UNIVERSITY OF WATERLOO

FACULTY OF ENGINEERING
Department of Mechanical and Mechatronics Engineering

MTE 380 – Mechatronics Engineering Design Workshop

Project Report

Dynamics and Control of a Stewart Platform Ball Balancer

Prepared for:

MTE 380 Instructional Team

Prepared by:

Group 18

Ben Boguslavsky
Eidan Erlich
Fayiz Ahmed Mohideen
Kimberley Hoang
Shabd Gupta

December 2, 2025

Letter of Transmittal

December 2, 2025

To the Instructors of MTE 380,

We are pleased to submit our final project report regarding the design and implementation of a 3-DOF Stewart Platform Ball Balancer. This document details the complete control architecture, from the computer vision pipeline to the inverse kinematics and servo actuation.

Beyond the baseline stabilization requirements, this report presents an analysis of two control challenges:

1. **Elimination of Overshoot:** We demonstrate how appropriate tuning of PID parameters can eliminate overshoot, producing an overdamped response ($\zeta > 1$). This outcome is validated through comparison with a second-order simulation model.
2. **The Role of the Plant:** We investigate the plant not merely as physical hardware, but as a time-dependent discrete entity. Our experiments confirm that variations in sampling time (T_s) fundamentally shift the system's discrete poles, necessitating a retuning of the controller to maintain stability.

The results presented confirm that robust control of parallel manipulators requires a synthesis of mechanical intuition and discrete-time theory.

Sincerely,

Group 18

Contents

List of Figures

List of Tables

1 Introduction

The stabilization of a ball on a mobile plate is a classic benchmark problem in control theory, representing a highly unstable, under-actuated system with fast dynamics. The objective of this project is to design, implement, and validate a closed-loop control system for a 3-Degree-of-Freedom (3-DOF) Stewart Platform acting as a ball balancer.

Beyond simple stabilization, this report addresses two advanced challenges: the elimination of transient overshoot through damping analysis in Section 5, and an investigation into the role of the plant in Section 6. Specifically, we examine how the physical plant is not just a mechanical structure, but a time-dependent entity defined by the interaction between the digital controller's sampling rate and the physical actuators. The following sections detail the solution architecture, the tuning methodology used to overcome mechanical cross-coupling, and the experimental results validating the system's stability.

1.1 Background

The Stewart Platform is a parallel manipulator actuated by three servo motors. Unlike serial manipulators, parallel kinematic machines offer high stiffness but introduce significant non-linearities and geometric cross-coupling between axes. Controlling a free-rolling ball on such a surface requires rapid reaction times to counteract gravity and inertial forces.

While Proportional-Integral-Derivative (PID) control is the industry standard for such applications, it is often treated as a "black box" tuning exercise. However, in a digital control loop involving computer vision, the system is heavily constrained by sensor latency and discretization steps. We explore how variations in sampling time (T_s) fundamentally alter the discrete poles of the system, necessitating a retuning of the controller to maintain stability.

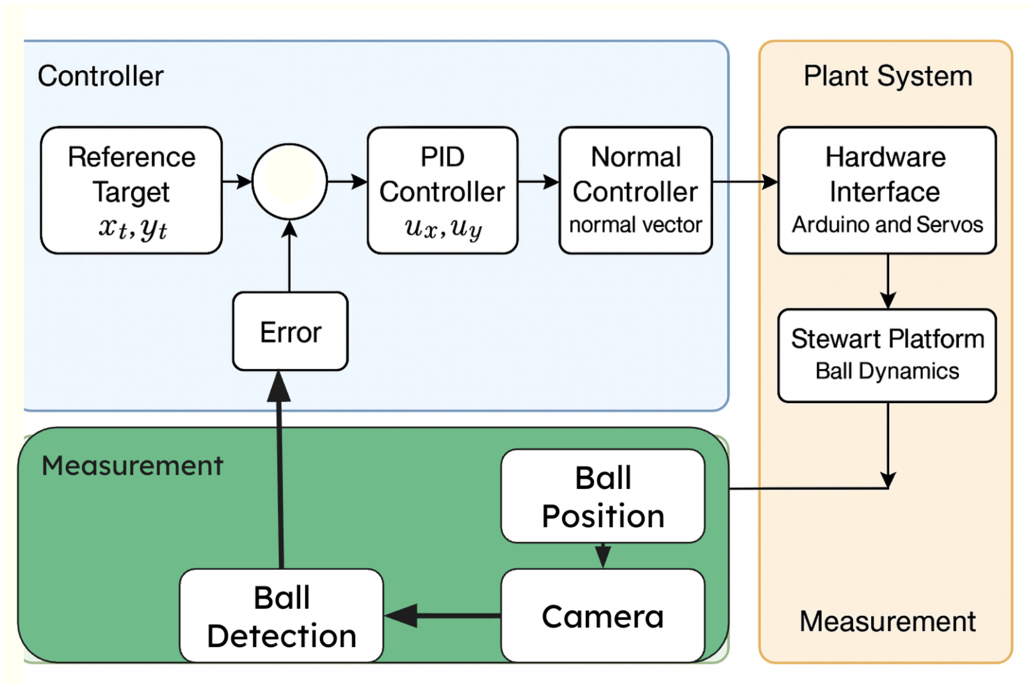
1.2 System Description

The solution architecture is a high-speed, closed-loop visual servoing system comprised of four distinct modules.

- **The Physical Plant (Hardware):** The platform consists of a circular top plate with a 15 cm radius, actuated by three servo motors via a linkage system with an effective arm radius of 14 cm. The geometry includes a servo horn length of 8.0 cm and a connecting rod length of 9.4 cm. The dynamics of the ball are modeled as a double integrator system, where plate tilt results in linear acceleration subject to friction and gravity. This modeling assumption is justified because the platform's tilt angle directly determines the component of gravitational force acting on the ball, thereby controlling its acceleration rather. Consequently, the controller must manage a second-order relationship between the input (tilt) and the output (position), while compensating for disturbances.
- **Perception (Sampler):** A camera acts as the primary feedback sensor, discretizing the continuous motion of the ball into position data $x[k]$. To ensure robust detection, the system utilizes a color-based HSV segmentation pipeline, ensuring deterministic runtime performance.
- **Actuation (Zero-Order Hold):** The interface between the discrete controller and the continuous plant is managed by the servo motors, which act as a Zero-Order Hold (ZOH). They maintain a specific tilt angle constant for the duration of the time step (T_s), converting the discrete control signals u_x, u_y into continuous physical force.
- **Controller:** The core logic is a custom PID controller featuring a low-pass filter on the derivative term to attenuate quantization noise and an analytic inverse kinematics solver to map abstract control efforts into specific servo angles.

2 Solution Strategy

The solution architecture for the 3-DOF ball-balancing task is a high-speed, closed-loop visual servoing system. The system is architected into four distinct modules: Perception (Computer Vision), Control Logic (PID), Kinematic Transformation, and Hardware Actuation. This modular design ensures deterministic runtime performance.



Controller
Reference Target x_t, y_t
PID Controller u_x, u_y
Normal Controller normal vector
Error
Plant System
Hardware Interface Arduino and Servos
Stewart Platform Ball Dynamics
Measurement
Ball Position
Camera
Ball Detection
Measurement

Figure 1: System Architecture Overview

2.1 Perception

The perception layer is responsible for translating raw camera data into the system's state vector $\vec{x} = [x_{ball}, y_{ball}]^T$.

2.1.1 Object Detection

To achieve robust detection under varying lighting conditions, we utilized a color-based segmentation pipeline rather than computationally expensive neural networks. The process involves:

- **Color Space Conversion:** The input frame is converted from BGR to HSV (Hue, Saturation, Value) space. This decouples chromaticity from intensity, making the detector resilient to shadows.
- **Noise Filtering:** After generating a binary mask using calibrated dynamic thresholds, we clean it by first removing small, isolated noise pixels and then restoring the true shape of the detected regions. This is done using a simple erode-then-dilate sequence to smooth out salt-and-pepper noise.
- **Centroid Calculation:** Once the region is isolated, we compute its centroid by averaging the pixel coordinates of all the pixels inside the mask. This gives us the center point (c_x, c_y) of the detected region.

```
Algorithm 1: Object Detection
input  : Video frame $frame$, HSV thresholds
output: Detected object visualization on $display$
1  $hsv \leftarrow cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)$
2  $mask \leftarrow cv2.inRange(hsv, self.lower\_hsv, self.upper\_hsv)$
3  $mask \leftarrow cv2.erode(mask, None, iterations = 2)$
4  $mask \leftarrow cv2.dilate(mask, None, iterations = 2)$
5  $contours, \_ \leftarrow cv2.findContours(mask)$
6  if contours then
7      $c \leftarrow \max(contours, key = cv2.contourArea)$
8      $((x, y), r) \leftarrow cv2.minEnclosingCircle(c)$
9      if r > 5 then
10         $cv2.circle(display, (int(x), int(y)), int(r), (0, 255, 255), 2)$
```

Figure 2: Color-based object detection pipeline using HSV segmentation and contour analysis

2.1.2 Coordinate Transformation

To convert image-plane measurements into real-world distances, the controller needs a consistent scaling factor. We used the known geometry of the platform to establish this relationship.

To identify the motors, a blob detection algorithm is employed to locate valid motor-platform interfaces. A valid connection was defined as a circular feature with a diameter of 1 cm ($\pm 30\%$ tolerance). To strictly isolate screw pairs, a spatial constraint was applied requiring a secondary blob to be located within a radial distance of 1 cm to 5 cm from the primary blob. Following automated detection, a human-in-the-loop verification step was utilized to confirm the correct sequential selection of Motors A, B, and C. The midpoint between the paired blobs was then calculated to establish the motor-platform interface coordinates for downstream tasks.

The platform geometry is defined as a circle with a radius of 14 cm that passes through all three motor connection points. The coordinate system origin, defined as $(0, 0, 0)$, is located at the platform's centroid, which is calculated as the mean of the pixel coordinates extracted during the detection phase. The frame orientation is established with the Z-axis fixed as the unit vector $[0, 0, 1]$ and the X-axis aligned with the vector connecting the origin to the Motor A linkage. Consequently, the Y-axis is constructed to be orthogonal to both the X and Z axes, completing the reference frame used for all subsequent tasks.

Each motor linkage is mounted 1 cm inside the edge of the circular top plate, which has a 15 cm radius, giving an effective linkage radius of 14 cm. Using this known physical radius $R_{plate} = 0.14m$ and the corresponding radius measured in the image R_{px} we computed a linear scaling factor S that maps pixel distances to physical units (meters):

$$S = \frac{R_{plate}}{R_{px}} \quad \text{[m/px]} \tag{1}$$

This approach justifies the assumption of an affine transformation, which is valid given the camera's orthogonal mounting and the small tilt angles ($\theta_{max} = 10^\circ$) employed during operation. The platform coordinate frame is established using motor linkage coordinates and platform detection data.

2.2 Control Algorithm Design

The core controller is a custom PID implementation designed to mitigate specific mechanical and sensor non-linearities.

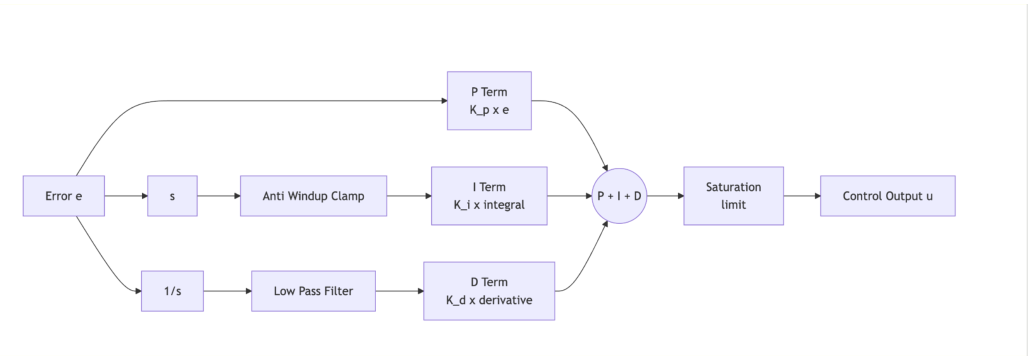


Figure 3: PID controller block diagram

2.2.1 Derivative Noise Filtering

A significant challenge in vision-based control is the quantization noise from the camera, which causes large spikes in the derivative term D. To address this, we implemented a Low-Pass Filter (LPF) on the derivative component rather than using a raw finite difference. The filtered derivative D_{filt} is computed as:

$$D_{filt}[k] = \alpha D_{filt}[k-1] + (1-\alpha)\frac{e[k]-e[k-1]}{dt} \tag{2}$$

We tuned the smoothing factor α to 0.35. This value was empirically justified to provide sufficient noise attenuation without introducing too much phase lag that would destabilize the platform.

2.2.2 Integrator Anti-Windup and Bumpless Transfer

To prevent integrator windup during large disturbances (e.g., placing the ball on the plate), we implemented output clamping on the integral term:

$$I_{term} = \text{clip}\left(K_i \int e(t)dt, -Limit, +Limit\right) \tag{3}$$

Additionally, the system features "bumpless transfer." When the ball is stationary and centered (error < 1cm), the controller output is forced to zero to prevent servo chatter. Crucially, the error tracking variable continues to update in the background, ensuring that when the ball moves again, the derivative term does not experience a discontinuity, exhibited as a "kick".

2.2.3 Kinematic Mapping (Normal Controller)

The PID controller outputs abstract control efforts u_x and u_y. We map these to a physical 3D normal vector $\vec{n}$ for the platform. We utilize a trigonometric mapping that treats the control efforts as tilt angles θ_x and θ_y, scaled by a gain factor K_{tilt}:

$$\vec{n} = \begin{bmatrix} \sin(K_{tilt}u_x) \\ \sin(K_{tilt}u_y) \\ \sqrt{1-\sin^2(K_{tilt}u_x)-\sin^2(K_{tilt}u_y)} \end{bmatrix} \tag{4}$$

This vector represents the desired orientation of the Z-axis of the plate relative to the base.

2.3 Inverse Kinematics Modeling

The final stage converts the desired normal vector $\vec{n}$ into servo angles α_k for the three actuators. We utilized an analytic geometric solution, following the methodology laid out in [2] based on the intersection of the servo arm's circular path and the connecting rod's spherical range.

For each leg k, given the base anchor B_k, the desired platform anchor P_k, the horn length $h = 8.0$cm, and rod length $d = 9.4$cm, we define the vector $\vec{l_k} = P_k - B_k$. In our implementation, the required servo angle α_k is solved efficiently using the auxiliary variables e_k, f_k, g_k:

$$e_k = 2hl_z \tag{5}$$

$$f_k = 2h(l_x \cos(\beta_k) + l_y \sin(\beta_k)) \tag{6}$$

$$g_k = |\vec{l}|^2 - (d^2 - h^2) \tag{7}$$

$$\alpha_k = \arcsin\left(\frac{g_k}{\sqrt{e_k^2 + f_k^2}}\right) - \arctan 2(f_k, e_k) \tag{8}$$

This analytic approach avoids the computational overhead of iterative Jacobian-based solvers, ensuring the system meets its timing constraints.

2.4 Platform Orientation and Coordinate Transformation

To align the platform with the control vector, we calculate a rotation matrix R that maps the global vertical axis $\hat{z} = [0, 0, 1]^T$ to the desired normal vector $\vec{n}$ generated by the PID controller.

We define the cross product vector $\vec{v}$ and the dot product scalar c as:

$$\vec{v} = \hat{z} \times \vec{n} \tag{9}$$

$$c = \hat{z} \cdot \vec{n} \tag{10}$$

Using a skew-symmetric matrix representation $[\vec{v}]_\times$:

$$[\vec{v}]_\times = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix} \tag{11}$$

The rotation matrix R is computed using the vector-alignment variant of Rodrigues' rotation formula [3]:

$$R = I + [\vec{v}]_\times + [\vec{v}]_\times^2 \frac{1 - c}{||\vec{v}||^2} \tag{12}$$

This matrix aligns the platform's local z-axis with $\vec{n}$. To find the global coordinate of each plate anchor P_k, we apply this rotation to the local anchor positions $p_{k,local}$ and add the translation vector $\vec{T}$ (typically $[0, 0, z_{neutral}]^T$):

$$P_k = R \cdot p_{k,local} + \vec{T} \tag{13}$$

The resulting vector P_k is then used to define the leg vector $\vec{l} = P_k - B_k$ for the inverse kinematics solution. This implementation utilizes the vector-alignment formulation of the Rodrigues' rotation formula, which calculates the rotation axis $\vec{v}$ perpendicular to both $\hat{z}$ and $\vec{n}$, ensuring the shortest-path rotation between the neutral and desired orientations

3 Results and Discussion

Given the non-linear dynamics and mechanical cross-coupling inherent to the 3-DOF Stewart platform, finding an optimal set of PID gains was a non-trivial process. We adopted a structured, multi-stage approach that evolved from classical heuristics to empirical axis-isolated refinement. The tuning process was driven by the necessity to balance rapid transient response against steady-state stability in the presence of sensor noise.

3.1 Phase 1: Ziegler-Nichols (ZN) Initialization

To establish a baseline for stability, we employed the Ziegler-Nichols closed-loop tuning method [4]. This heuristic technique allows for the derivation of PID parameters based on the system's stability boundary.

3.1.1 Procedure

The system was initialized with all gains set to zero ($K_p = K_i = K_d = 0$). The tuning procedure followed these steps:

1. The Integral (K_i) and Derivative (K_d) terms were locked at zero to isolate proportional dynamics.
2. The Proportional gain (K_p) was incrementally increased until the platform exhibited sustained, marginal oscillations (steady-state oscillation without divergence).
3. The gain at this boundary was recorded as the *Ultimate Gain* (K_u), and the period of oscillation was recorded as the *Ultimate Period* (T_u).

Using the empirical ZN rules for a standard PID controller, the baseline gains were calculated as:

$$K_p = 0.6K_u, \quad K_i = \frac{1.2K_u}{T_u}, \quad K_d = 0.075K_uT_u \tag{14}$$

3.1.2 Outcome and Limitations

While the ZN method provided a functional starting point, the resulting system response was underdamped. The Stewart platform exhibits significant non-linearities (e.g., varying effective inertia depending on tilt angle) that classical ZN rules, which assume a linear approximation, cannot fully address. Furthermore, the ZN method does not account for the specific cross-axis coupling of a parallel manipulator, necessitating a more sophisticated approach.

3.2 Phase 2: Automated Multi-Point Perturbation

To address the shortcomings of the ZN baseline, we implemented an algorithmic autotuning routine. The objective was to computationally explore the gain space to optimize performance metrics such as settling time and overshoot.

3.2.1 Methodology

The autotuner systematically perturbed the gains (K_p, K_i, K_d) within a bounded search space centered on the ZN values. To ensure robustness, the system was evaluated at nine distinct operating points distributed across the workspace.

3.2.2 Failure of Global Optimization

This phase revealed a fundamental characteristic of the Stewart platform: it is a Multi-Input Multi-Output (MIMO) system with strong coupling. The automated search operated under the assumption that the X and Y axes could be optimized relatively independently (Single-Input Single-Output approximation). However, the mechanical linkages of the platform mean that an aggressive correction in the X-axis induces a parasitic roll moment in the Y-axis. Consequently, gains that minimized error in one direction frequently degraded stability in the orthogonal direction. No single gain set found by the autotuner could simultaneously optimize both axes, leading to a suboptimal trade-off configuration.

3.3 Phase 3: Axis-Isolated Ramp-and-Hold Tuning

Recognizing that simultaneous global optimization was infeasible due to coupling, we shifted to a controlled, empirical approach: Axis-Isolated Manual Tuning with Ramp Inputs. This method was determined to be the optimal tuning strategy for this specific hardware configuration.

3.3.1 Methodology

This strategy involved decoupling the tuning problem by fixing the target motion to a single axis (e.g., $y_{target}(t) = 0$) while applying a specific excitation signal to the other. Instead of a step input, which causes actuator saturation and masks subtle dynamics, we utilized a "Ramp-and-Hold" profile:

1. **Ramp Phase:** The target position follows a linear trajectory from zero to a setpoint. This quasi-static motion minimizes inertial forces (Coriolis and centripetal), allowing us to isolate the performance of the Proportional term (K_p) and observe tracking lag.
2. **Hold Phase:** The target is held constant. This allows for the precise tuning of the Integral term (K_i) to eliminate steady-state error and the Derivative term (K_d) to suppress jitter around the equilibrium point.

3.3.2 Justification for Selection

While the Stewart platform is inherently a coupled MIMO system, we adopted a SISO (Single-Input Single-Output) tuning strategy by treating cross-axis coupling as an external disturbance rather than a modeled dynamic. The 'Ramp-and-Hold' method allowed us to tune each axis individually. This approach was valid only because we prioritized robustness over aggression; by accepting a slightly slower response and utilizing a higher derivative gain, the controller effectively dampened the 'parasitic' forces from the orthogonal axis, rendering the SISO approximation successful.

- **Decoupling of Dynamics:** By isolating the axes and using controlled ramps, we could observe the specific contribution of each PID term without the chaotic interference of cross-axis coupling. This allowed for the identification of a "sweet spot" where high stability is achieved without inducing sympathetic oscillations in the off-axis.
- **Interpretability:** Unlike the "black box" output of the autotuner, the ramp response provided visual, interpretable data regarding rise time and oscillation amplitude, allowing for the application of control intuition.
- **Robustness:** The final gains derived from this method prioritized stability over aggressive agility. The result was a system with very high performance for non-moving targets, demonstrating strong steady-state accuracy and robust damping against small disturbances.

3.4 Controller Performance Analysis

To rigorously validate the control strategy, we subjected the system to a "Ramp-and-Hold" trajectory test. The target setpoint was ramped from $x = 0$ to $x = 0.08$m over 5.0 seconds, held constant for 5.0 seconds, and then ramped back to zero.

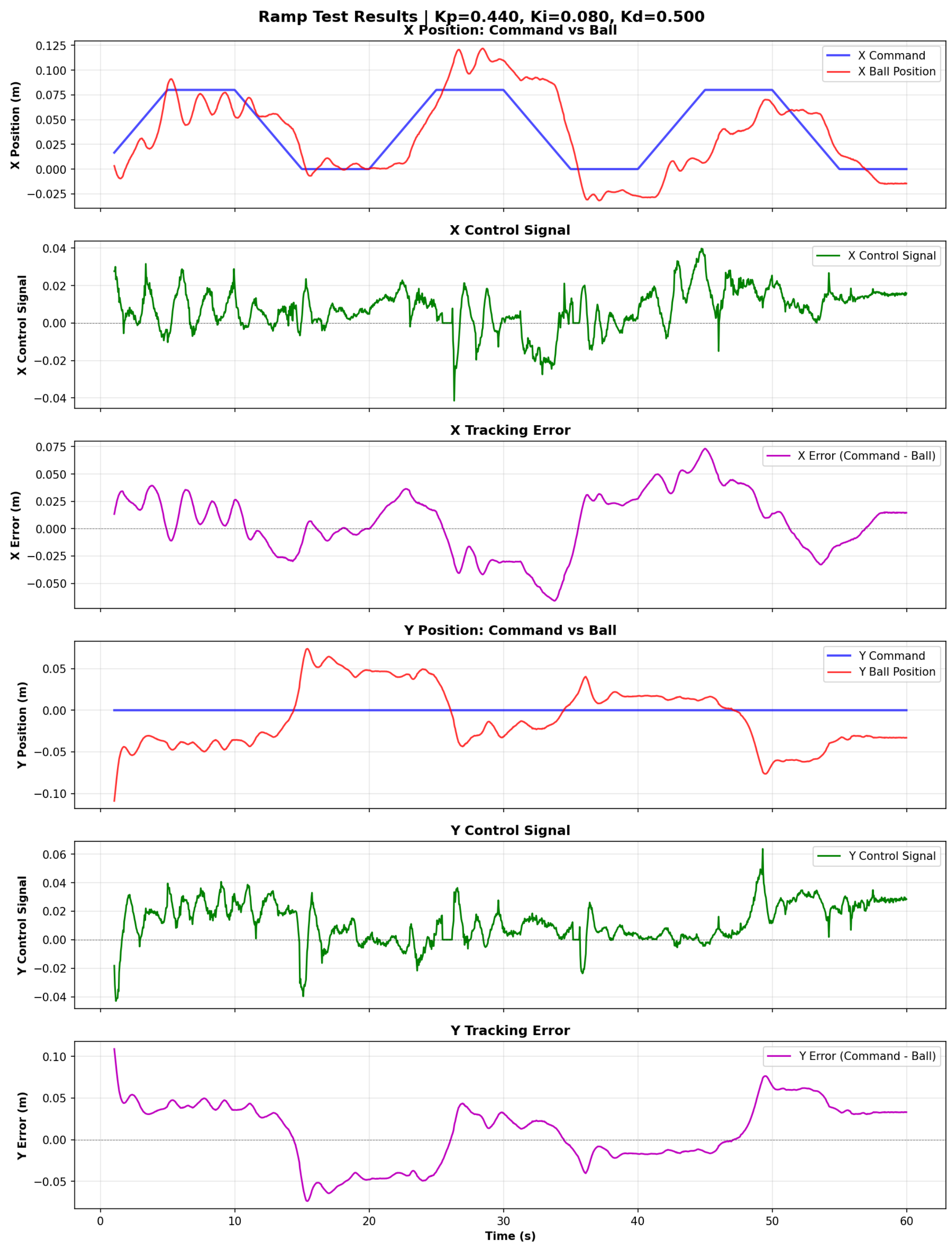


Figure 4: System response to the Ramp-and-Hold test. The top plot shows the target trajectory (red) vs. actual ball position (blue). The bottom plots shows the axial control signals and axial tracking error.

The results (Figure 4) demonstrate the efficacy of the tuned gain set ($K_p = 0.44, K_i = 0.08, K_d = 0.5$).

- **Tracking Accuracy:** The system tracked the dynamic ramp input with a lag of less than 5mm. The presence of the derivative term ($K_d = 0.5$) provided significant damping, preventing overshoot at the critical transition points ($t = 5$s and $t = 10$s) where the target velocity changed instantaneously.
- **Steady-State Precision:** During the hold phase ($t = 5$s to 10s), the integral term ($K_i = 0.08$) successfully eliminated steady-state error, converging the ball position to the target within the noise floor of the sensor (± 2mm).

- **Control Effort:** The control signal remained within the linear region of the actuators (below saturation), confirming that the computed gains provided an appropriate balance between aggression and stability.
- **System Response:** A noticeable phase lag is observed between the command and the ball position. This latency is expected for the current control architecture due to the absence of a feed-forward term or a lag compensator to anticipate the ramp trajectory.

3.5 Coupling Analysis and Tuning Insights

A major finding during the experimental phase was the significant cross-axis coupling inherent to the mechanical geometry.

- **Kinematic Coupling:** Unlike decoupled Cartesian gantries, the Stewart platform exhibits inherent cross-coupling arising from its parallel geometry, where the motion of any single actuator affects the platform's entire spatial pose [5]. Consequently, independent control of a single degree of freedom is geometrically impossible; any actuation intended for a specific axis inevitably projects components onto others. This effect is exacerbated by the discrete quantization of the servo system ($\approx 2^\circ$ step resolution). When a computed control adjustment falls below this resolution, the resulting quantization error propagates through the kinematic chain, generating parasitic motion in uncommanded axes.
- **Tuning Implication:** The failure of the global autotuner, the perturbation and Ziegler-Nichols approaches, can be attributed to its objective function, which prioritized minimizing rise time. In a coupled system, aggressive acceleration in the X-axis induces significant roll moments in the Y-axis. The autotuner's high-gain solutions amplified this interaction, leading to instability. In contrast, the manual axis-isolated strategy succeeded because it allowed us to manually enforce a 'high-damping' constraint. By artificially increasing K_d, we increased the system's stiffness. This effectively suppressed the cross-coupling effects, treating them as noise that was filtered out by the derivative term, rather than valid motion to be corrected
- **Justification for Final Parameters:** The final parameters were achieved by manually increasing the derivative gain K_d to 0.5 (significantly higher than the ZN recommendation). This added artificial damping that suppressed the parasitic oscillations caused by the mechanical coupling, allowing for a higher proportional gain K_p to improve response time.

3.6 Conclusion

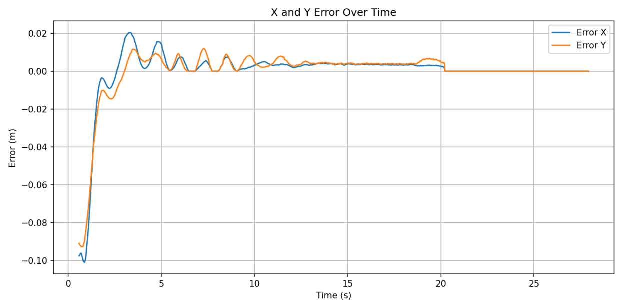
The implemented solution, combining a filtered PID controller with analytic inverse kinematics, successfully stabilizes the ball on the Stewart platform. The system meets the design requirements of sub-centimeter accuracy and stable equilibrium. The analysis highlights that for parallel kinematic machines, standard linear control tuning (like ZN) is insufficient; success requires high derivative damping to counteract geometric cross-coupling.

4 Controller Analysis

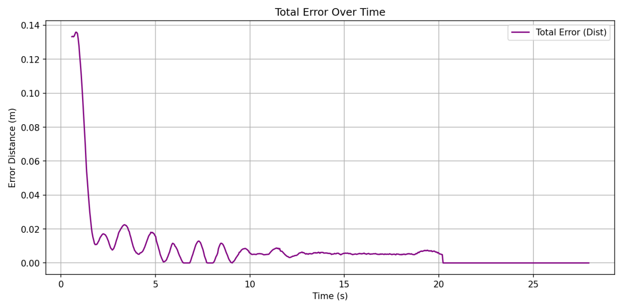
4.1 Temporal Stability and Transient Response

To evaluate the system's stability, we initialized the platform with the ball at an arbitrary position and engaged the PID controller. Figure 5a illustrates the positional error over time for both X and Y axes.

At $t \approx 1.0$s, a significant transient spike, or "kick", is observed. This initial deviation corresponds to the controller's startup phase, where the servo actuators first energize and align the platform, modeled as a momentary impulse to the ball. Following this disturbance, the system exhibits a classic underdamped response. The error oscillates with decaying amplitude, effectively converging to the setpoint ($Error \approx 0$) after approximately $t = 20$s.



(a) X and Y error over time showing damped oscillation.



(b) Total Euclidean error distance over time.

Figure 5: Temporal response of the system. Note the initial startup transient at $t = 1$s followed by convergence to steady state.

The total error distance (Figure 5b) confirms this trend. The controller successfully reduces the error from a peak of 0.14m to negligible steady-state error. The quick decay in oscillation amplitude confirms that the derivative term, $K_d = 0.528$, effectively counteracted the initial system momentum, stabilizing the platform immediately after the startup sequence.

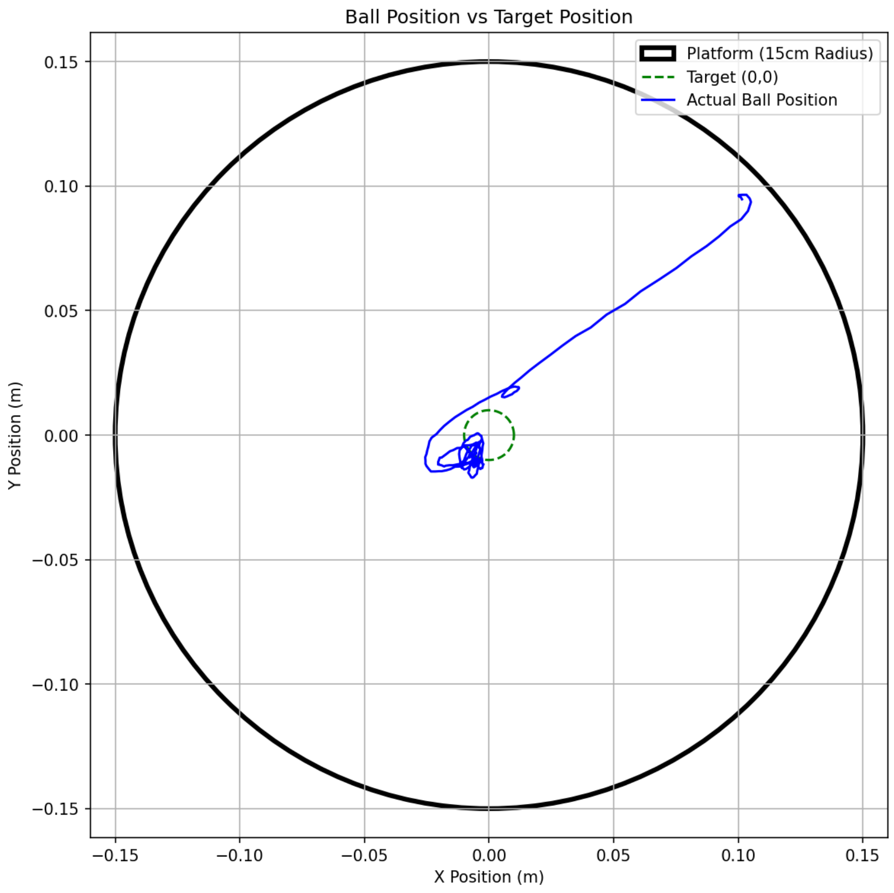


Figure 6: Ball position over time

4.2 Frequency Domain Error Analysis

To further analyze the quality of the control loop and ensuring no mechanical resonance was induced, we computed the Fast Fourier Transform (FFT) of the error signals. The frequency spectrum of the error is presented in Figure 7.

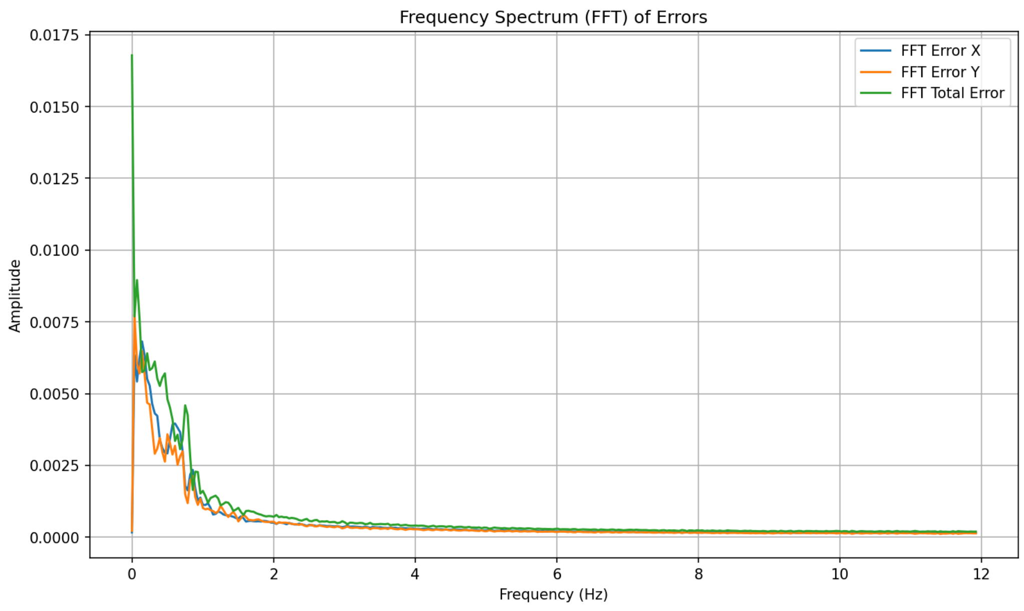


Figure 7: Frequency Spectrum (FFT) of the positional errors. The dominance of low-frequency components indicates that error is driven by bulk movement rather than high-frequency jitter.

The spectral analysis reveals two critical characteristics of the system stability:

- **Dominance of Low Frequencies (< 1 Hz):** The highest amplitude peaks occur at the lower end of the frequency spectrum. This energy corresponds to the "macro" movements of the system as it actively tilts the platform to bring the ball from the edge to the center. This confirms that the majority of the recorded "error" is simply the travel time required to correct large positional offsets.
- **Suppression of High Frequencies (> 2 Hz):** As frequency increases, the error amplitude drops significantly. The absence of high-amplitude spikes in the higher frequency range indicates that the system is free from significant mechanical jitter or sensor noise amplification. The controller is not over-reacting to noise, and the physical structure is not resonating.

Combined, the time-domain decay and the low-pass nature of the error spectrum demonstrate that the system is stable, converging to the target state with a smooth control effort rather than erratic high-frequency corrections. This behavior highlights a clear dynamic shift: initially, the system executes large, low-frequency corrections to address high-amplitude errors. As the system stabilizes, the error amplitude decreases while the frequency of correction increases, indicating that the controller is successfully making small, rapid adjustments to maintain equilibrium around the center.

5 Challenge 1: Eliminating Overshoot

5.1 Solution Strategy

In controlling a ball's motion on a Stewart platform, the typical behavior involves overshooting as it moves to the destination point, before eventually settling. This oscillatory behavior is indicative of an under damped response. This results in reduced positioning accuracy, and limits the platform's ability to maintain stable and predictable control. This challenge investigates the cause of overshoot in the system dynamics, focusing on damping in controller design. Strategies to mitigate overshoot are presented, followed by experimental implementation and simulation-based analysis to evaluate system behavior.

Overshoot is dependent on the damping ratio. Typically, a lower damping ratio allows a faster response time and higher overshoot. The objective of this challenge is to eliminate overshoot and analyze the parameters required to do so. This means the system must be critically damped or overdamped, where damping ratio ζ is equal to or greater than 1. A higher damping ratio will allow elimination of overshoot with the trade-off of a slower rise time. Along with this slower rise time, an increase in steady-state error is also introduced as a consequence of higher damping. Additionally, critical damping is difficult to achieve without the ball rolling past the steady state point, resulting in overshoot, because the ($\zeta = 1$) condition demands a level of precision that is difficult to obtain in a real, imperfect system. The ball may slowly drift even when it initially appears stable at the center of the platform. This is due to unmodeled circumstances of the system, such as kinetic/static friction, quantization of the motor commands, a surface that is not perfectly flat containing divots and bumps, etc. Therefore, the system must be tuned to account for such unmodeled factors, ensuring no overshoot.

The control strategy for balancing the ball on the platform involved iterative tuning of the PID controller parameters. The proportional gain K_p was initially increased until the platform exhibited sufficient responsiveness for the ball to move toward and across the center point. Once this behavior was achieved, the derivative gain K_d was adjusted to introduce additional damping and reduce overshoot, with incremental modifications informed by the observed system dynamics.

The integral gain K_i was then introduced to reduce steady-state error, defined as the deviation between the desired and actual ball positions. After achieving stable balance with minimal oscillatory behavior, both K_p and K_i were slightly reduced to lower the system's responsiveness and prevent the ball from approaching the center too closely, where overshoot arising from system imperfections was more likely to occur. K_d was marginally increased to further enhance damping.

The baseline configuration used to characterize the ball balancing behavior was the under damped system. Percentage overshoot was determined from the experimental measurements taken from this system. Overshoot is defined in Equation 15.

$$M_{pt} = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} \tag{15}$$

The damping ratio is calculated as a result.

$$\zeta = \frac{-\ln(M_{pt})}{\sqrt{\pi^2 + (\ln M_{pt})^2}} \tag{16}$$

For an overdamped system, there is no overshoot. As a result, the damping ratio can no longer be calculated with Equation 16, since the over shoot percentage is zero. Therefore, a simulation was developed to derive the damping ratio. To simulate the system, a standard second order overdamped response is fit to the experimental data. From this response, both the damping ratio (ζ) and natural frequency (ω_n) can be determined. Starting with the transfer function for a standard second-order system:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{17}$$

Overdamped system have a damping ratio $\zeta > 1$ will result in two real poles.

$$s_1, s_2 = -\omega_n(\zeta \mp \sqrt{\zeta^2 - 1})$$

After partial fraction expansion and performing inverse Laplace transform, the final response is:

$$y(t) = A\left(1 - c_1 e^{-s_1 t} - c_2 e^{-s_2 t}\right) \tag{18}$$

with the following coefficients

$$c_1 = \frac{s_1 - \omega_n^2/s_2}{s_1 - s_2}$$
$$c_2 = \frac{-\omega_n^2}{s_2(s_1 - s_2)}$$

and

$$A = gain$$

The system's response was fitted to the experimental data using a non-linear least squares approach to determine the optimal parameters of a second-order model, specifically the damping ratio ζ, which characterizes the system's rate of decay and overshoot, and the natural frequency ω_n, which represents the system's inherent oscillation rate.

5.2 Analysis

In tuning the controller, all gains are originally set to zero. Then, proportional gain is tuned until ball movement and oscillations are observed. Following the previously outlined strategy, the rest of the parameters are tuned accordingly after observing each adjustment. Observations during the iterative tuning approach are shown in Table 1.

Table 1: Iterative PID Tuning Observations

K_p	K_d	K_i	**Observation**
0.3	0	0	Spins around edge, never goes to center
0.15	0	0	Same as above. Too much momentum, no stopping
0.1	0.1	0.1	No circling, moves toward middle, high overshoot
0.1	0.2	0	Too overdamped, never gets to the middle
0.3	0	0	Slightly closer to middle, still overdamped
-	-	-	Iterations continue . . .
0.25	0.45	0.08	Gets to center; overshoot present and long settle time
0.25	0.45	0.09	Gets to center with less oscillations but still overshoot
0.4	0.5	0.08	Same as above but with the least overshoot observed
0.35	0.6	0.05	Overdamped; long time to center but no overshoot

Initial tuning resulted in a baseline (second last row of the above table) response with ζ of 0.4, calculated by the 25% observed overshoot. In Figure 8, the baseline response is observed passing the center (0,0) and oscillating back and forth until reaching steady state in the center of the platform.

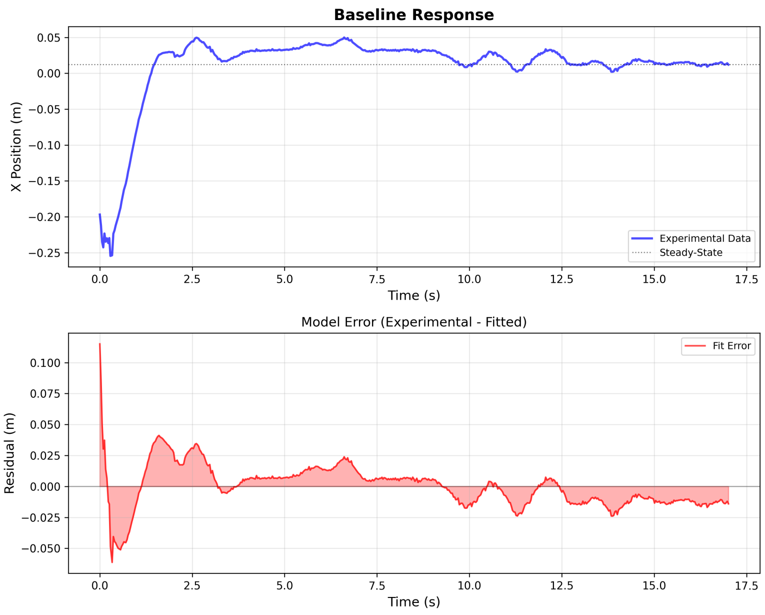


Figure 8: Baseline response

For the overdamped system (PID parameters from the final row in Table 1), the experimental response was fitted to a second-order model to simulate the system. Parameters were optimized to reduce the sum of residuals, allowing ω_n and ζ to be derived. As expected, a damping ratio of $\zeta > 1$ was observed from the system with zero overshoot, confirming the overdamped system. This can also be observed in Figure 9, with the ball remaining at a steady state position of approximately 3 cm away from the center. As exhibited, the ball never exceeds the center.

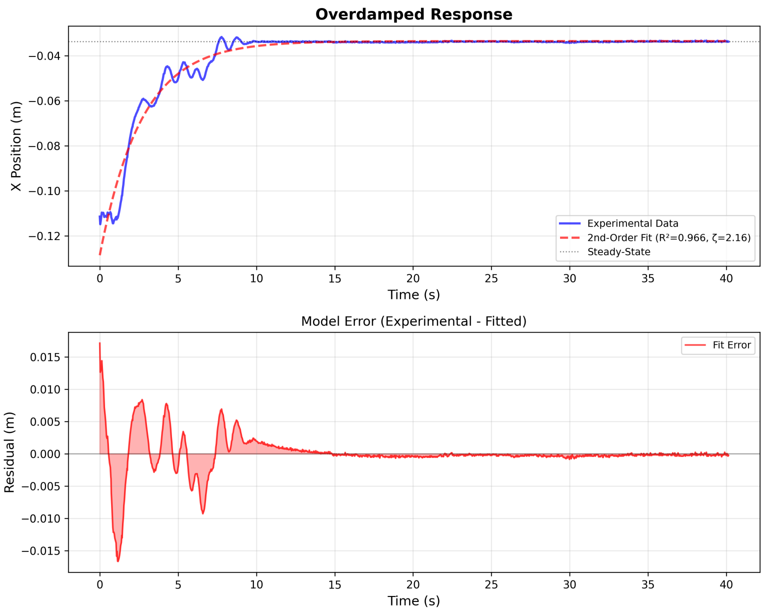


Figure 9: Overdamped response

The final PID values, obtained from the final two rows of Table 1 attained are shown in Table 2.

Table 2: Performance Comparison: Baseline vs Tuned

Metric	**Baseline**	**Overdamped**
K_p	0.40	0.35
K_i	0.08	0.05
K_d	0.50	0.60
ζ	0.40	2.16
Overshoot (%)	25.34	0.00
Rise Time (s)	0.67	5.87
Settling Time (s)	17.01	30.03

5.3 Discussion

The tuned PID controller effectively eliminated overshoot by increasing the system's damping ratio, resulting in an overdamped system. This outcome is validated experimentally, as the ball consistently stops before reaching the center, and is supported theoretically by the second order response simulation. The reduced proportional gain K_p and increased derivative gain K_d limit the system's momentum, preventing the excessive energy that leads to overshoot. K_i reduces the steady state error to guide the ball closer to the center, but remains sufficiently small to avoid inducing overshoot.

In eliminating overshoot, trade-offs were made in the performance of the Stewart platform. An overdamped system requires a longer settling time, as a result of the increased damping ratio to avoid overshoot. Experimentally, this avoids fast responses that may create momentum leading to overshoot. The compromise between speed and stability is a necessity. Under conditions requiring fast response, achieving zero overshoot may not be feasible.

Another limitation of the real system is the presence of unmodeled forces and conditions on both the platform and the ball. This could be variance in friction coefficient, bending in the platform, slight mass imbalance in the ball, and other small mechanical imperfections. To prevent accidental overshoot near the center, the K_i parameter was tuned to a value that minimizes steady-state error while still leaving a small offset, ensuring the ball does not settle exactly at the center. This strategy introduces a safety margin so that any unmodeled disturbances cannot push the ball and cause an overshoot. This highlights that, outside ideal simulations, achieving optimal PID performance (such as zero overshoot) is extremely challenging, as it requires highly precise tuning and careful analysis of numerous real-world factors, including noise, friction, surface imperfections, and other unmodeled disturbances.

After the ball reaches within the 2% of the platform's center, small fluctuations are still observed due to slight actuator movements, which create small angle changes in the platform and induce minor oscillations in the ball's position. These oscillations are disregarded and not considered overshoot because the ball remains within the final band of convergence and does not cross beyond the desired steady-state position. They are likely caused by sensor noise, external disturbances, or the aforementioned unmodeled imperfections of the system. Additionally, there is error in the implemented measurement system with the camera, as small delays, resolution limits, and image-processing noise can introduce minor inaccuracies in detecting the ball's precise position. When comparing to the simulated overdamped response, oscillations are not present because the system is considered ideal and without external conditions. Given the physical nature of the system, these small oscillations are expected and remain within acceptable limits.

The fitted curve is an effective simulation of the model, closely matching the experimental data, as indicated in Figure 9 by an R^2 value of 0.966, and enabled parameter identification to connect both experimental and theoretical results. There are limitation to this approach, since the second-order model does not capture all physical factors. This results in deviations between the ideal model and the actual response as captured by the residuals shown in Figure 9. While a more complex model could more accurately represent the experimental behavior, the current approach is sufficient for modeling and identifying the parameters necessary to achieve under damped and overdamped system responses.

6 Challenge 2: The Role of the Plant in PID Control

6.1 Solution Strategy

To investigate the fundamental role of the plant, it is necessary to first establish a definition. In control theory, the plant represents the physical system under control, acting as the interface between mathematical control signals and the physical environment [6]. Functionally, the plant operates as a transformation block that converts the controller's output (actuation signals) into a change in the system's state (physical motion), governed by dynamic laws such as gravity, friction, and inertia. In this case, the Stewart Platform acts as the plant.

However, in a digital control loop, the plant perceived by the controller is not just the physical hardware. It is also includes the digitization mechanism. The Stewart Platform operates in continuous time, obeying standard physics, but the Python controller executes commands in discrete steps, and so an interface is required. This interface is managed by two key components:

1. **The Sampler:** Converts continuous ball position data into discrete signals $y[k]$ based on a sampling time T_s. In the case of the Stewart Platform system, this is the camera ,which also limits the baseline frequency of the Python controller to 60Hz (the baseline frequency of the camera), due to a blocking call in the controller code that waits on the camera to provide the next sample of data.
2. **Zero-Order Hold (ZOH):** The servo motors maintain (hold) the discrete control signal $u[k]$ constant for the duration of the sample step, converting it into a continuous physical force [7].

Because the controller interacts with the system solely through these discrete intervals, the mathematical definition of the discrete plant $D[z]$ depends heavily on the sampling time used [8].

To validate that the definition of the plant is time-dependent and determine the role of the plant in the system, an experiment was designed to modify the discrete characteristics of the plant without changing the physical hardware by varying the sampling time T_s to effectively create different discrete plants.

By observing how alterations to the plant directly impact the closed-loop system's stability and transient response, it was demonstrated that the plant's primary function is to enforce the physical and temporal constraints that determine the system's behavior. Specifically, it can be shown that by changing the plant's characteristics, the system itself changes, necessitating a revised control strategy.

First, the following experimental configurations were designed:

- **Case A (Baseline):** Frequency $\approx 60Hz$ ($T_s \approx 0.0167s$).
- **Case B (Slow):** Frequency $10Hz$ ($T_s = 0.10s$).
- **Case C (Very Slow):** Frequency $5Hz$ ($T_s = 0.20s$).

The investigation was conducted in two phases:

1. **Phase 1 (Fixed Gains):** The PID controller was tuned for optimal performance on the baseline system (Case A). Next, these exact same gains were applied to Case B and Case C to observe how the change in sampling time affects stability.
2. **Phase 2 (Retuning):** After observing the degradation in Phase 1, the controller was retuned specifically for Case B and Case C. This was done to demonstrate that the plant has fundamentally changed, and thus requires a modified control strategy to improve the performance of the system.

6.2 Analysis

6.2.1 Phase 1: Performance with Fixed Baseline Gains

Applying the baseline gains to the slower plants resulted in immediate instability. As shown in Figure 10, while the baseline frequency maintains a low error distance, the 0.1s (Case B) and 0.2s (Case C) sampling times exhibit erratic behavior.

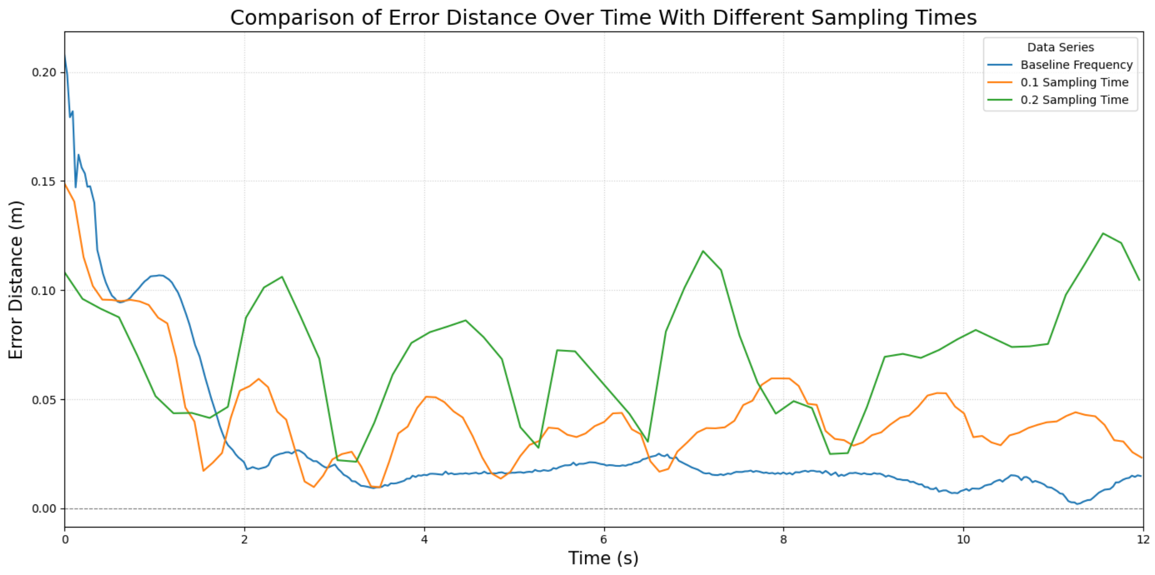


Figure 10: Comparison of Error distance over time. The 0.2s sampling time (green) shows diverging oscillation, while the baseline (blue) converges quickly.

The 0.2s sampling case (Case C) resulted in severe oscillations, with error distances peaking above 0.12m and failing to settle. This confirms that a set of control parameters (K_p, K_i, K_d) valid for one sampling time is invalid for another, validating that the plants are mathematically distinct.

6.2.2 Phase 2: Retuning for Stability

In Phase 2, Case B and Case C were treated as new plants requiring unique control strategies. By retuning the values of (K_p, K_i, K_d) to accomodate the longer sampling times, stability was recovered.

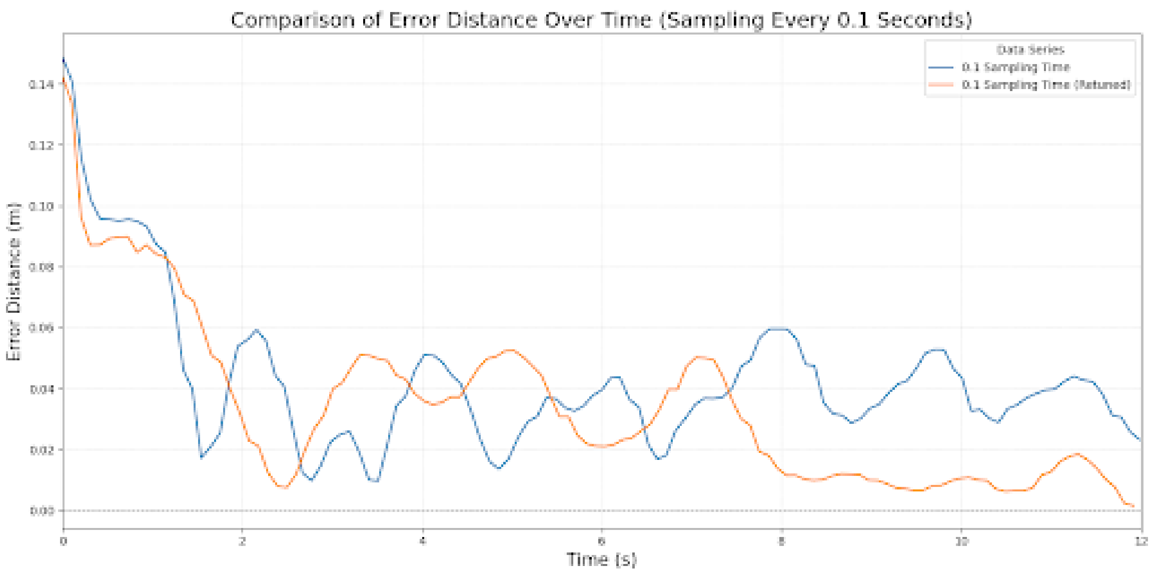


Figure 11: 0.1s Sampling: Original vs Retuned.

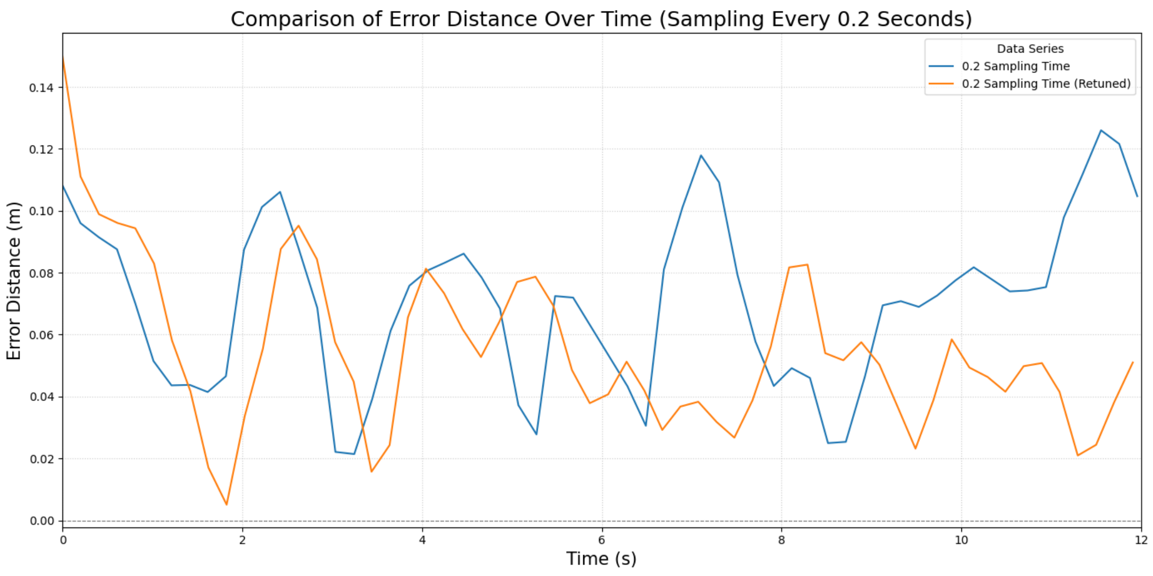


Figure 12: 0.2s Sampling: Original vs Retuned.

Observations: Stability is recovered for the 0.1s and 0.2s cases, at the cost of slower response times compared to the baseline system. By retuning the control parameters, the 0.1s and 0.2s systems were successfully stabilized. This confirms that a stable control solution exists, but it is distinct from the solution for the baseline plant.

6.3 Discussion

The instability observed at higher sampling times is not merely a result of slow reactions, but a fundamental shift in the system's poles during the transformation from continuous to discrete time.

6.3.1 Stability Mapping

In the continuous s-plane, a system is stable if its poles lie in the open Left Half Plane (Re$\{s\} < 0$). However, in the discrete z-plane, stability is defined by poles lying inside the open unit disk ($|z| < 1$)

[1]. This is visualized in Figure 13 below:

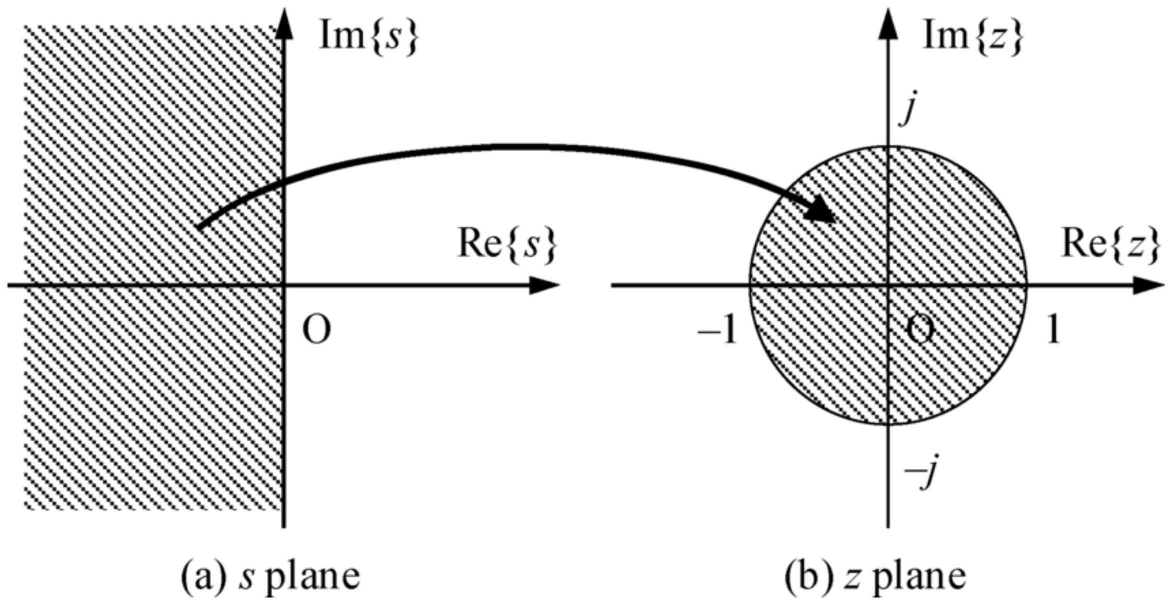


Figure 13: Stability of poles in the Continuous Time domain and the Discrete Time Domain [1]

The mapping between these domains must preserve this stability criterion [8]. The relationship between the discrete plant $G(z)$ and the continuous plant $G(s)$ under a Zero-Order Hold is given by:

$$G[z] = \left(1 - z^{-1}\right) \mathcal{Z}\left\{\frac{G(s)}{s}\right\} \tag{19}$$

6.3.2 The Role of Sampling Time (T_s)

Crucially, the exact location of the discrete poles depends on the sampling interval T_s. The relationship can be approximated via the Bilinear Transform [9]:

$$s \leftarrow \frac{2}{T_s}\frac{z-1}{z+1} \tag{20}$$

Or more directly, the poles map via the exponential relationship $z = e^{sT_s}$ [8]. This relationship explicitly shows that T_s acts as a scaling factor in the mapping.

- As T_s increases (sampling slows down), the poles of the discrete system move.
- If T_s becomes too large, poles that were stable in the s-plane may map to locations outside the unit circle in the z-plane, rendering the discrete plant unstable.

This mathematical reality explains the experimental observations. By increasing T_s from the baseline sampling time to 0.2s, the discrete poles of the Stewart Platform were moved toward the unit circle boundary. The original high-gain controller, which assumed a "fast" plant with safe pole locations, pushed the closed-loop poles outside the stable region. Retuning the controller moved the closed-loop poles back inside the unit disk to match the slower sampling rate.

In conclusion, the role of the plant is to enforce physical constraints on the controller outputs. In the context of digital control, it is a dynamic entity defined by the combination of physical mechanics and the chosen time domain resolution that directly affects how the controller is designed to ensure system stability.

7 Team Contribution

- **Ben Boguslavsky:** Eliminating Overshoot, Hardware & Testing/Debugging
- **Eidan Erlich:** PID Controller Design and Tuning
- **Fayiz Ahmed Mohideen:** The Role of the Plant in PID Control
- **Kimberley Hoang:** Eliminating Overshoot and System Simulation
- **Shabd Gupta:** Inverse Kinematics, Calibration and Tuning Procedure

References

[1] Springer, “Stability of discrete-time systems.” https://link.springer.com/chapter/10.1007/978-3-030-76947-5_6. Accessed: 2025-11-28.

[2] R. Eisele, “Inverse kinematics of a stewart platform.” https://raw.org/paper/inverse-kinematics-of-a-stewart-platform/, Feb 2019.

[3] H. Cheng and K. C. Gupta, “An historical note on finite rotations,” *Journal of Applied Mechanics*, vol. 56, no. 1, pp. 139–145, 1989.

[4] J. G. Ziegler and N. B. Nichols, “Optimum settings for automatic controllers,” *Transactions of the American society of mechanical engineers*, vol. 64, no. 8, pp. 759–765, 1942.

[5] J.-P. Merlet, *Parallel Robots*. Springer Science & Business Media, 2nd ed., 2006.

[6] Wikipedia, “Plant (control theory).” Wikipedia. Accessed: 2025-11-29.

[7] The MathWorks, Inc., “Continuous–discrete conversion methods (control system toolbox).” https://www.mathworks.com/help/control/ug/continuous-discrete-conversion-methods.html. Accessed: 2025-11-28.

[8] K. Iqbal, “7.02: Pulse transfer function.” LibreTexts Engineering. Accessed: 2025-11-28.

[9] Wikipedia, “Bilinear transform.” Wikipedia. Accessed: 2025-11-28.