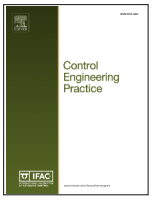
Control Engineering Practice 170 (2026) 106847

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac



Artificial intelligence based learning methods for the automatic tuning of fixed-parameter MIMO PID controllers for industrial applications: A review and comparison

J.A. van Niekerk [a,b], J.D. le Roux [b], I.K. Craig [b,*]

[a] Zutari, Pretoria, South Africa
[b] Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria, South Africa

ARTICLE INFO

Keywords:
Artificial intelligence
Automatic tuning
Bayesian optimisation
Grinding mills
Pareto front
Particle swarm optimisation
Proximal policy optimisation

ABSTRACT

This paper reviews and compares artificial intelligence (AI) methods for the automatic tuning of multi-input-multi-output (MIMO) proportional-integral-derivative (PID) controllers in industrial process applications. The study focuses on fixed-parameter PID tuning and introduces a generalised procedure that unifies diverse AI methods within a single autotuning framework. A Pareto-front-based weighting strategy is proposed to balance performance and actuator usage, enabling fair comparison of tuning outcomes across different algorithms. Within this framework, three representative approaches, particle swarm optimisation (PSO), proximal policy optimisation (PPO), and Bayesian optimisation (BO), are implemented and evaluated against the defining criteria of an ideal autotuner: versatility, global optimality, data efficiency, and safety. The analysis bridges computational intelligence and machine learning perspectives, providing a structured benchmark for assessing AI-based tuning performance. Results show that all AI-based tuners successfully identify high-performing controller parameters for multivariable nonlinear systems, confirming their applicability to industrial processes. Among them, BO achieves the best overall performance, offering superior convergence speed and data efficiency through surrogate-driven optimisation. By maximising information gained from each plant trial, BO provides a safe, robust, and computationally efficient tuning method ideally suited to practical industrial deployment.

1. Introduction

Artificial intelligence (AI) refers to the capability of digital computers to perform tasks typically associated with intelligent beings (Mukhamediev et al., 2022). This review examines the application of AI to the automatic tuning of proportional-integral-derivative (PID) controllers, which are widely used to regulate industrial processes in sectors such as manufacturing, chemical production, refining, distilling, mining, water treatment, energy generation, and food processing. In these contexts, control systems are designed to maintain efficient, safe, and reliable operation by regulating critical process variables such as temperature, pressure, flow, level, density, or composition. The objective of this review is to summarise the current state of the art in AI-based PID tuning, propose a generic framework for AI-driven PID autotuning, compare the tuning performance of methods across different AI classes, and provide a reference to guide practitioners in selecting the most suitable approach for their applications.

PID controllers are ubiquitously used in industrial process applications due to their simplicity and the interpretability of their tuning parameters (Borase et al., 2021). The proportional gain increases the controller output in proportion to the error, the integral term eliminates steady-state error over time, and the derivative term acts as a damping force that responds to the rate of change of the error (Patil et al., 2024). While PID controllers are structurally simple, effective tuning remains challenging, particularly for MIMO loops, and many industrial implementations therefore operate with suboptimal tuning (Lequesne, 2021; Shamsuzzoha & Skogestad, 2010). Studies report that up to 66% of control loops perform suboptimally, with approximately 33% operated in manual mode (Desborough & Miller, 2002). Given both their widespread use and the persistent prevalence of inadequate tuning, continued research into improved PID tuning methodologies remains well justified.

Research presents compelling arguments for completely replacing PID controllers with AI-based controllers that directly manipulate actuators in response to measured process variables and setpoint deviations (Beahr et al., 2024; Norlund et al., 2024; Shou et al., 2025; Zhang et al., 2023); however, such controllers, particularly neural-network-based controllers, are often black-box systems that are difficult to

* Corresponding author.
E-mail address: ian.craig@up.ac.za (I.K. Craig).

https://doi.org/10.1016/j.conengprac.2026.106847
Received 13 November 2025; Received in revised form 8 February 2026; Accepted 9 February 2026
Available online 17 February 2026

interpret and troubleshoot during faults (Lawrence et al., 2020a; Nian et al., 2020). Although explainable artificial intelligence (XAI) seeks to improve the transparency and interpretability of these models (Giudici & Raffinetti, 2021; Zhang et al., 2020), this paper instead advocates retaining the existing PID controllers embedded within programmable logic controllers (PLCs) and distributed control systems (DCSs) while enhancing their performance through AI-based optimisation. This strategy minimises modifications to plant control infrastructure, preserves existing actuator and sensor interfaces, and situates the AI to control-system interface at Level 2 (supervisory control) of the Purdue hierarchy (Williams, 1990). Once tuning is complete, the AI system is disconnected, and the optimised fixed-parameter PID controller continues to operate independently, without any further changes to the tuning parameters.

Although adaptive PID control can enhance the performance of nonlinear processes by adjusting controller parameters either continuously or at discrete intervals based on state observations (Guan & Yamamoto, 2021; Lawrence et al., 2020b) caution that implementing a fully adaptive scheme, where the PID gains are updated at every time step, may introduce additional nonlinearities into the closed-loop system, thereby complicating stability analysis (Malmborg et al., 1996). In contrast, the approach adopted in this study deliberately avoids continuous adaptation: once the AI-based tuning agent has identified suitable parameters, it can be disconnected, leaving the PID controller to operate independently.

Neumann-Brosig et al. (2020) define the criteria for an ideal autotuner as follows: it should be *versatile*, with applicability to a wide range of control structures; *globally optimal*, reliably identifying the best solution without becoming trapped in local optima; and *data efficient*, enabling online implementation with minimal experimental effort. Data efficiency is intended to maximise long-term gains while minimising the short-term costs associated with experimentation. In addition, the autotuner must be *safe*, permitting global optimisation over the parameter space while preserving closed-loop stability, such that the resulting system remains bounded-input bounded-output stable (Seborg et al., 2011).

The contributions of this paper are as follows:

- a Pareto front based procedure for weighting the priorities of variables and selecting the compromise between performance and actuator usage to support controller tuning and the evaluation of AI-based methods,
- a review of AI-based tuning methods, with a focus on fixed-parameter PID tuning,
- a comparison of tuning performance between computational intelligence and machine learning approaches, assessed against the criteria for an ideal autotuner, and
- a generalised AI-based PID tuning procedure.

The paper is structured as illustrated in Fig. 1. Section 1 provides an introduction, followed by Section 2, which formulates the automatic tuning problem to be addressed using AI-based algorithms. Section 3 reviews AI-based PID tuning methods capable of addressing the stated problem and concludes with the selection and motivation of representative algorithms for comparison. Section 4 presents a preliminary discussion of these algorithms, particle swarm optimisation (PSO), proximal policy optimisation (PPO), and Bayesian optimisation (BO), while Section 5 describes the proposed methodology for implementing the AI-based tuner. Section 6 presents the application of the methodology to an ore-milling circuit and provides a comparative analysis of the results. Section 7 concludes the paper.

2. Problem statement

A nonlinear multivariable multi-input-multi-output (MIMO) plant can be represented in state-space form as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{1a}$$

$$\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t)) \tag{1b}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u}(t) \in \mathbb{R}^m$ is the manipulated variable vector, and $\boldsymbol{y}(t) \in \mathbb{R}^p$ is the controlled variable vector. Disturbances are omitted from (1) because, in most industrial applications, it is impractical to introduce a timed and measurable step disturbance and reliably repeat it over the duration of AI-based tuning. This omission does not imply that AI-based methods cannot be tuned for disturbance rejection; as demonstrated by van Niekerk et al. (2025a), rejection of controller interactions in a MIMO system can serve as an effective proxy for disturbance-rejection capability.

To achieve setpoint tracking and disturbance rejection, the plant is regulated using diagonal PID controllers arranged in a MIMO loop. The controller action is described as

$$\boldsymbol{u}(t) = \boldsymbol{K}(\boldsymbol{e}(t), \boldsymbol{\eta}), \tag{2}$$

where $\boldsymbol{K} \in \mathbb{R}^{p\times m}$ denotes the controller matrix, $\boldsymbol{e}(t) \in \mathbb{R}^p$ is the tracking error vector defined as the difference between the setpoint vector $\boldsymbol{y}^{sp}(t)$ and the controlled variable vector $\boldsymbol{y}(t)$, and $\boldsymbol{\eta} \in \mathbb{R}^{3\times m}$ contains the PID tuning parameters assigned to the diagonal elements.

The challenge is to select tuning parameters for a MIMO controller that yield optimal controller performance. MIMO controllers, in which multiple manipulated variables interact with multiple controlled variables, are considerably more challenging to tune than single-input single-output (SISO) controllers due to loop interactions, differing variable sensitivities, and disparities in variable magnitudes and engineering units. Effective tuning therefore requires appropriate pairing of manipulated and controlled variables, as well as careful weighting of scales and performance trade-offs, as discussed in Sections 5.5 and 6.2. To achieve this, controller performance must be expressed through an objective function, in which weighting matrices are used to balance the relative importance of different variables. This objective function can be formulated using various performance criteria, typically including the integral absolute error (IAE), integral square error (ISE), time-weighted absolute error (ITAE), or the integral of the time-weighted squared error (ITSE) (Seborg et al., 2011).

The objective function is defined as

$$J : \mathcal{A} \subseteq \mathbb{R}^{3\times m} \rightarrow \mathbb{R}, \tag{3}$$

and must be minimised by selecting the tuning parameters

$$\boldsymbol{\eta} \in \mathcal{A} \in \mathbb{R}^{3\times m} \tag{4}$$

where $\mathcal{A}$ denotes the domain of tuning parameters. The optimisation problem can therefore be stated as

$$\min_{\boldsymbol{\eta}\in\mathcal{A}} \boldsymbol{J}(\boldsymbol{\eta}). \tag{5}$$

The direction of optimality may differ depending on the AI method applied. Reinforcement learning (RL) seeks to maximise cumulative rewards, whereas BO and nature-inspired algorithms generally minimise a cost function. In either case, the goal is to determine the optimum of the objective function by selecting the most effective tuning parameters.

3. Review of AI-based PID tuning in industrial processes

The purpose of this review is to identify AI-based methods in the literature that are capable of tuning fixed-parameter PID controllers by optimising the objective function defined in (5). From these methods, representative candidate algorithms are motivated and selected for comparative evaluation against the characteristics of the ideal autotuner defined in Section 1. Although the comparison is conducted on a MIMO process, the review also includes SISO applications, as methods developed for SISO systems can be extended to MIMO applications provided the objective function accounts for the additional complexity of multivariable control.

The review content was obtained from databases including IEEE Xplore, ScienceDirect, and Google Scholar using search expressions such

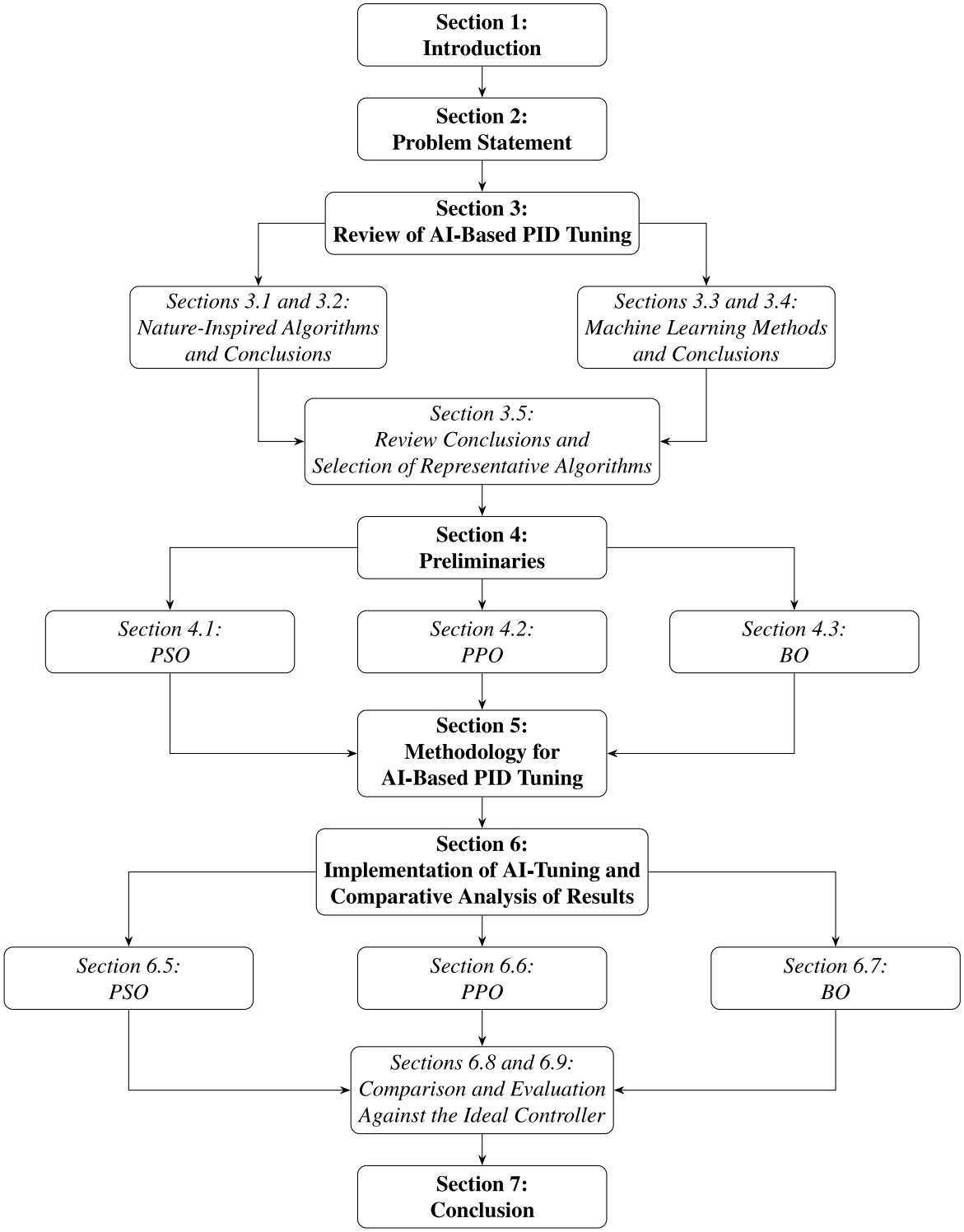


Fig. 1. Logical diagram representing the structure of the paper.

as "AI PID", "intelligent PID", "nature-inspired PID", "metaheuristic PID", "fuzzy PID", "reinforcement learning PID", "supervised learning PID", "unsupervised learning PID", and "Bayesian optimisation PID" for the period 2015–2025. Where relevant papers were identified prior to this time frame, they were also included in the study.

While there is no universally accepted classification of AI methods among scholars (Joseph et al., 2022; Mukhamediev et al., 2022; Nian et al., 2020; Vilone & Longo, 2021), a taxonomy of AI-based learning approaches for the automatic tuning of PID controllers is developed here by synthesising insights from several publications (Blondin et al., 2019; Engelbrecht, 2007; Nian et al., 2020; Sumar et al., 2010; Zhu, 2014). Accordingly, AI methods suitable for PID tuning can be broadly grouped into computational intelligence and machine learning.

Computational intelligence comprises two subclasses: intelligent algorithms and nature-inspired algorithms. Intelligent algorithms include techniques such as artificial neural networks (ANN) and fuzzy systems, whereas nature-inspired algorithms draw on mechanisms from natural processes, including swarm intelligence, genetic mutation, and

evolutionary strategies. Techniques such as fuzzy logic, ANN, and adaptive neuro-fuzzy inference systems (ANFIS) have been successfully applied to update PID parameters (Caiza et al., 2022; Chopra et al., 2014; El-Gendy et al., 2020; Ladjouzi & Grouni, 2020; Yan et al., 2025; Yusoff et al., 2017). In these approaches, parameters are adapted continuously based on the controller tracking error and its derivative, while the control signal is generated by the adaptive PID controller. Although nonlinear processes benefit from adaptive PID control, the objective of this study is to tune fixed PID parameters, as justified in Section 1, and therefore the intelligent algorithms will not be considered further in this review.

Machine learning methods are generally categorised into supervised learning (SL), unsupervised learning (UL), and RL (Nian et al., 2020). RL has been applied to both adaptive (Chowdhury et al., 2023; Guan & Yamamoto, 2021; Lakshmi & Manonmani, 2025; Mate et al., 2023; Meng et al., 2023; Qin et al., 2018; Wang & Ricardez-Sandoval, 2024) and fixed-parameter PID tuning, as detailed in Section 3.3.1. Only a limited number of studies have explored SL-based PID tuning, which are reviewed in Section 3.3.2. Finally, just a single case was found where UL has been employed for PID tuning, discussed in Section 3.3.3.

Patil et al. (2024) classify BO as a metaheuristic-probabilistic algorithm. For the purposes of this paper, however, BO is treated as a probabilistic machine learning method because it exhibits the core characteristics of machine learning: learning from data, employing probabilistic models, making predictions, and improving performance over time through objective function optimisation. BO has also been applied in adaptive PID control (König et al., 2021), although the emphasis here is on its applications for fixed-parameter tuning which is discussed in Section 3.3.4.

3.1. Nature-inspired algorithms

The nature-inspired algorithms of the computational intelligence subclass have attracted significant interest in the application of PID tuning evident from the number of review articles dedicated to this topic (Ghosal et al., 2012; Joseph et al., 2022; Patil et al., 2024). Joseph et al. (2022) report on 73 metaheuristic algorithms published during the period from 1975 to 2020. Genetic algorithms (GA) and PSO are among the most established and effective nature-inspired methods for PID controller tuning, particularly for nonlinear and poorly modelled systems. GA employs evolutionary operators such as selection, crossover, and mutation to perform global search, making it robust to local minima and well suited to complex optimisation landscapes. PSO, inspired by collective swarm behaviour, updates candidate solutions through shared individual and global best information, typically achieving faster convergence with lower computational overhead (Fister et al., 2016; Joseph et al., 2022; Patil et al., 2024).

Chopra et al. (2014) and Jayachitra and Vinodha (2014) investigate the use of the GA for PID tuning in the concentration control of a continuous stirred tank reactor (CSTR) linear model in simulation. Although the CSTR is inherently a multi-input-multi-output (MIMO) system, the analysis is simplified to a single transfer function describing the dynamics of the output concentration in response to the feed flow rate. Using the ISE criterion, the GA achieves satisfactory setpoint tracking and disturbance rejection after approximately 2000 iterations. In related work, Nekoui et al. (2010) compare the Ziegler-Nichols (ZN) and PSO methods for CSTR concentration control via coolant flow manipulation. Employing the integral of the ITAE objective, the PSO-tuned controller reduces overshoot and settling time relative to the ZN method, converging after about 400 iterations.

Sharma et al. (2016) employ GA and PSO algorithms to tune PID level controllers for an interacting two-tank system with hydraulic coupling. Regulating the feed flow rate to control tank level, they report that PSO eliminates overshoot but exhibits a longer settling time, whereas GA achieves faster convergence.

Younis et al. (2019) compare GA-based PID tuning with classical ZN, Cohen-Coon, and Chien-Hrones-Reswick methods for cascaded double-tank systems, concluding that GA achieves the lowest ISE despite the nonlinear, delay-like dynamics of the system. Extending this to more complex configurations, Cornejo et al. (2020) apply GA and the Imperialist Competitive Algorithm (ICA) to triple-tank systems, finding ICA superior in minimising ISE after approximately 3000 iterations.

In cascade control, Pramanik et al. (2021) compare GA and Harmony Search Algorithm (HSA) tuning for a single-tank system, where both optimisation methods outperform ZN, with HSA delivering better closed-loop performance after 4000 iterations on a pilot plant. More recently, Achu Govind and Mahapatra (2024) apply tree seed optimisation (TSO) with μ-analysis for coupled conical, spherical, quadruple, and sextuple tank configurations, demonstrating improved robustness and precision compared with PSO-, fuzzy-, and sliding-mode-based controllers, and highlighting its industrial applicability for nonlinear interconnected level systems.

Singh et al. (2016) employ the cuckoo search (CS) algorithm to tune a PID controller for regulating cooling-vessel pressure via feed-valve manipulation, showing improved setpoint step response compared with ZN tuning after 10,000 iterations. Similarly, Li and Feng (2020) apply an improved artificial bee colony (IABC) algorithm for water-distribution-header pressure control by adjusting pump speed, demonstrating that, within 100 iterations, the IABC minimises a composite cost function, combining ISE, actuator-usage penalty, and overshoot, more effectively than the standard ABC method.

El-Gendy et al. (2020) employ GA to optimise PID tuning parameters for temperature control in a divided-wall distillation column separating ethanol, propanol, and n-butanol. Three diagonally arranged PID loops regulate tray temperatures to maintain product purity, achieving superior disturbance rejection, measured by ISE, compared with the PI controllers of Khanam (2014), with convergence after about 2500 iterations. In a related study, Deulkar and Hanwate (2020) compare GA- and PSO-tuned PID controllers for temperature regulation in a 2×2 MIMO CSTR model, showing that PSO outperforms GA in reducing overshoot when using a composite objective function of steady-state error and peak overshoot, converging after approximately 600 of 1400 iterations. Similarly, Baruah and Dewan (2017) report that, for a CSTR with temperature controlled via coolant flow, the PSO-tuned controller yields significantly lower overshoot than GA while maintaining comparable rise and settling times. Extending beyond CSTRs, Anbumani et al. (2017) apply the grey wolf optimiser (GWO) for PID tuning in a shell-and-tube heat exchanger, demonstrating improved performance over PSO based on the (IAE) metric.

Behroozsarand and Shafiei (2010) apply the non-dominated sorting genetic algorithm II (NSGA-II) to tune 11 diagonal PID controllers for a MIMO amine gas-sweetening plant. Using a bi-objective optimisation of IAE and overshoot, they demonstrate that NSGA-II achieves well-balanced trade-offs between tracking performance and robustness. Similarly, dos Santos Coelho and Mariani (2012) propose the chaotic firefly algorithm (CFA) and benchmark it against GA and PSO for tuning multiloop PID controllers in the Wood-Berry distillation column and a polymerisation reactor. Across both MIMO case studies, CFA yields lower objective function values and superior control performance using a population of 15 fireflies over 100 generations. Extending this line of research, Reynoso-Meza et al. (2012) employ the differential evolution (DE) algorithm to design decentralised PI controllers for the Wood-Berry column, formulating a multi-objective problem that constrains sensitivity, complementary sensitivity, and closed-loop log modulus. Their results show that DE-based tuning provides improved disturbance rejection and robustness compared with the largest log modulus method of Luyben (1986).

Fister et al. (2016) evaluate six population-based optimisation methods, bat algorithm (BA), hybrid bat algorithm (HBA), DE, PSO, GA, and CS, for automatic PID tuning in a two-degree-of-freedom robotic arm. They assess suitability for online applications requiring fast convergence

and low computational cost, concluding that reactive nature-inspired algorithms, particularly PSO and BA, offer the most promising performance in terms of convergence speed and robustness.

3.2. Nature-inspired review conclusions

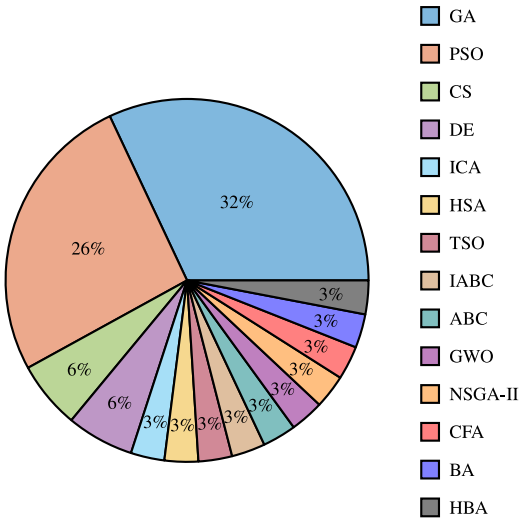
The literature demonstrates the broad applicability of nature-inspired methods for PID tuning across diverse process domains. Techniques such as GA and PSO consistently outperform classical heuristic rules (e.g., ZN, Cohen-Coon, Chien-Hrones-Reswick) in both transient and steady-state performance. However, these classical methods have largely been superseded in modern process-control practice, with contemporary texts such as Seborg et al. (2011) recommending alternatives like IMC (Garcia & Morari, 1982) and SIMC (Skogestad, 2003). Consequently, comparing AI-based methods with tuning techniques developed more than seventy years ago offers limited practical value beyond establishing a historical performance baseline.

Fig. 2 presents the distribution of nature-inspired fixed-parameter PID autotuning studies. The data are extracted from the publications reviewed in Section 3.1 by identifying the algorithms employed and the control structures considered in each study.

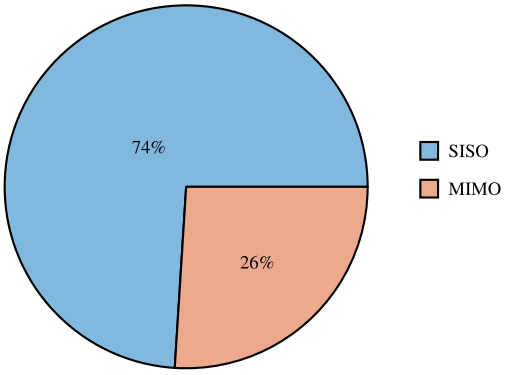
The distribution of nature-inspired algorithms, as illustrated in Fig. 2(a), reveals that the research community has predominantly favoured population-based metaheuristics. GA and PSO together account for approximately two-thirds of the studies surveyed, with shares of 32% and 26%, respectively. This strong representation highlights the sustained appeal of these methods owing to their ease of implementation and minimal problem-specific assumptions. The remaining studies are distributed across a diverse set of algorithms, each contributing between 6% and 3% of the total. These include DE, CS, HSA, the CFA, and other swarm- and evolution-inspired methods such as the ABC, IABC, and GWO. The comparatively small representation of these algorithms suggests that, while they have demonstrated potential in specific process-control applications, their use remains exploratory and less standardised. Overall, the dominance of GA and PSO indicates that most AI-based PID tuning research continues to rely on relatively mature evolutionary and swarm-intelligence frameworks, rather than more recent or problem-specific variants.

The control-structure analysis in Fig. 2(b) shows that the majority of nature-inspired-based autotuning studies have focused on SISO systems, which comprise approximately 74% of the reported applications. This prevalence reflects the relative simplicity and tractability of SISO processes, where the decoupling between loops allows direct evaluation of algorithmic performance without the added complexity of loop interactions. In contrast, MIMO systems represent around 26% of the studies, highlighting a growing but still limited interest in extending AI-based tuning techniques to more complex industrial systems.

Collectively, these studies reinforce the potential of nature-inspired approaches to bridge the limitations of traditional PID tuning, offering improved precision, robustness, and adaptability for modern industrial control applications.



(a) Nature-inspired algorithm distribution.



(b) Distribution of nature-inspired PID tuning studies by control structure (SISO vs MIMO).

Fig. 2. Distribution of nature-inspired fixed-parameter PID autotuning studies: (a) autotuning algorithms and (b) control-structure type.

3.3. Machine learning methods

3.3.1. Reinforcement learning methods

In RL, an agent interacts with a stochastic environment that can be modelled as a Markov decision process (MDP). At each time step t, the agent observes the current state $\boldsymbol{s}_t \in \mathbb{R}^p$, selects an action $\boldsymbol{a}_t \in \mathbb{R}^{3m}$ according to its policy π, and the environment responds with a new state $\boldsymbol{s}_{t+1}$ and a scalar reward r_{t+1}. Over time, this repeated exchange generates sequences of states, actions, and rewards, which may be episodic or continue indefinitely. The objective of the agent is to learn an optimal policy that maps states to actions in order to maximise the long-term cumulative reward, known as the return G_t (Lee et al., 2018; Sutton & Barto, 2018).

RL is a promising approach for automatic PID tuning, providing a data-driven framework applicable to both linear and nonlinear systems. Early work by Berger and da Fonseca Neto (2013) shows that RL-based tuning can exploit closed-loop performance over entire trajectories, enabling controllers to account for long-term behaviour and improving flexibility in both online and offline settings.

Building on this foundation, Lawrence et al. (2020b) present a finite-difference random-search method for automatic PID tuning. Their approach perturbs controller gains, evaluates the resulting step responses, and iteratively updates the policy without requiring gradients or process models. By leveraging full closed-loop trajectories rather than incremental updates, their results show that even simple RL search strategies achieve competitive PID tuning performance. A related contribution by Lawrence et al. (2020a) further demonstrates that RL can incorporate practical features such as anti-windup while preserving the interpretability of the underlying PID structure.

Bridging probabilistic inference and control design, Jesawada et al. (2022) introduce a Kullback-Leibler divergence (KLD)-based framework that maps optimal policies learned via the probabilistic inference for learning control (PILCO) algorithm into interpretable PID parameters. This integration of the data efficiency of PILCO with structured PID mapping provides a practical link between advanced RL formulations and industrial control. Similarly, Dogru et al. (2022) reformulate PI tuning as a contextual bandit problem, eliminating the need for full Markovian state transitions. Offline pre-training on simple step-response models enables

the agent to approximate process dynamics before online fine-tuning, thereby reducing training risk and computational cost. The resulting policy, updated via a variance-reduced policy gradient method, is validated on a nonlinear multi-modal tank system, confirming its data efficiency and practical relevance.

Experimental progress in deep reinforcement learning (DRL) further extends the reach of RL to real-world applications. Lawrence et al. (2022) demonstrate that a shallow policy structure embedded in a modified twin delayed deep deterministic policy gradient (TD3) framework can tune PID parameters directly. Applied to a two-tank laboratory process, the study confirms that interpretable PID structures can be retained within DRL, bridging the gap between black-box learning and traditional control practice. Likewise, Aksland et al. (2022) recast PID tuning as a gradient-based constrained optimisation problem incorporating actuator and state limits, demonstrating scalability on a multivariable thermal management system with five coupled PI controllers. Their results show that gradient-based RL provides a systematic approach to anti-windup and constrained MIMO control.

Recent work introduces hybrid algorithms that enhance both exploration and convergence. Chowdhury and Lu (2023) propose the entropy-maximising twin delayed deep deterministic policy gradient (EMTD3) algorithm, which combines entropy-based stochastic exploration with deterministic exploitation. This hybrid approach mitigates local optima and accelerates convergence compared with conventional TD3 and DDPG. Similarly, van Niekerk et al. (2025b) employ PPO for automatic PID tuning in a multivariable nonlinear grinding mill circuit. By incorporating explicit safety constraints into the reward formulation, their method ensures closed-loop stability while matching or surpassing manual tuning performance. In another recent contribution, Bujgoi and Sendrescu (2025) apply TD3 to PID tuning for a nonlinear fed-batch bioreactor. The agent, trained to maximise tracking performance in simulation, achieves superior results to classical tuning rules, with twin critics effectively mitigating value overestimation.

Overall, the reviewed body of work confirms that RL provides a versatile and scalable framework for PID tuning across nonlinear, multivariable, and constrained systems. Approaches range from simple random search and contextual bandits to advanced actor-critic algorithms such as DDPG, TD3, and PPO, all capable of producing robust, well-tuned controllers without explicit process models. Crucially, these methods preserve the interpretability of PID structures while leveraging the optimisation strengths of modern RL, bridging classical control theory with data-driven intelligence. This synthesis of familiarity, adaptability, and performance positions RL-based PID tuning as a practical and forward-looking solution for next-generation industrial process control.

3.3.2. Supervised learning methods

SL learns explicit input-output mappings from labelled data and is therefore well suited to regression and classification tasks in process systems engineering. By exploiting historical or simulated datasets, SL enables accurate prediction and controller parameter inference without explicit process models, but remains fundamentally offline and distinct from decision-making approaches based on interaction with the environment (Lee et al., 2018; Shin et al., 2019).

Lee and Jang (2021) explore SL for automatic PID tuning of a second-order mass-spring-damper system by learning data-driven mappings between process responses and controller parameters. The authors propose a two-stage framework that employs sequential neural networks to estimate system dynamics from PID response data and then predict suitable controller gains for the identified model. Because the method relies on labelled datasets generated through simulation, it is classified as a supervised approach. Its key advantage is that it automates tuning while preserving the familiar PID structure widely used in industry.

Further expanding SL-based design, Wang et al. (2021b) develop a time-varying reservoir parameter echo state network (TVRP-ESN) for tuning PID parameters in computer numerical control (CNC) servo systems. By learning a mapping from system inputs, outputs, and errors to PID gains, the TVRP-ESN provides a fast and adaptive means of computing suitable parameters for discrete-time systems with delays. Complementing this, Lakshminarayanan et al. (2025) propose a direct inverse supervised learning (ISL) approach that eliminates intermediate modelling stages such as system identification or data compression. Their method directly learns a mapping from input-output trajectories to controller parameters, achieving more efficient and robust tuning than traditional identification or virtual reference feedback approaches.

SL has also been applied to multivariable control problems. Mumuni et al. (2025) design a MIMO PID controller with an integrated decoupler and formulate gain tuning as a supervised classification task for a generic, abstract MIMO plant. Their results show that incorporating machine-learning classifiers into the tuning process improves performance, yielding faster settling, enhanced stability, and more effective gain selection compared with conventional methods.

Overall, these studies underscore the growing relevance of supervised learning for automating and improving PID controller tuning. By leveraging labelled datasets derived from simulations, digital twins, or measured responses, SL approaches can reliably infer suitable controller parameters.

3.3.3. Unsupervised learning methods

UL comprises data-driven methods that infer structure, patterns, or representations directly from unlabelled data, without predefined input-output pairs. In process systems engineering, UL is commonly applied to tasks such as clustering, dimensionality reduction, and feature extraction, enabling the discovery of inherent process characteristics, operating regimes, or latent variables from historical measurements, while remaining independent of explicit performance objectives or decision-making policies (Lee et al., 2018; Shin et al., 2019).

Das et al. (2018) propose a robust PID tuning framework for second-order plus time-delay (SOPTD) processes by combining analytical dominant pole placement with data-driven clustering. Using Padé approximations, multiple algebraic expressions for PID gains are derived under different non-dominant pole assumptions, and Monte Carlo simulations are employed to sample stabilising solutions. To extract a single robust set of controller parameters, the authors apply the unsupervised k-means clustering algorithm to the stabilising gain sets, selecting the cluster centroid as the robust solution. This approach is considered unsupervised learning because the clustering operates on unlabeled stabilising data, identifying patterns in the parameter space without explicit optimisation or reward signals. Simulation results across nine benchmark SOPTD processes demonstrate that the clustering-based centroid method provides static PID parameters that deliver robust stability and acceptable performance against uncertainty, thereby offering a systematic, data-driven alternative to conventional tuning rules for time-delay systems

While clustering organises the solution space efficiently, it does not directly optimise an objective function. The result may be robust but not necessarily optimal for all operating conditions.

3.3.4. BO methods

BO is a sample-efficient, model-based optimisation framework designed for problems where objective evaluations are expensive, noisy, or experimentally constrained. By constructing a probabilistic surrogate model of the objective function and selecting new evaluation points via an acquisition function that balances exploration and exploitation, BO enables systematic and data-efficient tuning of controller parameters, making it particularly well suited to data-driven control and experimental PID tuning applications (Brochu et al., 2010; Fujimoto et al., 2023).

Neumann-Brosig et al. (2020) introduce a general BO-based framework and demonstrate its effectiveness through throttle valve control using active disturbance rejection. The objective function is modelled as a Gaussian process (GP) with a Matérn 5/2 kernel, while the expected improvement (EI) criterion serves as the acquisition function. Their results highlight the strong data efficiency and generalisability of BO across different controller structures and tuning objectives.

Building on this foundation, Chen et al. (2019) apply BO to tune a PID controller for servo motor position control. The GP employs a composite kernel comprising constant, squared exponential, and white noise components, and optimisation is guided by an upper confidence bound (UCB) acquisition function with a randomised exploration term. The approach proves effective in both simulation and hardware implementation, demonstrating consistent closed-loop improvements. Extending this idea, Roveda et al. (2020) integrate BO into a robotic control setting, simultaneously tuning both the link-mass parameters, representing system dynamics, and PID gains for trajectory tracking. Using a squared exponential kernel with EI as the acquisition function, their results show substantial reductions in tracking error relative to baseline controllers.

A constrained formulation of BO is adopted by Khosravi et al. (2022) for cascade PID tuning in CNC grinding machines. In this configuration, the outer-loop controller regulates position, while the inner loop ensures precise velocity tracking through a brushless permanent magnet ac motor. The objective function is modelled as a GP with a Matérn 3/2 kernel and optimised using EI, yielding a notable reduction in position error compared with the nominal controller. Similarly, Coutinho et al. (2023) benchmark BO-tuned diagonal PID controllers against the SIMC rules (Skogestad, 2003) for a three-loop evaporation system, demonstrating that BO provides a superior trade-off between setpoint tracking and actuator effort using a GP with a Matérn 5/2 kernel.

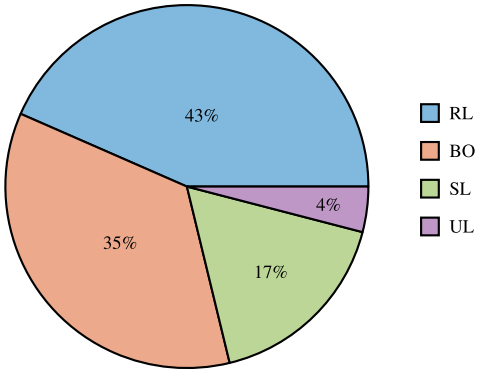
Further refinement of BO for practical applications is presented by Fujimoto et al. (2023), who accelerate convergence by incorporating a model-based prior mean derived from a linear servo motor model. This modification reduces exploration requirements compared with the common zero-mean assumption, allowing faster tuning when using a Gaussian kernel and UCB acquisition strategy. In the context of multivariable systems, van Niekerk et al. (2023) apply BO to tune diagonal PID controllers for a linear grinding mill model. Employing a Matérn 5/2 kernel and EI acquisition, the study demonstrates that BO can identify near-optimal tuning parameters within only a few iterations, confirming its potential for efficient online tuning.

Finally, in an advanced manufacturing context, Kavas et al. (2025) employ BO for laser power regulation in laser powder bed fusion (LPBF) additive manufacturing. By modelling the objective function with a radial basis function kernel and using a lower confidence bound (LCB) acquisition strategy, they achieve effective stabilisation of the melt pool, a process that traditionally operates in open loop, thereby mitigating overheating and improving product quality.

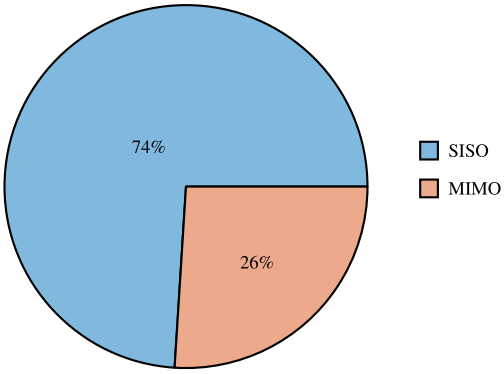
3.4. Machine learning review conclusions

The distribution of machine-learning-based autotuning algorithms presented in Fig. 3(a) shows that RL has become the dominant paradigm in this category, representing approximately 43% of reported studies. This strong presence reflects the growing shift towards data-driven control frameworks capable of learning optimal tuning behaviour directly through interaction with process models or real plants. BO follows closely, with 35% of studies, and has emerged as a complementary alternative where sample efficiency is prioritised.

SL and UL together account for less than one quarter of the studies, underscoring that while data-driven models have been successfully trained to infer PID parameters from labelled datasets, such approaches have yet to demonstrate the adaptability and autonomy achieved by RL and BO. SL offers rapid inference and low computational demand once trained; however, its reliance on labelled datasets and prior tuning knowledge limits its generality across different plant types. In contrast, RL learns directly from closed-loop interaction, enabling systematic exploration of the tuning space without requiring labelled data. Policy-gradient-based RL methods can infer performance trends from the effect of small parameter perturbations on the reward signal, making them better suited to data-efficient online tuning. Consequently, SL is excluded from the comparative analysis, as it lacks online adaptability and depends on extensive labelled datasets to identify near-optimal tuning configurations.



(a) Machine learning algorithm distribution.



(b) Distribution of machine-learning-based PID tuning studies by control structure (SISO vs MIMO).

Fig. 3. Distribution of machine learning fixed-parameter PID autotuning studies: (a) autotuning algorithms and (b) control-structure type.

UL methods remain largely exploratory in the context of PID tuning, with only a single paper applying clustering or representation learning to controller parameter selection. Because UL does not explicitly optimise a performance objective or interact dynamically with the control environment, its practical applicability to automatic PID tuning is currently limited. Consequently, UL is excluded from the comparative analysis, although future developments in self-supervised or hybrid approaches may enhance its relevance.

The control-structure distribution shown in Fig. 3(b) indicates that 74% of the machine-learning-based autotuning studies have been conducted on SISO systems, with only 26% addressing MIMO configurations. Coincidentally, the same ratio is observed for the nature-inspired structures. This imbalance underscores that practical applications to coupled process systems remain limited. MIMO control introduces challenges such as controller interactions, increased training complexity, priority weighting and the need for multi-objective reward or cost formulations.

3.5. AI-based PID tuning review conclusions

The literature demonstrates that AI provides an effective alternative for automatic PID tuning across a wide range of industrial applications. Compared with classical tuning rules, AI-based methods consistently achieve improved transient and steady-state performance and represent a shift toward optimisation-driven control frameworks with enhanced scalability and robustness.

Despite this progress, the literature lacks a systematic comparison of PID autotuning methods across different AI classes. This study

addresses this gap by evaluating representative algorithms from the nature-inspired and machine learning sub-classes against key attributes of an ideal automatic tuner. The following sections motivate the selection of the candidate algorithms considered in the comparative analysis.

3.5.1. Motivation of PSO

GAs have received significant attention for PID tuning within the nature-inspired subclass of AI algorithms; however, they are frequently criticised for relatively slow convergence (Wang et al., 2021a). By contrast, PSO often exhibits faster convergence and improved robustness, although it remains susceptible to premature convergence and entrapment in local optima (Elbeltagi et al., 2005; Valluru & Singh, 2018). In this paper, the tuning objective is strictly convex (Section 5.4), thereby eliminating concerns related to local optima. Unlike GAs, which rely on crossover and mutation, PSO employs a memory-based search mechanism in which each particle updates its trajectory using both individual and collective experience, allowing information about high-quality solutions to persist throughout the search. Under these conditions, PSO is well suited to controller tuning and is therefore selected as the representative nature-inspired algorithm in this study. For problems where convexity cannot be guaranteed, hybrid GA-PSO approaches offer an effective alternative by enhancing global exploration while mitigating premature convergence and improving local search performance (Araz, 2025).

3.5.2. Motivation of PPO

RL algorithms fall broadly into two categories: dynamic programming and policy optimisation. Dynamic programming approaches rely on estimating value functions before choosing actions, which becomes challenging in continuous action spaces due to the need to approximate infinitely many values. Policy optimisation, by contrast, directly adjusts the policy parameters using stochastic gradient ascent, moving in the direction that increases the expected return. This class includes policy gradient methods, which are particularly well suited for PID tuning because they operate naturally in continuous and high-dimensional parameter spaces (Sutton & Barto, 2018; Sutton et al., 1999).

Although effective, policy gradient methods can be unstable: small parameter changes may cause large swings in performance or even collapse the learned policy (Schulman et al., 2015). PPO mitigates this by constraining updates with a clipped surrogate objective (Schulman et al., 2017). PPO is selected as the representative RL algorithm in this paper due to its demonstrated effectiveness for multivariable PID tuning and its ability to learn directly from closed-loop interaction, as shown in van Niekerk et al. (2025b).

3.5.3. Motivation of BO

BO is a data-efficient approach for controller tuning that is well suited to problems where objective function evaluations are expensive. By modelling the tuning objective probabilistically using Gaussian processes and guiding the search through acquisition functions such as EI, UCB, and LCB, BO enables rapid convergence with minimal experimentation.

Its successful application to PID and controller tuning in domains such as servo drives, robotics, grinding mills, and additive manufacturing motivates its inclusion as the representative probabilistic, machine-learning-based optimisation method in this comparative study.

4. Preliminaries

This section presents a concise discussion of PSO, PPO, and BO, as motivated in the previous section, in preparation for their inclusion in the comparative analysis.

4.1. Particle swarm optimisation

PSO is a population-based metaheuristic introduced by Eberhart and Kennedy (1995), inspired by the collective behaviour of bird flocking and fish schooling. Belonging to the broader class of swarm intelligence methods, PSO has been successfully applied to optimisation tasks across many domains, including control engineering and PID tuning (Joseph et al., 2022). Olorunda and Engelbrecht (2008) attribute the success of PSO to keeping particles widely dispersed early (exploration) and progressively focusing near good solutions (exploitation).

A common formulation of PSO is the global best (**gbest**) version with inertia, in which every particle is influenced by the best solution discovered by the entire swarm (Khanduja & Bhushan, 2016; Nekoui et al., 2010). The inertia weight w multiplies the previous velocity, acting as a momentum term that smooths particle trajectories and regulates the exploration-exploitation balance: larger w encourages broad search, while smaller w damps motion for local refinement. In practice, a decaying schedule yields rapid early exploration with stable late convergence. This formulation is computationally simple, converges rapidly, and has been widely applied in control optimisation problems.

Let the objective function to be minimised be denoted by

$$J(\boldsymbol{\eta}), \quad \boldsymbol{\eta} \in \mathbb{R}^{3m}, \tag{6}$$

where $\boldsymbol{\eta}$ represents a candidate solution vector in a $3m$-dimensional decision space. The **gbest** PSO variant proceeds as shown in Algorithm 1.

Algorithm 1 **Gbest** PSO with inertia.

Require: Swarm size S, cognitive acceleration coefficient c_1, social acceleration coefficient c_2, inertia weight w

1: Initialise a swarm of S particles with random positions $\boldsymbol{\eta}_i^{(0)}$ and velocities $\mathbf{v}_i^{(0)}$, $i = 1, \dots, S$.
2: Evaluate $J(\boldsymbol{\eta}_i^{(0)})$ for each particle and set $\mathbf{pbest}_i = \boldsymbol{\eta}_i^{(0)}$
3: Identify the global best solution $\mathbf{gbest} = \arg\min_i J(\mathbf{pbest}_i)$
4: **while** termination criterion not met **do**
5: **for** each particle $i = 1, \dots, S$ **do**
6: Update velocity:

$$\mathbf{v}_i^{(t+1)} = w\,\mathbf{v}_i^{(t)} + c_1 r_1 \left(\mathbf{pbest}_i - \boldsymbol{\eta}_i^{(t)}\right) + c_2 r_2 \left(\mathbf{gbest} - \boldsymbol{\eta}_i^{(t)}\right) \tag{7}$$

where $r_1, r_2 \sim \mathcal{U}([0,1])$

7: Update position:

$$\boldsymbol{\eta}_i^{(t+1)} = \boldsymbol{\eta}_i^{(t)} + \mathbf{v}_i^{(t+1)} \tag{8}$$

8: Evaluate $J(\boldsymbol{\eta}_i^{(t+1)})$
9: Update $\mathbf{pbest}_i$ if $J(\boldsymbol{\eta}_i^{(t+1)}) < J(\mathbf{pbest}_i)$
10: **end for**
11: Update **gbest** as the best $\mathbf{pbest}_i$ across the swarm
12: **end while**

The PSO algorithm requires few parameters and uses only basic vector operations. This simplicity, combined with its memory-based search and capacity for rapid convergence, makes PSO a practical option for engineering optimisation problems such as automatic controller tuning.

4.2. Proximal policy optimisation

The PPO-Clip variant updates the policy by maximising the expected surrogate loss with respect to the parameter vector θ

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathbb{E}}_{s,a\sim\pi_{\theta_k}} \left[\mathcal{L}(s, a, \theta_k, \theta)\right]. \tag{9}$$

Here, θ_k denotes the parameters of the previous policy, θ represents the candidate parameters of the new policy, and θ_{k+1} is the updated parameter vector obtained after optimisation.

When the actor and critic are represented jointly in a neural network, the PPO loss function is

$$\mathcal{L}(s, a, \theta_k, \theta) = \mathcal{L}_{\text{CLIP}} - \mathcal{L}_{\text{VF}} + \mathcal{H}[\pi_\theta] \tag{10}$$

where $\mathcal{L}_{\text{CLIP}}$ is the clipped surrogate objective that stabilises learning, $\mathcal{L}_{\text{VF}}$ is a squared-error loss, and $\mathcal{H}[\pi_\theta]$ is an entropy bonus promoting exploration. The clipped surrogate objective is defined as

$$\mathcal{L}_{\text{CLIP}} = \frac{1}{M}\sum_{i=1}^{M}\left[\min\left(c_i(\boldsymbol{\theta})A_i^{\pi_{\theta_k}}, d_i(\boldsymbol{\theta})A_i^{\pi_{\theta_k}}\right)\right] \tag{11}$$

which is the summation over M mini-batches $\mathcal{B}$, where

$$c_i(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_i|\boldsymbol{s}_i)}{\pi_{\boldsymbol{\theta}_k}(\boldsymbol{a}_i|\boldsymbol{s}_i)} \tag{12}$$

is the importance sampling ratio of the ith sample in $\mathcal{B}$ and

$$d_i(\boldsymbol{\theta}) = \text{clip}(c_i(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon) \tag{13}$$

restricts policy updates to a trust region controlled by the hyperparameter ϵ. The importance sampling ratio exploits trajectories generated by the previous policy to provide an unbiased estimate of expectations under the new policy. PPO then clips this ratio, ensuring that policy updates remain small and stable.

To enhance the bias-variance trade-off of the advantage estimation, generalised advantage estimation (GAE) is employed (Schulman et al., 2017). The advantage at time t is computed as

$$A_t^{\pi_{\theta_k}} = (\gamma\lambda)^l\,\delta_{t+l}, \tag{14}$$

where γ is the discount factor, λ is the GAE factor controlling the exponential decay of past temporal-difference errors, l is a time-step offset from t, and δ_t denotes the one-step temporal-difference residual

$$\delta_t = r_{t+1} + \gamma V_{\boldsymbol{\phi}}(\boldsymbol{s}_{t+1}) - V_{\boldsymbol{\phi}}(\boldsymbol{s}_t). \tag{15}$$

The corresponding return is then estimated as

$$G_t = A_t^{\pi_{\theta_k}} + V_{\boldsymbol{\phi}_k}(\boldsymbol{s}_t), \tag{16}$$

where

$$V_{\boldsymbol{\phi}}(\boldsymbol{s}_t) = \mathbb{E}_{\pi_{\theta_k}}\left[\sum_{l=0}^{N-t-1}\gamma^l r_{t+l+1} \mid \boldsymbol{s}_t\right] \tag{17}$$

is the value function with parameter vector $\boldsymbol{\phi}$.

The critic parameters are trained by minimising the value function loss

$$\mathcal{L}_{VF}(\boldsymbol{\phi}) = \frac{1}{2M}\sum_{i=1}^{M}\left(G_i - V_{\boldsymbol{\phi}}(\boldsymbol{s}_i)\right)^2, \tag{18}$$

while exploration is encouraged through the entropy term

$$\mathcal{H}[\pi_\theta] = \frac{w}{2}\sum_{j=1}^{C}\ln\left(2\pi\, e\,\sigma_{ji}^2\right), \tag{19}$$

where C is the number of continuous action dimensions, w the entropy weight, and σ_{ji} is the standard deviation of action j.

In this paper, the actor and critic are implemented as separate neural networks. Their respective losses are

$$\mathcal{L}_{actor} = \mathcal{L}_{\text{CLIP}} + \mathcal{H}[\pi_\theta] \tag{20}$$

and

$$\mathcal{L}_{critic} = \mathcal{L}_{\text{VF}} \tag{21}$$

with the actor updating the policy and the critic evaluating state values to guide learning (Sutton & Barto, 2018).

The PPO-Clip variant, without deviation from Schulman et al. (2017) is implemented as shown in Algorithm 2.

Algorithm 2 PPO-Clip.

Require: γ, ϵ, w, M, GAE factor λ,
transitions N, epochs K_e, training episodes E,
actor learn rate α_{actor}, and critic learn rate α_{critic}

Initialise actor parameters $\boldsymbol{\theta}_0$ and critic parameters $\boldsymbol{\phi}_0$
for each episode $= 1, \dots, E$ **do** ▷ Training loop
  Collect a trajectory of N transitions with policy π_{θ_k} to obtain $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_{t+1}, \boldsymbol{s}_{t+1}, \dots, \boldsymbol{s}_{N-1}, \boldsymbol{a}_{N-1}, r_N, \boldsymbol{s}_N)$
  for $l = 0, \dots, N-t-1$ **do**
    $A_t^{\pi_{\theta_k}} = (\gamma\lambda)^l\,\delta_{t+l}$ ▷ Advantage estimate
    $G_t = A_t^{\pi_{\theta_k}} + V_{\boldsymbol{\phi}_k}(\boldsymbol{s}_t)$ ▷ Return
  end for
  for each epoch in K_e **do**
    Partition data into mini-batches $\mathcal{B}$ of size M
    for $i = 1$ **to** M **do**
      For $(\boldsymbol{s}_i, \boldsymbol{a}_i, A_i, G_i) \in \mathcal{B}$:
      Calc $c_i(\boldsymbol{\theta})$ (12) ▷ Importance sampling ratio
      Calc $d_i(\boldsymbol{\theta})$ (13) ▷ Apply clip
      Calc $\mathcal{L}_{actor}(\boldsymbol{\theta})$ (20) ▷ Update actor loss
      Calc $\mathcal{L}_{critic}(\boldsymbol{\phi})$ (21) ▷ Update critic loss
    Update parameter vectors:
      $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha_{actor}\nabla_{\boldsymbol{\theta}}L_{actor}(\boldsymbol{\theta})$
      $\boldsymbol{\phi}_{k+1} = \boldsymbol{\phi}_k - \alpha_{critic}\nabla_{\boldsymbol{\phi}}L_{critic}(\boldsymbol{\phi})$
    end for
  end for
end for

4.2.1. PPO actor neural network

The actor network defines a stochastic policy that maps the observed plant states to the parameters of a Gaussian action distribution. Within the PPO framework, this network must support continuous action spaces and allow stochastic exploration while maintaining differentiability with respect to the policy parameters. To achieve this, the actor outputs both the mean and standard deviation of a Gaussian distribution that represents the conditional probability of selecting an action given the current state, denoted by

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{a} \mid \boldsymbol{s}) = \mathcal{N}\big(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s}), \text{diag}\big(\boldsymbol{\sigma}_{\boldsymbol{\theta}}^2(\boldsymbol{s})\big)\big), \tag{22}$$

where $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{s})$ and $\boldsymbol{\sigma}_{\boldsymbol{\theta}}(\boldsymbol{s})$ are the state-dependent mean and standard deviation vectors predicted by the actor network.

The network is designed with a shared feature-extraction path followed by two output branches that separately estimate the mean and variance of the action distribution. The shared path extracts nonlinear features from the state observations, ensuring that the subsequent mean and variance predictions are conditioned on the same latent representation of the environment. The mean branch defines the deterministic component of the policy, corresponding to the most likely action under the current policy parameters, while the variance branch regulates the stochastic exploration behaviour during training.

The output of the mean branch is scaled using a bounded activation function (typically tanh) and then linearly transformed to match the domain of tuning parameters $\mathcal{A}$. The variance branch applies a softplus activation to ensure positive standard deviations, which are further scaled to maintain a consistent level of exploration noise across all action dimensions.

4.2.2. PPO critic neural network

The critic network approximates the state-value function $V_{\boldsymbol{\phi}}(\boldsymbol{s}_t)$, which represents the expected cumulative discounted reward obtained from state $\boldsymbol{s}_t$ under the current policy. Within the PPO framework, this value estimate serves as a baseline for computing the advantage function, thereby reducing variance in the policy-gradient estimate and improving training stability.

The critic must provide smooth, differentiable estimates of $V_{\boldsymbol{\phi}}(\boldsymbol{s})$ that generalise well across the state space. Its accuracy directly influences the quality of the advantage estimates, which in turn determine the direction and magnitude of the policy updates of the actor. Hence, the critic network is designed as a continuous, feed-forward function approximator with sufficient expressive capacity to model the nonlinear mapping from plant states to the expected return.

4.3. Bayesian optimisation

BO provides a systematic framework for optimising unknown black-box objective functions when evaluation is costly (Brochu et al., 2010). These costs may stem from computational load, production interruptions, the need for specialist expertise, or the capital required for experimental trials. BO has demonstrated strong performance across a wide spectrum of domains, from hyperparameter optimisation in machine learning, where it has matched or exceeded human expert-level tuning (Turner et al., 2021), to aerospace engineering applications where only a limited budget of evaluations is available (Lam et al., 2018). Furthermore, evidence from black-box optimisation competitions highlights its superiority over uninformed strategies such as random search when tuning high-dimensional parameter spaces (Turner et al., 2021).

The core strength of BO lies in its ability to combine prior information with data collected from previous evaluations through a probabilistic surrogate model (Wilson et al., 2018). This surrogate provides a computationally inexpensive approximation of the objective function, which is used in conjunction with an acquisition function to guide the search towards the global optimum. In contrast to brute-force strategies such as grid search or random search, which disregard past evaluations, BO systematically exploits historical performance data to make informed decisions about where next to sample. This process embodies its hallmark efficiency: extracting maximum information from each evaluation and thus reducing the number of costly objective function assessments.

In this paper, the surrogate model is represented by a GP, which provides both predictions for unseen inputs and a measure of uncertainty. Acquisition functions exploit these properties to determine the most informative next sample. GPs are attractive due to their relatively few training parameters, and although their computational cost grows cubically with the number of samples (Liu et al., 2013), the focus on minimising expensive evaluations makes this issue negligible.

4.3.1. Gaussian processes

A GP prior placed over the objective function to be minimised, $J(\boldsymbol{\eta})$, is given by

$$J(\boldsymbol{\eta}) \sim \mathcal{GP}\left(\mu(\boldsymbol{\eta}), k(\boldsymbol{\eta}, \boldsymbol{\eta}')\right), \tag{23}$$

where $\mu(\boldsymbol{\eta})$ denotes the mean function, which can be set to zero without loss of generality, and $k(\boldsymbol{\eta}, \boldsymbol{\eta}')$ is the covariance function (or kernel). The latter defines how function values at two input points, $\boldsymbol{\eta}$ and $\boldsymbol{\eta}'$, are correlated. Following the recommendation of Snoek et al. (2012), the automatic relevance determination (ARD) Matérn 5/2 kernel is employed in this study.

Function evaluations are generally corrupted by noise and are modelled as

$$\widehat{J} = J(\boldsymbol{\eta}) + \varepsilon, \tag{24}$$

where $\widehat{J}$ denotes the noisy observation and ε is additive Gaussian noise with zero mean and variance σ^2,

$$\varepsilon \sim \mathcal{N}(0, \sigma^2). \tag{25}$$

The training dataset $\mathcal{D}$, consisting of a budget of N noisy evaluations of $J(\boldsymbol{\eta})$, is therefore

$$\mathcal{D} = \{(\boldsymbol{\eta}_i, \widehat{J}_i)\}_{i=1}^{N}. \tag{26}$$

Under the GP prior, the observed values and the predicted (unknown) function values follow a joint Gaussian distribution

$$\begin{bmatrix} \widehat{\boldsymbol{J}} \\ \boldsymbol{J}_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} k(\boldsymbol{A}, \boldsymbol{A}) + \sigma_N^2 I & k(\boldsymbol{A}, \boldsymbol{A}_*) \\ k(\boldsymbol{A}_*, \boldsymbol{A}) & k(\boldsymbol{A}_*, \boldsymbol{A}_*) \end{bmatrix}\right), \tag{27}$$

where $\boldsymbol{A}$ is the design matrix of inputs $\boldsymbol{\eta}_i$, $\widehat{\boldsymbol{J}}$ is the vector of observations, $\boldsymbol{J}_*$ are the predictions at test inputs $\boldsymbol{A}_*$, and $k(\cdot,\cdot)$ denotes the covariance function.

Conditioning on the training data yields the predictive distribution

$$\boldsymbol{J}_* \mid \boldsymbol{A}, \widehat{\boldsymbol{J}}, \boldsymbol{A}_* \sim \mathcal{N}(\bar{\boldsymbol{J}}_*, \mathrm{cov}(\boldsymbol{J}_*)), \tag{28}$$

with mean and covariance

$$\bar{\boldsymbol{J}}_* = \boldsymbol{k}_*^\top [\boldsymbol{k} + \sigma_N^2 I]^{-1} \widehat{\boldsymbol{J}}, \tag{29a}$$

$$\mathrm{cov}(\boldsymbol{J}_*) = \boldsymbol{k}_{**} - \boldsymbol{k}_*^\top [\boldsymbol{k} + \sigma_N^2 I]^{-1} \boldsymbol{k}_*, \tag{29b}$$

where $\boldsymbol{k} = k(\boldsymbol{A}, \boldsymbol{A})$, $\boldsymbol{k}_* = k(\boldsymbol{A}, \boldsymbol{A}_*)$, and $\boldsymbol{k}_{**} = k(\boldsymbol{A}_*, \boldsymbol{A}_*)$. The mean $\bar{\boldsymbol{J}}_*$ represents the GP prediction of the objective function, while the predictive variance $\sigma^2(\boldsymbol{\eta})$ is obtained from the diagonal entries of $\mathrm{cov}(\boldsymbol{J}_*)$, quantifying the associated predictive uncertainty.

The BO algorithm with GP surrogate is implemented as shown in Algorithm 3.

Algorithm 3 BO with GP surrogate.

Require: Search domain $\mathcal{A}$, kernel $k(\cdot,\cdot)$, acquisition $EI(\boldsymbol{\eta} \mid \mathcal{D})$, initial design size n_0, budget N

Initialise: Draw n_0 points $\{\boldsymbol{\eta}_i\}_{i=1}^{n_0}$ in $\mathcal{A}$ using random sampling
for $i = 1, \ldots, n_0$ **do**
  $\widehat{J}_i = J(\boldsymbol{\eta}_i) + \varepsilon_i$ ▷ Evaluate objective (noisy)
end for
$\mathcal{D} = \{(\boldsymbol{\eta}_i, \widehat{J}_i)\}_{i=1}^{n_0}$ ▷ Training data
Fit GP hyperparameters $\boldsymbol{\theta}$ by maximising the log marginal likelihood on $\mathcal{D}$
for $i = n_0 + 1, \ldots, N$ **do**
  Build GP posterior $p(J(\cdot) \mid \mathcal{D})$ with mean $\mu(\boldsymbol{\eta})$ and standard deviation $\sigma(\boldsymbol{\eta})$
  $\boldsymbol{\eta}_* = \arg\max_{\boldsymbol{\eta} \in \mathcal{A}} EI(\boldsymbol{\eta} \mid \mathcal{D})$ ▷ Acquisition maximisation
  $\widehat{J}_* = J(\boldsymbol{\eta}_*) + \varepsilon$ ▷ Query objective at next point
  $\mathcal{D} = \mathcal{D} \cup \{(\boldsymbol{\eta}_*, \widehat{J}_*)\}$ ▷ Augment dataset
  Re-fit GP hyperparameters $\boldsymbol{\theta}$ on $\mathcal{D}$.
end for
Return: $\boldsymbol{\eta}_{\min} = \arg\min_{(\boldsymbol{\eta}_i, \widehat{J}_i) \in \mathcal{D}} \widehat{J}_i$ and the final GP posterior

4.3.2. Acquisition function

In BO, acquisition functions guide the selection of new sampling points by exploiting the predictive mean and variance of the surrogate model. Their purpose is not to reconstruct the objective function in its entirety, but rather to identify its global extremum within the feasible domain (Shahriari et al., 2015). A key property of acquisition functions is the balance between exploration and exploitation. Sampling in regions where the predicted mean and standard deviation is low exploits current knowledge, whereas sampling in regions of high uncertainty promotes exploration (Shahriari et al., 2015).

Among the commonly used acquisition functions for Gaussian processes are probability of improvement, Gaussian process upper confidence bound, and expected improvement (EI) (Snoek et al., 2012). This work employs EI (Mockus, 1975), as it is more robust than probability of improvement, avoids extra tuning parameters required by the upper confidence bound, and has been shown to escape local minima effectively (Emmerich et al., 2006).

The EI acquisition function measures the expected reduction below the current best observed objective value. It is defined as

$$EI(\boldsymbol{\eta}_*) = \mathbb{E}\left[\max\left(0, \widehat{J}_{\min} - J_*\right)\right], \tag{30}$$

where $\widehat{J}_{\min}$ is the best (minimum) observed noisy objective value and J_* incorporates both the predictive mean and the uncertainty.

The next evaluation point is determined by maximising the acquisition function

$$\boldsymbol{\eta}_* = \mathrm{argmax}_{\boldsymbol{\eta} \in \mathcal{A}} EI(\boldsymbol{\eta} | \mathcal{D}), \tag{31}$$

where $\boldsymbol{\eta}_*$ denotes the next candidate input. Since the EI is differentiable, it can be efficiently maximised using gradient-based methods.

When the posterior is Gaussian, EI has the closed-form solution (Jones et al., 1998)

$$EI(\boldsymbol{\eta}) = \begin{cases} \Delta(\boldsymbol{\eta})\,\psi\,Z(\boldsymbol{\eta}) + \sigma(\boldsymbol{\eta})\,\phi\,Z(\boldsymbol{\eta}), & \sigma(\boldsymbol{\eta}) > 0 \\ 0, & \sigma(\boldsymbol{\eta}) = 0 \end{cases} \tag{32}$$

with

$$\Delta(\boldsymbol{\eta}) = \hat{J}_{\min} - \bar{\boldsymbol{J}}_*(\boldsymbol{\eta}) - \xi, \tag{33a}$$

$$Z(\boldsymbol{\eta}) = \Delta(\boldsymbol{\eta})/\sigma(\boldsymbol{\eta}), \tag{33b}$$

Here, $\sigma(\boldsymbol{\eta})$ is the predictive standard deviation, ξ is the exploration offset, and ϕ and ψ are the probability density and cumulative distribution functions of the standard normal distribution.

The EI algorithm is shown in Algorithm 4.

Algorithm 4 EI acquisition function.

Require: GP posterior mean $\bar{\boldsymbol{J}}_*(\boldsymbol{\eta})$, standard deviation $\sigma(\boldsymbol{\eta})$; best observed value $\hat{J}_{\min}$; exploration offset $\xi \geq 0$

```
function EI(η)
    if σ(η) = 0 then
        return 0
    else
        EI(η) = Δ(η) ψ Z(η) + σ(η) φ Z(η)
        return EI(η)
    end if
end function
```

5. Methodology for AI-based PID tuning

This section presents a generalised methodology for AI-based PID tuning, which is subsequently implemented, for illustrative purposes, on an ore-milling circuit in Section 6. The methodology is founded on a two-stage tuning procedure. In the first stage, pre-tuning is performed on a linearised plant model, enabling the systematic selection of tuning and algorithm hyperparameters in an environment unconstrained by production priorities. In the second stage, the resulting controller parameters and hyperparameters are used to initialise the final training on the actual plant, represented in this paper by a validated nonlinear model, thereby addressing the model mismatch between the linear model and the plant. This staged approach reduces the duration of the final training phase while mitigating the operational risk associated with direct tuning on the plant.

Prerequisites for implementing the methodology are that step tests can be conducted on the plant to obtain a linear model and to facilitate AI-based tuning, that the controller is parameterisable by a fixed set of tuning parameters, and that the desired control performance can be expressed through an objective function.

5.1. General AI tuning framework

A general framework for AI-based controller tuning is illustrated in Fig. 4. At the base layer of this framework is the plant, controlled by a controller $\boldsymbol{K}$ in a unity feedback configuration. During pre-tuning, the plant is represented by a linear model in simulation, while final tuning is conducted on the actual plant if available; otherwise, the plant is approximated by a nonlinear model, such as (1), in simulation. The tracking error vector $\boldsymbol{e}(t)$, defined as the difference between the setpoint vector $\boldsymbol{y}^{sp}(t)$ and the controlled variable vector $\boldsymbol{y}(t)$, serves as an input to the controller. The manipulated variables, $\boldsymbol{u}(t)$, which depend on $\boldsymbol{e}(t)$ and the tuning parameters $\boldsymbol{\eta}$, actuate the plant with the objective of driving $\boldsymbol{e}(t)$ towards zero.

The AI-tuner operates as a temporary add-on to the baseline controller (highlighted in blue in Fig. 4) and is activated by an operator when poor performance is observed. It may consist of a RL agent, a nature-inspired optimiser, or a Bayesian optimiser. The AI-tuner receives

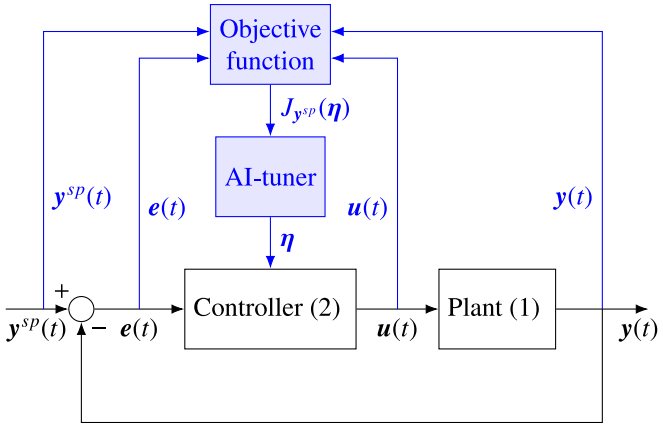


Fig. 4. Generalised framework for AI-based controller tuning with objective function $J_{y^{sp}}(\boldsymbol{\eta})$. The temporary tuning layer, comprising the AI-tuner and objective function, is shown in blue. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the objective function $J_{y^{sp}}(\boldsymbol{\eta})$ as input and produces the tuning parameter vector $\boldsymbol{\eta}$, as an output to $\boldsymbol{K}$. The objective function, $J_{y^{sp}}(\boldsymbol{\eta})$, is chosen arbitrarily but typically depends on $\boldsymbol{e}(t)$, $\boldsymbol{u}(t)$, and $\boldsymbol{y}(t)$, in line with the desired performance criteria.

The setpoint $\boldsymbol{y}^{sp}$ provides the context for evaluating the objective function on the nonlinear model or plant; hence, $J_{y^{sp}}(\boldsymbol{\eta})$ is conditional on the chosen setpoint. For different setpoints, the same controller parameters $\boldsymbol{\eta}$ may yield different values of $J_{y^{sp}}(\boldsymbol{\eta})$ on the nonlinear plant model. Consequently, if the plant is required to operate at multiple operating points, tuning must be performed at each operating point to obtain $J_{y^{sp}}(\boldsymbol{\eta})$ values relative to the respective $\boldsymbol{y}^{sp}$. For linear plant models, the "sp" superscript is redundant.

A single tuning iteration consists of the AI-tuner observing the value of $J_{y^{sp}}(\boldsymbol{\eta})$ and mapping the candidate parameters $\boldsymbol{\eta}$ to the controller. The setpoint $\boldsymbol{y}^{sp}$ is stepped from the nominal operating condition to a new operating point, with all vector elements updated simultaneously. The controller responds by adjusting $\boldsymbol{u}(t)$ to reduce the resulting error $\boldsymbol{e}(t)$ in accordance with the current tuning parameters $\boldsymbol{\eta}$. The trajectories of $\boldsymbol{e}(t)$, $\boldsymbol{u}(t)$, and $\boldsymbol{y}(t)$ are monitored over the duration of the tuning iteration and assessed using $J_{y^{sp}}(\boldsymbol{\eta})$. The scalar outcome of this evaluation is then observed by the AI-tuner to inform the next iteration. Tuning continues until the required number of iterations has been completed or until a stopping criterion is satisfied.

5.2. General AI tuning procedure

The proposed AI tuning procedure is a two-step process consisting of pre-tuning and final tuning. Pre-tuning is performed offline on a linear plant model with the objective of identifying optimal tuning parameters, or, in the case of RL, training an agent, that can be used to seed the final tuning stage. This seeding provides the AI-tuner with a head start, thereby reducing the number of tuning iterations required on the actual plant, where step testing may be expensive.

A prerequisite for pre-tuning is the availability of a linear plant model. The linear model serves multiple purposes, including the selection of the controller structure. Specifically, the plant transfer-function matrix determines the pairing of manipulated and controlled variables, the arrangement of the controller matrix according to the number of variables, and the choice of controller type (P, PI, or PID) based on the characteristics of the transfer functions. The linear model is also essential for conducting μ-analysis, which establishes the parameter thresholds beyond which closed-loop instability occurs. Finally, the linear model is used to determine the Pareto front, from which the weighting matrices are derived.

Once the controller structure, parameter constraints, and the objective function with its weighting matrices have been defined, pre-tuning can begin. As pre-tuning incurs only computational cost, the optimisation may proceed until no further improvement in $J_{y^{sp}}(\boldsymbol{\eta})$ is observed. Under a strictly convex objective function, this condition corresponds to convergence to the global optimum of the linear model.

Upon completion of pre-tuning, the parameters associated with the global optimum, or the pre-trained agent in the case of RL, are used to initialise the final tuning stage, ensuring that the process begins at the pre-tuned solution. Owing to the inevitable mismatch between the linear model and the actual plant, final tuning is necessary to account for these discrepancies. However, tuning on the actual plant is costly, primarily because process interruptions reduce production rates and product quality. Consequently, final tuning cannot continue until absolute convergence is reached, but rather only until an acceptable level of improvement has been achieved or the allocated budget of tuning iterations has been exhausted.

5.3. Safety constraints

Safe reinforcement learning (Safe RL) is concerned with training agents that not only maximise long-term returns but also satisfy safety constraints during learning and deployment (Garcia & Fernández, 2015). Early surveys by Garcia and Fernández (2015) emphasised two main strategies: modifying the optimisation criterion and altering the exploration process. The first includes worst-case optimisation, risk-sensitive objectives, and constrained formulations that penalise unsafe behaviour. The second relies on incorporating prior knowledge, teacher demonstrations, or risk metrics to guide exploration, thereby reducing the likelihood of catastrophic states in domains where unsafe actions can be costly.

More recent work by Gu et al. (2024) has expanded these foundations into a structured framework addressing five core challenges: safety policy, safety complexity, safety applications, safety benchmarks, and safety challenges. Safe RL methods are broadly divided into model-based and model-free categories. Model-based approaches such as Lyapunov-based methods, control barrier functions (CBFs), and GPs offer stronger safety guarantees but depend on accurate models. Model-free approaches, including constrained policy optimisation (CPO), projection-based CPO, safety critics, and formal verification layers, are more flexible in high-dimensional settings but face difficulties in ensuring convergence and maintaining guarantees.

As both reviews conclude, adoption of RL by industry will depend on developing unified frameworks that integrate robust control theory, probabilistic reasoning, and scalable learning algorithms to enable trustworthy deployment of RL in real-world, safety-critical applications

Safe Bayesian optimisation (Safe BO) extends standard BO by embedding safety guarantees into the search process, thereby preventing the selection of unsafe parameters during learning. The central approach relies on GP surrogates to model both performance and safety constraints, enabling algorithms to balance exploration and exploitation while respecting operational limits. Pioneering contributions, such as SafeOpt (Berkenkamp et al., 2016), formalised probabilistic guarantees that all candidate evaluations remain within the safe set, laying the foundation for safety-critical applications.

Applications in control have shown that Safe BO is particularly effective for automatic controller tuning, where unsafe parameters may induce instability or excessive overshoot. For example, in combustion engine calibration, Safe BO has been proposed to tune PI controllers directly on test vehicles. These approaches learn loss function models through closed-loop measurements and restrict parameter exploration to safe regions, ensuring that unsafe dynamics are never tested on physical systems (Schillinger et al., 2017). Similarly, in quadrotor and cascade PID control, constrained BO frameworks introduce barrier-like penalties and data-driven stability constraints that enforce safe operation while converging rapidly to high-performance solutions (Khosravi et al., 2022).

Safe AI-based controller tuning in industrial plants requires the exclusion of parameters that would lead to closed-loop instability. Although such instability may not necessarily cause equipment damage or harm to personnel due to the presence of safety instrumented systems in high-risk scenarios (Seborg et al., 2011), the oscillations resulting from instability can still cause production delays through trips triggered by interlock violations and degrade product quality through deviations from setpoints. To enable safe AI-based controller tuning in industrial applications in general, the model-based approach proposed by van Niekerk et al. (2025b) is adopted. In this framework, robust stability analysis, specifically μ-analysis, is applied to determine a safe search space, $\mathcal{A}_{safe}$, from which tuning parameters can be sampled. This approach holds the plant parameters fixed and quantifies the maximum structured uncertainty in the tuning parameters that can be tolerated before stability is lost. Its main limitation, namely that μ-analysis is conducted on a linear plant model, can be mitigated provided that automated tuning occurs within the same linear region of the plant from which the model was derived.

The safe search space, $\mathcal{A}_{safe}$, required for safe AI-based controller tuning is defined as

$$\boldsymbol{\eta} \in \mathcal{A}_{safe} \subseteq \mathcal{A} \in \mathbb{R}^{2\times m} \tag{34}$$

and accordingly (5) is revised as

$$\min_{\boldsymbol{\eta}\in\mathcal{A}_{safe}} J_{y^{sp}}(\boldsymbol{\eta}). \tag{35}$$

5.4. Objective function

In AI-based controller tuning, the objective function plays a pivotal role, as it defines the performance criteria against which candidate controllers are evaluated. With the exception of UL approaches, all reviewed AI-based tuning methods share the common goal of optimising this objective function. While the optimisation algorithm determines how efficiently and accurately the optimum is located, it is the formulation of the objective function itself that ultimately dictates the quality of the resulting controller.

For the task of tuning diagonal PID controllers in a MIMO loop, Wang and Ricardez-Sandoval (2024) employ the inverted deep deterministic policy gradient (IDDPG) algorithm with a reward based on a weighted ITAE function. The case study involves the control of a CSTR subject to time-varying uncertainty, where the effluent concentration is paired with the coolant flow rate, and the reactor hold-up is paired with the outlet flow rate. The reward function is defined as

$$R_e(\boldsymbol{s},\boldsymbol{a}) = -\sum_{i=1}^{p} \omega_i \int_t^T t\,|e_i(t)|\,dt, \tag{36}$$

where R_e is the episodic reward, p is the number of loops in the system, ω_i is the penalty weight of the i^{th} loop, e_i is tracking error and T is the episode duration. The implementation of IDDPG in conjunction with the reward defined in (36) is shown to improve system performance.

dos Santos Coelho and Mariani (2012) tune multiloop PID controllers of the Wood-Berry distillation column and a polymerisation reactor, which are both MIMO systems, using the CFA. The cost function adopted to provide the required controller performance is given by

$$J(k) = \sum_{i=1}^{p}\sum_{k=1}^{T} k_i\,|e_i(k)| \tag{37}$$

which represents a discrete formulation of the ITAE performance metric, where $J(k)$ denotes the cost function evaluated at step k. Reward or cost formulations, such as (36) and (37) have been shown to improve closed-loop performance; however, the omission of actuator-usage penalties may result in excessive control effort and premature actuator wear or failure.

Mate et al. (2023) tune and supervise adaptive diagonal PID controllers in a MIMO system using deep reinforcement learning (DRL). The immediate reward for tuning the PID controllers of a nonlinear, quadruple-tank system is defined as

$$R(k) = \sum_{i=1}^{p} \alpha_i^{sp} R_i^{sp}(k) + \sum_{i=1}^{p} \alpha_i^{u} R_i^{u}(k), \tag{38}$$

where

$$R_i^{sp}(k) = \exp\left(-0.5\left(\frac{e_i(k)}{\beta_i\, y_i^{sp}(k)}\right)^2\right) \tag{39}$$

and

$$R_i^{u}(k) = \exp\left(-0.5\left(\frac{\Delta u_i(k)}{u_{\max,i}}\right)^2\right). \tag{40}$$

Here, $R(k)$ denotes the immediate reward at step k, R_i^{sp} is the setpoint reward of the i^{th} control loop, and β_i is a penalty factor applied to the tracking error e_i. The term R_i^{u} represents the control reward, $u_{\max,i}$ is the upper bound of the manipulated variable $u_i(k)$, and Δu_i denotes the change in the controller output between two consecutive steps, defined as total variation (TV), serving as a measure of control effort. The weighting factors α_i^{sp} and α_i^{u} impose additional penalties on the setpoint and actuator rewards, respectively. Although reward (38) is formulated as an immediate reward, it can be extended to an episodic reward by integrating the immediate rewards over the duration of an episode.

Coutinho et al. (2023) make use of BO to tune diagonal PID controllers for a nonlinear evaporator. The performance of each loop is evaluated using the IAE to quantify tracking accuracy, and TV to measure actuator effort. The discrete expressions of these indices are

$$J^{IAE} = \sum_{k=1}^{T} |e(k)|\,\Delta t \tag{41}$$

$$J^{TV} = \sum_{k=1}^{T} |\Delta u(k)| \tag{42}$$

where Δt denotes the sampling period. Since accurate tracking and smooth control are conflicting objectives, selecting the controller parameters constitutes a multi-objective optimisation problem. This is addressed by defining the cost function as a weighted sum

$$J = \sum_{i=1}^{p} \left(\alpha_i J_i^{IAE} + \beta_i J_i^{TV}\right) \tag{43}$$

where α and β are weights that determine the trade-off between setpoint tracking and actuator usage in each loop, as well as the relative importance of the loops.

Kavas et al. (2025) employ the following Euclidean cost function to tune a BO-based PID controller for a laser powder bed fusion application

$$J = \sqrt{J^{MSE^2} + t_{\text{rise}}^2 + \sigma_{\text{laser}}^2}. \tag{44}$$

Here, the mean squared error J^{MSE} penalises deviations from the setpoint, t_{rise} rewards faster rise times, and the standard deviation σ_{laser} penalises fluctuations in the laser power signal. A high standard deviation reflects large or frequent controller-induced variations in the laser power. By incorporating this term, the optimisation not only improves tracking performance and response speed but also encourages stable, smooth, and practical actuator behaviour. Minimising the standard deviation of the controller output serves as an alternative to the TV performance criterion used in (43).

The literature references reveal that objective functions are arbitrarily chosen by control engineers from a wide range of available performance indices. This choice is critical, as the selected objective function, not the optimisation method, sets the upper bound on achievable controller performance. Optimisation algorithms, whether nature-inspired heuristics or machine learning-based methods, merely provide mechanisms to approach the global optimum of the defined function.

Consequently, comparisons between controller tuning algorithms should focus on their efficiency and reliability in locating the global optimum of the objective function. The closed-loop performance of the controller is determined by the objective function rather than the optimisation algorithm. When benchmarked against conventional methods such as ZN tuning, it is not the optimiser itself that inherently outperforms the baseline, but the underlying objective function which enables superior control performance.

Similar to the objective function proposed by Coutinho et al. (2023), which account for both error tracking and actuator usage, the objective function adopted in this paper is defined as

$$J_{\boldsymbol{y}^{sp}}(\boldsymbol{\eta}) = \sum_{t=0}^{T} \left(\boldsymbol{e}_t^{\top} \boldsymbol{\Omega}_e \boldsymbol{e}_t + \Delta \boldsymbol{u}_t^{\top} \boldsymbol{\Omega}_u \Delta \boldsymbol{u}_t\right) \tag{45}$$

where the sampling interval t must be selected to adequately capture the dynamics of the plant (1), and the total evaluation horizon is T, which must be set at least to equal the settling times of the observed step responses. Here, $\boldsymbol{\Omega}_e$ and $\boldsymbol{\Omega}_u$ are diagonal weighting matrices applied to the tracking error vector and actuator TV vector, respectively.

$J_{\boldsymbol{y}^{sp}}(\boldsymbol{\eta})$ is a multi-objective quadratic cost function which is strictly convex, provided that the weighting matrix on the actuator TV is positive definite ($\boldsymbol{\Omega}_u > 0$) and the weighting matrix on the tracking error, is positive semidefinite ($\boldsymbol{\Omega}_e \geq 0$). Strict convexity ensures that the cost function has a unique global optimum, but does not by itself guarantee closed-loop stability, which must be verified separately (Perez-Ramirez et al., 2025; Rawlings et al., 2017). Quadratic terms naturally penalise large deviations more heavily, $\boldsymbol{\Omega}_e$ penalises deviations from setpoints, $\boldsymbol{\Omega}_u$ penalises excessive actuator usage. This provides a balanced trade-off between tracking performance and control effort.

5.5. Weighting matrices

Controller tuning often involves conflicting objectives that cannot be optimised simultaneously. In the milling circuit, the multi-objective function (45) embodies several such conflicts: prioritising one controlled variable can increase interactions with others; performance must be balanced against actuator usage; and favouring one manipulated variable may come at the expense of the rest. Traditional approaches collapse these objectives into a single scalar cost via *a priori* weighting, but this is inherently subjective and empirical, and it risks biasing the optimisation away from promising solutions.

The Pareto front provides a rigorous alternative by characterising the set of nondominated solutions, where no objective can be improved without sacrificing at least one other. This allows control system designers to first explore the boundary of achievable trade-offs, and only then apply weights according to operational or economic priorities.

Recent studies demonstrate the versatility of Pareto-based approaches across diverse control and optimisation domains. In advanced control design, Kumar et al. (2020) apply multi-objective evolutionary algorithms to tune linear quadratic Gaussian/loop transfer recovery (LQG/LTR) controllers, generating Pareto fronts that reveal the trade-offs between robustness recovery, control effort, and closed-loop performance. In chemical process industries, Behroozsarand and Shafiei (2010) optimise the control of an amine-based natural gas sweetening plant using non-dominated sorting genetic algorithm (NSGA-II), obtaining Pareto fronts that balance overshoot against the IAE, thereby guiding the tuning of multiple PID loops.

In offshore engineering, Yin et al. (2025) combines deep learning surrogate modelling with BO to derive Pareto fronts for ultra-deepwater semi-submersible mooring systems, explicitly capturing trade-offs between platform offset, mooring line tension, and cost. Similarly, in carbon dioxide capture and amine plant operation, multi-objective optimisation with NSGA-II has been used to reduce regeneration energy while

maintaining capture efficiency, offering clear insights into the conflicting nature of energy use and separation performance (Behroozsarand & Shafiei, 2010).

Combined, these works show how Pareto-based methods enhance weighting strategies: instead of fixing subjective weights at the outset, the Pareto front enables post-optimisation weighting, where decision-makers can select solutions aligned with contextual priorities. This two-stage approach, front discovery followed by preference articulation, ensures transparency, adaptability through flexible weighting strategies, and robustness in complex multi-objective optimisation problems.

The multi-objective optimisation problem is

$$\min_{\boldsymbol{\eta}\in\mathcal{A}_{safe}} \boldsymbol{J}(\boldsymbol{\eta}) = \big(J_1(\boldsymbol{\eta}),\, J_2(\boldsymbol{\eta}),\, \ldots,\, J_n(\boldsymbol{\eta})\big) \tag{46}$$

where $\boldsymbol{J}(\boldsymbol{\eta})$ is the vector of n objective functions and each $J_i(\boldsymbol{\eta})$ measures a chosen performance criterion. Here, the dependence on the setpoint (denoted previously by the subscript $\boldsymbol{y}^{sp}$) is omitted from $\boldsymbol{J}(\boldsymbol{\eta})$ because the Pareto front is computed using a linear plant model obtained at the target operating point. Once linearised, and working in deviation variables, the step responses are setpoint-independent due to the linearity of the transfer function.

The set $\mathcal{P}$ is the Pareto-optimal set within the domain of safe tuning parameters. Formally

$$\mathcal{P} = \Big\{\, \boldsymbol{\eta}\in\mathcal{A}_{safe} \;\Big|\; \nexists\, \boldsymbol{\eta}'\in\mathcal{A}_{safe} \text{ such that } \boldsymbol{J}(\boldsymbol{\eta}')\prec\boldsymbol{J}(\boldsymbol{\eta}) \,\Big\}, \tag{47}$$

where $\boldsymbol{J}(\boldsymbol{\eta}')\prec\boldsymbol{J}(\boldsymbol{\eta})$ means that $\boldsymbol{\eta}'$ *dominates* $\boldsymbol{\eta}$

$$\begin{cases} J_i(\boldsymbol{\eta}')\le J_i(\boldsymbol{\eta}) & \text{for all } i,\\ J_j(\boldsymbol{\eta}')< J_j(\boldsymbol{\eta}) & \text{for at least one } j. \end{cases} \tag{48}$$

Intuitively, $\mathcal{P}$ contains all tuning parameters for which there exists no alternative that improves at least one objective without worsening another.

The Pareto front $\mathcal{F}$ is the image of the Pareto-optimal parameters in objective space:

$$\mathcal{F} = \{\, \boldsymbol{J}(\boldsymbol{\eta}) \mid \boldsymbol{\eta}\in\mathcal{P} \,\}. \tag{49}$$

Thus, $\mathcal{F}$ is the Pareto front in objective space – the visible curve (for two objectives), surface (for three objectives), or hypersurface (for $n>3$) of non-dominated compromises.

To obtain weights from the local gradients at the selected compromise on $\mathcal{F}$, the objectives must be normalised to ensure that compromises reflect preference rather than engineering units or numerical ranges. The normalisation is computed from the minimum and maximum values of each objective

$$\hat{J}_i = \frac{J_i - J_{i,\min}}{J_{i,\max} - J_{i,\min}}, \tag{50}$$

so that the normalised objectives $\hat{J}_i$ are dimensionless and lie in the interval $[0,1]$.

For problems with two objectives, the local gradient of $\mathcal{F}$ at the given compromise solution $\eta^\star$ directly relates to the ratio of weights in an equivalent weighted-sum scalarisation. In essence, the marginal rate of trade-off (the slope of the Pareto curve) indicates how many units of one objective must be sacrificed for a unit gain in the other - and this trade-off equals the ratio of optimal weights (Das & Dennis, 1997). The trade-off rate between the normalised objectives $\hat{J}_1$ and $\hat{J}_2$ is given by

$$m \;\approx\; \frac{\partial\hat{J}_2}{\partial\hat{J}_1}\Big|_{\eta^\star} \;\approx\; \frac{\hat{J}_2(\eta_{k+1})-\hat{J}_2(\eta_{k-1})}{\hat{J}_1(\eta_{k+1})-\hat{J}_1(\eta_{k-1})}, \tag{51}$$

where $\{\eta_{k-1},\,\eta_k=\eta^\star,\,\eta_{k+1}\}$ are neighbouring Pareto points ordered along the front. With a weighted-sum scalariser $J_w = w_1\hat{J}_1 + w_2\hat{J}_2$ subject to $w_1, w_2 \ge 0$ and $w_1 + w_2 = 1$, the weights are chosen to match the local tangent

$$\frac{w_1}{w_2} \;\approx\; \left|\frac{\partial\hat{J}_2}{\partial\hat{J}_1}\Big|_{\eta^\star}\right| = |m|, \tag{52}$$

Table 1
Process variable descriptions, constraints and nominal operating values (van Niekerk et al., 2025b).

Manipulated variables		Unit	Min	Max	Nominal
u_{MIW}	Flow-rate of water to the mill	m^3/h	–	–	4.17
u_{MFO}	Feedrate of ore to the mill	t/h	0	100	66.9
u_{MFB}	Feedrate of steel balls to the mill	t/h	–	–	6.43
u_{SFW}	Flow-rate of water to the sump	m^3/h	0	150	67.1
u_{CFF}	Flow-rate of slurry to the hydrocyclone	m^3/h	100	500	267
Controlled variables					
y_{JT}	Fraction of the mill filled	–	0.25	0.45	0.31
y_{SVOL}	Volume of slurry in the sump	m^3	1.0	8.0	5.90
y_{PSE}	Fraction of particles within specification	–	0.5	0.8	0.60

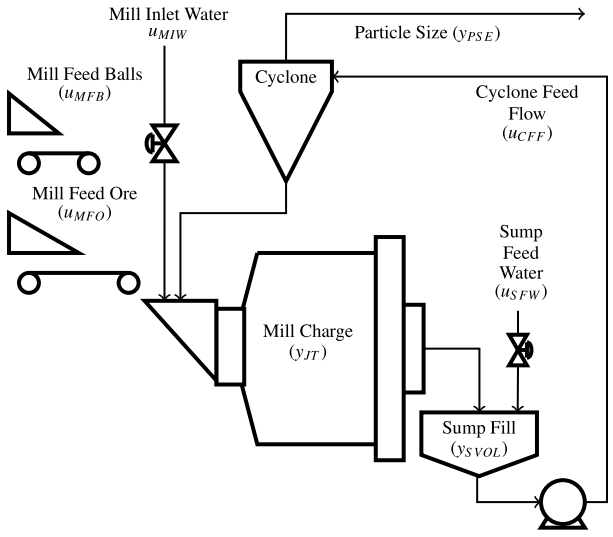


Fig. 5. ROM ore milling circuit (le Roux et al., 2013).

$$w_1 = \frac{|m|}{1+|m|}, \quad w_2 = \frac{1}{1+|m|}. \tag{53}$$

For $n>2$ objectives, the local gradients are estimated $\nabla\hat{J}_i(\eta^\star)$ and the weights computed by aligning the normal of the scalariser with the front via the simplex-constrained least-squares problem:

$$\min_{\mathbf{w}\in\Delta} \left\| \sum_{i=1}^{n} w_i\, \nabla\hat{J}_i(\eta^\star) \right\|_2^2, \qquad \Delta = \Big\{\, \mathbf{w}\in\mathbb{R}^n_{\ge 0} \;:\; \textstyle\sum_{i=1}^n w_i = 1 \,\Big\}. \tag{54}$$

This yields $\mathbf{w}$ such that the induced scalariser is locally tangent to the Pareto front at $\boldsymbol{\eta}^\star$, providing an interpretable, data-driven weight vector.

6. Implementation of AI-tuning and comparative analysis of results

6.1. Ore milling circuit

The plant selected for control is a single-stage run-of-mine (ROM) ore-milling circuit, represented in Fig. 5. The nonlinear plant model used in this paper was validated against data from an operational mill obtained during a sampling campaign by le Roux et al. (2013), and was originally introduced by Coetzee et al. (2010) for controller design and simulation. The use of a validated ore-milling circuit model is essential to ensure the credibility of the results presented.

A concise process description is provided here to define the variables, while a detailed account of the milling process is documented in Coetzee et al. (2010), Craig and MacLeod (1995), le Roux et al. (2013). The mill

is fed with ore u_{MFO}, steel balls u_{MFB}, feed water u_{MIW}, and the out-of-specification particles from the hydrocyclone underflow. Although the addition of steel balls is performed in batches in practice, the simulation assumes a constant feed rate. The water supply u_{MIW} is controlled to maintain a ratio of $0.06u_{MFO}$. The mill charge is measured and expressed as a fraction of the total mill volume, denoted by y_{JT}.

The milled content is discharged as slurry through a grate and collected in the sump, where it is diluted with sump feed water u_{SFW}. The sump volume is represented by y_{SVOL}. The diluted slurry is pumped from the sump to the cyclone at a variable flow rate u_{CFF}. The cyclone separates the in-specification particles, denoted by y_{PSE}, for downstream processing, while the out-of-specification particles are returned to the mill for further grinding. The constraints and nominal operating values provided in Table 1 originate from the model validation (le Roux et al., 2013).

6.2. Ore milling circuit controller

The linear plant model of the ore milling circuit, $\boldsymbol{G}_{nom}(s)$, derived from step tests conducted on the validated nonlinear model around the nominal operating points provided in Table 1, is given by van Niekerk et al. (2025b). While reducing model order can simplify controller synthesis and tuning, over-reduction risks discarding higher-order or weakly damped dynamics that materially affect closed-loop behaviour. Oversimplification reduces the fidelity of the AI-tuning obtained on the linear model when it is transferred to the actual process. The pairing of manipulated variables

$$\boldsymbol{u} = [u_{CFF}, u_{SFW}, u_{MFO}]^\top, \tag{55}$$

with controlled variables

$$\boldsymbol{y} = [y_{PSE}, y_{SVOL}, y_{JT}]^\top, \tag{56}$$

as suggested by Coetzee (2009), promoting robustness to feed disturbances, enables the implementation of a diagonal controller $\boldsymbol{K}$,

$$\boldsymbol{K} = \begin{bmatrix} k_{11} & 0 & 0 \\ 0 & k_{22} & 0 \\ 0 & 0 & k_{33} \end{bmatrix} \tag{57}$$

where each diagonal element k_{jj} corresponds to a PI controller with the form of

$$k_{jj} = k_{Pjj}\left(1 + \frac{1}{\tau_{Ijj}s}\right),\ j = 1, 2, 3. \tag{58}$$

where k_{Pjj} is the proportional gain, and τ_{Ijj} (in hours) is the integral time constant. The optimal values of k_{Pjj} and τ_{Ijj} are unknown and must be determined by the AI-based automatic tuner. The constraints defining $\mathcal{A}_{safe}$ for the milling circuit tuning parameters are provided in van Niekerk et al. (2025b).

6.3. Ore milling circuit objective function

Obtaining the diagonal weighting matrices, $\boldsymbol{\Omega}_e$ and $\boldsymbol{\Omega}_u$, in (45) requires generating the Pareto fronts for the controlled and manipulated variable objectives, respectively. The priority among the variables is determined by selecting an appropriate compromise point on the Pareto front, which informs the corresponding weight vector. Once these weights have been derived and applied to the controlled and manipulated variable objectives, the overall trade-off point that balances performance and actuator usage can be selected.

To determine the weights of the controlled variable objectives that capture the desired priority allocation, the NSGA-II algorithm (Behroozsarand & Shafiei, 2010) is employed to generate the Pareto front from the linear plant model $\boldsymbol{G}_{\text{nom}}(s)$. Fig. 6 illustrates the objective space of the ISE values for the controlled variables, normalised according to (50). The normalised ISEs are represented along the axes as $\hat{J}_{PSE}$, $\hat{J}_{SVOL}$, and $\hat{J}_{JT}$. The Pareto-optimal points, indicated by blue markers,

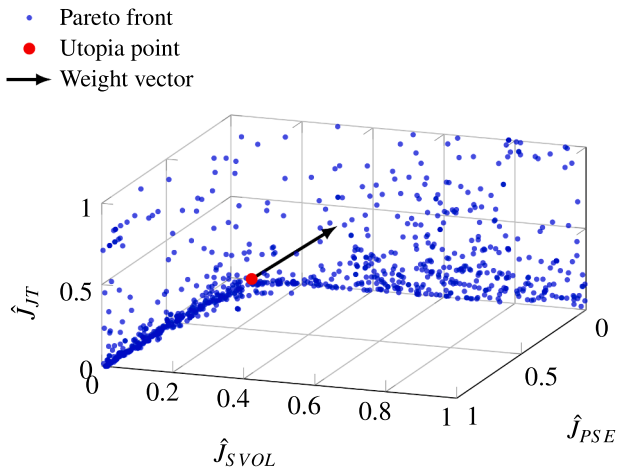


Fig. 6. Normalised objective space with axes defined by the ISE for each controlled variable. The Pareto front is shown in blue, while the utopia point (red) identifies the most balanced trade-off, and the weight vector (black arrow) indicates the compromise among objectives at this point. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

collectively form the Pareto front, which constitutes a surface in the three-dimensional objective space. The compromise solution is selected at the utopia point of the front, shown as a red marker. Selection of the utopia point is appropriate when all objectives are considered equally important, as it represents the theoretical optimum where each objective attains its minimum value (Marler & Arora, 2004), i.e.,

$$\boldsymbol{J}_{ISE}^{\text{utopia}} = \left(\hat{J}_{PSE,\min}, \hat{J}_{SVOL,\min}, \hat{J}_{JT,\min}\right). \tag{59}$$

The corresponding weight vector is defined as

$$\boldsymbol{w}_{ISE} = [w_{PSE}, w_{SVOL}, w_{JT}] \tag{60}$$

and is represented by the black arrow, computed by aligning the vector normal with the tangent surface passing through the utopia point. Here, w_{PSE}, w_{SVOL}, and w_{JT} denote the relative importance of each controlled variable objective, ensuring a balanced improvement across all objectives. As discussed in van Niekerk et al. (2023), less emphasis is often placed on y_{SVOL} because the sump volume can tolerate the deviation from setpoint; however in this paper, equal emphasis is assigned to its weight w_{SVOL} so that controller-induced interaction on y_{SVOL} can be rejected. If interaction rejection is not a priority, w_{SVOL} may be reduced comparatively, provided the normalisation $\sum_{i=1}^{n} w_i = 1$ is maintained.

Similarly, Fig. 7 presents the normalised objective space of the integral of the squared total variation (TV^2) for the manipulated variables, with the axes denoted by $\hat{J}_{CFF}$, $\hat{J}_{SFW}$, and $\hat{J}_{MFO}$. As before, the utopia point is used to derive the weight vector

$$\boldsymbol{w}_{TV^2} = [w_{CFF}, w_{SFW}, w_{MFO}] \tag{61}$$

where w_{CFF}, w_{SFW}, and w_{MFO} represent the relative weighting assigned to each manipulated variable objective.

To establish the trade-off between controller performance and actuator effort, the normalised objectives are combined to form a two-dimensional Pareto front, as shown in Fig. 7, with $\hat{J}_{ISE}$ and $\hat{J}_{TV^2}$ represented on the axes. The trade-off point is selected at the knee point, the region of maximum curvature, where small improvements in one objective result in large sacrifices in the other (Branke et al., 2004). This point represents the most advantageous compromise, where overall performance is maximised without excessive actuator demand. In Fig. 7, the knee point is shown in red, the Pareto front in blue, and the dominated solutions in grey, illustrating how the objective space is sampled to identify the non-dominated frontier.

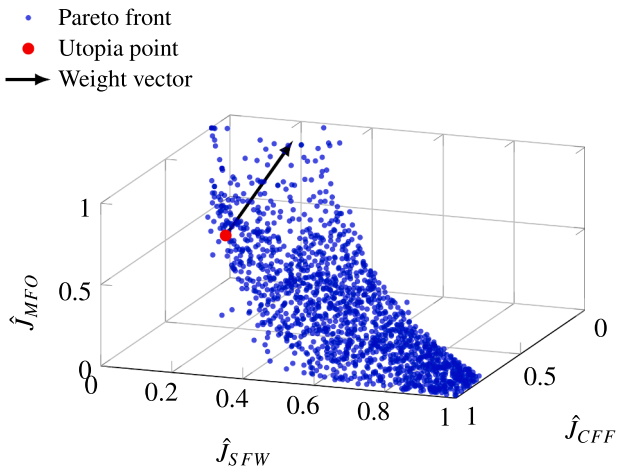


Fig. 7. Normalised objective space with axes defined by the TV^2 for each manipulated variable. The Pareto front is shown in blue, while the utopia point (red) identifies the most balanced trade-off, and the weight vector (black arrow) indicates the compromise among objectives at this point. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

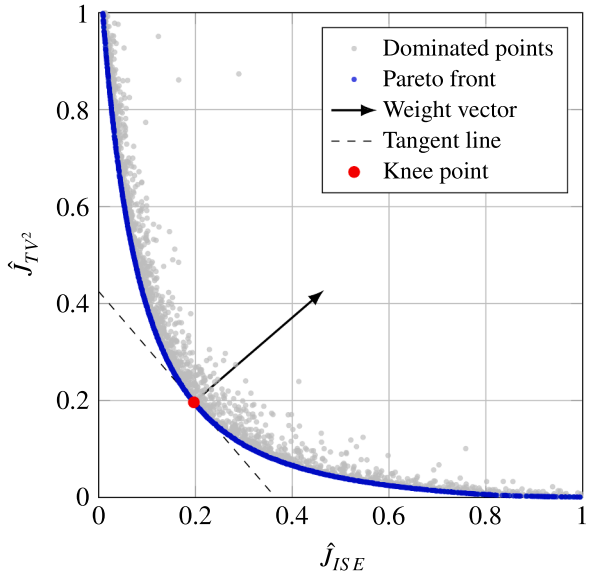


Fig. 8. Normalised objective space with axes defined by the squared TV, representing actuator usage, and the ISE, representing performance. The Pareto front is shown in blue, while the dominated points behind the front are shown in grey. The knee point (red) identifies the optimal trade-off, and the weight vector (black arrow), which is normal to the dashed tangent line passing through the knee point, indicates the compromise among objectives at this point. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The performance-actuator trade-off weight vector is defined as

$$\boldsymbol{w}_{TO} = [w_{ISE}, w_{TV^2}] \tag{62}$$

and is represented in Fig. 8 by the black arrow, computed by aligning its normal with the tangent line passing through the knee point. Here, w_{ISE} and w_{TV^2} represent the relative weighting factors used to balance control performance against actuator utilisation.

Finally, the diagonal weighting matrices are computed as

$$\boldsymbol{\Omega}_e = w_{ISE}\begin{bmatrix} w_{PSE} & 0 & 0 \\ 0 & w_{SVOL} & 0 \\ 0 & 0 & w_{JT} \end{bmatrix}, \tag{63}$$

Table 2
Hyperparameter values selected for PSO tuning.

Parameter	Description	Value
S	Swarm size	20
E	Iterations (termination criterion)	50
c_1	Cognitive acceleration coefficient	1.49 (default)
c_2	Social acceleration coefficient	1.49 (default)
w	Inertia weight	[0.1, 1.1] (decaying)

and

$$\boldsymbol{\Omega}_u = w_{TV^2}\begin{bmatrix} w_{CFF} & 0 & 0 \\ 0 & w_{SFW} & 0 \\ 0 & 0 & w_{MFO} \end{bmatrix}. \tag{64}$$

Here, $\boldsymbol{\Omega}_e$ and $\boldsymbol{\Omega}_u$ represent the final diagonal weighting matrices for the controlled and manipulated variable objectives, respectively. The scalar multipliers w_{ISE} and w_{TV^2} capture the overall trade-off between control performance and actuator effort, while the diagonal elements assign the relative priorities among individual variables within each objective group.

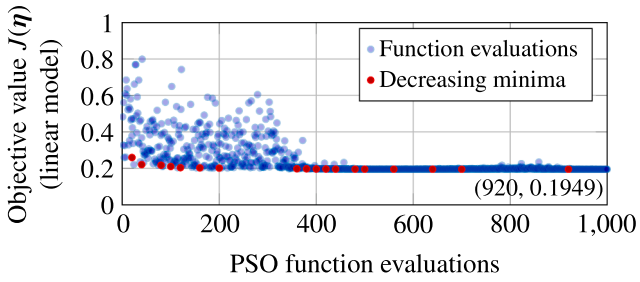
6.4. Sampling and update frequencies

The sampling interval of the ore-milling circuit is set to $t = 10$ s, which is sufficient to capture the system dynamics. This interval applies to the update period of the controlled variables and manipulated variables, as well as to the integration of the objective function $J_{\boldsymbol{y}^{sp}}(\boldsymbol{\eta})$ in (45). The duration of a single AI-tuner iteration, including the step test, tuning parameter update period, and evaluation horizon of $J_{\boldsymbol{y}^{sp}}(\boldsymbol{\eta})$, is $T = 2$ h. This duration is selected to exceed the observed settling time of the step responses, while remaining as short as possible to limit the overall tuning campaign duration. Consequently, the ISE and TV terms of $J_{\boldsymbol{y}^{sp}}(\boldsymbol{\eta})$ are integrated at a sampling rate of 10 s, but the resulting objective value is revealed to the AI-tuner only upon completion of the evaluation horizon, at which point the algorithm parameters are updated.

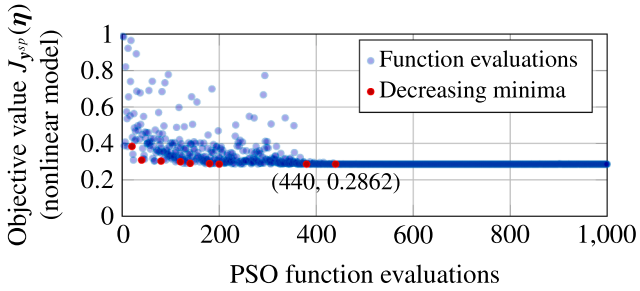
6.5. Analysis of PSO-tuning results

PSO tuning is conducted in accordance with the generalised framework shown in Fig. 4, where the AI tuner is represented by the PSO algorithm. The tuning results shown in Fig. 9 are obtained by applying Algorithm 1 with the hyperparameter values listed in Table 2. The swarm size S is selected by considering the evaluation budget, problem dimensionality, and swarm diversity. A value of $S = 20$ provides adequate diversity during the initial iterations for exploration of the search space, followed by a gradual convergence toward the global minimum. The acceleration coefficients are retained at the default values of MATLAB, which are generally not varied among applications (Eberhart & Kennedy, 1995). The inertia weight w decays linearly within the range $[1.1, \ldots, 0.1]$ to encourage rapid exploration during early iterations, diminishing progressively as convergence is approached. The number of iterations E is selected to be sufficiently large to allow the algorithm to demonstrate convergence. Consequently, PSO hyperparameter selection is relatively straightforward, with the primary consideration being the choice of S.

Fig. 9(a) illustrates the pre-tuning progress in the linear plant model. The scatter plot reports the objective function value $J(\boldsymbol{\eta})$ from (45) as blue markers on the y-axis at each function call. Owing to the normalisation applied via (50), the magnitude of $J(\boldsymbol{\eta})$ is bounded between [0, 1] on the linear plant model. A single PSO iteration comprises S function calls, resulting in the total of 1000 function evaluations shown on the x-axis. As evident from the plot, effective exploration of the search space occurs during the first 400 evaluations, after which the swarm converges rapidly. After each iteration, the global best solution (**gbest**) is evaluated;

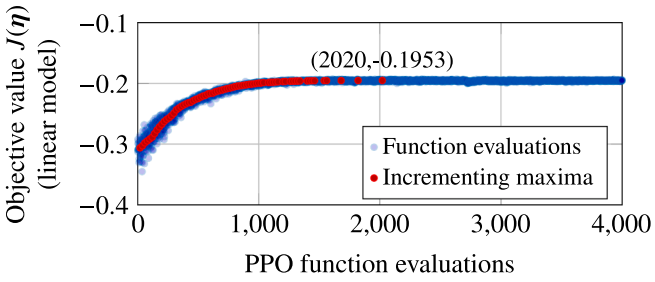


(a) PSO tuning on the linear plant model. All function evaluations are shown in blue, while new global minima, identified at the end of each swarm evaluation, are marked in red.

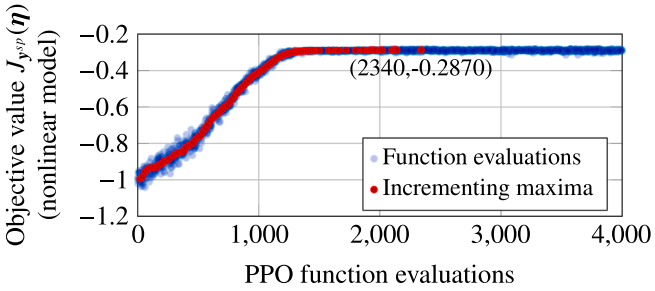


(b) PSO tuning on the nonlinear plant model. All function evaluations are shown in blue, while new global minima, identified at the end of each swarm evaluation, are marked in red.

Fig. 9. Objective function values during PSO tuning of the (a) linear and (b) nonlinear plant model controllers.



(a) PPO tuning on the linear plant model. All function evaluations are shown in blue, while new global maxima, sampled after every 20 episodes, are marked in red.



(b) PPO tuning on the nonlinear plant model. All function evaluations are shown in blue, while new global maxima, sampled after every 20 episodes, are marked in red.

Fig. 10. Objective function values during PPO tuning of the (a) linear and (b) nonlinear plant model controllers.

if an improvement greater than a tolerance of 10^{-4} is achieved, the point is marked in red. The figure shows how **gbest** gradually decreases until no further improvements are observed after the 920$^{\text{th}}$ function evaluation.

Fig. 9(b) presents the final tuning progress on the nonlinear plant model. The first iteration of the final stage of tuning is seeded with the best parameter vector $\boldsymbol{\eta}$ obtained during pre-training. In this plot, $J_{\boldsymbol{y}^{sp}}(\boldsymbol{\eta})$ is shown on the y-axis, as the objective function now has contextual dependence on the setpoint. Similar to tuning on the linear model, the swarm converges after 400 function evaluations. The best value of **gbest** is achieved after 440 evaluations. Notably, the minimum objective value achieved, $J_{\boldsymbol{y}^{sp}}(\boldsymbol{\eta}) = 0.2862$, on the nonlinear model is greater than that on the linear model, $J(\boldsymbol{\eta}) = 0.1949$, attributable to the inherent plant-model mismatch between the linearised and nonlinear dynamics.

In Fig. 9(b), tuning is allowed to continue to illustrate the duration required for PSO to minimise the objective function on the nonlinear plant model. In practice, such a conservative tuning budget may not be practical. More realistic tuning budgets are discussed in Section 6.8.

6.6. Analysis of PPO-tuning results

PPO tuning follows the same generalised framework (Fig. 4), with the AI tuner represented by the PPO agent. To reduce computational complexity, PPO tuning is formulated as a contextual bandit problem, limiting each episode to a single decision step and reward. This approach improves training efficiency (van Niekerk et al., 2025b) by removing the need for multi-step MDP formulation. In this setup, the setpoint is observed as the context and mapped to the initial state, $\boldsymbol{y}^{sp} \rightarrow \boldsymbol{s}_0$. The agent observes the state and samples an action from the stochastic policy, $\boldsymbol{a}_0 \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_0 \mid \boldsymbol{s}_0)$, which is mapped to the controller tuning parameters, $\boldsymbol{a}_0 \rightarrow \boldsymbol{\eta}$. Hence, the tuning parameters are sampled from the policy conditioned on the setpoint context, $\boldsymbol{\eta} \sim \pi_{\boldsymbol{\theta}}(\boldsymbol{\eta} \mid \boldsymbol{y}^{sp})$. The episodic reward, computed from the objective function, is then used to update the policy parameters with the goal of maximising the expected return.

Table 3
Hyperparameter values selected for PPO tuning.

Parameter	Description	Value
γ	Discount rate	0
λ	GAE factor	0.95 (default)
w	Entropy loss weight	0.02
N	Trajectory steps	1
M	Mini-batch size	1
ϵ	Clip factor	0.2
K_e	Number of epochs	Empirical
α_{actor}	Actor learning rate	5×10^{-5}
α_{critic}	Critic learning rate	5×10^{-4}
E	Training episodes	4,000

The PPO tuning results in Fig. 10 are obtained using Algorithm 2 with the hyperparameters listed in Table 3. The discount rate is set to $\gamma = 0$ since only the immediate reward is required, while the trajectory length $N = 1$ and mini-batch size $M = 1$ is used because the task involved a single decision step. The clip factor $\epsilon = 0.2$ and the GAE factor $\lambda = 0.95$ is selected according to the recommendations of Schulman et al. (2017). The learning rates α_{actor} and α_{critic}, the entropy loss weight w, and the number of epochs K_e are determined empirically. The number of training episodes E is chosen to ensure that convergence can be observed.

Additional architectural hyperparameters, including the number of layers and neurons, are required to define the structure of the actor and critic networks.

The actor network is implemented as a feed-forward fully connected model comprising a shared feature path and two parallel output branches corresponding to the mean and standard deviation of the Gaussian policy. The shared path begins with a feature input layer whose size matches the dimensions of $\boldsymbol{y}^{sp}$, followed by two fully connected layers with 64 and 32 neurons, respectively, each employing rectified linear unit (ReLU) activations.

The mean branch contains an additional fully connected layer with 32 neurons and a ReLU activation, followed by a fully connected layer that outputs a vector matching the dimensionality of the safe action space, $\mathcal{A}_{safe}$. A hyperbolic tangent activation constrains the outputs to

Table 4
Hyperparameter values selected for BO tuning.

Parameter	Description	Value
n_o	Initial design size	4 (default)
ξ	EI exploration offset	0.05 (default)
N	Optimisation budget	1,000

the range $[-1, 1]$, and a subsequent scaling layer applies affine transformations (scale and bias) to constrain the outputs to $\mathcal{A}_{safe}$.

The standard-deviation branch comprises a single fully connected layer whose output dimension equals the dimensionality of $\mathcal{A}_{safe}$, followed by a softplus activation to enforce positivity. The resulting standard deviations are scaled to 10% of the scale used in the mean branch to maintain an appropriate level of exploration noise.

The critic network is implemented as a feed-forward fully connected model that maps $\boldsymbol{y}^{sp}$ to a scalar estimate of the state-value function, $V_{\boldsymbol{\phi}}(\boldsymbol{y}^{sp})$. The input is processed through two hidden layers containing 64 and 32 neurons, respectively, each followed by a ReLU activation. The final layer consists of a single fully connected neuron that outputs the estimate of $V_{\boldsymbol{\phi}}(\boldsymbol{y}^{sp})$.

The architectural hyperparameters, including the number of hidden layers, the number of neurons per layer, and the choice of ReLU activations, were selected empirically to balance approximation accuracy, computational efficiency, and training stability. This empirical workflow involved manually adjusting values, retraining, and inspecting reward trends over multiple sessions to progressively tune hyperparameters and optimise learning outcomes.

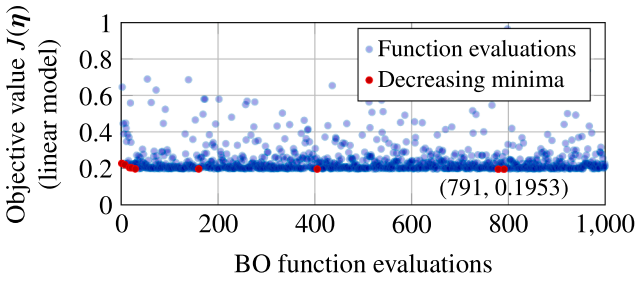
Fig. 10(a) shows the pre-tuning progress on the linear plant model. As PPO seeks to maximise the reward, the scatter plot reports the negative of the objective function value, $-J_y(\boldsymbol{\eta})$, as blue markers at each episode. A steep increase in reward is observed during the first 1000 episodes, followed by convergence near the -0.2 level. To monitor progress and facilitate comparison with PSO, the agent is sampled every 20 episodes; each sample is evaluated in deterministic mode to extract the corresponding tuning parameters without exploration. Improvements over previous samples are indicated with red markers. A period of substantial exploration is observed during the initial 500 episodes, after which the policy stabilises around the maximum reward. The best reward, $J_y(\boldsymbol{\eta}) = -0.1953$, is achieved after 2020 episodes.

Following pre-training, the agent is transferred to the nonlinear plant environment for final tuning, as shown in Fig. 10(b). The initial reward obtained by the pre-trained agent is well below the achievable maximum, necessitating substantial further training to adapt to the nonlinear dynamics. Because PPO employs policy-clipping for stability, large updates are restricted, and extensive training is required to compensate for the plant-model mismatch. The maximum reward achieved on the nonlinear model is $J_{y^{sp}}(\boldsymbol{\eta}) = -0.2870$ after 2340 function evaluations.

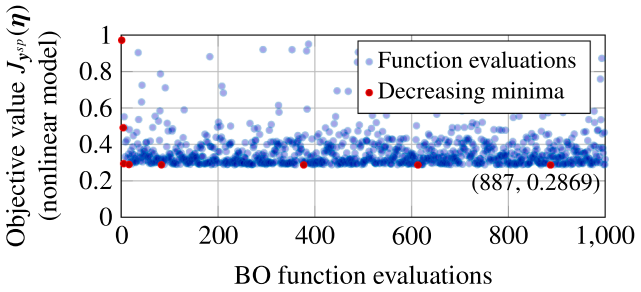
Notably, PPO requires substantially more function evaluations to achieve convergence than PSO. In these results, PSO converged after roughly 400 evaluations, compared with about 1000 for PPO. A detailed comparative analysis is provided in Sections 6.8 and 6.9.

6.7. Analysis of BO-tuning results

BO tuning is performed in accordance with the generalised framework in Fig. 4, with the AI tuner represented by the BO engine. The BO tuning results presented in Fig. 11 are obtained using Algorithms 3 and 4 with the hyperparameters listed in Table 4. Unless specified, the default initial design size of MATLAB is $n_o = 4$. The total number of iterations, referred to as the optimisation budget N, is selected to ensure sufficient evaluations for convergence. During pre-tuning, the budget is limited primarily by simulation time, whereas during online tuning, it is constrained by operational tolerances. The EI exploration offset ξ is retained at the MATLAB default of 0.05. Consequently, when using MATLAB defaults, the BO-EI implementation is effectively hyperparameter-free, representing a notable advantage in terms of ease of application. The only practical consideration is the selection of the evaluation budget, which can readily be determined through offline simulations on the linear plant model.



(a) BO tuning on the linear plant model. All function evaluations are shown in blue, while new global minima, identified at the end of each iteration, are marked in red.



(b) BO tuning on the nonlinear plant model. All function evaluations are shown in blue, while new global minima, identified at the end of each iteration, are marked in red.

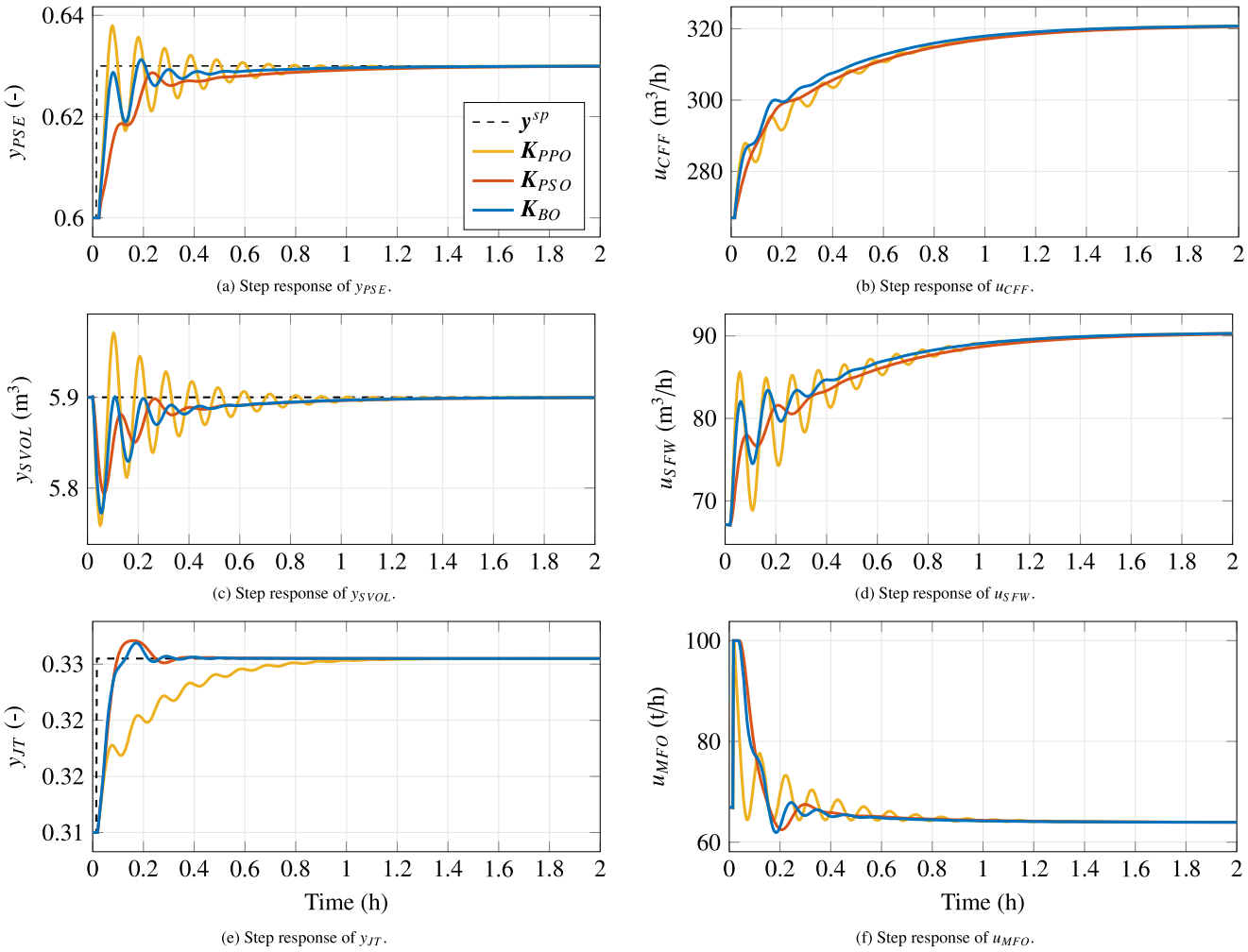
Fig. 11. Objective function values during BO tuning of the (a) linear and (b) nonlinear plant model controllers.

The pre-tuning progress of BO is shown in Fig. 11(a). Unlike PSO, the BO algorithm continues to explore the search space even after the minimum is located. Red markers indicate iterations where a new global minimum is identified. The figure shows that minima near the asymptote are discovered early, within the first 200 iterations, while the overall best minimum, $J(\boldsymbol{\eta}) = 0.1953$, is attained after 791 iterations.

Fig. 11(b) illustrates the final tuning progress, initialised with the best parameter vector $\boldsymbol{\eta}$ obtained during pre-training. The GP surrogate developed for the linear plant model cannot be directly transferred to the nonlinear model, as the underlying objective function differs in the nonlinear environment. Consequently, only the optimal tuning parameters identified in the pre-tuning phase are used to initialise the final optimisation. Similar to tuning on the linear model, good results are achieved within the first 200 iterations, with the optimal minimum value of $J_{y^{sp}}(\boldsymbol{\eta}) = 0.2869$ reached after 887 iterations.

6.8. Comparison

Executing the number of function evaluations shown in Figs. 9, 10, and 11 on an actual plant, such as a milling circuit, would be unreasonably optimistic. With each function evaluation requiring approximately two hours, completing 1000 evaluations would demand about 2000 hours (roughly 83 days) of continuous tuning. Such an extended tuning campaign would be unacceptable, particularly if tuning disturbances resulted in suboptimal production. A far more practical tuning budget is 40 function evaluations, which corresponds to approximately 3.5 days of tuning. Table 5 lists the best objective function values achieved after 40 evaluations on the nonlinear plant model, where BO is shown to minimise the objective function most efficiently within the restricted evaluation budget.



(a) Step response of y_{PSE}.
(b) Step response of u_{CFF}.
(c) Step response of y_{SVOL}.
(d) Step response of u_{SFW}.
(e) Step response of y_{JT}.
(f) Step response of u_{MFO}.

Fig. 12. Step responses of the nonlinear ore milling circuit model using AI-based tuned controllers, achieved after 40 function evaluations. Panels (a) through (f) show the controlled variable $\boldsymbol{y}$ (left) and its paired manipulated variable $\boldsymbol{u}$ (right) for PPO-, PSO-, and BO-based tuning.

Table 5
Best objective function value achieved by the AI-based tuners after 40 function evaluations.

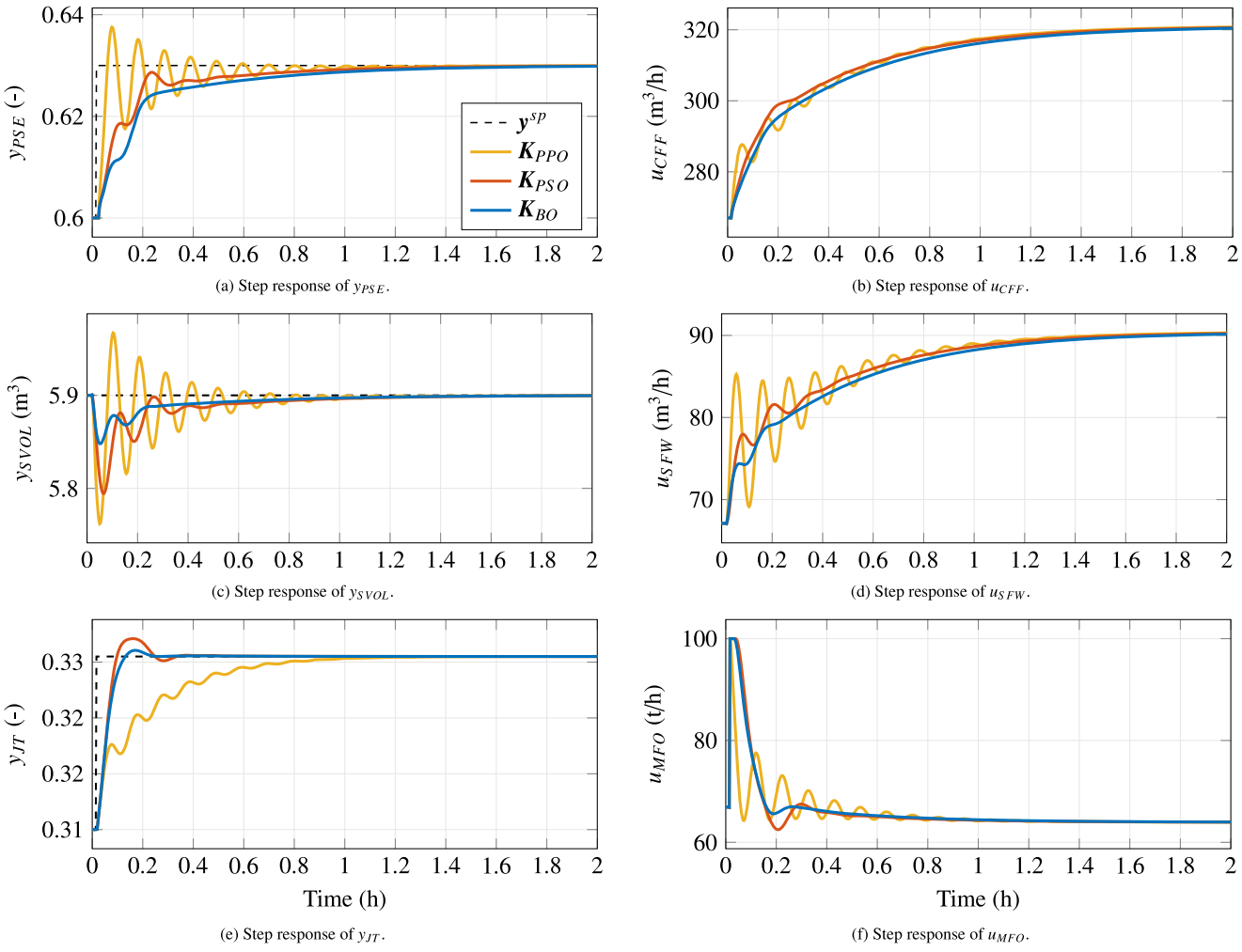
AI-tuner	$J_{\boldsymbol{y}^{sp}}(\boldsymbol{\eta})$
BO	0.2897
PSO	0.3085
PPO	0.9838

Fig. 12 shows the step responses of the PPO-, PSO-, and BO-tuned controllers, with the controlled variables of the ore milling circuit in the left column and their paired manipulated variables in the right column. The setpoints of y_{PSE} and y_{JT} are shown to be stepped simultaneously. Stepping the setpoints concurrently reduces the total testing period by half compared with sequentially stepping and evaluating y_{PSE} and y_{JT} individually. In response to the step change in y_{JT}, u_{MFO} immediately increases the ore feed to meet the new setpoint. Similarly, in response to the step in y_{PSE}, u_{CFF} increases to achieve the setpoint, which in turn reduces y_{SVOL}. To counteract this reduction in y_{SVOL}, u_{SFW} increases, thereby lowering the particle density in the sump. This, however, re-interacts with y_{PSE}, causing u_{CFF} to throttle back, which again affects y_{SVOL} and introduces oscillations when the controllers are tuned too aggressively.

The variable y_{SVOL} is not directly stepped, as in practice the sump level setpoint would typically remain fixed. Nevertheless, suppression of controller interactions affecting y_{SVOL} remains necessary, and therefore the ISE of y_{SVOL} must be included in the overall objective function.

Given the limited tuning budget, the PPO-tuned controller $\boldsymbol{K}_{PPO}$ performs poorly as expected as the agent requires at least 1000 training episodes to converge. The response exhibits weak setpoint tracking for y_{JT} and severe oscillations originating from controller interactions, which manifest in the controlled variables y_{PSE} and y_{SVOL}. The $\boldsymbol{K}_{PSO}$ and $\boldsymbol{K}_{BO}$ controllers show improved setpoint tracking for y_{JT}, with reduced, but still unsatisfactory, oscillations in y_{PSE} and y_{SVOL}.

As evident from Fig. 12, the trade-off between performance and actuator usage chosen at the knee of the Pareto front in Fig. 8 yields overly aggressive controllers that induce excessive oscillations due to control interactions. Interestingly, such oscillations were not observed during tuning on the nonlinear model, owing to the elimination of higher-order dynamics during system identification, simplifications that ultimately increase the plant-model mismatch. To mitigate these oscillations, the weighting vector is shifted away from the Pareto knee toward a point that penalises actuator usage more strongly. The adjusted trade-off weight vector is selected as $\boldsymbol{w}_{TO} = [0.2, 0.8]$, where the larger weight on actuator usage discourages overly aggressive control action.



(a) Step response of y_{PSE}.
(b) Step response of u_{CFF}.
(c) Step response of y_{SVOL}.
(d) Step response of u_{SFW}.
(e) Step response of y_{JT}.
(f) Step response of u_{MFO}.

Fig. 13. Step responses of the nonlinear ore milling circuit model using AI-based tuned controllers, achieved after 40 function evaluations. The weighting vector $\boldsymbol{w}_{TO} = [0.2, 0.8]$ is selected to impose a stronger penalty on actuator usage. Panels (a) through (f) show the controlled variable $\boldsymbol{y}$ (left) and its paired manipulated variable $\boldsymbol{u}$ (right) for PPO-, PSO-, and BO-based tuning.

Table 6
Best objective function value achieved by the AI-based tuners after 40 function evaluations at the revised performance-actuator weighting.

AI-tuner	$J_{\boldsymbol{y}^{sp}}(\boldsymbol{\eta})$
BO	0.1188
PSO	0.1198
PPO	0.4095

The results of the reweighting are presented in Fig. 13. While $\boldsymbol{K}_{PPO}$ and $\boldsymbol{K}_{PSO}$ exhibit negligible improvement, the oscillations in the response of $\boldsymbol{K}_{BO}$ are substantially reduced and can be considered acceptable. Table 6 confirms that BO once again achieves the lowest objective function value under the revised performance-actuator weighting and reduced tuning budget. The PSO algorithm performs surprisingly well in minimising the objective function, especially considering that, when unconstrained, PSO typically requires approximately 400 function evaluations to converge on this plant model. It should be noted that the objective function values in Tables 5 and 6 are not directly comparable, as the weighting matrices of the underlying objective functions differ.

6.9. Ideal AI-based autotuner

The criteria of the ideal autotuner, as established in Section 1, require the controller to be *versatile*, in that it is applicable to a wide range of control structures and processes with minimal operator intervention; *globally optimal*, meaning that it can reliably locate the global optimum without becoming trapped in local optima; *data-efficient*, such that the global optimum is identified using a minimal number of objective function evaluations; and *safe*, in that closed-loop instability is not introduced.

Subject to the qualifications that the controller can be parameterised by a finite set of tuning parameters, that performance can be quantified through an objective function, and that step tests can be conducted on the plant, all three autotuners considered in this study may be regarded as versatile.

As shown in Figs. 9, 10, and 11, all three autotuners are capable of optimising the objective function on the nonlinear plant. The PSO-tuner achieved the best value of $J_{\boldsymbol{y}^{sp}}(\boldsymbol{\eta}) = 0.2862$, compared with $J_{\boldsymbol{y}^{sp}}(\boldsymbol{\eta}) = 0.2869$ obtained by BO optimisation and $J_{\boldsymbol{y}^{sp}}(\boldsymbol{\eta}) = 0.2870$ achieved by PPO. These values were obtained under a conservative budget of function evaluations. Therefore, in the category of global-optimisation under a conservative budget, the PSO-tuner performs best, albeit by a small margin.

A caveat to this observation is that, while PSO and PPO have been shown to become trapped in local optima in general settings (Elbeltagi et al., 2005; Xue et al., 2022), the performance reported in this paper is specific to the ore-milling circuit considered and to the use of the strictly convex objective function in (45).

The results presented in Tables 5 and 6 indicate that the BO-tuner exhibits the highest data efficiency, achieving the most effective optimisation of the objective function within the limited evaluation budget of 40 function calls. Consequently, the BO-tuner is the most cost-effective, as it requires the least amount of process disruption to perform the step testing necessary for tuning.

During training, it is inevitable that the AI-based tuner will occasionally select poorly performing tuning parameters, as such exploration is a necessary step toward improved parameter selection. These selections may result in poor and perhaps oscillatory responses which, although stable, may temporarily disrupt the process. A further advantage of BO, arising from its superior data efficiency, is that the occurrence of such disruptions is minimised."

With respect to safety, tuning in all three cases was performed using the same model-based μ-analysis approach to define the safe search space $\mathcal{A}_{safe}$, which is safe provided that tuning is performed within the operating region for which the linear model was derived. During both the pre-tuning and final tuning stages, no closed-loop unstable iterations were observed. Although the tuning progress figures are not presented in this paper, the evolution of the tuning process using the μ-analysis approach is documented in van Niekerk et al. (2023, 2025a,b). On the basis of the safety criterion, all three methods are therefore considered equally effective.

While PPO and BO are inherently capable of handling uncertain objective functions in stochastic environments (Richter et al., 2025; Yau et al., 2024), PSO is more susceptible to performance degradation when objective function evaluations are corrupted by noise. Zhang et al. (2018) report that PSO, although effective for deterministic optimisation problems, experiences a loss of accuracy in the presence of noise. As the noise magnitude increases, solution quality degrades, manifesting as slower convergence, reduced accuracy, and less stable behaviour (Pan et al., 2006). Nevertheless, Moravec and Pošík (2014) demonstrate that PSO can deliver stable performance at low noise levels when evaluated under sensor noise. These findings indicate that, while PSO can tolerate limited noise, significant noise results in substantial performance degradation, often necessitating specialised mitigation strategies. Hybrid PSO variants incorporating statistical techniques to improve noise resilience have been proposed (Pan et al., 2006; Zhang et al., 2018); however, such variants are not considered in this study.

A limitation of BO is its scalability, as it is typically restricted to optimisation problems involving at most approximately 20 parameters (Moriconi et al., 2020). Gui et al. (2024) characterise the limited scalability of BO from two complementary perspectives: sample-efficiency scalability, associated with the curse of dimensionality, and computational scalability. As dimensionality increases, the sampling efficiency of BO deteriorates rapidly, since the search space expands exponentially, resulting in poor surrogate model accuracy and degraded optimisation performance. From a computational perspective, scalability is constrained primarily by GP regression, whose core operations scale cubically with the number of function evaluations. Moriconi et al. (2020) propose mitigating these limitations by learning a nonlinear low-dimensional feature space in which the optimisation is performed; however, such mitigation strategies are not considered in this paper.

PPO can be applied to high-dimensional continuous action spaces; however, its scalability is limited primarily by sample inefficiency and optimisation bias that worsen with increasing action dimensionality. As an on-policy method, PPO requires large numbers of fresh interactions, which becomes impractical for expensive systems, and in high-dimensional settings the clipped likelihood-ratio objective is more likely to suppress gradients, leading to reduced learning efficiency (Han & Sung, 2019). Consequently, PPO performance in high-dimensional action spaces often degrades unless additional structural assumptions or algorithmic modifications are introduced (Wang et al., 2023).

PSO can also be applied to high-dimensional optimisation problems; however, as dimensionality increases, the swarm must explore an exponentially larger search space (Xu et al., 2019). This often leads to a loss of swarm diversity, premature convergence, and slow optimisation progress unless the swarm size and iteration budget are increased substantially, thereby raising the overall computational burden (Tian et al., 2023).

PPO, and RL-based autotuners more generally, offer the advantage, demonstrated by Dogru et al. (2022), that the agent can be trained over a continuous setpoint range and can therefore provide tuning parameters for any configuration of the setpoint vector. A limitation of PPO-based tuners, however, is that the selection of suitable neural-network architectures and reinforcement-learning hyperparameters introduces significant complexity and requires substantial user intervention. PSO, by contrast, requires the selection of only a small number of parameters, while BO is particularly attractive in this regard, as it can be implemented using default settings without the need for manual hyperparameter tuning.

Table 7 provides a comparative evaluation of the AI-based tuners against the criteria of the ideal autotuner. Overall, the BO-tuner most closely represents the characteristics of an ideal autotuner, based on the ore milling application.

Table 7
Evaluation of AI-based tuners against the criteria of the ideal autotuner.

Criteria	PSO	PPO	BO
Versatility	Good	Good	Good
Globally optimal	Best	Good	Good
Data efficient	Good	Poor	Best
Safe	Good	Good	Good
Noise resilience	Poor	Good	Good
Scalability	Good	Good	Poor
Hyperparameter complexity	Complex	Simple	Simple

7. Conclusion

All the AI-based tuning methods reviewed, PSO representing the nature-inspired subclass, and PPO and BO representing machine learning, demonstrate the capability to automatically determine high-performing PID parameters. Each method advances classical rule-based tuning by enabling systematic, data-driven optimisation that improves tracking and reduces controller interaction as demonstrated on the ore milling circuit. Collectively, these approaches confirm that AI provides a viable and powerful framework for modern autotuning.

Among these methods, BO emerges as the most effective and practical approach for tuning the ore-milling circuit controller. Unlike population-based heuristics that require hundreds of evaluations, or RL approaches that depend on extensive exploration and hyperparameter tuning, BO achieves comparable or superior performance with far fewer plant trials. Its probabilistic GP surrogate enables informed sampling of the parameter space, balancing exploration and exploitation to locate near-optimal solutions rapidly.

In the context of autotuning, these characteristics of BO are especially advantageous. The tuning task is inherently expensive due to the need for online experiments, which may involve production losses during evaluation. By replacing large-scale experimentation with surrogate-driven optimisation, BO maximises the utility of every trial while accelerating convergence towards high-performing controller parameters. This makes it ideally suited for online or semi-supervised deployment, where only a limited number of safe, informative evaluations can be performed without interrupting production.

Overall, BO provides a versatile, data-efficient, safe, and globally optimal autotuning methodology that aligns with the practical constraints

of industrial process control. While all AI-based tuners demonstrate potential, BO uniquely combines the interpretability of model-based reasoning with the adaptability of data-driven learning. Its ability to deliver high-quality controller parameters within minimal experimental budgets establishes it as the preferred choice for next-generation automatic PID tuning in industrial applications.

Potential directions for future research include validating the broader applicability of the AI-based tuning methodology on other types of industrial processes, developing mechanisms for the automatic detection of suboptimal controller performance to trigger AI-based tuning, and conducting a systematic comparison of PSO, PPO, and BO under objective function uncertainty.

CRediT authorship contribution statement

J.A. van Niekerk: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **J.D. le Roux:** Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization; **I.K. Craig:** Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Achu Govind, K. R., & Mahapatra, S. (2024). Improving precision and robustness in level control of coupled tank systems: A tree seed optimization and μ-analysis approach. *Results in Control and Optimization*, *15*, 100410.

Aksland, C. T., Lupp, C. A., Clark, D. L., & Alleyne, A. G. (2022). Gradient-based optimization for anti-windup PID controls. In *2022 American control conference (ACC)* (pp. 1588–1594). IEEE.

Anbumani, K., Ranihemamalini, R., & Pechinathan, G. (2017). GWO based tuning of PID controller for a heat exchanger process. In *2017 Third international conference on sensing, signal processing and security* (pp. 417–421). IEEE.

Araz, O. (2025). Influence of PGA/PGV ratio on the seismic performance of buildings with different slenderness ratios considering soil-structure interaction. *Structures*, *79*, 109653.

Baruah, S., & Dewan, L. (2017). A comparative study of PID based temperature control of CSTR using genetic algorithm and particle swarm optimization. In *2017 International conference on emerging trends in computing and communication technologies* (pp. 1–6). IEEE.

Beahr, D., Bhattacharyya, D., Allan, D. A., & Zitney, S. E. (2024). Development of algorithms for augmenting and replacing conventional process control using reinforcement learning. *Computers & Chemical Engineering*, *190*, 108826.

Behroozsarand, A., & Shafiei, S. (2010). Optimal control of amine plant using non-dominated sorting genetic algorithm-II. *Journal of Natural Gas Science and Engineering*, *2*(6), 284–292.

Berger, M. A. R., & da Fonseca Neto, J. V. (2013). Neurodynamic programming approach for the PID controller adaptation. *IFAC Proceedings Volumes*, *46*(11), 534–539.

Berkenkamp, F., Schoellig, A. P., & Krause, A. (2016). Safe controller optimization for quadrotors with Gaussian processes. In *2016 IEEE international conference on robotics and automation (ICRA)* (pp. 491–496). IEEE.

Blondin, M. J., Pardalos, P. M., & Sáez, J. S. (2019). Computational intelligence and optimization methods for control engineering. Springer.

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

Branke, J., Deb, K., Dierolf, H., & Osswald, M. (2004). Finding knees in multi-objective optimization. In *International conference on parallel problem solving from nature* (pp. 722–731). Springer.

Brochu, E., Cora, V., & Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Department of Computer Science, University of British Columbia, Vancouver, BC, Canada, Technical Report UBC.

Bujgoi, G., & Sendrescu, D. (2025). Tuning of PID controllers using reinforcement learning for nonlinear system control. *Processes*, *13*(3), 735.

Caiza, L., Benítez, D. S., & Camacho, O. (2022). Non-linear PID controller optimization using the artificial bee colony algorithm applied to a small-scale pasteurization plant. In *2022 IEEE International autumn meeting on power, electronics and computing* (pp. 1–6). IEEE (vol. *6*).

Chen, H., Bowels, S., Zhang, B., & Fuhlbrigge, T. (2019). Controller parameter optimization for complex industrial system with uncertainties. *Measurement and Control*, *52*(7-8), 888–895.

Chopra, V., Singla, S. K., & Dewan, L. (2014). Comparative analysis of tuning a PID controller using intelligent methods. *ACTA Polytechnica Hungarica*, *11*(8), 235–249.

Chowdhury, M. A., Al-Wahaibi, S. S. S., & Lu, Q. (2023). Entropy-maximizing TD3-based reinforcement learning for adaptive PID control of dynamical systems. *Computers & Chemical Engineering*, *178*, 108393.

Chowdhury, M. A., & Lu, Q. (2023). A novel entropy-maximizing TD3-based reinforcement learning for automatic PID tuning. In *2023 American control conference* (pp. 2763–2768).

Coetzee, L. C. (2009). Robust nonlinear model predidictive control of a closed run-of-mine ore milling circuit. Ph.D. thesis. University of Pretoria.

Coetzee, L. C., Craig, I. K., & Kerrigan, E. C. (2010). Robust nonlinear model predictive control of a run-of-mine ore milling circuit. *IEEE Transactions on Control Systems Technology*, *18*(1), 222–229.

Cornejo, E. R. F., Diaz, R. C., & Alama, W. I. (2020). PID tuning based on classical and metaheuristic algorithms: A performance comparison. In *2020 IEEE engineering international research conference* (pp. 1–4). IEEE.

Coutinho, J. P. L., Santos, L. O., & Reis, M. S. (2023). Bayesian optimization for automatic tuning of digital multi-loop PID controllers. *Computers & Chemical Engineering*, *173*, 108211.

Craig, I. K., & MacLeod, I. M. (1995). Specification framework for robust control of a run-of-mine ore milling circuit. *Control Engineering Practice*, *3*(5), 621–630.

Das, I., & Dennis, J. E. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural Optimization*, *14*(1), 63–69.

Das, S., Halder, K., & Gupta, A. (2018). Performance analysis of robust stable PID controllers using dominant pole placement for SOPTD process models. *Knowledge-Based Systems*, *146*, 12–43.

Desborough, L., & Miller, R. (2002). Increasing customer value of industrial control performance monitoring - Honeywell's experience. In *Aiche symposium series* 326 (pp. 169–189).

Deulkar, P., & Hanwate, S. (2020). Analysis of PSO-PID controller for CSTR temperature control. In *2020 IEEE First international conference on smart technologies for power, energy and control* (pp. 1–6). IEEE.

Dogru, O., Velswamy, K., Ibrahim, F., Wu, Y., Sundaramoorthy, A. S., Huang, B., Xu, S., Nixon, M., & Bell, N. (2022). Reinforcement learning approach to autonomous PID tuning. *Computers & Chemical Engineering*, *161*, 107760.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Mhs'95. Proceedings of the sixth international symposium on micro machine and human science* (pp. 39–43). IEEE.

El-Gendy, E. M., Saafan, M. M., Elksas, M. S., Saraya, S. F., & Areed, F. F. G. (2020). Applying hybrid genetic-PSO technique for tuning an adaptive PID controller used in a chemical process. *Soft Computing*, *24*(5), 3455–3474.

Elbeltagi, E., Hegazy, T., & Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, *19*(1), 43–53.

Emmerich, M. T. M., Giannakoglou, K. C., & Naujoks, B. (2006). Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, *10*(4), 421–439.

Engelbrecht, A. P. (2007). Computational intelligence: an introduction (vol. 2). Wiley Online Library.

Fister, D., Fister Jr, I., Fister, I., & Šafarič, R. (2016). Parameter tuning of PID controller with reactive nature-inspired algorithms. *Robotics and Autonomous Systems*, *84*, 64–75.

Fujimoto, Y., Sato, H., & Nagahara, M. (2023). Controller tuning with Bayesian optimization and its acceleration: Concept and experimental validation. *Asian Journal of Control*, *25*(3), 2408–2414.

Garcia, C. E., & Morari, M. (1982). Internal model control. A unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development*, *21*(2), 308–323.

Garcia, J., & Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, *16*(1), 1437–1480.

Ghosal, S., Darbar, R., Neogi, B., Das, A., & Tibarewala, D. N. (2012). Application of swarm intelligence computation techniques in PID controller tuning: A review. In *Proceedings of the international conference on information systems design and intelligent applications* (pp. 195–208). Springer.

Giudici, P., & Raffinetti, E. (2021). Shapley-Lorenz eXplainable artificial intelligence. *Expert systems with applications*, *167*, 114104.

Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., & Knoll, A. (2024). A review of safe reinforcement learning: Methods, theories and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *46*(12), (pp. 11216–11235).

Guan, Z., & Yamamoto, T. (2021). Design of a reinforcement learning PID controller. *IEEJ Transactions on Electrical and Electronic Engineering*, *16*(10), 1354–1360.

Gui, Y., Zhan, D., & Li, T. (2024). Taking another step: A simple approach to high-dimensional Bayesian optimization. *Information Sciences*, *679*, 121056.

Han, S., & Sung, Y. (2019). Dimension-wise importance sampling weight clipping for sample-efficient reinforcement learning. In *International conference on machine learning* (pp. 2586–2595). PMLR.

Jayachitra, A., & Vinodha, R. (2014). Genetic algorithm based PID controller tuning approach for continuous stirred tank reactor. *Advances in Artificial Intelligence*, *2014*(1), 791230.

Jesawada, H., Yerudkar, A., Del Vecchio, C., & Singh, N. (2022). A model-based reinforcement learning approach for robust PID tuning. In *2022 IEEE 61st conference on decision and control* (pp. 1466–1471). IEEE.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*, 455–492.

Joseph, S. B., Dada, E. G., Abidemi, A., Oyewola, D. O., & Khammas, B. M. (2022). Metaheuristic algorithms for PID controller parameters tuning: Review, approaches and open problems. *Heliyon*, *8*(5), e09399.

Kavas, B., Balta, E. C., Tucker, M. R., Krishnadas, R., Rupenyan, A., Lygeros, J., & Bambach, M. (2025). In-situ controller autotuning by Bayesian optimization for closed-loop feedback control of laser powder bed fusion process. *Additive Manufacturing*, *99*, 104641.

Khanam, A. (2014). Control strategies for divided wall (Petlyuk) columns. Ph.D. thesis. Norwegian University of Science and echnology Trondheim, Norway.

Khanduja, N., & Bhushan, B. (2016). Intelligent control of CSTR using IMC-PID and PSO-PID controller. In *2016 IEEE 1st international conference on power electronics, intelligent control and energy systems* (pp. 1–6). IEEE.

Khosravi, M., König, C., Maier, M., Smith, R. S., Lygeros, J., & Rupenyan, A. (2022). Safety-aware cascade controller tuning using constrained Bayesian optimization. *IEEE Transactions on Industrial Electronics*, *70*(2), 2128–2138.

König, C., Turchetta, M., Lygeros, J., Rupenyan, A., & Krause, A. (2021). Safe and efficient model-free adaptive control via Bayesian optimization. In *2021 IEEE International conference on robotics and automation* (pp. 9782–9788). IEEE.

Kumar, L., Kumar, P., & Dhillon, S. S. (2020). A multiobjective optimization approach for linear quadratic Gaussian/loop transfer recovery design. *Optimal Control Applications and Methods*, *41*(4), 1267–1287.

Ladjouzi, S., & Grouni, S. (2020). PID controller parameters adjustment using a single memory neuron. *Journal of the Franklin Institute*, *357*(9), 5143–5172.

Lakshmi, S., & Manonmani, A. (2025). Optimizing thermal power plant efficiency through washout filter based proportional integral derivative controller. *Heliyon*, *11*(4), e42864.

Lakshminarayanan, B., Dettú, F., Rojas, C. R., & Formentin, S. (2025). Inverse supervised learning of controller tuning rules. *Automatica*, *178*, 112356.

Lam, R., Poloczek, M., Frazier, P., & Willcox, K. E. (2018). Advances in Bayesian optimization with applications in aerospace engineering. In *2018 AIAA non-deterministic approaches conference* (p. 1656).

Lawrence, N. P., Forbes, M. G., Loewen, P. D., McClement, D. G., Backström, J. U., & Gopaluni, R. B. (2022). Deep reinforcement learning with shallow controllers: An experimental application to PID tuning. *Control Engineering Practice*, *121*, 105046.

Lawrence, N. P., Stewart, G. E., Loewen, P. D., Forbes, M. G., Backstrom, J. U., & Gopaluni, R. B. (2020a). Optimal PID and antiwindup control design as a reinforcement learning problem. *IFAC-PapersOnLine*, *53*(2), 236–241.

Lawrence, N. P., Stewart, G. E., Loewen, P. D., Forbes, M. G., Backstrom, J. U., & Gopaluni, R. B. (2020b). Reinforcement learning based design of linear fixed structure controllers. *IFAC-PapersOnLine*, *53*(2), 230–235.

le Roux, J. D., Craig, I. K., Hulbert, D. G., & Hinde, A. L. (2013). Analysis and validation of a run-of-mine ore grinding mill circuit model for process control. *Minerals Engineering*, *43*, 121–134.

Lee, J. H., Shin, J., & Realff, M. J. (2018). Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers & Chemical Engineering*, *114*, 111–121.

Lee, Y.-S., & Jang, D.-W. (2021). Optimization of neural network-based self-tuning PID controllers for second order mechanical systems. *Applied Sciences*, *11*(17), 8002.

Lequesne, D. (2021). Practical PID Handbook. EDP Sciences.

Li, M., & Feng, X. (2020). Application of improved artificial bee colony algorithm in constant pressure water supply system. In *2020 5th international conference on automation, control and robotics engineering* (pp. 521–525). IEEE.

Liu, B., Zhang, Q., & Gielen, G. G. E. (2013). A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. *IEEE Transactions on Evolutionary Computation*, *18*(2), 180–192.

Luyben, W. L. (1986). Simple method for tuning SISO controllers in multivariable systems. *Industrial & Engineering Chemistry Process Design and Development*, *25*(3), 654–660.

Malmborg, J., Bernhardsson, B., & Åström, K. J. (1996). A stabilizing switching scheme for multi controller systems. *IFAC Proceedings Volumes*, *29*(1), 2627–2632.

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, *26*(6), 369–395.

Mate, S., Pal, P., Jaiswal, A., & Bhartiya, S. (2023). Simultaneous tuning of multiple PID controllers for multivariable systems using deep reinforcement learning. *Digital Chemical Engineering*, *9*, 100131.

Meng, Q., Anandan, P. D., Rielly, C. D., & Benyahia, B. (2023). Multi-agent reinforcement learning and RL-based adaptive PID control of crystallization processes. *Computer Aided Chemical Engineering*, *52*, 1667–1672.

Mockus, J. (1975). On the bayes methods for seeking the extremal point. *IFAC Proceedings Volumes*, *8*(1, Part 1), 428–431.

Moravec, J., & Pošík, P. (2014). Global robot localization under noise stress utilizing EA methods and semisemantic classification of a known environment. *Applied Artificial Intelligence*, *28*(4), 360–417.

Moriconi, R., Deisenroth, M. P., & Sesh Kumar, K. S. (2020). High-dimensional Bayesian optimization using low-dimensional feature spaces. *Machine Learning*, *109*(9), 1925–1943.

Mukhamediev, R. I., Popova, Y., Kuchin, Y., Zaitseva, E., Kalimoldayev, A., Symagulov, A., Levashenko, V., Abdoldina, F., Gopejenko, V., Yakunin, K. et al. (2022). Review of artificial intelligence and machine learning technologies: Classification, restrictions, opportunities and challenges. *Mathematics*, *10*(15), 2552.

Mumuni, Q. A., Olaniyan, O. M., Ipinnimo, O., Akinyemi, L. A., & Olayiwola-Mumuni, A. I. (2025). Intelligent PID gain selection via supervised machine learning approach for decoupled MIMO control systems. *Journal of Future Artificial Intelligence and Technologies*, *2*(1), 163–181.

Nekoui, M. A., Khameneh, M. A., & Kazemi, M. H. (2010). Optimal design of PID controller for a CSTR system using particle swarm optimization. In *Proceedings of 14th international power electronics and motion control conference 2010* (pp. 63–66). IEEE.

Neumann-Brosig, M., Marco, A., Schwarzmann, D., & Trimpe, S. (2020). Data-efficient autotuning with Bayesian optimization: An industrial control study. *IEEE Transactions on Control Systems Technology*, *28*(3), 730–740.

Nian, R., Liu, J., & Huang, B. (2020). A review on reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, *139*, 106886.

van Niekerk, J. A., le Roux, J. D., & Craig, I. K. (2023). On-line automatic controller tuning of a multivariable grinding mill circuit using Bayesian optimisation. *Journal of Process Control*, *128*, 103008.

van Niekerk, J. A., le Roux, J. D., & Craig, I. K. (2025a). Automated tuning of an inverse controller for a MIMO bulk tailings treatment plant using reinforcement learning. *IFAC-PapersOnLine*, *59*, 150–155.

van Niekerk, J. A., le Roux, J. D., & Craig, I. K. (2025b). Reinforcement learning based automatic tuning of PID controllers in multivariable grinding mill circuits. *Control Engineering Practice*, *165*, 106522.

Norlund, F., Tammia, R., Hägglund, T., & Soltesz, K. (2024). Linear-quadratic level control for flotation through reinforcement learning. *IFAC-PapersOnLine*, *58*(14), 799–804.

Olorunda, O., & Engelbrecht, A. P. (2008). Measuring exploration/exploitation in particle swarms using swarm diversity. In *2008 IEEE Congress on evolutionary computation (IEEE world congress on computational intelligence)* (pp. 1128–1134). IEEE.

Pan, H., Wang, L., & Liu, B. (2006). Particle swarm optimization for function optimization in noisy environment. *Applied Mathematics and Computation*, *181*(2), 908–919.

Patil, R. S., Jadhav, S. P., & Patil, M. D. (2024). Review of intelligent and nature-inspired algorithms-based methods for tuning PID controllers in industrial applications. *Journal of Robotics and Control*, *5*(2), 336–358.

Perez-Ramirez, J., Montoya-Acevedo, D., Gil-González, W., Montoya, O. D., & Restrepo, C. (2025). Shunt active power filter using model predictive control with stability guarantee. *e-Prime-Advances in Electrical Engineering, Electronics and Energy*, (101029).

Pramanik, S., Sengupta, A., & Roy, N. (2021). PID flow-level control tuned by genetic algorithm and harmony search algorithm. In *2021 IEEE second international conference on control, measurement and instrumentation* (pp. 172–177). IEEE.

Qin, Y., Zhang, W., Shi, J., & Liu, J. (2018). Improve PID controller through reinforcement learning. In *2018 IEEE CSAA guidance, navigation and control conference (CGNCC)* (pp. 1–6). IEEE.

Rawlings, J. B., Mayne, D. Q., & Diehl, M. (2017). Model predictive control: theory and design. (2nd ed.). Nob Hill Publishing, LLC.

Reynoso-Meza, G., Sanchis, J., Blasco, X., & Herrero, J. M. (2012). Multiobjective evolutionary algorithms for multivariable PI controller design. *Expert Systems with Applications*, *39*(9), 7895–7907.

Richter, A. V., le Roux, J. D., & Craig, I. K. (2025). Bayesian optimization for automatic tuning of a MIMO controller of a flotation bank. *Journal of process control*, *147*, 103388.

Roveda, L., Forgione, M., & Piga, D. (2020). Robot control parameters auto-tuning in trajectory tracking applications. *Control Engineering Practice*, *101*, 104488.

dos Santos Coelho, L., & Mariani, V. C. (2012). Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. *Computers & Mathematics with Applications*, *64*(8), 2371–2382.

Schillinger, M., Hartmann, B., Skalecki, P., Meister, M., Nguyen-Tuong, D., & Nelles, O. (2017). Safe active learning and safe Bayesian optimization for tuning a PI-controller. *IFAC-PapersOnLine*, *50*(1), 5967–5972.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning* (pp. 1889–1897). Proceedings of Machine Learning Research.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv:1707.06347.

Seborg, D. E., Edgar, T. F., Mellichamp, D. A., & Doyle, F. J. (2011). Process dynamics and control. (4th ed.). John Wiley & Sons.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & De Freitas, N. (2015). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, *104*(1), 148–175.

Shamsuzzoha, M., & Skogestad, S. (2010). The setpoint overshoot method: A simple and fast closed-loop approach for PID tuning. *Journal of Process Control*, *20*(10), 1220–1234.

Sharma, M., Verma, P., & Mathew, L. (2016). Design an intelligent controller for a process control system. In *2016 International conference on innovation and challenges in cyber security* (pp. 217–223). IEEE.

Shin, J., Badgwell, T. A., Liu, K.-H., & Lee, J. H. (2019). Reinforcement learning- Overview of recent progress and implications for process control. *Computers & Chemical Engineering*, *127*, 282–294.

Shou, Z., Hong, X., Sun, J., Yang, Y., Wang, J., Yang, Y., & Liao, Z. (2025). A two-stage expert-guided reinforcement learning controller training strategy for large time-delay systems. *Computers & Chemical Engineering*, (109250).

Singh, K. J., Elamvazuthi, I., Shaari, K., & Lima, F. V. (2016). PID tuning control strategy using cuckoo search algorithm for pressure plant. In *2016 6th international conference on intelligent and advanced systems* (pp. 1–6). IEEE.

Skogestad, S. (2003). Simple analytic rules for model reduction and PID controller tuning. *Journal of Process Control*, *13*(4), 291–309.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Proceedings of the 25th international conference on neural information processing systems-volume 2* (pp. 2951–2959).

Sumar, R. R., Coelho, A. A. R., & dos Santos Coelho, L. (2010). Computational intelligence approach to PID controller design using the universal model. *Information Sciences*, *180*(20), 3980–3991.

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. (2nd ed.). MIT press.

Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, *12*, 1057–1063.

Tian, J., Hou, M., Bian, H., & Li, J. (2023). Variable surrogate model-based particle swarm optimization for high-dimensional expensive problems. *Complex & Intelligent Systems*, *9*(4), 3887–3935.

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., & Guyon, I. (2021). Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS 2020 competition and demonstration track* (pp. 3–26). Proceedings of Machine Learning Research.

Valluru, S. K., & Singh, M. (2018). Performance investigations of APSO tuned linear and nonlinear PID controllers for a nonlinear dynamical system. *Journal of Electrical Systems and Information Technology*, *5*(3), 442–452.

Vilone, G., & Longo, L. (2021). Classification of explainable artificial intelligence methods through their output formats. *Machine Learning and Knowledge Extraction*, *3*(3), 615–661.

Wang, H., & Ricardez-Sandoval, L. A. (2024). A deep reinforcement learning-based PID tuning strategy for nonlinear MIMO systems with time-varying uncertainty. *IFAC-PapersOnLine*, *58*(14), 887–892.

Wang, X., Jin, Y., Schmitt, S., & Olhofer, M. (2023). Recent advances in Bayesian optimization. *ACM Computing Surveys*, *55*(13s), 1–36.

Wang, Z., Qin, C., Wan, B., & Song, W. W. (2021a). A comparative study of common nature-inspired algorithms for continuous function optimization. *Entropy*, *23*(7), 874.

Wang, Z., Yao, X., Li, T., & Zhang, H. (2021b). Design of PID controller based on echo state network with time-varying reservoir parameter. *IEEE Transactions on Cybernetics*, *52*(7), 6615–6626.

Williams, T. J. (1990). A reference model for computer integrated manufacturing from the viewpoint of industrial automation. *IFAC Proceedings Volumes*, *23*(8), 281–291.

Wilson, J. T., Hutter, F., & Deisenroth, M. P. (2018). Maximizing acquisition functions for Bayesian optimization. *32nd Conference on Neural Information Processing Systems*, *20*(5), 645–651.

Xu, G., Cui, Q., Shi, X., Ge, H., Zhan, Z.-H., Lee, H. P., Liang, Y., Tai, R., & Wu, C. (2019). Particle swarm optimization based on dimensional learning strategy. *Swarm and Evolutionary Computation*, *45*, 33–51.

Xue, W., Wu, H., Ye, H., & Shao, S. (2022). An improved proximal policy optimization method for low-level control of a quadrotor. In *Actuators* (p. 105). MDPI (vol. *11*).

Yan, C., Zhang, H., Chang, X., Gao, P., Fan, Q., & Fu, L. (2025). IRIME-MFuzzyN-PID: A novel method for MIMO temperature control system. *Measurement*, *256*, 118352.

Yau, H.-T., Kuo, P.-H., Luan, P.-C., & Tseng, Y.-R. (2024). Proximal policy optimization-based controller for chaotic systems. *International Journal of Robust and Nonlinear Control*, *34*(1), 586–601.

Yin, Q., Zhu, H., Yang, J., Ni, H., & Gao, B. (2025). Research and application of optimization method for semi-submersible platform mooring system based on deep learning. *Ocean Engineering*, *329*, 121084.

Younis, A.-S. A., Moustafa, A. M., & Moness, M. (2019). Experimental benchmarking of PID empirical and heuristic tuning for networked control of double-tank system. In *2019 15th international computer engineering conference* (pp. 162–167). IEEE.

Yusoff, Z. M., Muhammad, Z., Abidin, A. F. Z., Aziz, M. A. A., Ismail, N., & Rahiman, M. H. F. (2017). Self-tuning fuzzy PID controller using online method in essential oil extraction process. In *Proceedings of the 6th international conference on computing and informatics* (pp. 25–27).

Zhang, G., Zhang, C., Wang, W., Cao, H., Chen, Z., & Niu, Y. (2023). Offline reinforcement learning control for electricity and heat coordination in a supercritical CHP unit. *Energy*, *266*, 126485.

Zhang, J., Zhu, X., Wang, Y., & Zhou, M. (2018). Dual-environmental particle swarm optimizer in noisy and noise-free environments. *IEEE Transactions on Cybernetics*, *49*(6), 2011–2021.

Zhang, K., Xu, P., & Zhang, J. (2020). Explainable AI in deep reinforcement learning models: A SHAP method applied in power system emergency control. In *2020 IEEE 4th conference on energy internet and energy system integration (EI2)* (pp. 711–716). IEEE.

Zhu, X. (2014). Computational intelligence techniques and applications. In *Computational intelligence techniques in earth and environmental sciences* (pp. 3–26). Springer.