# LARGE LANGUAGE MODEL ASSISTED INTERACTIVE DESIGN OF PID CONTROLLERS

Lappeenranta-Lahti University of Technology LUT

Bachelor's Programme in Technology and Engineering Science, Electrical Engineering

2026

Farah Noor

Examiners: Associate Professor, Niko Nevaranta

# Abstract

Lappeenranta–Lahti University of Technology LUT

LUT School of Energy Systems

Technology and Engineering Science, Electrical Engineering

Farah Noor

**LARGE LANGUAGE MODEL ASSISTED INTERACTIVE DESIGN OF PID CONTROLLERS**

Bachelor's thesis

2026

45 pages, 10 figures and 10 tables

Examiner: Associate professor Niko Nevaranta, D.Sc. (Electrical Engineering)

Keywords: PID controller; Controller tuning; Large language models (LLMs)

Proportional-Integral-Derivative (PID) controllers remain widely used in industrial applications due to their simplicity and effectiveness. However, designing and tuning PID parameters to meet specific performance objectives often requires expert knowledge and iterative adjustment. Recent advances in large language models (LLMs) present new opportunities for interactive and knowledge-driven support in engineering design tasks.

This bachelor's thesis investigates the feasibility of using LLMs as interactive assistants for PID design and tuning. Two frameworks were considered: *SmartControl*, with a structured graphical interface and *MATLAB Co-Pilot*, guided through carefully designed prompts. Simulation-based benchmarking was conducted on a second-order, moderately underdamped system, comparing LLM-assisted tuning against analytical and classical PID methods. Key performance metrics, including rise time, settling time, overshoot, PID gain values and reproducibility across repeated tuning runs, were evaluated.

The results indicate that LLM-assisted tuning can achieve comparable and in some cases, faster nominal transient responses with reduced overshoot relative to conventional analytical tuning rules for the considered benchmark system. However, these performance improvements are often obtained through comparatively high controller gains and exhibit variability

between repeated tuning runs. This raises important considerations regarding robustness, control effort and practical implementability. Overall, these findings demonstrate that LLMs can serve as valuable tools in the controller design process, while reinforcing the continued necessity of human oversight and constraint-aware validation to ensure that the resulting controllers are both feasible and reliable in real-world applications.

# Declarations

## Turnitin

The originality of this thesis has been reviewed with the Turnitin similarity checking service.

## AI usage

The author of the thesis, Farah Noor, used the following AI tools during the preparation of the thesis:

1. Google NotebookLM

    a. Purpose of use: To assist in finding, organizing and analyzing relevant information from the literature sources.

    b. Explanation of the use of the tool: NotebookLM was used to summarize key articles, extract important findings and structure the literature review efficiently. All generated content was critically reviewed and integrated by the author.

2. QuillBot

    a. Purpose of use: To improve phrasing, paraphrase sentences and enhance the clarity of the written text.

    b. Explanation of the use of the tool: QuillBot was employed to refine language and improve readability. All paraphrased text was carefully reviewed and verified by the author to ensure accuracy and fidelity to the source material.

# Symbols and Abbreviations

### Symbols

| | |
|---|---|
| $\mathbf{K_p}$ | Proportional gain |
| $\mathbf{K_i}$ | Integral gain |
| $\mathbf{K_d}$ | Derivative gain |
| $\zeta$ | Damping ratio |
| $\omega_n$ | Natural frequency |
| $\mathbf{t_r}$ | Rise time |
| $\mathbf{M_p}$ | Maximum overshoot |
| $\mathbf{t_s}$ | Settling time |
| $\mathbf{e_{ss}}$ | Steady-state error |
| $\omega_L$ | Loop bandwidth |
| $\beta_b$ | Integral boost parameter |

### Abbreviations

| | |
|---|---|
| **PID** | Proportional-Integral-Derivative |
| **HVAC** | Heating, Ventilation and Air Conditioning |
| **IoT** | Internet of Things |
| **LLM** | Large Language Model |
| **PLC** | Programmable Logic Controller |
| **AI** | Artificial Intelligence |
| **MSE** | Mean Squared Error |
| **ISE** | Integrated Squared Error |
| **IAE** | Integrated Absolute Error |
| **ITAE** | Integrated Time-Weighted Absolute Error |
| **HIL** | Human-in-the-Loop |
| **HITL** | Hardware-in-the-Loop |

| | |
|---|---|
| **PSO** | Particle Swarm Optimization |
| **GA** | Genetic Algorithm |
| **DE** | Differential Evolution |
| **RL** | Reinforcement Learning |
| **FSF** | Full State Feedback |
| **DC** | Direct Current |
| **LAC** | LLM-based Agents as the Controller |
| **MJSCHM** | Markov Jump Systems with Controlled and Hidden Modes |
| **ZN** | Ziegler-Nichols |
| **CN** | Cohen-Coon |
| **ITSE** | Integral of time multiplied by the squared error |
| **GUI** | Graphical User Interface |

# Table of contents

Declarations

Symbols and Abbreviations

1 Introduction 10

1.1 Research objectives . . . . . 12

2 Advances in interactive PID design 14

2.1 PID control background (brief) . . . . . 15

2.2 Human-in-the-loop and interactive control approaches . . . . . 18

2.3 LLMs in control and engineering design . . . . . 19

2.4 Summary of literature gaps . . . . . 25

3 Simulation-based case study of LLM-PID tuning 27

3.1 Selected plant model and benchmark setup . . . . . 27

3.2 Selected LLMs, targets and tuning workflow . . . . . 28

3.2.1 Interactive workflow . . . . . 29

3.2.2 Prompt in natural language . . . . . 30

4 Results and discussion 32

4.1 Closed loop performance with step-responses . . . . . 32

4.2 Reproducibility of LLM-Assisted PID Tuning . . . . . 38

5 Conclusions 41

References 43

# List of Figures


1 General structure of PID controller *(Almabrok, Psarakis & Dounis, 2018)*

2 Illustrative example of one possible approach for an interactive human-LLM-PID control loop.

3 SmartControl user-level workflow employed in this study.

4 MATLAB Co-pilot workflow employed in this study.

5 Prompt used for SmartControl.

6 Two prompts used for MATLAB Co-Pilot (left: PSO-based solution, right:DE-based solution).

7 SmartControl-based results from PSO and DE optimization methods compared against the benchmark case.

8 MATLAB Co-Pilot results with PSO and DE optimization method compared against benchmark (BM) case.

9 Closed-loop responses of the LLM-based PID controller compared to ones obtained using analytical tuning methods.

10 Graphical representation of transient performance metrics for LLM-assisted PID controllers compared to the benchmark and design constraints.


# List of Tables


1 Effects of PID parameters on system response

2 Performance metrics and equations for a second-order system. $e(t)$ represents the error signal shown in Fig. 1, calculated as the difference between the reference input $r(t)$ and the system output $y(t)$.

3 Directly relevant works on LLM-assisted PID tuning and control integration.

4 Contextual and supporting works for LLM-assisted PID control research.

5 Summary of identified research gaps in LLM-assisted control frameworks

6 Controller parameters and metrics for initial and benchmark cases

7 Comparison of PID parameters and time domain metrics.

8 SmartControl reproducability

9 MATLAB Co-pilot reproducability

10 Summary of key insights from LLM-assisted PID tuning compared to the analytical benchmark

# 1 Introduction

If one were to search for the common principle that connects the regulation of temperature in home HVAC systems, the precision of smart thermostats and IoT devices, the motion control of autonomous vehicles and the stability of industrial power systems and manufacturing processes, the answer would almost certainly be the PID controller *(Kamenko, Ilic & Congradac, 2025)*. Celebrated as the "workhorse" of the process industry *(Bassi, Mishra & Omizegba, 2011)*, PID controllers are employed for regulatory tasks in nearly 97% of process industries *(Pandey et al., 2020)*. Their appeal lies in their balance of simplicity, robustness and effectiveness *(Bassi, Mishra & Omizegba, 2011)*, while their integration in modern applications highlights their adaptability, enhancing stability, noise resilience and overall system performance *(Kamenko, Ilic & Congradac, 2025)*.

Like many widely adopted technologies, PID controllers are not without their limitations. As illustrated in Figure 1, the proportional term $K_p$ responds to current error, the integral term $K_i$ accumulates past errors and the derivative term $K_d$ predicts future trends *(Karn, 2025)*. Striking the right balance among these terms requires careful parameter tuning, which is often complex and system-specific.

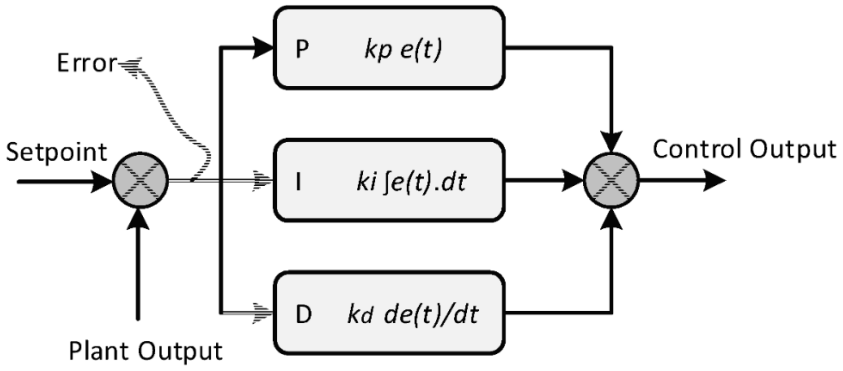


Figure 1: General structure of PID controller *(Almabrok, Psarakis & Dounis, 2018)*

Poorly chosen gains can result in oscillations, instability or slow responses *(Pandey et al., 2020)*. Although heuristic methods such as Ziegler-Nichols provide useful starting points,

they are not universally reliable, particularly for nonlinear or time-varying systems *(Moran & Önsten, 2025)*, *(Lakhani, Chowdhury & Lu, 2021)*. As a result, tuning continues to rely heavily on human expertise and iterative trial-and-error adjustments, which can be time-consuming *(Moran & Önsten, 2025)*, *(Guo et al., 2024)*, *(Kamenko, Ilic & Congradac, 2025)*.

In this context, Large Language Models (LLMs) are increasingly being explored as transformative tools in control engineering, offering capabilities that extend beyond traditional tuning methods. LLMs have been shown in recent studies to assist in parameter adjustment and under certain conditions, achieve reduced overshoot and improved transient performance *(Kamenko, Ilic & Congradac, 2025)*. Crucially, LLMs also bring interactive reasoning to the process: frameworks like *SmartControl* allow users to specify system performance goals in natural language, after which the model suggests initial PID gains, evaluates responses and iteratively refines parameters based on feedback *(Guo et al., 2024)*, *(Tohma et al., 2025)*. Recent work has also focused on multi-agent architectures that expand the role of LLMs in control engineering. *AgenticControl (Narimani & Emami, 2025)* proposes an automated control design framework where LLM agents, combined with domain expertise, are applied to tasks ranging from PID tuning to robot manipulation and even power electronics, demonstrating how LLMs can synthesize interpretable control policies. Similarly, *ControlAgent (Lakhani, Chowdhury & Lu, 2021)* introduces a collaborative framework that integrates LLM agents with engineering knowledge to automate more complex design challenges, including PID and circuit design, by leveraging iterative refinement and reasoning. Together, these systems illustrate how multiple LLMs can collaborate to optimize PID and adaptive controllers, perform scenario-based testing, generate MATLAB or PLC code and even propose compensator designs for unknown systems. In robotics, LLM-based frameworks decompose high-level commands into sequences of sub-tasks and generate control policies, while in autonomous driving they enhance trajectory planning, control supervision and decision-making for edge cases *(Zhu et al., 2025)*, *(Narimani & Emami, 2025)*. Compared to other AI-based approaches such as reinforcement learning, evolutionary algorithms and neural

networks, which often face challenges of high sample complexity, limited transferability and lack of interpretability, LLMs stand out for their combination of natural language reasoning, explainability, and adaptability, along with the ability to generate executable code and refine strategies interactively *(Moran & Önsten, 2025)*, *(Kamenko, Ilic & Congradac, 2025)*, *(Pandey et al., 2020)*, *(Lakhani, Chowdhury & Lu, 2021)*. This positions them not merely as black-box optimizers, but as more transparent design partners that complement human expertise.

Despite this growing body of work, most existing studies continue to frame LLMs primarily as automated optimizers, validated through numerical benchmarks such as rise time, settling time, overshoot and mean squared error (MSE) *(Moran & Önsten, 2025)*, *(Guo et al., 2024)*, *(Kamenko, Ilic & Congradac, 2025)*, *(Tohma et al., 2025)*, *(Pandey et al., 2020)*, *(Lakhani, Chowdhury & Lu, 2021)*. Frameworks like *AgenticControl* and *ControlAgent* exemplify this trend, demonstrating competitive or superior performance against classical tuning methods by refining controller parameters automatically. More recent tools, such as *SmartControl* and LLM-guided adaptive compensators, signal a shift toward interactive augmentation, enabling natural language feedback, clearer reasoning paths and reduced expertise requirements.

Nevertheless, a critical gap remains: there has been no systematic comparison between LLMs and human expert-derived benchmark controllers in PID tuning, particularly in evaluating not only performance outcomes but also the design process itself.

## 1.1 Research objectives

The objective of this thesis is to evaluate the feasibility and repeatability of large language model (LLM) assisted PID controller tuning using prompt-based interactions. LLMs are considered as design assistants that propose PID parameters based on given system models and performance requirements, rather than as autonomous control designers.

The evaluation is conducted using tools such as *SmartControl* and *MATLAB Copilot* and is restricted to a standard second-order linear system. Controller performance is assessed

exclusively using time-domain metrics. The LLM-assisted tuning procedure is executed a limited number of times for each tuning scenario. This deliberate limitation of scope enables a controlled and reproducible comparison between LLM-assisted tuning and established analytical PID tuning rules.

The thesis is guided by the following research questions:

1. Can LLM-assisted interactive PID tuning achieve equal or better transient performance compared to classical analytical PID tuning rules for a benchmark linear system?
2. How reproducible and consistent are the PID parameters generated by LLM-assisted tuning across repeated tuning sessions?
3. What practical limitations and engineering considerations emerge when applying LLM-assisted PID design, particularly regarding controller parameter values, time-domain performance and feasibility of the resulting controllers?

These questions are addressed through simulation-based benchmarking, comparative performance evaluation and reproducibility analysis. Analytical tuning rules are used as the reference baseline to ensure a consistent and repeatable comparison within the defined scope of this thesis.

# 2 Advances in interactive PID design

While recent advances in LLMs have introduced new forms of dialogue and automation into controller design, the underlying principles of interactive PID tuning predate these technologies. Contemporary LLM-assisted approaches are best understood as the latest stage in the evolution of human-computer interaction in control engineering. This evolution has been driven largely by the transition from analog to digital control systems, which enabled computation, simulation and real-time feedback beyond the capabilities of early mechanical implementations *(Franklin et al., 2002)*.

The foundations of interactive digital control were established during the microprocessor revolution of the 1970s, which enabled the widespread replacement of analog and pneumatic controllers with programmable digital devices *(Franklin et al., 2002) (Bennett, 2001)*. This shift moved controller design towards computational analysis and simulation, supported by early numerical tools such as MATLAB. However, interactions at this stage remained largely text-based or batch-oriented, limiting rapid design iteration *(Franklin et al., 2002)*.

By the turn of the century, these developments had matured into standardized interactive PID tuning environments. Graphical tools such as MATLAB's sisotool and later the pidTuner app allowed users to adjust controller gains or manipulate poles and zeros while directly visualizing changes in time and frequency-domain responses *(Franklin et al., 2002)*. In parallel, industrial PID controllers were increasingly deployed with software-based interfaces or physical tuning elements, reinforcing the role of the human operator as an active participant in iterative tuning loops *(Moran & Önsten, 2025)*.

Against this backdrop, LLM-assisted PID design can be viewed as a natural extension of established interactive control practices, building upon rather than replacing existing design paradigms. Whereas earlier tools required users to express design intent through graphical manipulation or direct numerical parameter adjustment *(Franklin et al., 2002)*, LLM-based systems enable the specification of control objectives through natural language. By translat-

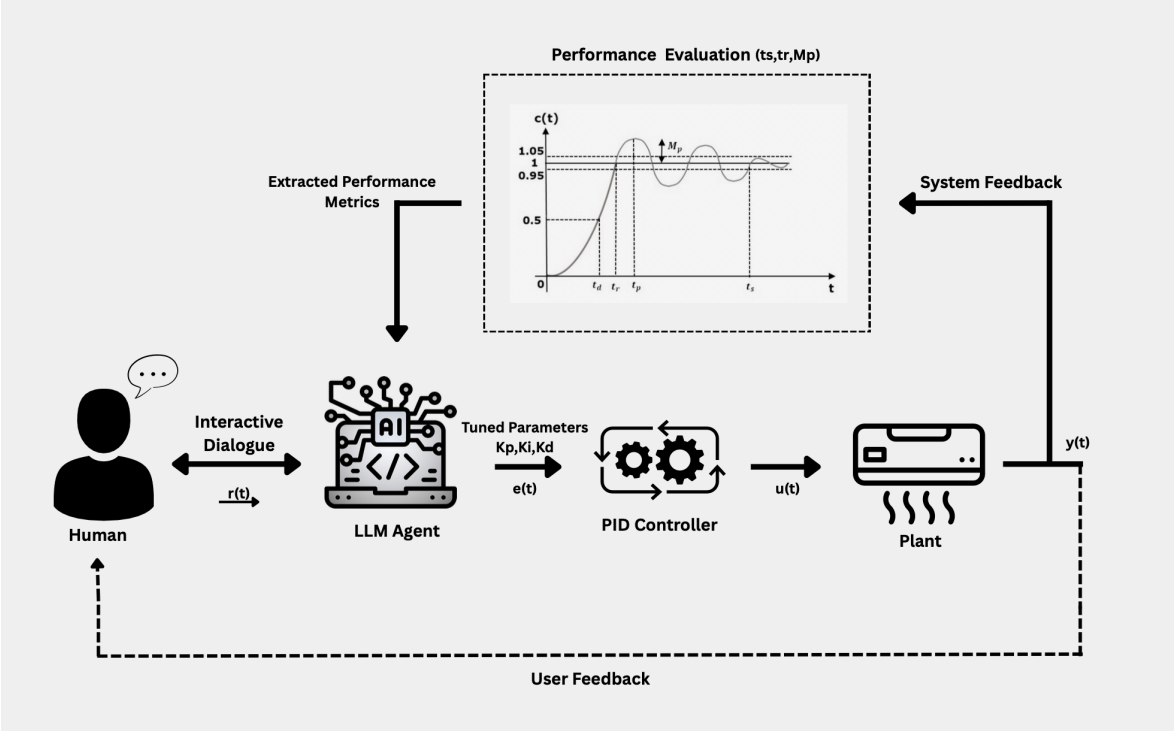


Figure 2: Illustrative example of one possible approach for an interactive human-LLM-PID control loop.

ing qualitative performance goals into quantitative tuning actions, LLMs introduce a higher-level interface layer that augments existing human-in-the-loop workflows. As illustrated in Figure 2, this interaction forms a co-operative design loop in which the human provides objectives or qualitative feedback and the LLM iteratively interprets, evaluates and refines controller parameters based on the resulting closed-loop response. This shift lowers the barrier to effective controller design while preserving the iterative and interactive principles that have characterized digital PID tuning for decades.

In order to add to the context behind these ideas, the following section reviews the basic principles and performance measures of PID tuning before exploring recent LLM-based advancements in interactive controller design.

## 2.1 PID control background (brief)

A second-order system was selected as the representative model in this study because it is standard, well-understood structure that provides interpretable foundation for analyzing and

tuning PID controller. As the simplest system that captures essential dynamic characteristics such as oscillation, damping and transient response, it gives the possibility to derive analytical tuning rules for controllers *(Haidekker, 2020; Franklin et al., 2002)*. This model establishes a direct and intuitive relationship between physical parameters like damping ratio $\zeta$, natural frequency $\omega_n$ and performance metrics such as rise time $t_r$, overshoot $M_p$ and settling time $t_s$, while maintaining analytical simplicity compared to higher-order systems *(Guo et al., 2024; Borase et al., 2021)*. The general time-domain form of a second-order system is expressed as

$$\ddot{y}(t) + 2\zeta\omega_n\dot{y}(t) + \omega_n^2 y(t) = \omega_n^2 r(t), \tag{1}$$

where $y(t)$ is the system output, $r(t)$ is reference input, $\omega_n$ is natural frequency and $\zeta$ represent the damping ratio. Applying the Laplace transform to (1) (assuming zero initial conditions) yields the transfer function

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

where $G(s)$ is system transfer function, $Y(s)$ is Laplace transforms of the output, $R(s)$ is Laplace transforms of the input and $s$ is complex frequency variable in Laplace domain. A PID controller has the following transfer function

$$G_{PID}(s) = K_p + \frac{K_i}{s} + K_d s, \tag{3}$$

where $K_p$, $K_i$ and $K_d$ are the proportional, integral and derivative gains, respectively. The closed-loop system is then obtained by combining $G_{PID}(s)$ with $G(s)$ in a standard feedback configuration. The tuning of the controller can be based on loop-shaping techniques or on parameters selected from analytical approximations and further refined through trial-and-error simulations. As an illustrative example, Table 1 summarizes the qualitative influence of the PID controller gains on key performance aspects of a feedback control system.

As it is observed, the proportional term ($K_p$) primarily affects the system's responsiveness

Table 1: Effects of PID parameters on system response

| **Parameter** | **Rise time ($t_r$)** | **Overshoot ($M_p$)** | **Settling time ($t_s$)** | **Steady-state error ($e_{ss}$)** | **Stability** |
|---|---|---|---|---|---|
| Increase $K_p$ | Decrease | Increase | Small Increase | Decrease | May Degrade |
| Increase $K_i$ | Slight Decrease | Increase | Increase | Large Decrease | May Degrade |
| Increase $K_d$ | Small Decrease | Decrease | Decrease | Minor Change | Improve |

by reducing rise time ($t_r$), but excessive gain can increase overshoot ($M_p$) and potentially degrade stability. The integral term ($K_i$) accumulates error over time, effectively eliminating steady-state error ($e_{ss}$) but at the cost of increased overshoot and slower settling ($t_s$). The derivative term ($K_d$), by predicting the future trend of the error, introduces a damping effect that reduces overshoot and improves settling behavior. Building on the qualitative insights summarized in Table 1, the performance metrics presented in Table 2 provide a quantitative framework for evaluating how PID parameters influence the transient and steady-state behavior of a second-order system.

Table 2: Performance metrics and equations for a second-order system. $e(t)$ represents the error signal shown in Fig. 1, calculated as the difference between the reference input $r(t)$ and the system output $y(t)$.

| **Metric** | **Symbol** | **Equation / Expression** |
|---|---|---|
| Rise time | $t_r$ | $t_r \approx \dfrac{\pi-\theta}{\omega_n\sqrt{1-\zeta^2}}$, where $\theta = \tan^{-1}\left(\dfrac{\sqrt{1-\zeta^2}}{\zeta}\right)$ |
| Peak time | $t_p$ | $t_p = \dfrac{\pi}{\omega_n\sqrt{1-\zeta^2}}$ |
| Maximum overshoot | $M_p$ | $M_p = e^{\left(-\dfrac{\pi\zeta}{\sqrt{1-\zeta^2}}\right)} \times 100\%$ |
| Settling time (2% criterion) | $t_s$ | $t_s \approx \dfrac{4}{\zeta\omega_n}$ |
| Steady-state error | $e_{ss}$ | $e_{ss} = \lim_{t\to\infty}[r(t)-y(t)]$ |
| Mean squared error | MSE | $\text{MSE} = \dfrac{1}{T}\int_0^T e^2(t)\,dt$ |
| Integrated squared error | ISE | $\text{ISE} = \int_0^\infty e^2(t)\,dt$ |
| Integrated absolute error | IAE | $\text{IAE} = \int_0^\infty \lvert e(t)\rvert\,dt$ |
| Integrated time-weighted absolute Error | ITAE | $\text{ITAE} = \int_0^\infty t\,\lvert e(t)\rvert\,dt$ |

In Table 2 the time-domain measures ($t_r$, $t_p$, $M_p$ and $t_s$) capture how quickly and smoothly the system responds to a change in input, reflecting the influence of proportional and derivative actions on responsiveness and damping. The steady-state error $e_{ss}$ represents long-term accuracy and is primarily corrected through integral action. In addition, the error-based indices

(MSE, ISE, IAE, ITAE) provide numerical measures for evaluating and optimizing controller performance, supporting systematic tuning approaches that balance speed, stability and control effort. These performance indices form the basis for the evaluation metrics used in the simulation study.

## 2.2 Human-in-the-loop and interactive control approaches

Human participation has always been central to control engineering, evolving from manual parameter tuning to adaptive and intelligent frameworks. The notion of human-in-the-loop (HIL) control captures this progression, referring to systems that embed human expertise within feedback and automation processes *(Mabrok et al., 2020)*, *(Franklin et al., 2002)*. Manual tuning methods such as iterative trial-and-error once relied entirely on operator intuition *(Guo et al., 2024)*, *(Kamenko, Ilic & Congradac, 2025)*, whereas modern adaptive and collaborative approaches now enable shared control between humans and intelligent agents *(Lakhani, Chowdhury & Lu, 2021)*, *(Wu & Zhang, 2020)*, *(Liu et al., 2024)*.

Despite these advancements, tuning a PID controller remains a non-trivial task, as it involves reconciling multiple and often conflicting objectives such as rise time, overshoot and steady-state accuracy as summarized by the performance metrics and equations presented in Table 1 and Table 2. Achieving an optimal balance among these parameters typically requires iterative experimentation and interpretive reasoning, underscoring the continued importance of human insight in the control loop.

LLMs represent the latest stage in this evolution of human-machine interaction, extending HIL principles into the domain of intelligent automation *(Zhu et al., 2025)*. Rather than replacing human expertise, LLMs augment it by combining adaptive reasoning with natural-language interfaces capable of translating qualitative objectives into quantitative control actions *(Narimani & Emami, 2025)*, *(Moran & Önsten, 2025)*, *(Cui et al., 2024)*. This linguistic interactivity lowers the expertise barrier for complex tuning tasks while offering adaptability through in-context learning and interpretability through transparent, language-based feed-

back *(Zhu et al., 2025)*, *(Narimani & Emami, 2025)*, *(Luketina et al., 2019)*, *(Liang et al., 2022)*.

LLMs further distinguish themselves from conventional artificial intelligence (AI) methods for PID tuning by combining high-level reasoning with intuitive, language-based interaction *(Narimani & Emami, 2025)*. Classical approaches such as Ziegler-Nichols rely on fixed empirical tuning rules, while optimization-based methods including particle swarm optimization (PSO) and genetic algorithms (GA) adjust controller parameters through predefined search mechanisms driven by explicit objective functions *(Bassi, Mishra & Omizegba, 2011)*, *(Lakhani, Chowdhury & Lu, 2021)*. Reinforcement learning (RL) based controllers similarly depend on formally defined reward structures and typically require repeated interaction with the system during training*(Moran & Önsten, 2025)*. In contrast, LLMs iteratively reason over system feedback, refining control parameters through natural-language interaction and adapting in real time without explicit retraining *(Narimani & Emami, 2025)*, *(Moran & Önsten, 2025)*, *(Tohma et al., 2025)*. This transparency and flexibility address the "black-box" limitations of other AI models *(Zhu et al., 2025)* and have shown practical superiority, frameworks such as *AgenticControl* reduced PID tracking error by 55 % compared to MATLAB's PIDTuner *(Narimani & Emami, 2025)*, *(Guo et al., 2024)*.

In essence, these developments trace the evolution of control design towards intelligent, language-centered collaboration between humans and machines. The next section reviews emerging LLM-based PID tuning frameworks that extend these ideas into practical and experimental forms of interactive control design.

## 2.3 LLMs in control and engineering design

Building on the historical and conceptual developments discussed previously, the emergence of LLMs can be seen as the latest stage in a long trajectory toward increasingly interactive control design. Early PID tuning relied heavily on heuristic operator judgment: engineers interpreted system behavior and manually adjusted gains. This approach, originating from

pre-1930s operator-based control methods *(Bennett, 2001)* and shaped by intuitive attempts to model human corrective actions *(Franklin et al., 2002)*, was limited by its dependence on expertise and the time-consuming nature of manual tuning, often resulting in suboptimal performance *(Bassi, Mishra & Omizegba, 2011)*. Over time, these intuition-driven practices evolved into structured human-in-the-loop (HIL) and adaptive control frameworks, explicitly modeling the human as part of the feedback loop. Examples include Markov jump systems with controlled and hidden modes (MJSCHM) in driver-assistance architectures, which represent uncertainty in human decision processes *(Wu & Zhang, 2020)* and more recent LLM-driven systems such as ControlAgent and AgenticControl, which replicate engineer-like iterative refinement by combining natural-language guidance, embedded knowledge and computational feedback *(Guo et al., 2024) (Kamenko, Ilic & Congradac, 2025)*. Across these developments, a recurring theme is the role of interaction between human and system, desired behavior and controller parameters, qualitative judgment and quantitative adjustment.

Motivated by this shift, recent studies have explored integrating LLMs into control workflows, either as autonomous tuning agents or as components in broader engineering pipelines (e.g., autonomous driving planning *(Zhu et al., 2025)*, PLC code generation *(Koziolek, Gruener & Ashiwal, 2023)* and robot policy synthesis *(Liang et al., 2022)*). This thesis presents a structured literature review of existing studies on PID tuning, with particular emphasis on approaches that incorporate LLMs. To provide a clear and systematic overview, the reviewed literature is organized into two complementary categories: LLMs functioning as direct participants in PID design and LLMs serving as contextual or supervisory agents within broader control system architectures.

The first category, summarized in Table 3, includes studies in which LLMs directly generates, tunes or refines PID gains, often through iterative reasoning or multi-agent collaboration. Frameworks such as SmartControl *(Tohma et al., 2025)* demonstrate how LLMs can translate natural-language performance goals into precise numerical targets and coordinate with optimization algorithms like PSO and differential evolution (DE). ControlAgent *(Guo*

*et al., 2024)* takes a different approach, introducing a fully automated PID tuning workflow in which a central LLM coordinates with task-specific agents and a Python computation module to iteratively refine controller parameters. Validated across 500 simulation tasks, ControlAgent shows effective PID and loop-shaping optimization, systematically adjusting the loop bandwidth $\omega_L$, which governs the speed of the system response and directly affects settling time, and the integral boost parameter $\beta_b$, which shapes low-frequency gain to enhance robustness and phase margin. By iteratively tuning these two parameters, the agent balances the trade-off between fast performance (small settling time) and robust stability (large phase margin), directly addressing the performance metrics outlined in Table 2. However, the framework has yet to be tested on nonlinear systems or hardware, limiting conclusions about real-world applicability.

Table 3: Directly relevant works on LLM-assisted PID tuning and control integration.

| **Paper** | **PID Relevance** | **LLM Role** | **Validation** | **Key Limitation** |
|---|---|---|---|---|
| *SmartControl Tohma et al., 2025* | Focuses on interactive PID design using LLMs and optimization (PSO/DE) | LLM interprets user goals in natural language, provides feedback, and generates MATLAB code | Simulation | Validated only in simulation; lacks nonlinear and real-world testing |
| *AgenticControl Narimani & Emami, 2025* | Automates PID and FSF design with multi-agent LLM coordination | Six LLM agents collaborate (Actor-Critic structure) for controller selection, tuning, and testing | Four case studies(DC Motor, Ball & Beam, Pendulums) | No hardware implementation |
| *ControlAgent Guo et al., 2024* | Fully automates PID tuning and controller design | Central + task-specific LLMs with Python agent for iterative parameter refinement | Simulation benchmark | Limited to PID/loop-shaping; needs nonlinear and hardware validation |
| *LLM based PID controller optimization Kamenko, Ilic & Congradac, 2025* | Direct LLM-driven PID tuning using GPT-4o & DeepSeek-V2 | Uses prompts to iteratively refine $K_p$, $K_i$, $K_d$ based on performance metrics | Simulation | No physical tests; ignores noise/nonlinear effects |
| *LLMs-guided adaptive compensator (Zhou et al., 2025)* | Adds PID-like adaptive compensator to enhance existing controllers | LLM designs compensator via zero-shot prompts aligning system and reference response | Simulation + prototype | Ineffective for structurally mismatched systems |
| *LLM Multi-Agent Control for Power Electronics (Cui et al., 2024)* | Uses LLM agents and PSO for PID design in DC-DC boost converter | Multi-agent reasoning converts high-level objectives to PID coefficients | Simulation | Early prototype; lacks industrial validation |

Similarly, the LLM-based PID controller optimization *(Kamenko, Ilic & Congradac, 2025)* leverages advanced models such as GPT-4o and DeepSeek-V2 to automate tuning through iterative, performance-based prompts. While effective in simulation, it deliberately ignores system noise and nonlinearities, which constrains its practical deployment. The LLMs-guided adaptive compensator *(Zhou et al., 2025)* extends these capabilities by designing a PID-like compensator to reconcile discrepancies between an unknown system and a reference model. Tested in both simulation and prototype environments, the framework demonstrates improved adaptability, but its effectiveness decreases for structurally mismatched systems, limiting generalizability. Likewise, LLM multi-agent control for power electronics considered in *(Cui et al., 2024)* applies a multi-agent LLM-PSO workflow to objective-driven PID design in DC-DC boost converters. While promising, this approach remains at an experimental stage without validation on real systems. Beyond individual frameworks, multi-agent architectures such as AgenticControl *(Narimani & Emami, 2025)* illustrate the benefits of distributing specialized roles among LLM agents, facilitating end-to-end automation across controller selection, tuning, and evaluation.

The second category, summarized in Table 4, focuses on the broader role of LLMs within control engineering, emphasizing cognitive capabilities, architectural paradigms and hybrid frameworks rather than PID design alone. This line of work includes reasoning and capability benchmarks showing that LLMs often struggle with tasks requiring precise numerical accuracy or the interpretation of visual control representations such as Bode or Nyquist plots *(Kevian et al., 2024)*. These plots, widely used in control theory, represent system behavior in the frequency domain and summarize key performance metrics such as gain, phase margin, and stability characteristics, as discussed in Section 2.1 (see Table 2 for details). The *LLM-based agents as the controller* framework or in short, *LAC* framework *(Xu, Zheng & Wang, 2024)*, which deploys LLMs as online controllers, illustrates challenges in slow inference and long-term reasoning, with one model taking 118.7 seconds for a single task in robot application considered. Additional studies investigate LLMs as high-level planners layered above classical PID control, such as NeuroLift, where natural-language commands

are translated into drone maneuvers executed by low-level PID loops *(Joshi, Sanyal & Roy, 2025)*. Hybrid approaches, like the PID-based self-healing mechanism *(Chen et al., 2024)*, leverage control-theoretic principles to stabilize LLM outputs, demonstrating how classical controllers can mitigate LLM inconsistencies. While this second group is more conceptually diverse and includes early real-world testing, it also presents persistent limitations in reasoning fidelity, robustness to noisy feedback and stability of long-term interactions.

Table 4: Contextual and supporting works for LLM-assisted PID control research.

| **Paper** | **PID Relevance** | **LLM Role** | **Validation** | **Key Limitation** |
|---|---|---|---|---|
| *LLM capabilities in control engineering (Kevian et al., 2024)* | Evaluates LLMs on classic control and PID tasks | Problem-solving and reasoning benchmark | Simulation benchmark | Struggles with visual plots; limited reasoning accuracy |
| *LAC (Xu, Zheng & Wang, 2024)* | PID as a traditional offline control law, contrasting LAC's novelty | LLMs act as online controllers via feedback dialogue | Simulation | LAC remains slow, LLMs struggle with long-term reasoning and feedback |
| *NeuroLift (Joshi, Sanyal & Roy, 2025)* | Integrates LLMs with low-level PID flight control (EV-PID) | Converts natural language to drone commands executed by PID | Real-world drone | Training overfitting, false positives, poor real-world performance |
| *PID based self-healing (Chen et al., 2024)* | Uses PID logic to stabilize LLM outputs during inference | Internal PID controller corrects trajectory errors | Simulation | Gains empirical; linear approximation limits |

Across both tables, a rapidly evolving but still fragmented research landscape is revealed. Table 3 captures practical, early-stage efforts to automate classical PID workflows, treating LLMs as intelligent design tools capable of producing and refining controller gains. Table 4, by contrast, represents the exploratory and architectural frontier, where LLMs are evaluated as reasoning engines, high-level planners or components of hybrid intelligent-control systems. Despite their different emphases, both lines of research reflect a transition from

passive advisory roles of LLMs toward active involvement in control system design and execution. Nonetheless, significant challenges persist, particularly with respect to robustness, reliability, real-time consistency, interpretability and deployment in real-world contexts, reinforcing the necessity for systematic, human-centered evaluation of LLM-assisted control frameworks.

## 2.4 Summary of literature gaps

A review of LLM-assisted control frameworks, benchmarking studies and adaptive architectures indicates recurring limitations that limit their practical applicability. While prior work demonstrates the potential of LLMs for parameter tuning, iterative design and human-in-the-loop collaboration, several interconnected gaps remains which are classified in table 5.

In summary, the literature on LLM-assisted PID and interactive control highlights significant progress in automating iterative design, integrating human-in-the-loop reasoning and leveraging natural-language interfaces. However, persistent gaps remain as most frameworks are restricted to linear or simplified systems, lack real-world validation, struggle with real-time reliability and multimodal inputs and rarely evaluate usability or benchmark performance against human experts. These limitations collectively constrain both the practical deployment of LLM-assisted control and the ability to rigorously assess its advantages over traditional tuning methods.

This thesis addresses the aspects of interactivity, usability and human comparison through a systematic benchmarking study. In this study, selected LLM-assisted PID tuning algorithms are evaluated against human-designed controllers with respect to accuracy, efficiency and interpretability. The results provide empirical insights into the performance of LLM-based approaches in realistic control scenarios and support user-centered evaluation in interactive control system design.

Table 5: Summary of identified research gaps in LLM-assisted control frameworks

| **Gap Category** | **Key Limitations Identified in Literature** |
|---|---|
| Domain scope and complexity | Most frameworks are limited to linear time-invariant systems and conventional controller structures, primarily focusing on PID tuning. While exceptions such as AgenticControl extend to nonlinear and unstable systems using full-state feedback and multi-agent optimization, systematic methods for nonlinear, high-order or time-varying systems remain largely unexplored. Emerging work on LLM-guided adaptive compensation suggests potential pathways, but no generalizable framework exists. |
| Simulation-to-practical validation | The majority of studies remain simulation-bound, neglecting sensor noise, actuator constraints, unmodeled dynamics and nonlinearities. Only a few approaches provide real-world validation with theoretical stability guaranteed, while hardware-in-the-loop experiments and deployment in realistic environments are largely absent. |
| Reliability and real-time feasibility | Benchmarking studies reveal frequent calculation errors and inconsistencies in LLM outputs, while real-time frameworks are constrained by high inference latency. Although fine-tuning can improve performance, it may introduce overfitting and unsafe behavior. Approaches integrating control principles directly into LLMs show improved reliability with low computational overhead, but adoption remains limited. |
| Interactivity, usability, and human comparison | Despite claims of human-in-the-loop design, few studies systematically evaluate usability, interpretability or efficiency. Direct comparisons between LLM-assisted and human-only tuning are largely missing, leaving the practical advantages of LLM assistance unquantified. |
| Industrial benchmarking and safety | Existing frameworks rely on human-defined objectives and constraints and lack evaluation against industrial standards, performance envelopes or regulatory safety requirements. As a result, most approaches remain experimental and insufficiently validated for industrial deployment. |

# 3 Simulation-based case study of LLM-PID tuning

Building on the state of the art in LLM-assisted PID tuning, this section presents a simulation-based case study to evaluate PID controller design using LLM frameworks. Two frameworks, *MATLAB Co-Pilot* and *SmartControl*, are applied to an underdamped second-order system, with their results benchmarked against the mathematically derived tuning method proposed in *Albatran, Smadi & Bataineh, 2020*. Details on the system models, simulation setup, LLM prompting strategies, resulting controller parameters, and performance metrics are provided here, establishing the foundation for the comparative analysis presented in Section 4.

## 3.1 Selected plant model and benchmark setup

For the simulation-based evaluation, the test plant is chosen directly from *Albatran, Smadi & Bataineh, 2020*, as the benchmark PID tuning analysis in that study was conducted on this system. The plant is described by a standard second-order transfer function, as introduced in (2). By substituting the natural frequency $\omega_n = 1.73$ rad/s and damping ratio $\zeta = 0.288$ into (2), the resulting transfer function is

$$G(s) = \frac{2.9929}{s^2 + 0.99648s + 2.9929}. \tag{4}$$

This transfer function corresponds to a system with a moderately underdamped response, exhibiting approximately 39% overshoot and a rise time of about 1.13 seconds, making it well-suited for evaluating PID controllers and highlighting differences between tuning methods.

Table 6 summarizes the system's open-loop performance metrics, including rise time, settling time, and maximum overshoot, alongside the PID controller parameters and the resulting closed-loop performance. The closed-loop results are presented in Table 6 for the benchmark PID tuning method adopted from *Albatran, Smadi & Bataineh, 2020*, as well as for the classical Ziegler-Nichols (ZN), Cohen-Coon (CC), and integral of time-weighted

squared error (ITSE) tuning approaches. For the initial conditions, rise time ($t_r$), settling time ($t_s$) and maximum overshoot ($M_p$) were determined using the equations outlined in Table 2. For the remaining approaches, both controller parameters and performance metrics were directly adopted from *Albatran, Smadi & Bataineh, 2020*, ensuring consistency in the baseline for comparison.

Table 6: Controller parameters and metrics for initial and benchmark cases

| **Method / Condition** | $K_p$ | $K_i$ | $K_d$ | $t_r$ (s) | $t_s$ (s) | $M_p$ (%) |
|---|---|---|---|---|---|---|
| Initial condition (open-loop) | - | - | - | 1.124 | 8 | 44.80 |
| Benchmark | 12.32 | 55.80 | 25.17 | 0.029 | 0.057 | 0.468 |
| ZN | 3.98 | 7.79 | 0.509 | 0.315 | 8.66 | 47.54 |
| CC | 1.38 | 1.85 | 0.178 | 0.607 | 8.95 | 22.38 |
| ITSE | 2.537 | 2.39 | 1.77 | 0.403 | 3.56 | 0.152 |

This table establishes the reference values against which the performance of LLM-assisted PID tuning frameworks will be evaluated in the following subsections.

## 3.2 Selected LLMs, targets and tuning workflow

For the simulations in this study, two LLM frameworks were selected: *SmartControl*, which uses the GPT-4 model and *MATLAB Co-Pilot*, which operates on the GPT-5 mini model. *SmartControl* was selected for its successful setup and reliable operation on a personal laptop, configured with an OpenAI API key. *MATLAB Co-Pilot* was chosen due to its accessibility and novelty, offering an opportunity to explore a recently introduced interactive tool in the context of PID tuning. Since *SmartControl* supports two optimization techniques, Particle Swarm Optimization (PSO) and Differential Evolution (DE), *MATLAB Co-Pilot* was instructed to conduct separate simulations using each method. This methodology facilitated a comprehensive comparison of the performance characteristics of the two algorithms.

For the LLM-assisted tuning experiments, the closed-loop performance targets were defined as a maximum overshoot ($M_p$) below 0.4%, a settling time ($t_s$) under 0.057 s, and a rise time ($t_r$) less than 0.029 s. These targets were selected based on the benchmark PID results reported in *Albatran, Smadi & Bataineh, 2020* and are incorporated in the design prompts

presented in Figure 5 and 6. Although the selected performance targets place strong emphasis on rapid and highly damped transient behavior, they were deliberately chosen to replicate the response achieved by the benchmark controller, thereby establishing a controlled and consistent basis for comparison with the LLM-assisted tuning approach. The benchmark PID controller adopted from *Albatran, Smadi & Bataineh, 2020* does not explicitly define numerical bounds or constraints on the proportional, integral or derivative gains. In order to preserve equivalence with these assumptions and to isolate the effect of the tuning methodology itself, no additional gain constraints were imposed on the LLM-assisted controllers. This decision intentionally prioritizes comparative transient performance over practical implementability considerations.

### 3.2.1 Interactive workflow

*SmartControl* was used in this study according to the workflow shown in Figure 3. Although detailed architectural and algorithmic descriptions of *SmartControl* are available in the original publication *(Tohma et al., 2025)*, this work concentrates on the user-level workflow for LLM-assisted PID tuning, emphasizing interactivity and usability in line with the research gaps identified in Section 2. Likewise, Figure 4 presents the corresponding workflow implemented for *MATLAB Co-Pilot*.

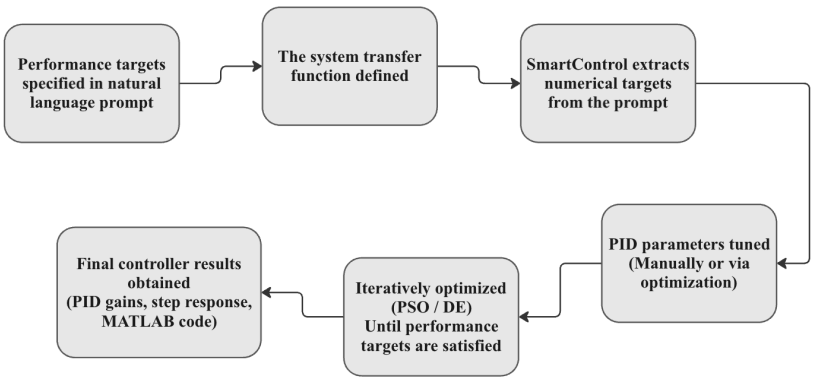


Figure 3: SmartControl user-level workflow employed in this study.

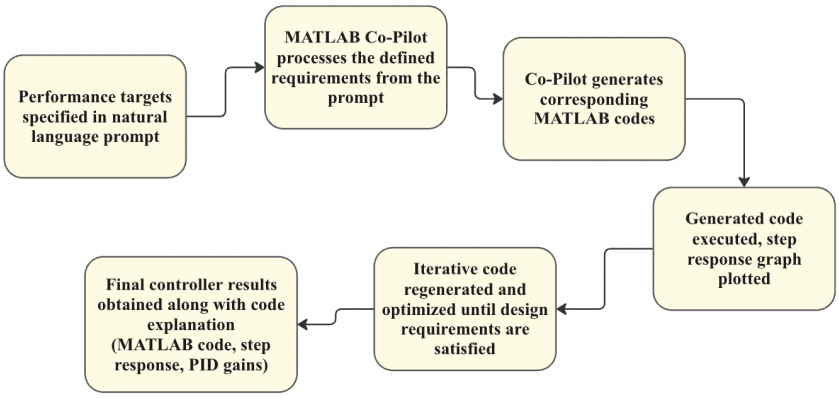


Figure 4: MATLAB Co-pilot workflow employed in this study.

### 3.2.2 Prompt in natural language

Figure 5 presents the prompt snippet used for testing *SmartControl*. In this framework, the LLM functions as an embedded interpretive layer that translates relatively simple user prompts into executable actions, while relying on predefined interface settings such as the system transfer function, performance objectives, and the selected optimization algorithm. These inputs are specified through dedicated fields in the graphical user interface and remain consistent across simulations, reducing the complexity of the prompt itself.

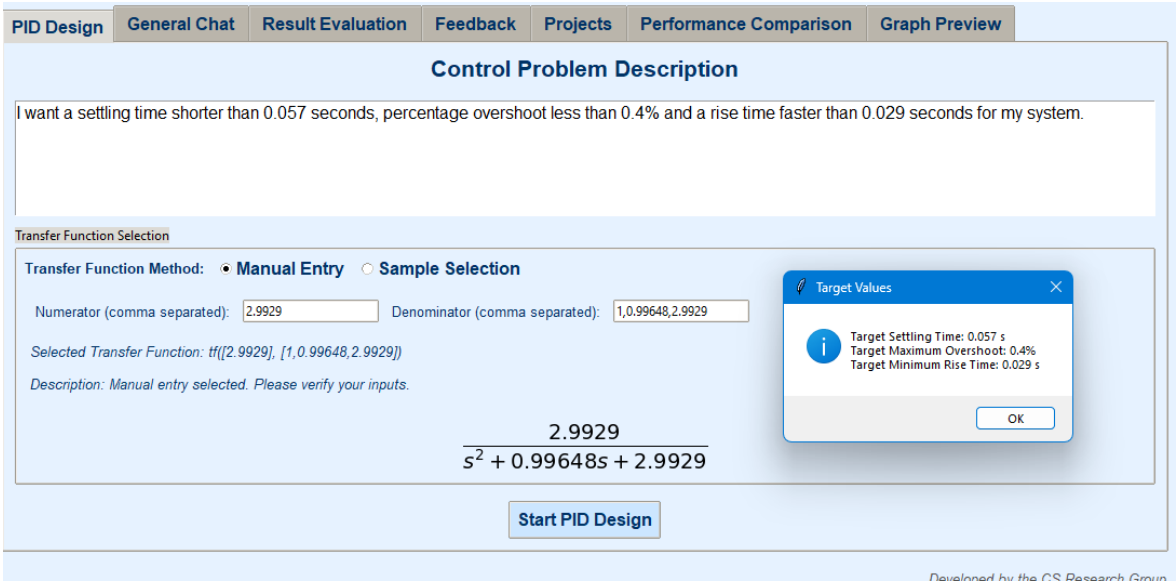


Figure 5: Prompt used for SmartControl.

In contrast, *MATLAB Co-Pilot* introduces additional complexity due to the absence of a similar structured interface. To obtain results that are reliable and comparable to those from *SmartControl*, the prompt must be carefully designed to replicate the same formulation, including the system model, performance objectives, and optimization approach. As presented in Figure 6, two distinct prompts were used for *MATLAB Co-Pilot*: one for the PSO-based solution and one for the DE-based solution.

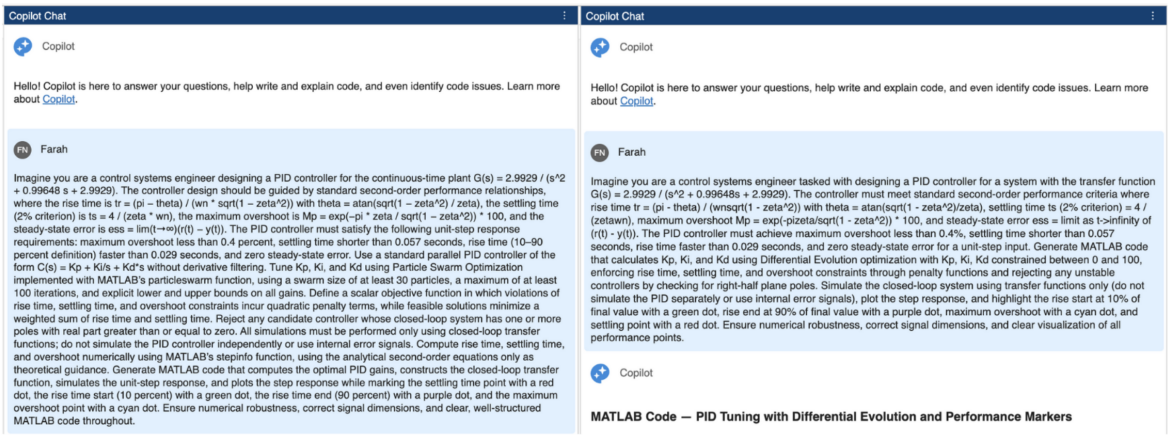


Figure 6: Two prompts used for MATLAB Co-Pilot (left: PSO-based solution, right:DE-based solution).

These differences reflect the distinct design goals of the two frameworks. *SmartControl* offers a dedicated, user-friendly graphical user interface (GUI) specifically designed for PID controller tuning, with clearly defined fields for the system transfer function, performance objectives and optimization method. By comparison, *MATLAB Co-Pilot* operates as a conversational assistant within the *MATLAB* environment,designed to support a variety of *MATLAB*-related tasks (coding, debugging, explaining functions, data analysis, etc.) through a chat-based interface. All relevant information, including the system model, performance criteria and optimization approach, must therefore be explicitly and comprehensively specified within the chat interface. Consequently, the quality and effectiveness of the generated code are highly dependent on prompt design, with more detailed, precise and well-structured prompts yielding improved performance.

# 4 Results and discussion

This section presents a detailed comparison of several PID controller designs obtained using both LLM-assisted and analytical tuning approaches. The controllers considered include: (i) the proposed PID tuning method from *Albatran, Smadi & Bataineh, 2020*, adopted here as a benchmark; (ii) PID controllers tuned using the *SmartControl* and *MATLAB Co-Pilot* LLM-based frameworks; and (iii) PID controllers designed using established classical tuning rules, namely Ziegler-Nichols (ZN), Cohen-Coon (CC) and integral of time-weighted squared error (ITSE) minimization, with the results for these classical methods also taken from *Albatran, Smadi & Bataineh, 2020*.

## 4.1 Closed loop performance with step-responses

Figure 7 compares the closed-loop step responses of the PID controllers obtained using *SmartControl* with PSO and DE optimization, alongside the benchmark PID controller from *Albatran, Smadi & Bataineh, 2020*. The controller parameters used in *SmartControl* were obtained using the prompt shown previously in Figure 5.

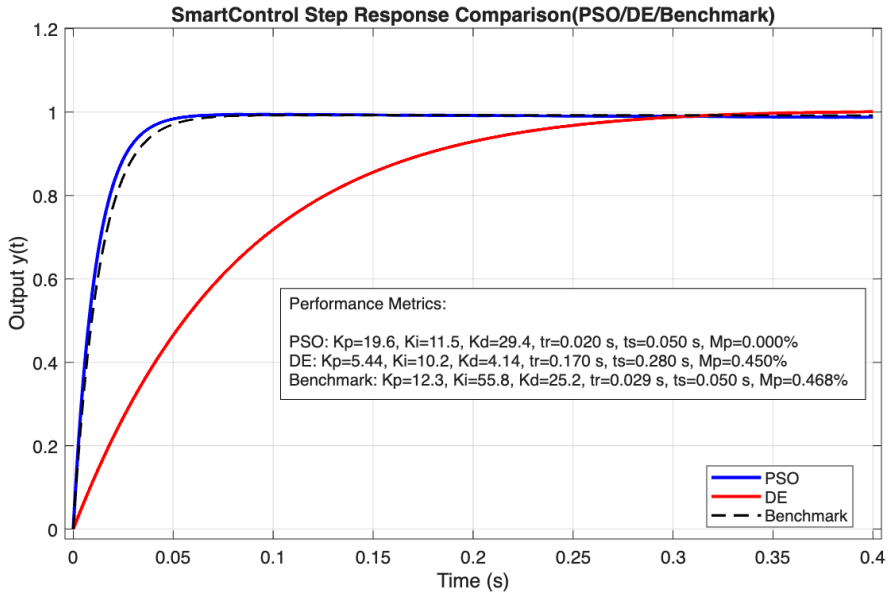


Figure 7: SmartControl-based results from PSO and DE optimization methods compared against the benchmark case.

It is observed that the *SmartControl*-PSO controller significantly improves transient performance relative to the benchmark. The rise time is reduced to $t_r = 0.020$ s, settling time remains similar and overshoot is completely eliminated ($M_p = 0\%$). These results align with classical PID principles: the higher proportional gain ($K_p = 19.591$) accelerates the rise, the increased derivative gain ($K_d = 29.450$) suppresses overshoot, and the moderate integral gain ($K_i = 11.454$) limits oscillations.

In contrast, the *SmartControl*-DE controller exhibits slower dynamics, with $t_r = 0.170$ s, $t_s = 0.280$ s and a small overshoot ($M_p = 0.45\%$). The slower response reflects the lower proportional ($K_p = 5.438$) and derivative ($K_d = 4.142$) gains which reduce acceleration and damping.

For both PSO and DE, two independent runs were conducted using identical prompts (details in Table 8). The best-performing results are presented in Figure 7. Notably, *SmartControl* operates with fully automated optimization and does not enforce explicit bounds on $K_p$, $K_i$ or $K_d$. While user settings such as iteration count and population size are adjustable, the default automated outputs were used in this study.

Figure 8 presents the *MATLAB Co-Pilot*-assisted PID tuning results obtained using the prompt shown in Figure 6. The DE-based controller achieves a fast rise time ($t_r = 0.0205$ s) and settling time ($t_s = 0.0366$ s) while eliminating overshoot. The relatively high proportional and derivative gains provide rapid acceleration and improved damping and the large integral gain ($K_i = 100$) ensures negligible steady-state error.

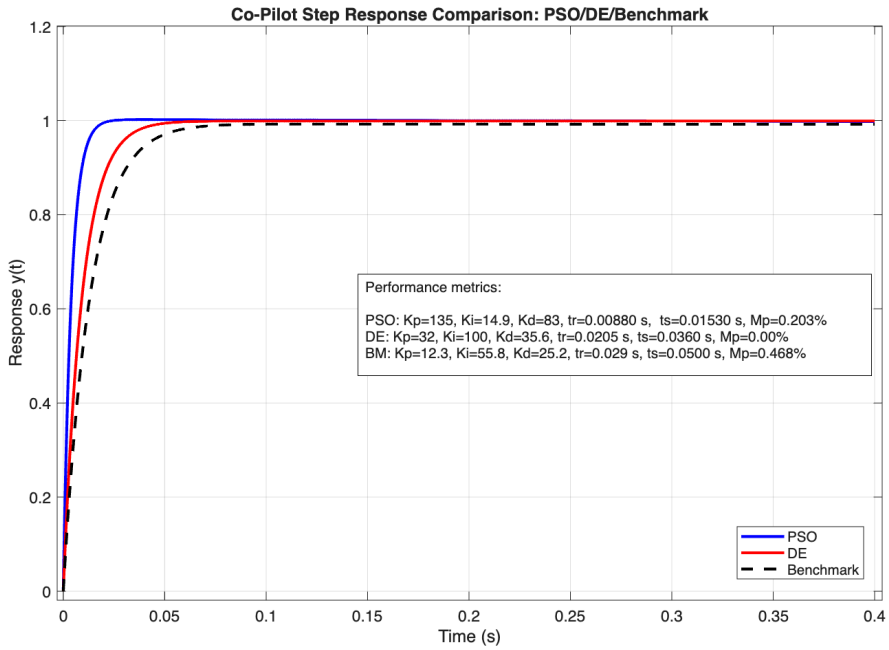


Figure 8: MATLAB Co-Pilot results with PSO and DE optimization method compared against benchmark (BM) case.

The PSO-based controller produces an even faster response, with $t_r = 0.0088$ s, $t_s = 0.0153$ s and a small overshoot of $M_p = 0.203\%$. The exceptionally high proportional and derivative gains ($K_p = 135$, $K_d = 83$) reduce rise time and overshoot but may compromise robustness. These results illustrate the trade-off between aggressive transient performance and practical implementability, highlighting the importance of carefully designed prompts in LLM-assisted tuning.

Figure 9 compares the best-performing LLM-assisted controllers (*SmartControl*-PSO and *MATLAB Co-Pilot*-DE) with the benchmark, ZN, CC and ITSE-tuned controllers.

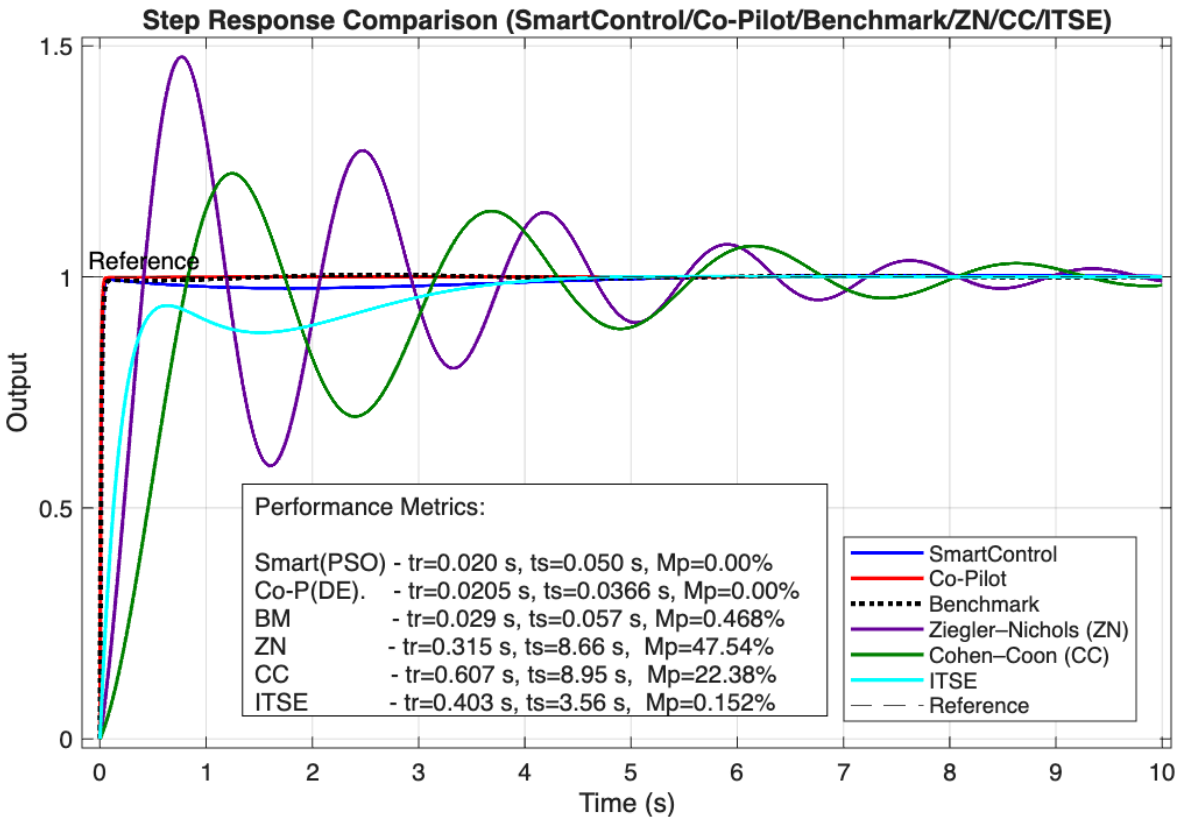


Figure 9: Closed-loop responses of the LLM-based PID controller compared to ones obtained using analytical tuning methods.

The closed-loop responses reveal clear and systematic differences in transient behavior, damping and convergence characteristics. The Ziegler–Nichols (ZN) and Cohen–Coon (CC) controllers exhibit pronounced oscillations, large overshoot, and extended settling times, indicative of underdamped dynamics resulting from limited derivative action and aggressive integral behavior. Although these controllers eventually reach the reference value, their transient performance is poor and unsuitable for applications with strict performance constraints. The ITSE-tuned controller demonstrates improved damping and substantially reduced overshoot relative to ZN and CC. However, this improvement is achieved at the expense of slower rise and settling times, reflecting the objective of ITSE-based tuning, which prioritizes long-term error minimization rather than aggressive transient shaping.

In contrast, the selected LLM-assisted controllers (SmartControl-PSO and MATLAB Co-Pilot-DE) exhibit faster transient responses with negligible overshoot. Both controllers reach

the reference rapidly and display step responses that are closely aligned with the benchmark trajectory in terms of rise and settling behavior. Notably, the MATLAB Co-Pilot-DE controller achieves near-identical transient performance while further suppressing overshoot, reflecting enhanced damping characteristics. These results indicates the capability of LLM-assisted optimization to produce PID controllers with significantly enhanced transient performance relative to both classical and analytically derived designs.

Table 7 reports the PID gains and corresponding time-domain performance metrics underlying the closed-loop responses shown in Figure 9. This table enables a quantitative comparison of how different tuning approaches translate into gain magnitudes and measurable performance outcomes.

Table 7: Comparison of PID parameters and time domain metrics.

| **Method** | $K_p$ | $K_i$ | $K_d$ | $t_r$ (s) | $t_s$ (s) | $M_p$ **(%)** |
|---|---|---|---|---|---|---|
| Benchmark | 12.32 | 55.80 | 25.17 | 0.029 | 0.057 | 0.468 |
| MATLAB Co-Pilot (DE) | 32 | 100 | 35.6 | 0.0205 | 0.0366 | 0.00 |
| SmartControl(PSO) | 19.591 | 11.454 | 29.450 | 0.020 | 0.050 | 0.00 |
| ZN | 3.98 | 7.79 | 0.509 | 0.315 | 8.66 | 47.54 |
| CC | 1.38 | 1.85 | 0.178 | 0.607 | 8.95 | 22.38 |
| ITSE | 2.537 | 2.39 | 1.77 | 0.403 | 3.56 | 0.152 |

The classical ZN and CC tuning rules yield relatively small derivative gains combined with modest proportional and integral action, which explains their poor transient metrics specifically, long rise and settling times and excessive overshoot. The ITSE-based controller significantly reduces overshoot by increasing derivative action; however, this improvement is achieved at the cost of slower system dynamics, as reflected by its comparatively large $t_r$ and $t_s$. These results shows the inherent limitations of classical tuning rules in simultaneously optimizing speed and damping.

The LLM-assisted controllers exhibit an evidently different gain structure. Both *SmartControl* (PSO) and *MATLAB Co-Pilot* (DE) employ substantially higher proportional and derivative gains, which directly contribute to reduced rise times and improved damping. Notably, *MATLAB Co-Pilot* (DE) further increases integral action, effectively eliminating steady-state

error while maintaining fast transient response. These gain configurations reflect the ability of optimization-based approaches to explore more aggressive regions of the PID parameter space that are not typically accessible through analytical or rule-based tuning, while still satisfying stability and performance objectives.

The implications of these numerical trends are further clarified in Figure 10, which visualizes rise time, settling time and maximum overshoot for the benchmark controller alongside the *SmartControl* (PSO and DE) and *MATLAB Co-Pilot* (PSO and DE) solutions. By presenting the same metrics graphically, the bar chart enables direct visual assessment of relative performance with respect to predefined design targets.

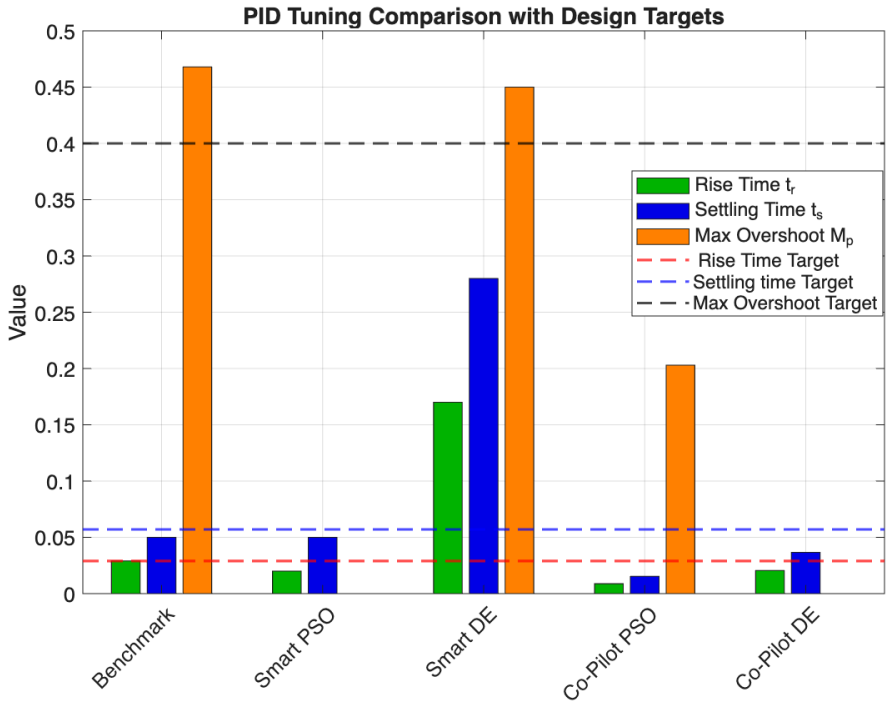


Figure 10: Graphical representation of transient performance metrics for LLM-assisted PID controllers compared to the benchmark and design constraints.

From Figure 10, it is evident that all LLM-assisted controllers, except for *SmartControl*-DE, satisfy the overshoot constraint, with bars remaining well below the specified limit. The *SmartControl*-DE controller performs the worst, exhibiting both slower rise time, settling time and higher overshoot relative to the other LLM-assisted solutions. Rise-time performance is consistently improved relative to the benchmark for all other controllers, particu-

larly for *MATLAB Co-Pilot*-based solutions. Settling-time performance shows smaller variations, with *SmartControl*-PSO closely matching the benchmark and *MATLAB Co-Pilot*-DE achieving the fastest convergence. The bar chart reinforces the quantitative results from Table 7 and clearly illustrates the compromises inherent in aggressive gain selection.

While the LLM-assisted controllers demonstrate superior transient performance in terms of rise time, settling time and overshoot, these improvements must be interpreted with caution. The observed performance gains are achieved through comparatively aggressive gain selection, particularly in the proportional and derivative terms. It should be noted that higher proportional and derivative gains generally imply increased control effort and sensitivity to noise, which were not evaluated in this study but would be critical factors in practical applications. In the absence of constraints on control effort, actuator saturation or noise sensitivity, such gain configurations may not be suitable for practical or industrial deployment. Therefore, the results should be understood as evidence of the optimization capability of LLM-assisted tuning under idealized conditions rather than as definitive indicators of real-world superiority.

## 4.2 Reproducibility of LLM-Assisted PID Tuning

To assess the reproducibility of LLM-assisted PID tuning, each framework and optimization method was run twice using the same prompt and algorithmic settings. The resulting controller parameters and corresponding transient responses were then compared.

Tables 8 and 9 summarize the outcomes for *SmartControl* and *MATLAB Co-Pilot*, respectively. These results provide insight into the variability inherent in stochastic optimization and the consistency of LLM-guided tuning.

For *SmartControl* (Table 8), PSO-generated controllers consistently achieved very fast rise times ($t_r$ = 0.020 s) and low overshoot, while the proportional and integral gains varied considerably between runs ($K_p$ = 8.00 – 19.591, $K_i$ = 11.454 – 36.204). DE-based tuning exhibited wider variability in both gains and transient performance, producing slower rise

and settling times ($t_r$ = 0.170 – 0.650 s, $t_s$ = 0.280 – 1.00 s). These results indicate that the optimization method influences the variability of the tuned gains as well as the transient response, with DE tending toward parameter combinations that yield slower system responses in these runs.

Table 8: SmartControl reproducability

| **Test no.** | $K_p$ | $K_i$ | $K_d$ | $t_r$ (s) | $t_s$ (s) | $M_p$ (%) |
|---|---|---|---|---|---|---|
| PSO 1 | 8.00 | 36.204 | 29.454 | 0.020 | 0.060 | 0.72 |
| PSO 2 | 19.591 | 11.454 | 29.450 | 0.020 | 0.050 | 0.00 |
| DE 1 | 1.173 | 3.226 | 0.996 | 0.650 | 1.00 | 0.26 |
| DE 2 | 5.438 | 10.230 | 4.142 | 0.170 | 0.280 | 0.45 |

In the case of *MATLAB Co-Pilot* (Table 9), PSO runs produced extremely fast rise and settling times ($t_r = 0.0072$ - 0.0088 s, $t_s = 0.0124$ – 0.0153 s) but exhibited substantial variability in the proportional and derivative gains. Notably, one PSO run resulted in $K_p = 0$, corresponding to an unfeasible controller that lacks proportional action.

Table 9: MATLAB Co-pilot reproducability

| **Test no.** | $K_p$ | $K_i$ | $K_d$ | $t_r$ (s) | $t_s$ (s) | $M_p$ (%) |
|---|---|---|---|---|---|---|
| PSO 1 | 135 | 14.9 | 83 | 0.0088 | 0.0153 | 0.203 |
| PSO 2 | 0 | 9.42 | 100 | 0.0072 | 0.0124 | 0.40 |
| DE 1 | 32 | 100 | 35.6 | 0.0205 | 0.036 | 0.00 |
| DE 2 | 83.31 | 100 | 53 | 0.0137 | 0.023 | 0.398 |

A key factor underlying these trends is that neither *SmartControl* nor *MATLAB Co-Pilot* enforce explicit constraints or analytical relationships on the PID parameters. Consequently, stochastic optimization runs may converge to different local optima, producing variable gain sets and transient responses even under identical prompts. While more carefully structured prompts or explicit parameter bounds could reduce this variability, the current results

demonstrate that LLM-assisted tuning generally produces controllers with realistic and implementable behavior. In many cases, aggressive transient performance can be achieved without violating fundamental stability constraints, though exceptions such as the $K_p = 0$ case must be considered in practical implementations.

While the limited number of repeated runs restricts statistical generalization, the observed variability provides useful qualitative insight into the consistency of LLM-assisted tuning under identical prompts.

In summary, the key insights derived from the analysis of LLM-assisted PID tuning, particularly in comparison with the analytically derived benchmark, are presented in Table 10.

Table 10: Summary of key insights from LLM-assisted PID tuning compared to the analytical benchmark

| **Aspect** | **Key Insights** |
|---|---|
| Transient performance relative to the benchmark | Both *SmartControl* and *MATLAB Co-Pilot* generated controllers with generally faster rise and settling times while maintaining low or negligible overshoot compared to the analytically derived benchmark. LLM-assisted tuning can achieve transient performance that is competitive with, and in some cases slightly superior to, the benchmark system. |
| Practical limitations and engineering considerations | The absence of explicit constraints on PID gains contributed to variability and occasionally unrealistic gain combinations. Careful prompt design or bounded optimization can improve practical reliability. Despite these limitations, most LLM-generated controllers were realistic and implementable, demonstrating that fundamental engineering requirements can generally be satisfied. |
| Reproducibility and consistency of PID parameters | Variability was observed across repeated runs: PSO exhibited higher variability in MATLAB Co-Pilot, whereas DE showed greater variability in SmartControl. Notably, one MATLAB Co-Pilot PSO run produced $K_P = 0$, resulting in an unfeasible controller. DE-based optimization within MATLAB Co-Pilot demonstrated higher consistency, producing reliable and repeatable closed-loop responses across runs. |
| Comparative performance of LLM-assisted approaches | Considering both performance and reproducibility, MATLAB Co-Pilot combined with DE optimization provided the most favorable trade-off between rise time, settling time, overshoot, and parameter consistency. Among the approaches evaluated, it represents the most effective LLM-assisted strategy for PID tuning within the context of the considered linear system. |

# 5 Conclusions

This thesis investigated the use of large language models as interactive assistants for PID controller design and tuning, with a focus on two representative frameworks: the GUI-based *SmartControl* and the chat-based *MATLAB Co-Pilot*. A carefully designed prompting strategy was employed to specify system dynamics, performance objectives, and optimization methods. The LLM-assisted approaches were systematically evaluated against an analytically derived benchmark PID controller from *Albatran, Smadi & Bataineh, 2020*, as well as classical tuning rules (ZN, CC, and ITSE), using a moderately underdamped second-order plant.

With respect to transient performance, the results demonstrate that LLM-assisted interactive PID tuning can achieve performance comparable to, and in some cases slightly better than, classical and analytical tuning methods for the defined time-domain objectives of this study. Under the idealized, nominal simulation conditions considered, several LLM-assisted configurations, most notably the *MATLAB Co-Pilot*-DE and *SmartControl*-PSO approaches exhibited faster rise times and reduced overshoot relative to the reference controllers. These improvements were enabled by stochastic optimization-driven selection of PID gains, which allowed systematic exploration of parameter regions not typically examined by analytical formulas or classical heuristics. In particular, increased proportional and derivative gains contributed to improved transient speed and damping, while integral action influenced overall transient shaping, as reflected in the closed-loop step responses and associated time-domain metrics.

The reproducibility and consistency of LLM-assisted tuning were evaluated through repeated optimization runs (two per configuration) for both *SmartControl* and *MATLAB Co-Pilot*, with PSO and DE applied separately within each framework. Although the limited number of runs restricts statistical generalization, the results indicate that *SmartControl*-PSO and *MATLAB Co-Pilot*-DE produced more consistent PID parameters across repeated sessions compared

to their respective alternatives. Overall, *SmartControl* exhibited superior reproducibility, which can be attributed to its purpose-built design for PID tuning. In contrast, the variability observed in *MATLAB Co-Pilot* results was found to depend strongly on prompt formulation, suggesting that prompt design and iterative refinement play a critical role in achieving consistent outcomes when using general-purpose LLM-based assistants.

From a practical and engineering perspective, the findings also indicates important limitations of LLM-assisted PID tuning within the scope of this study. While the tuned controllers successfully met the defined time-domain performance targets, this was often achieved through relatively high proportional, integral and derivative gains. Such gain magnitudes raise concerns regarding control effort, noise sensitivity, actuator saturation and overall feasibility in real-world implementations. Consequently, the results emphasize that LLM-assisted tuning should be viewed as a design-support tool rather than an autonomous replacement for classical control engineering judgment, with human oversight and constraint-aware validation remaining essential prior to practical deployment.

Future work should extend this investigation beyond the linear second-order system examined here. Frequency-domain analysis could be incorporated to assess robustness and stability margins under model uncertainty and external disturbances. Hardware-in-the-loop (HITL) experiments would further support validation of the simulation-based findings and enable evaluation of practical implementation issues such as actuator saturation, noise sensitivity, and real-time computational constraints. In addition, integrating explicit safety, robustness and gain constraints into the LLM prompting strategy or optimization process would improve reliability and broaden applicability to real-world control systems with more complex dynamics.

# References

Albatran, S., Smadi, I. A. & Bataineh, H. A. (2020). Generalized optimal and explicit PI/PID tuning formulas for underdamped second-order systems. *International Journal of Control, Automation and Systems* 18(4), pp. 1023–1032.

Almabrok, A., Psarakis, M. & Dounis, A. (2018). Fast tuning of the PID controller in an HVAC system using the big bang–big crunch algorithm and FPGA technology. *Algorithms* 11(10), p. 146.

Bassi, S., Mishra, M. & Omizegba, E. E. (2011). Automatic tuning of proportional-integral-derivative (PID) controller using particle swarm optimization (PSO) algorithm. *International Journal of Artificial Intelligence & Applications* 2(4), p. 25.

Bennett, S. (2001). The past of PID controllers. *Annual reviews in control* 25, pp. 43–53.

Borase, R. P., Maghade, D., Sondkar, S. & Pawar, S. (2021). A review of PID control, tuning methods and applications. *International Journal of Dynamics and Control* 9(2), pp. 818–827.

Chen, Z., Wang, Z., Yang, Y., Li, Q. & Zhang, Z. (2024). PID Control-Based Self-Healing to Improve the Robustness of Large Language Models. *arXiv preprint arXiv:2404.00828*.

Cui, C., Liu, J., Feng, J., Hui, P., Ghias, A. M. & Zhang, C. (2024). Large language models based multi-agent framework for objective oriented control design in power electronics. *arXiv preprint arXiv:2406.12628*.

Franklin, G. F., Powell, J. D., Emami-Naeini, A. & Powell, J. D. (2002). *Feedback control of dynamic systems*. Vol. 4. Prentice hall Upper Saddle River.

Guo, X., Keivan, D., Syed, U., Qin, L., Zhang, H., Dullerud, G., Seiler, P. & Hu, B. (2024). Controlagent: Automating control system design via novel integration of llm agents and domain expertise. *arXiv preprint arXiv:2410.19811*.

Haidekker, M. A. (2020). *Linear feedback controls: the essentials*. Elsevier.

Joshi, A., Sanyal, S. & Roy, K. (2025). Neuro-LIFT: A neuromorphic, LLM-based interactive framework for autonomous drone flight at the edge. *arXiv preprint arXiv:2501.19259*.

Kamenko, I., Ilic, S. & Congradac, V. (2025). LLM-based PID controller optimization. In: *2025 10th International Conference on Smart and Sustainable Technologies (SpliTech)*, pp. 1–6. DOI: 10.23919/SpliTech65624.2025.11091752.

Karn, R. R. (2025). Linear Feedback Control Systems for Iterative Prompt Optimization in Large Language Models. *arXiv preprint arXiv:2501.11979*.

Kevian, D., Syed, U., Guo, X., Havens, A., Dullerud, G., Seiler, P., Qin, L. & Hu, B. (2024). Capabilities of large language models in control engineering: A benchmark study on gpt-4, claude 3 opus, and gemini 1.0 ultra. *arXiv preprint arXiv:2404.03647*.

Koziolek, H., Gruener, S. & Ashiwal, V. (2023). Chatgpt for plc/dcs control logic generation. In: *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, pp. 1–8.

Lakhani, A. I., Chowdhury, M. A. & Lu, Q. (2021). Stability-preserving automatic tuning of PID control with reinforcement learning. *arXiv preprint arXiv:2112.15187*.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P. & Zeng, A. (2022). Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*.

Liu, H., Zhu, Y., Kato, K., Tsukahara, A., Kondo, I., Aoyama, T. & Hasegawa, Y. (2024). Enhancing the llm-based robot manipulation through human-robot collaboration. *arXiv preprint arXiv:2406.14097*.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S. & Rocktäschel, T. (2019). A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*.

Mabrok, M. A., Mohamed, H. K., Abdel-Aty, A.-H. & Alzahrani, A. S. (2020). Human models in human-in-the-loop control systems. *Journal of Intelligent & Fuzzy Systems* 38(3), pp. 2611–2622.

Moran, S. & Önsten, A. (2025). *Autonomous Forklift Path Following Using Large Language Model Agents and Reinforcement Learning*.

Narimani, M. & Emami, S. A. (2025). AgenticControl: An Automated Control Design Framework Using Large Language Models. *arXiv preprint arXiv:2506.19160*.

Pandey, S. K., Veeranna, K., Kumar, B. & Deshmukh, K. (2020). A robust auto-tuning scheme for PID controllers. In: *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, pp. 47–52.

Tohma, K., Okur, H. İ., Gürsoy-Demir, H., Aydın, M. N. & Yeroğlu, C. (2025). SmartControl: Interactive PID controller design powered by LLM agents and control system expertise. *SoftwareX* 31, p. 102194.

Wu, H.-N. & Zhang, X.-M. (2020). Stochastic stability analysis and synthesis of a class of human-in-the-loop control systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52(2), pp. 822–832.

Xu, J., Zheng, Z. & Wang, Z. (2024). LAC: Using LLM-based Agents as the Controller to Realize Embodied Robot. In: *2024 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, pp. 1894–1899.

Zhou, Z., Lu, Y., Zhu, Y., Zhao, Y., He, B., He, L., Yu, W. & Iwasawa, Y. (2025). LLMs-guided adaptive compensator: Bringing Adaptivity to Automatic Control Systems with Large Language Models. *arXiv preprint arXiv:2507.20509*.

Zhu, Y., Wang, S., Zhong, W., Shen, N., Li, Y., Wang, S., Li, Z., Wu, C., He, Z. & Li, L. (2025). A Survey on Large Language Model-Powered Autonomous Driving. *Engineering*.