of the motion. Although this behavior does not compromise the stability of the simulated system, it may lead to undesirable oscillatory behavior or mechanical stress in real robotic systems. Therefore, from a control design perspective, it would be advisable to incorporate strategies that smooth the initial control action, reducing both the initial acceleration and the jerk of the motion.

Additional quantitative indicators of the angular response are summarized in Table 1. The table presents the theoretical time constant τ and the corresponding settling time $t_s = 4\tau$, derived from the ideal continuous-time first-order model, together with the steady-state angular error obtained from the implemented system. As expected from the analytical model, both τ and t_s decrease as K_p increases, reflecting the predicted acceleration of the closed-loop response. The simulation results, in turn, show a progressive increase in steady-state error for larger gains, particularly for $K_p = 10$, indicating the growing influence of non-ideal implementation effects such as discretization and actuator saturation. This behavior highlights the limits of validity of the simplified analytical model under high-gain conditions.

An interesting phenomenon appears for the largest gain tested. Although the ideal continuous-time closed-loop model discussed in Section 4 exhibits first-order dynamics, the implemented response associated with $K_p = 10$ in Figure 9 exhibits a behavior reminiscent of overshoot typically observed in second-order systems. This apparent contradiction can be attributed to the digital nature of the implementation. In sampled-data control systems, high controller gains combined with finite sampling and communication delays may modify the effective closed-loop dynamics, potentially driving the system toward oscillatory behavior. This effect should therefore be interpreted as a consequence of non-ideal implementation conditions, rather than as a contradiction of the analytical first-order model itself.

Hence, the controller design must consider not only the proportional gain but also the sampling frequency of the control loop. An excessively large gain relative to the sampling rate may push the system toward oscillatory or unstable behavior. Consequently, the control engineer must determine an appropriate compromise between gain magnitude and sampling frequency, ensuring that the system remains stable while still satisfying the desired performance specifications.

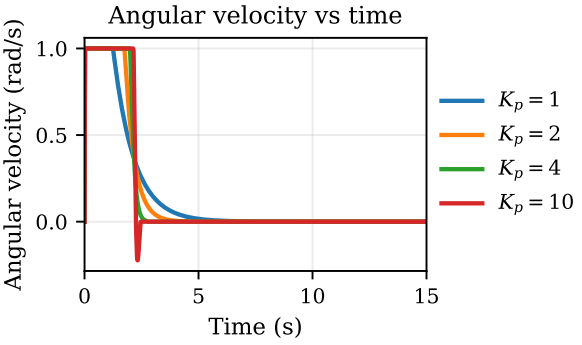

Figure 10: Angular velocity ω of the turtle (rad/s) for different proportional gains. For higher values of K_p, the control signal exhibits initial saturation, resulting in a clipped velocity profile. As the error decreases, the response transitions back to the proportional region, illustrating the combined effect of actuator limits and feedback control.

Figure 10 complements the angular-position analysis by showing the corresponding angular velocity profiles. The initial clipping of the control action is evi-

Table 1: Performance metrics obtained for the angular and linear motions, comparing theoretical predictions from the ideal continuous-time model with results from the implemented system. The table presents the proportional gain K_p, the theoretical time constant $\tau = 1/K_p$, the corresponding settling time $t_s = 4\tau$ (2% criterion), and the steady-state error measured at the end of each simulation. For the angular motion the error is expressed as the absolute angular error $|e_\theta(\infty)|$ in degrees, while for the linear motion the Euclidean norm of the position error $\|\mathbf{e}(\infty)\|$ is reported in meters.

Angular motion				
Experiment	K_p	τ (s)	t_s (s)	$\lvert e_\theta(\infty)\rvert$ (deg)
$K_p = 1$	1.00	1.00	4.00	0.03
$K_p = 2$	2.00	0.50	2.00	0.01
$K_p = 4$	4.00	0.25	1.00	0.09
$K_p = 10$	10.00	0.10	0.40	1.07
Linear motion				
Experiment	K_p	τ (s)	t_s (s)	$\|\mathbf{e}(\infty)\|$ (m)
$K_p = 1$	1.00	1.00	4.00	0.08
$K_p = 2$	2.00	0.50	2.00	0.00
$K_p = 4$	4.00	0.25	1.00	0.01
$K_p = 10$	10.00	0.10	0.40	0.11

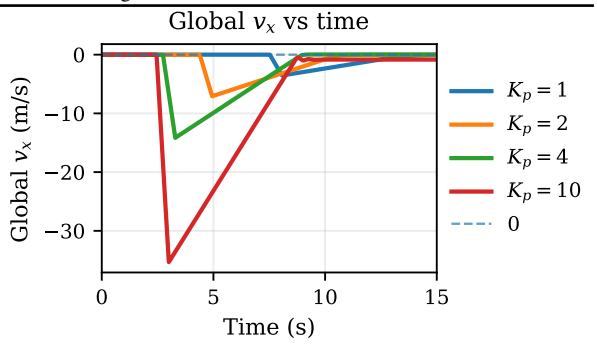

Figure 11: Global velocity of the turtle along the x-axis (m/s) for different proportional gains. The velocity is expressed in the inertial (global) reference frame. Variations in the velocity profile reflect the combined effects of translational control and the robot orientation, with higher gains producing faster responses and more abrupt changes due to non-ideal implementation effects.

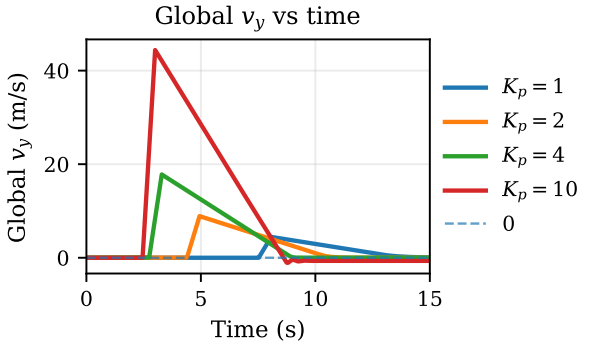

Figure 12: Global velocity of the turtle along the y-axis (m/s) for different proportional gains. The velocity is expressed in the inertial (global) reference frame and depends on both the commanded motion and the turtle orientation. Deviations from smooth profiles, especially for higher gains, reflect the influence of discrete-time implementation and coupling between angular and translational dynamics.

dent, especially for the larger gains, confirming that saturation plays a decisive role in shaping the practical transient response. The larger the gain, the longer the velocity remains at the imposed limit before returning to the proportional region. This behavior explains why the reduction in settling time is not proportional to the theoretical decrease in τ, even though higher gains still produce faster responses in relative terms.

Linear motion analysis. After the orientation stage, the translational controller becomes active and drives the turtle toward the desired point by regulating the linear position error in the global frame. Figures 11 and 12 present the global velocity components along the x- and y-axes, respectively. A first relevant observation is that the general shape of the velocity profiles remains similar both between the two axes and among different gains for a same axis. This behavior can be explained by the geometric characteristics of the motion and by the structure of the adopted control law. During the translational stage, the turtle is approximately aligned with the direction of the position error, and the commanded velocity in the global frame is defined as $\mathbf{v}_G = K_p \mathbf{e}_G$. Consequently, the velocity vector is colinear with the position error vector, which continuously points toward the target position. As a result, the motion occurs approximately along a straight line in the Cartesian plane. Under these conditions, the velocity components v_x^G and v_y^G correspond merely to the projections of the same velocity vector onto the global axes. Therefore, their temporal profiles remain similar in shape, differing only in magnitude according to the relative displacement required along each coordinate. In all cases, the signals exhibit the characteristic response of a proportional position controller acting on a first-order plant: an initially large control action, followed by a progressive reduction as the position error decreases.

What changes substantially with K_p is the amplitude of the commanded velocity. As expected from the adopted proportional law, increasing K_p produces larger velocity magnitudes and therefore faster translational responses.

The signs of the velocity components are also physically consistent with the direction of the position error. Since the turtle moves from $(5.5, 5.5)$ to $(2.0, 10.0)$, the displacement required along the global x-axis is negative, whereas the displacement required along the global y-axis is positive. Accordingly, Figure 11 shows negative values of v_x^G, while Figure 12 shows positive values of v_y^G. This result is a direct consequence of the adopted feedback law, in which the commanded velocity vector in the global frame is proportional to the position error vector. Therefore, the controller naturally generates a motion directed toward the target quadrant without requiring any additional switching logic for the translational phase.

A second important aspect concerns the relative magnitudes of the two global velocity components. Because the absolute displacement required along y is greater than that required along x, namely $|\Delta y| = 4.5 > |\Delta x| = 3.5$, the magnitude of v_y^G is expected to exceed that of v_x^G for the same gain. This trend is confirmed by comparing Figures 11 and 12, in which the peak positive values of v_y^G are greater than the peak negative values of v_x^G. Thus, although the velocity profiles are qualitatively similar, their amplitudes reflect the anisotropy of the required displacement in the Cartesian plane.

Figure 13 presents the magnitude of the linear velocity vector in the global frame, that is, $\|\mathbf{v}_G\| = \sqrt{(v_x^G)^2 + (v_y^G)^2}$. This representation is particularly useful because it condenses the translational motion into a single scalar measure of speed, independent of direction. As expected, the global speed magnitude is highest at the beginning of the translational stage, when the Euclidean norm of the position error is maximal, and then decays as the turtle approaches the target. In this sense, Figure 13 provides a compact visualization of the convergence process already implied by Figures 11 and 12. Moreover, it makes clear that larger proportional gains do not alter only the individual Cartesian components, but also increase the overall translational aggressiveness of the controller. Consequently, the gain K_p can be interpreted not only as a parameter governing convergence rate, but also as a direct scaling factor for the global intensity of the commanded motion.

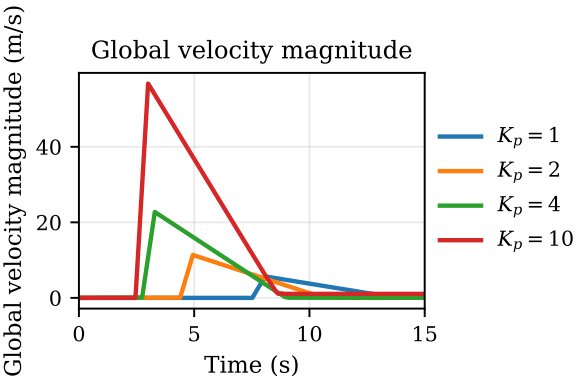

Figure 13: Magnitude of the turtle velocity in the global (inertial) reference frame (m/s) for different proportional gains. This quantity corresponds to the Euclidean norm of the velocity vector, combining the components shown in Figures 11 and 12. Higher gain values result in faster motion, while deviations from smooth profiles reflect the influence of discrete-time implementation effects and coupling with the turtle orientation.

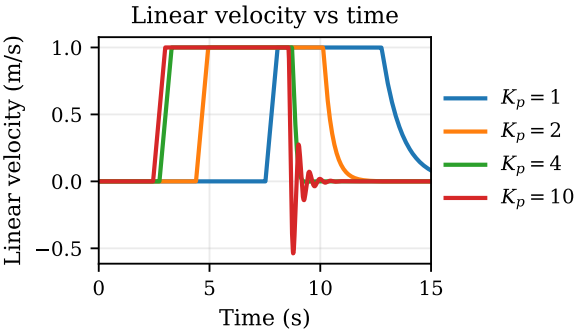

Figure 14: Linear velocity in the turtle body-fixed frame (m/s) for different proportional gains. For high gain values ($K_p = 10$), oscillatory behavior and temporary sign reversal can be observed, reflecting the influence of angular misalignment and discrete-time implementation effects.

The linear velocity expressed in the turtle frame is shown in Figure 14. In principle, during the translational stage, the turtle is expected to move predominantly along its body-fixed longitudinal axis, so that the relevant command is the scalar quantity v_T^x, given by Equation (16). By combining this equation with the peak values observed in Figures 11 and 12, one can estimate the approximate maximum longitudinal velocity in the body-fixed frame. Since, after the orientation stage, the turtle heading is nearly aligned with the direction of the global velocity vector, the longitudinal speed becomes approximately equal to the magni-

tude of the global velocity, that is, $v_T^x \approx \|\mathbf{v}_G\|$. At the initial instant of translation, the position errors are approximately $e_x^G = -3.5$ and $e_y^G = 4.5$, so that $\|\mathbf{e}_G\| = \sqrt{(-3.5)^2 + (4.5)^2} \approx 5.70$. Because the global proportional law yields $\|\mathbf{v}_G\| = K_p\|\mathbf{e}_G\|$, the approximate peak longitudinal velocities are 5.70, 11.40, 22.80, and 57.00 for $K_p = 1$, 2, 4, and 10, respectively. However, Figure 14 appears truncated because the plotted signal is limited by the admissible range imposed by the simulator or by the visualization scale. Thus, the graph does not fully display the actual peak value predicted from the global-frame components, even though such value can be inferred from Equation (16) together with the initial error magnitude.

An additional phenomenon deserves attention in Figure 14: for $K_p = 10$, the longitudinal velocity exhibits oscillatory behavior and even temporary sign reversal near the final stage of motion. This effect is consistent with the discussion introduced in Section 4, where reverse motion may emerge when the projection of the global velocity vector onto the turtle longitudinal axis becomes negative. In the present case, the oscillation is associated with the combined influence of angular misalignment, discrete-time implementation effects, and high controller gain. Because the turtle is not perfectly aligned with the target direction during the entire translational phase, especially for the most aggressive gain, the projected velocity in the body-fixed frame may fluctuate significantly. This behavior indicates that, for high proportional gains, the implemented closed-loop dynamics may depart from the idealized first-order continuous-time approximation adopted for analytical purposes. Accordingly, the oscillatory behavior observed in simulation should be understood as evidence of the limits of validity of the simplified model under non-ideal execution conditions, not as an internal inconsistency of the theoretical development. From a control perspective, the oscillation observed for $K_p = 10$ is undesirable because it reduces motion smoothness, may increase settling time, and suggests that the system is operating near a numerically sensitive regime.

The position responses shown in Figures 15 and 16 confirm the expected convergence of the translational motion. In both coordinates, increasing K_p generally reduces the time required to approach the target value, which is consistent with the first-order closed-loop behavior discussed previously. In the x-coordinate, the turtle moves from 5.5 toward 2.0, while in the y-coordinate it moves from 5.5 toward 10.0. The overall

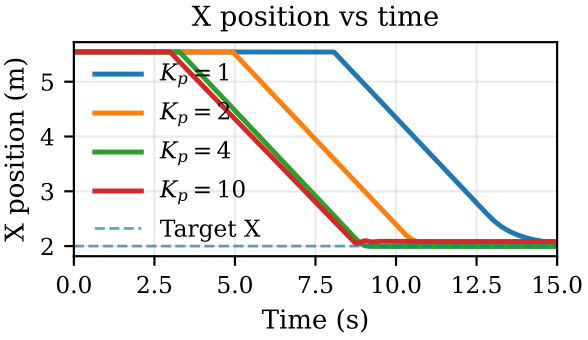

Figure 15: Global position of the turtle along the x-axis (m) for different proportional gains. The position is expressed in the inertial (global) reference frame. Higher gain values lead to faster convergence toward the target position, while small deviations from smooth trajectories reflect the influence of non-ideal implementation effects and coupling with the turtle orientation.

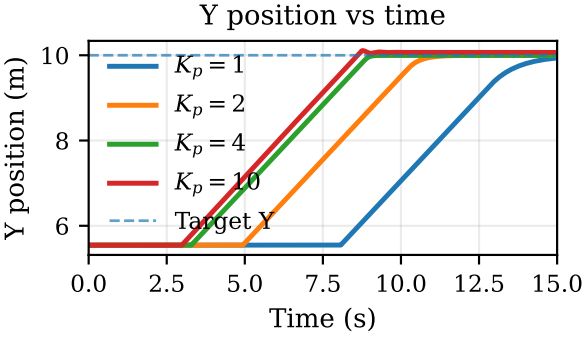

Figure 16: Global position of the turtle along the y-axis (m) for different proportional gains. The position is expressed in the inertial (global) reference frame and reflects the combined effects of translational control and robot orientation. For higher gain values, slight deviations from monotonic convergence can be observed due to discrete-time implementation effects and coupling between angular and translational dynamics.

interpretation is therefore analogous to that previously developed for the angular response: higher gains accelerate convergence but also amplify practical deviations from the ideal model, especially when actuator limitations, sampling effects, and frame-transformation sensitivities become relevant. This tendency is also reflected in Table 1, where the linear steady-state error remains very small for moderate gains but increases again for $K_p = 10$, indicating a degradation in practical performance despite the theoretically faster response.

Figure 17 provides a geometric synthesis of the translational behavior by depicting the trajectory of the

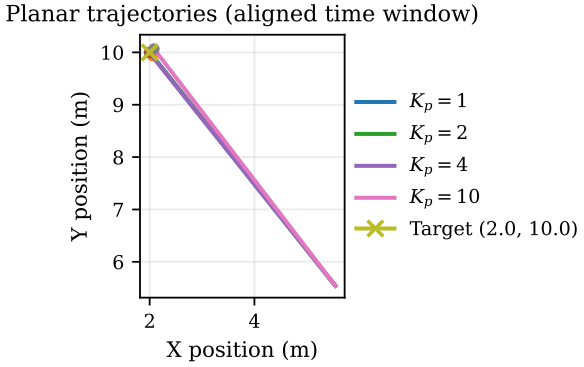

Figure 17: Time evolution of the turtle position in the Cartesian plane, global frame. The trajectories illustrate convergence toward the target position, with higher gains leading to faster motion but potentially less smooth paths due to non-ideal control effects.

turtle in the xy plane. For $K_p = 1$, 2, and 4, the trajectories remain close to the straight-line direction connecting the initial and desired positions, indicating that the orientation stage was sufficiently accurate to allow the translational controller to operate as intended. By contrast, for $K_p = 10$, the trajectory deviates more significantly and even moves away from the target region before attempting to recover. This result is compatible with the larger steady-state angular error reported in Table 1. Because the turtle begins the translational phase with a non-negligible orientation mismatch, the linear controller projects the commanded motion along a direction that does not coincide with the true line-of-sight to the target. Consequently, the robot is initially driven toward an incorrect direction, which explains the larger trajectory deviation observed in Figure 17. This result highlights the coupling between the angular and linear stages: although the control architecture is sequential, the quality of the translational motion depends directly on the accuracy achieved in the prior orientation phase.

From a broader perspective, the results demonstrate that the proposed approach is highly effective as a didactic framework. Its main strengths lie in its conceptual simplicity, direct visual interpretability, and close correspondence between classical control analysis and observable robotic motion. Students can clearly relate proportional gain variation to transient speed, steady-state accuracy, and trajectory shape, while also being introduced to essential robotics notions such as coordinate frames, vector projection, and the distinction between global and body-fixed variables. In addition, because the entire implementation relies on an accessible simulation environment and open-source tools, the methodology is reproducible and suitable for laboratory classes, guided exercises, or project-based learning activities.

At the same time, the experiments reveal important limitations of the adopted control strategy. First, the use of a pure proportional controller makes the motion sensitive to saturation, sampling effects, and residual orientation errors. Second, the sequential architecture—orientation first, translation afterward—simplifies the interpretation but may produce discontinuities at the switching instant and does not guarantee optimal smoothness. Third, high gains may lead to oscillatory responses, non-smooth velocity profiles, and trajectory deviations, as observed for $K_p = 10$. These effects indicate that, although the continuous-time analytical model predicts a stable first-order behavior, the implemented digital closed loop is also influenced by communication delays, finite sampling, numerical truncation, switching logic, and command saturation. Therefore, the analytical stability discussion presented in Section 4 should be interpreted as valid for the idealized model, whereas the practical simulation results reflect the behavior of the educational implementation under non-modeled effects, especially for high gain values.

Several improvements can therefore be proposed, both from control and pedagogical viewpoints. To obtain smoother and more stable motion, one may introduce velocity ramps, low-pass filtering, or explicit acceleration and jerk limits, thereby avoiding abrupt changes in the control signal. From the controller-design perspective, a PD or PID structure could be explored to improve damping and reduce oscillatory behavior, while gain scheduling could be used to decrease control aggressiveness near the target. Another relevant extension would consist of replacing the sequential switching logic with a continuous posture-control law, in which angular and linear velocities are adjusted simultaneously. Such an approach would reduce discontinuities and better reflect more advanced mobile-robot control strategies.

Pedagogically, these limitations should not be regarded as drawbacks alone, but rather as opportunities for deeper academic exploration. The present system is particularly suitable for classroom activities in which students first implement the basic proportional controller, then experimentally identify its limitations, and finally propose refinements grounded in control theory. Possible educational developments include varying the sampling frequency to study digital-control

effects, comparing P, PD, PI, and PID controllers, implementing saturation compensation, adding velocity smoothing, or extending the algorithm to track trajectories instead of performing only point-to-point motion. In this sense, the methodology not only illustrates the foundations of classical control, but also creates a natural bridge toward digital control, mobile robotics, and more advanced controller design topics. Consequently, the proposed approach fulfills a dual role: it is both an effective introductory teaching tool and a flexible platform for progressively more sophisticated academic activities.

From a broader perspective, it is important to situate the proposed framework in relation to existing tools and platforms commonly used in control and robotics education. Traditional environments such as MATLAB/Simulink provide powerful resources for analytical modeling, system design, and numerical simulation, being widely adopted in control engineering curricula. However, these tools often emphasize mathematical representation and block-diagram abstraction, with limited direct correspondence to physical motion and spatial behavior in robotic systems.

In the context of robotics-oriented simulation, platforms such as Gazebo and CoppeliaSim (formerly V-REP) offer high-fidelity physics engines, detailed sensor modeling, and support for complex three-dimensional environments. These characteristics make them highly suitable for advanced robotics research and system validation. Nevertheless, their level of detail, computational requirements, and multi-layered architectures may introduce additional complexity for students in introductory courses, potentially shifting the focus away from fundamental control concepts.

Recent educational frameworks based on ROS, including web-based environments and modular learning platforms, have sought to improve accessibility and scalability by enabling remote experimentation and structured learning workflows (CAñAS et al., 2020; ROLDáN-ÁLVAREZ; MAHNA; CAñAS, 2021; ROLDáN-ÁLVAREZ et al., 2023; RAUDMäE et al., 2023; RYALAT et al., 2025). While these approaches provide valuable contributions to robotics education, they are often oriented toward general robotics training, system integration, or platform-specific learning.

In contrast, the approach proposed in this work is specifically designed to support the teaching of classical control concepts through a simplified and visually intuitive robotic environment. By employing a lightweight two-dimensional simulator, the framework prioritizes conceptual clarity, direct visualization of system behavior, and a clear mapping between analytical formulations and observable motion. Rather than competing with high-fidelity simulators or comprehensive robotics platforms, the proposed methodology complements existing tools by addressing the specific needs of introductory control education.

6 Educational Applications

Beyond the analytical and simulation results presented in this work, the proposed framework can also serve as a practical educational platform for teaching fundamental concepts of control systems and mobile robotics. Because the system is simple, visually intuitive, and implemented using widely accessible open-source tools, it is particularly suitable for classroom demonstrations, guided laboratory exercises, and project-based learning activities in undergraduate engineering courses.

The Turtlesim environment provides an immediate visual representation of system behavior, allowing students to directly relate theoretical control concepts to observable motion in the plane. By modifying controller parameters and observing the resulting trajectories and velocity profiles, learners can develop an intuitive understanding of transient response, gain tuning, and the influence of implementation constraints such as actuator saturation and sampling effects.

In addition to the experiments described in this paper, several complementary learning activities can be developed around the same framework. These activities allow instructors to gradually increase the conceptual depth of the exercises, progressing from basic proportional control toward more advanced topics in digital control and mobile robotics. Table 2 summarizes examples of suggested classroom or laboratory activities that can be implemented using the proposed system.

These activities demonstrate that the proposed framework extends beyond a simple illustrative example. It can serve as a flexible educational platform in which students progressively explore increasingly sophisticated control concepts while maintaining a direct connection between theoretical analysis and observable robotic behavior. By combining classical control theory with an interactive simulation environment, the methodology helps bridge the gap between mathematical modeling, algorithm implementation, and physical interpretation of control systems.

Table 2: Suggested classroom activities based on the proposed Turtlesim control framework.

Activity	Learning objective	Description
Proportional gain tuning	Understand the influence of controller gain on transient response	Students vary the proportional gain K_p and analyze its effect on response speed, settling time, and trajectory behavior. The exercise reinforces classical concepts such as time constant and gain–performance trade-offs.
Velocity saturation analysis	Investigate practical implementation constraints in control systems	Learners analyze the effect of actuator saturation on the control signal and transient response. By observing clipped velocity signals, students understand how physical limits modify theoretical system behavior.
Sampling-rate exploration	Introduce the impact of discretization in digital control systems	Students modify the sampling frequency of the control loop and observe how digital implementation affects response smoothness, stability margins, and oscillatory behavior.
Controller structure comparison	Compare classical control strategies	The basic proportional controller can be extended to PD, PI, or PID controllers. Students evaluate how each structure influences transient performance, damping, and steady-state accuracy.
Trajectory tracking extension	Introduce concepts of motion control in mobile robotics	Instead of performing point-to-point positioning, students implement algorithms that allow the robot to follow predefined trajectories, such as lines, polygons, or circular paths.

7 Conclusions

This work presented a didactic framework for teaching classical feedback control concepts using the ROS 2 Turtlesim simulator. The proposed approach combines a simple proportional controller with a visual robotic environment, allowing students to directly observe how theoretical control principles translate into motion within a simulated robotic system.

The results demonstrate a clear relationship between the analytical predictions derived from an ideal continuous-time first-order model and the behavior observed in simulation. In particular, the influence of the proportional gain on transient response, convergence speed, and trajectory characteristics was systematically analyzed. At the same time, the experiments revealed important practical aspects, including actuator saturation, discrete-time implementation effects, and the coupling between rotational alignment and translational motion. These effects highlight the limits of validity of the simplified analytical model and provide valuable insight into the differences between idealized theory and implemented control systems.

From an educational perspective, this duality between analytical modeling and observed behavior constitutes a central contribution of the proposed framework. By explicitly exposing both agreement and deviation between theory and practice, the approach enables a deeper understanding of feedback control concepts, supporting the transition from purely mathematical formulations to realistic system behavior.

Future work may extend the proposed framework in several directions. Possible developments include the implementation of more advanced controllers such as PD, PI, or PID structures, the investigation of digital control effects through sampling-frequency analysis, and the extension of the algorithm from point-to-point positioning to trajectory tracking tasks. Additional studies may also explore the integration of the methodology with physical robotic platforms, allowing students to transition from simulation-based learning to real-world control experiments.

References

ARAÚJO, A. M. N. et al. Labvcon: A virtual laboratory for control engineering education. In: *Proceedings of the Brazilian Congress of Automation (CBA)*. Fortaleza, Ceará, Brazil: Brazilian Society of Automatics, 2022. p. 888–892.

ASSIS, W. O.; COELHO, A. D.; LIMA, F. R. G. A didactic program for teaching control systems in engineering laboratories. In: *Proceedings of the Brazilian Congress of Automation (CBA)*. Juiz de Fora, Minas Gerais, Brazil: Brazilian Society of Automatics, 2008.

BELHOT, R. V.; FIGUEIREDO, R. S.; MALAVÉ, C. O. The use of simulation in engineering education. In: *Proceedings of the Brazilian Congress on Engineering Education (COBENGE)*. Porto Alegre, RS, Brazil: Associação Brasileira de Educação em Engenharia, 2001. p. 447–451.

BREGANON, R. et al. Development of inverted pendulum systems as didactic tools in control engineering education. *Holos*, v. 37, n. 5, p. 1–12, 2021.

CAñAS, J. M.; PERDICES, E.; GARCíA-PéREZ, L.; FERNáNDEZ-CONDE, J. A ros-based open tool for intelligent robotics education. *Applied Sciences*, MDPI AG, v. 10, n. 21, p. 7419, out. 2020. ISSN 2076-3417.

CHEN, H. *Development of teaching materialfor Robot Operating System (ROS):creation and control of robots*. Dissertação (mathesis) — Aalto university - School of Engineering, Espoo, Finland, 2022. Disponível em: <https://urn.fi/URN:NBN:fi:aalto-202208285037>.

COELHO, A. A. R.; ALMEIDA, O. M.; SANTOS, J. E. S.; SUMAR, R. R. Simulation laboratories in the teaching of signals and linear systems. In: *Proceedings of the Brazilian Congress on Engineering Education (COBENGE)*. Porto Alegre, RS, Brazil: Associação Brasileira de Educação em Engenharia, 2001. p. 1–8.

COONEY, M.; YANG, C.; SIVA, P.; ARUNESH, S.; DAVID, J. Teaching robotics with robot operating system (ros): A behavior model perspective. In: . [S.l.: s.n.], 2018.

DORF, R. C.; BISHOP, R. H. *Modern Control Systems*. 13. ed. Rio de Janeiro: LTC, 2018.

FARZAN, S. *Project-Based Learning for Robot Control Theory: A Robot Operating System (ROS) Based Approach*. [S.l.]: arXiv, 2023.

KERSCHBAUMER, R.; LIMA, C. R. E.; SIMÃO, J. M. Using the v-rep robot simulator in the teaching of autonomous robot control techniques. In: *Proceedings of the Brazilian Congress on Engineering Education (COBENGE)*. Curitiba, PR, Brazil: Associação Brasileira de Educação em Engenharia, 2014.

KIRK, D. E. *Optimal Control Theory: An Introduction*. Englewood Cliffs: Prentice-Hall, 1970.

KLUEVER, C. A. *Dynamic Systems: Modeling, Analysis, and Simulation*. Rio de Janeiro: LTC, 2018.

NISE, N. S. *Engineering Control Systems*. 6. ed. Rio de Janeiro: LTC, 2013.

RAUDMäE, R.; SCHUMANN, S.; VUNDER, V.; OIDEKIVI, M.; NIGOL, M. K.; VALNER, R.; MASNAVI, H.; SINGH, A. K.; AABLOO, A.; KRUUSAMäE, K. Robotont – open-source and ros-supported omnidirectional mobile robot for education and research. *HardwareX*, Elsevier BV, v. 14, p. e00436, 2023. ISSN 2468-0672.

RENARD, E. *ROS 2 from Scratch*. Birmingham: Packt Publishing, 2024.

ROLDáN-ÁLVAREZ, D.; CAñAS, J. M.; VALLADARES, D.; ARIAS-PEREZ, P.; MAHNA,

S. Unibotics: open ros-based online framework for practical learning of robotics in higher education. *Multimedia Tools and Applications*, Springer Science and Business Media LLC, v. 83, n. 17, p. 52841–52866, nov. 2023. ISSN 1573-7721.

ROLDáN-ÁLVAREZ, D.; MAHNA, S.; CAñAS, J. M. A ros-based open web platform for intelligent robotics education. In: ____. *Robotics in Education*. [S.l.]: Springer International Publishing, 2021. p. 243–255. ISBN 9783030825447.

RYALAT, M. Empowering engineering education with the robot operating system (ros): An open source platform for robotics learning. In: *2025 IEEE 12th International Conference on E-Learning in Industrial Electronics (ICELIE)*. [S.l.]: IEEE, 2025. p. 1–6.

RYALAT, M.; ALMTIREEN, N.; AL-REFAI, G.; ELMOAQET, H.; RAWASHDEH, N. Research and education in robotics: A comprehensive review, trends, challenges, and future directions. *Journal of Sensor and Actuator Networks*, MDPI AG, v. 14, n. 4, p. 76, 2025. ISSN 2224-2708.

SANTOS, T. M. B.; FAVORETO, D. G. S. B.; CARNEIRO, M. M. d. O.; PINTO, M. F.; ZACHI, A. R. L.; GOUVEA, J. A.; MANHãES, A.; ALMEIDA, L. F.; SILVA, G. R. Introducing robotic operating system as a project-based learning in an undergraduate research project. In: *2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)*. [S.l.]: IEEE, 2023. p. 585–590.

SESSHINI, I.; GALVEZ, J. M. A virtual laboratory for automation and control education. In: *Proceedings of the Brazilian Congress on Engineering Education (COBENGE)*. Curitiba, PR, Brazil: Associação Brasileira de Educação em Engenharia, 2007.

SILVA, E. A.; QUIRINO, R. B.; GASOTO, M. A. A didactic computational environment for the study of dynamic systems in engineering. In: *Proceedings of the Brazilian Congress on Engineering Education (COBENGE)*. Belém, PA, Brazil: Associação Brasileira de Educação em Engenharia, 2012.