Control Engineering Practice 165 (2025) 106522

Contents lists available at ScienceDirect

Control Engineering Practice

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

Reinforcement learning based automatic tuning of PID controllers in multivariable grinding mill circuits

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:
Automatic tuning
Comminution
Grinding mills
Proximal policy optimisation
Reinforcement learning
Robust stability analysis

ABSTRACT

Process controllers are extensively utilised in industry and necessitate precise tuning to ensure optimal performance. While tuning controllers through the basic trial-and-error method is possible, this approach typically leads to suboptimal results unless performed by an expert. This study investigates the use of reinforcement learning (RL) for the automatic tuning of proportional–integral–derivative (PID) controllers that control a grinding mill circuit represented by a multivariable nonlinear plant model which was verified using industrial data. By employing the proximal policy optimisation (PPO) algorithm, the RL agent adjusts the controller parameters to enhance closed-loop performance. The problem is formulated to maximise a reward function specifically designed to achieve the desired controller performance. Agent actions are analytically constrained to minimise the risk of closed-loop instability and unsafe behaviours during training. The simulation results indicate that the automatically tuned controller outperforms the manually tuned controller in setpoint tracking. The proposed approach presents a promising solution for real-time controller tuning in industrial processes, potentially increasing productivity and product quality while reducing the need for manual intervention. This research contributes to the field by establishing a robust framework for applying RL in process control, designing effective reward functions, constraining the agent to a safe operational space, and demonstrating its potential to address the challenges associated with PID controller tuning in grinding mill circuits.

1. Introduction

Proportional–integral–derivative (PID) controllers have been widely used in the process industry since the 1950s, when the barriers to their adoption, including appropriate tuning, controllable processes, and complex mechanical linkages, were overcome (Åström & Hägglund, 1995; Bennett, 2001; Desborough & Miller, 2002). A survey by Olivier and Craig (2017) indicates that over 90% of controllers in the mineral processing industry are of the PID type, many being PI-only (Boubertakh, Tadjine, Glorennec, & Labiod, 2010). Their widespread use is attributed to their simplicity, robustness, and accessibility. PID controllers are intuitive, with the proportional, integral, and derivative terms each contributing distinctively to control performance (Wang, Cheng, & Wei, 2007). They are widely applied in systems with benign dynamics and moderate performance requirements, and can be robustly tuned to accommodate process uncertainties (Garpinger, Hägglund, & Åström, 2014).

Their versatility and ubiquity make PID controllers well-suited for testing automated tuning strategies. Proper tuning is critical to achieving closed-loop performance objectives such as setpoint tracking and disturbance rejection, leading to increased productivity, reduced energy consumption, improved product quality, and minimised operator intervention (Craig & MacLeod, 1995; Xue et al., 2019). Established techniques like Ziegler–Nichols (Ziegler & Nichols, 1942), internal model control (IMC) (Garcia & Morari, 1982), simplified internal model control (SIMC) (Skogestad, 2003), approximate M-constrained integral gain optimisation (AMIGO) (Åström & Hägglund, 2004), and others have been developed for this purpose.

However, up to 66% of control loops remain poorly tuned, with around 33% operated in manual mode (Åström & Hägglund, 1995; Bauer, Auret, Bacci di Capaci, Horch, & Thornhill, 2019; Desborough & Miller, 2002). Factors such as changes in operating conditions, ageing equipment, and the high cost of domain expertise often result in suboptimal tuning (Spielberg, Tulsyan, Lawrence, Loewen, & Bhushan Gopaluni, 2019). Automated tuning methods are increasingly important to address these challenges, particularly those capable of online adaptation to minimise plant and model mismatch. An ideal tuner should optimise performance while ensuring long-term gains outweigh

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

https://doi.org/10.1016/j.conengprac.2025.106522
Received 1 April 2025; Received in revised form 24 July 2025; Accepted 26 July 2025
Available online 19 August 2025

the temporary costs of tuning (Neumann-Brosig, Marco, Schwarzmann, & Trimpe, 2020).

Reinforcement learning (RL) has emerged as a promising framework for automated tuning. While RL has excelled in fields such as gaming (Mnih et al., 2013), article recommendations (Li, Chu, Langford, & Schapire, 2010), and board games (Silver et al., 2016), its applications in process control remain limited. RL combines principles from optimal control and behavioural psychology to optimise long-term rewards through trial-and-error interactions within Markov decision processes (MDPs) (Sutton & Barto, 2018; Thorndike, 1898). Advances in deep learning have extended the capabilities of RL to continuous state and action spaces, enabling process control applications (Lillicrap et al., 2015; Silver et al., 2014).

In simulation studies, RL has demonstrated success in controlling processes such as paper machines, distillation columns, and heating, ventilation, and air conditioning (HVAC) systems using deterministic policy gradient (DPG) and deep deterministic policy gradient (DDPG) algorithms (Spielberg et al., 2019). Similarly, Ma, Zhu, Benton, and Romagnoli (2019) applied DDPG to a nonlinear, time-delayed polymerisation process, while Dogru, Wieczorek, Velswamy, Ibrahim, and Huang (2021) utilised asynchronous advantage actor–critic (A3C) algorithms for hybrid tank systems. Despite these advances, RL-based control systems face challenges in interpretability and stability due to the black-box nature of neural network controllers (Lawrence et al., 2020a; Nian, Liu, & Huang, 2020).

To address these challenges, this research explores using RL to automatically tune PID controllers instead of replacing them. Retaining PID controllers preserves their interpretability and stabilising properties while leveraging RL to optimise their parameters. This avoids the uncertainties of direct RL-based control and reduces disruptions during deployment.

Recent studies highlight the potential of RL in PID tuning. Lawrence et al. (2020a) proposed a method embedding PID controllers within actor networks, with weights corresponding to the tuning parameters. A gradient-free RL approach introduced by Lawrence et al. (2020b) further simplified stability analysis while achieving competitive performance.

Although the real-world application of RL in process industries is limited, pilot-scale studies have demonstrated its potential. For instance, Dogru et al. (2022) optimised a PI controller for a water pumping system, while Lawrence et al. (2022) employed RL to tune a PID controller for a laboratory-scale two-tank system.

The RL-based tuning of PID controllers described in Dogru et al. (2022), Lakhani, Chowdhury, and Lu (2022) and Lawrence et al. (2022, 2020a, 2020b) apply to single-input single-output (SISO) systems. In contrast, this paper investigates RL-based PID tuning for a multi-input multi-output (MIMO) system, a substantially more complex problem due to controller interaction, differences in the magnitudes and priorities of the input and output variables, and the increased complexity of the reward function associated with the multivariable system dynamics.

Mate, Pal, Jaiswal, and Bhartiya (2023) present a case study in which RL is used to tune PID controllers for a MIMO simulated quadruple tank system, while Carlucho, De Paula, and Acosta (2020) apply RL to tune MIMO PID controllers for a simulated mobile robot. However, unlike the approach adopted in this paper, where the tuning parameters are selected based on the controller setpoints and held constant for the duration of the training episode, both studies implement adaptive PID controllers, in which the tuning parameters are continuously updated based on the observed system state. Notably, Lawrence et al. (2020b) caution against implementing the policy as an adaptive controller, where the PID tuning parameters are updated at each time step. Such continuous switching introduces nonlinearity into the closed-loop system, thereby complicating its overall stability analysis (Stewart, 2012).

A key challenge in implementing RL in process control applications is the requirement for safe exploration (Lakhani et al., 2022). In the context of automatically tuning process controllers, safe exploration involves selecting tuning parameters that do not lead to closed-loop instability. Although unstable controllers are unlikely to cause equipment damage or fatalities, thanks to protections such as emergency shutdown systems and safety instrumented systems for high-risk applications (Seborg, Edgar, Mellichamp, & Doyle, 2011), instabilities can still result in production interruptions or degraded product quality.

A comprehensive review of safe RL methods is provided by Gu et al. (2024). Among these methods are control theory-based approaches that utilise a plant model to constrain the action space. This paper contributes to this area by demonstrating how μ-analysis can be used to define the boundaries of the safe action space. While μ-analysis is a well-established technique for evaluating the tolerable uncertainty in plant parameters (Skogestad & Postlethwaite, 2007), this work applies it in a novel way: by keeping the plant parameters fixed and instead analysing the uncertainty in the action space.

This methodology is demonstrated in a multivariable nonlinear ore milling circuit simulation. The tuning problem is framed as a contextual bandit to reduce complexity, limiting each episode to a single decision step and improving training efficiency. This approach eliminates the need for multi-step MDP formulations, enhancing the feasibility of RL in industrial settings.

A related study by van Niekerk, le Roux, and Craig (2023) demonstrates the application of Bayesian optimisation for the automatic tuning of a diagonal controller in a linear, multivariable ore milling circuit model. Building on van Niekerk et al. (2023), the primary contributions of this work include:

- constraining the RL agent to select actions within analytically derived safe action boundaries;
- applying RL for the automatic tuning of a diagonal PID controller with fixed parameters in a nonlinear, MIMO, ore milling circuit model;
- demonstrating the capability of training an agent to provide tuning parameters for controlling the plant at different operating points; and
- developing a procedure for pre-training an agent prior to online training and deployment.

These advancements address key challenges related to stability, interpretability, and safety, aiming to bridge the gap between RL theory and its practical deployment in process industries.

The structure of this paper is organised as follows: Section 2 defines the problem by framing it as the maximisation of a reward function. Section 3 offers a theoretical overview of the reinforcement learning algorithm chosen for automatic tuning. Section 4 details the ore milling circuit, the controller setup, the robust stability analysis, and the rationale behind the design of the reward function to reflect the performance requirements. Section 5 presents the simulation results, and Section 6 provides the concluding remarks.

2. Problem statement

Consider the nonlinear MIMO process described by the following equations

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

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

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ represents the process states, $\boldsymbol{u}(t) \in \mathbb{R}^m$ denotes the manipulated variables, and $\boldsymbol{y}(t) \in \mathbb{R}^p$ are the controlled variables. This process is to be controlled using a feedback controller in a unity feedback configuration. The controller can be expressed as

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

where $\boldsymbol{K} \in \mathbb{C}^{m\times m}$ denotes the controller, $\boldsymbol{e}$ represents the setpoint tracking errors, specifically, the differences between the setpoints $\boldsymbol{y}^{sp}(t)$

and the controlled variables, and $\boldsymbol{\eta}$ are the tuning parameters of the controller. For a controller $\boldsymbol{K}$ consisting of PID-controllers on the diagonal, the dimensions of the tuning parameters are $\boldsymbol{\eta} \in \mathbb{R}^{3\times m}$.

The optimal tuning parameters of the controller are unknown and must be determined. These parameters are part of a broader domain of possible values $\mathcal{A}$, but they must be constrained to a subset $\mathcal{A}_{\text{safe}}$ that satisfies the safety and performance constraints. The safe parameter search space is defined as

$$\boldsymbol{\eta} \in \mathcal{A}_{\text{safe}} \subseteq \mathcal{A} \in \mathbb{R}^d \tag{3}$$

where d is the dimension of $\mathcal{A}$.

To assess the performance of the controller tuning parameters $\boldsymbol{\eta}$, closed-loop step tests are conducted online for each setpoint $\boldsymbol{y}^{sp}(t)$. The step responses with controller $\boldsymbol{K}$ in closed-loop, parameterised by $\boldsymbol{\eta}$, are evaluated across the entire trajectory of the controlled variables $\boldsymbol{y}(t)$ as the process dynamics move from one steady state to the next. Once the system has settled, the performance score is recorded as an episodic reward, represented as a scalar quantity. The reward function captures the desired trajectory of the step response and can be constructed from various performance indices that are suitable for time-domain evaluation. These performance indices are typically applied to $\boldsymbol{e}$ and may include the integral of the absolute error (IAE), the integral of the squared error (ISE), the integral of the time-weighted absolute error (ITAE), the integral of the time-weighted squared error (ITSE). Other performance indices may include settling time, overshoot, actuator usage, decay ratio, rise time, among others (Pongfai, Su, Zhang, & Assawinchaichote, 2020; Seborg et al., 2011; Skogestad & Postlethwaite, 2007).

In a MIMO system, it is necessary to scale the contribution of each controlled and manipulated variable to the reward to prevent any single variable from disproportionately influencing the outcome. Additionally, the performance index of each variable can be weighted according to its relative importance. The episodic reward $R_i(\boldsymbol{\eta})$ for a MIMO controller can be expressed as (van Niekerk et al., 2023)

$$R_i(\boldsymbol{\eta}) = \sum_{i=1}^{N} \omega_i \left(\sum_{j=1}^{P} \beta_j \phi_j(\boldsymbol{\eta}) \right) \tag{4}$$

where N is the total number of variables, ω_i represents the performance weighting of the variable, P is the total number of performance indices selected per variable, ϕ_j is the performance index, and β_j is a scaling factor that adjusts the contribution of each performance index.

The problem of determining the optimal tuning parameters can be formulated as an optimisation task, where the objective is to maximise the reward with respect to the tuning parameters

$$\max_{\boldsymbol{\eta}\in\mathcal{A}_{\text{safe}}} R_i(\boldsymbol{\eta}). \tag{5}$$

RL is well-suited for addressing this problem, as it operates online, interacts directly with the process to mitigate plant-model mismatches, and is inherently designed to maximise rewards.

3. Reinforcement learning

In the RL framework, an agent interacts with a stochastic environment modelled as an MDP. The agent, acting as the learner and decision-maker, continuously interacts with the environment, with the agent selecting actions and the environment responding with new situations and rewards. At each time step t, the agent receives a representation of the state $\boldsymbol{s}_t \in \mathbb{R}^p$ of the environment and selects an action $\boldsymbol{a}_t \in \mathbb{R}^{3\times m}$ based on the policy π. One time step later, due to the action taken, the environment transitions to a new state $\boldsymbol{s}_{t+1}$, and the agent receives a scalar reward r_{t+1}, which guides the selection of the next action. The MDP and the agent create a sequence of states, actions, and rewards, which can be episodic or continue indefinitely. The agent learns an optimal decision policy, mapping the system state to the optimal action (Lee, Shin, & Realff, 2018). The objective of the agent is to find the best policy to maximise the long-term cumulative reward, known as the return G_t, through interactions with the environment (Sutton & Barto, 2018).

RL methods for maximising return include policy optimisation and dynamic programming. Policy optimisation is a class of algorithms aimed at directly optimising the policy, which defines the behaviour of the agent. In contrast to dynamic programming methods, which first estimate the value function and then select actions based on these estimates, policy gradient methods focus on learning a policy that maximises the expected return directly, and then samples actions from the policy without relying on a value function (Sutton, McAllester, Singh, & Mansour, 1999). Policy optimisation is especially effective for continuous action spaces because it parameterises the policy as a probability distribution over actions (Sutton & Barto, 2018). In contrast, dynamic programming methods must approximate an infinite number of possible action values. Policy optimisation policies for continuous and complex action spaces are therefore often simpler compared to those of dynamic programming.

The goal of policy gradient methods, a subset of the policy optimisation class, is to learn the policy parameter vector by using stochastic gradient ascent by taking steps proportional to the policy gradient in the direction that increases the objective function. The policy parameter is updated by

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}_k}) \tag{6}$$

where $\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}_t}) \in \mathbb{R}^d$ is the gradient of the objective function with respect to the policy parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$, α is the learning rate and k is the policy update iteration.

A common policy gradient method defines the gradient of the objective function as (Sutton et al., 1999)

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = \mathop{\mathbb{E}}_{\tau\sim\pi_{\boldsymbol{\theta}}} \left[\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t) A^{\pi_{\boldsymbol{\theta}}(\boldsymbol{s}_t,\boldsymbol{a}_t)} \right] \tag{7}$$

where the gradient is represented by the expected sum over time steps t of the product between the gradient of the log-probability of the selected action $\boldsymbol{a}_t$ given the state $\boldsymbol{s}_t$ under the policy $\pi_{\boldsymbol{\theta}}$, and the advantage function $A^{\pi_{\boldsymbol{\theta}}(\boldsymbol{s}_t,\boldsymbol{a}_t)}$ which measures how much better or worse an action $\boldsymbol{a}_t$ is compared to the average action at state $\boldsymbol{s}_t$. Here, τ is the state–action trajectory, and T denotes the final time step.

One of the limitations of this policy gradient method is that small changes in the parameter space can lead to significant fluctuations in performance, potentially causing the policy to collapse (Schulman, Levine, Abbeel, Jordan, & Moritz, 2015). Trust region policy optimisation (TRPO) developed by Schulman et al. (2015) addresses this issue by using Kullback–Leibler (KL) divergence to constrain policy updates, ensuring a guaranteed monotonic improvement in the policy. Proximal policy optimisation (PPO) introduced by Schulman, Wolski, Dhariwal, Radford, and Klimov (2017) shares the same objective as TRPO, i.e. to take the largest possible policy update step without degrading performance, but simplifies implementation by eliminating TRPO's second-order methods. PPO is widely regarded as one of the most successful RL methods, achieving state-of-the-art results (Wang, He, & Tan, 2020), is the primary variant employed at OpenAI (OpenAI, 2025), and is used without modification in this paper.

The PPO-Clip variant updates the current policy parameter vector, $\boldsymbol{\theta}_k$, by finding the optimal policy vector that maximises the expectation of the PPO loss function over state–action pairs sampled from the current policy $\pi_{\boldsymbol{\theta}_k}$

$$\boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\boldsymbol{s},\boldsymbol{a}\sim\pi_{\boldsymbol{\theta}_k}} \left[L(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_k,\boldsymbol{\theta}) \right]. \tag{8}$$

When the actor-critic architecture is represented as a single neural network, the PPO loss function is given by

$$L(\boldsymbol{s},\boldsymbol{a},\boldsymbol{\theta}_k,\boldsymbol{\theta}) = L_{\text{CLIP}} - L_{\text{VF}} + \mathcal{H}[\pi_{\boldsymbol{\theta}}] \tag{9}$$

where L_{CLIP} is the clipped surrogate objective function, L_{VF} is a squared-error loss, and $\mathcal{H}[\pi_\theta]$ represents the entropy bonus added to ensure sufficient exploration. The surrogate objective function, maximised by updating the policy parameter vector $\boldsymbol{\theta}$, is defined as

$$L_{\text{CLIP}} = \frac{1}{M}\sum_{i=1}^{M}\left[\min\left(c_i(\boldsymbol{\theta})A_i^{\pi_{\theta_k}}, d_i(\boldsymbol{\theta})A_i^{\pi_{\theta_k}}\right)\right] \tag{10}$$

which is the summation over M mini-batches, where

$$c_i(\boldsymbol{\theta}) = \frac{\pi_\theta(\boldsymbol{a}_i|\boldsymbol{s}_i)}{\pi_{\theta_k}(\boldsymbol{a}_i|\boldsymbol{s}_i)} \tag{11}$$

is the importance sampling ratio, representing the ratio of the probabilities of taking the ith sampled action $\boldsymbol{a}_i$ at state $\boldsymbol{s}_i$ under the current policy π_θ compared to the previous policy π_{θ_k}. The clipping function

$$d_i(\boldsymbol{\theta}) = \text{clip}(c_i(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon) \tag{12}$$

uses a hyperparameter ϵ to limit how far the new policy can deviate from the old policy while still contributing to the objective. The advantage function is defined as

$$A_i^{\pi_{\theta_k}} = G_{ti} - V_{\boldsymbol{\phi}}(s_i) \tag{13}$$

where G_{ti} represents the return of the ith sample, and $V_{\boldsymbol{\phi}}(\boldsymbol{s}_i)$ denotes the value function evaluated at state sample $\boldsymbol{s}_i$ with parameter vector $\boldsymbol{\phi}$.

The squared-error loss function, which is minimised by updating the parameter vector $\boldsymbol{\phi}$ is given by

$$L_{VF}(\boldsymbol{\phi}) = \frac{1}{2M}\sum_{i=1}^{M}\left(G_{ti} - V_{\boldsymbol{\phi}}(\boldsymbol{s}_i)\right)^2. \tag{14}$$

Exploration is encouraged by incorporating an entropy loss term, $\mathcal{H}[\pi_\theta]$. Maximising the entropy loss term increases the agent's uncertainty, thereby promoting exploration and aiding in escaping local optima.

$$\mathcal{H}[\pi_\theta] = \frac{w}{2}\sum_{j=1}^{C}\ln\left(2\pi\, e\, \sigma_{ji}^2\right). \tag{15}$$

In (15), C represents the number of continuous actions produced by the actor, w the entropy loss weight, and σ_{ji} denotes the standard deviation of action j under the current policy π_θ.

In this paper, separate actor and critic neural networks are adopted. The actor and critic loss functions are defined as

$$L_{\text{actor}} = L_{\text{CLIP}} + \mathcal{H}[\pi_\theta] \tag{16}$$

and

$$L_{\text{critic}} = L_{\text{VF}}. \tag{17}$$

The training algorithm for the PPO agent is as follows:

1. Initialise the weights of the actor neural network with a random policy parameter vector $\boldsymbol{\theta}_0$.
2. Initialise the weights of the value function neural network with a random value function parameter vector $\boldsymbol{\phi}_0$.
3. Generate a trajectory consisting of N state–action-reward steps by executing the policy π_{θ_k} in the environment.
4. For each step in the trajectory, calculate the return G_t and advantage $A^{\pi_{\theta_k}}$.
5. Sample a mini-batch of size M from the trajectory.
6. Update the policy parameters by maximising L_{actor} across the sampled mini-batch for K_e epochs, using stochastic gradient ascent, as follows

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha_{actor}\nabla_\theta L_{\text{actor}}(\boldsymbol{\theta}_k). \tag{18}$$

7. Update the value function parameters by minimising L_{critic} across the sampled mini-batch for K_e epochs using stochastic gradient descent, as follows

$$\boldsymbol{\phi}_{k+1} = \boldsymbol{\phi}_k - \alpha_{critic}\nabla_{\boldsymbol{\phi}} L_{\text{critic}}(\boldsymbol{\phi}_k). \tag{19}$$

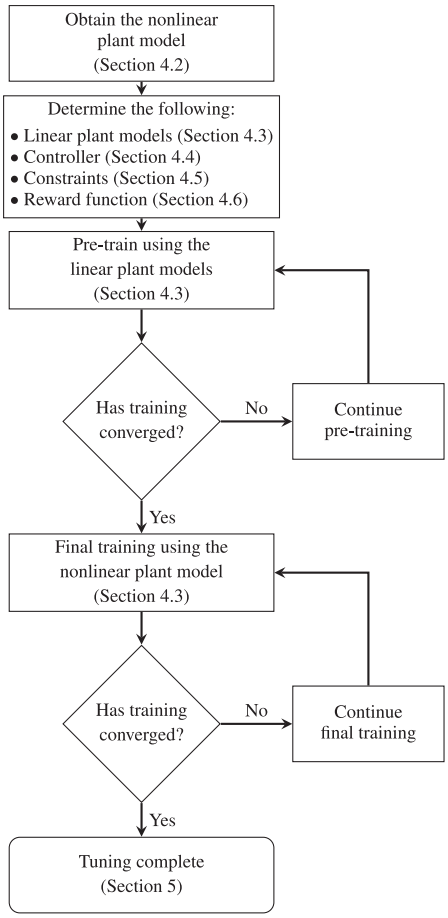

Fig. 1. Flowchart of the RL-based tuning procedure.

8. Repeat steps 3 through 7 until the number of training episodes are complete.

4. Simulation

The simulations are conducted on a grinding mill circuit represented by a multivariable nonlinear plant model to demonstrate how PPO can be used to automatically tune a MIMO controller.

4.1. Tuning procedure

The flowchart in Fig. 1 provides an overview of Section 4 and outlines the RL-based tuning procedure. The process begins with obtaining the nonlinear plant model, as described in Section 4.2. Subsequently, the key components required for the RL tuning framework are determined, including the linear plant models (Section 4.3), the controller (Section 4.4), the constraints defining the safe parameter search space (Section 4.5), and the reward function (Section 4.6).

Once these elements are established, pre-training is performed using the linear plant models (Section 4.3) with the agent constrained to the safe search space. A decision point then follows, where the convergence of training is assessed. If training has not yet converged to the maximum reward, pre-training continues; otherwise, the procedure advances to the final training phase using the nonlinear plant model (Section 4.3).

At the next decision point, the convergence of the final training phase is evaluated. If training has not yet converged, it continues; otherwise, the tuning process is considered complete, with the results presented in Section 5. This structured approach ensures a systematic

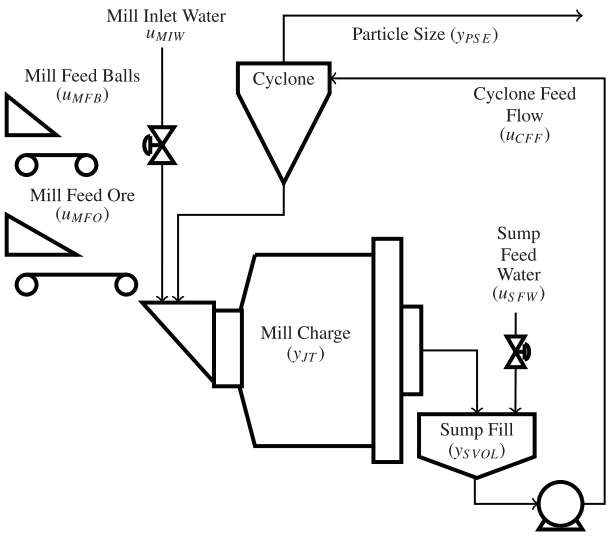

Fig. 2. ROM ore milling circuit (le Roux et al., 2013).

transition from pre-training to final training, ultimately leading to the successful and safe tuning of the controller.

4.2. Plant

The simulation focuses on the control of a single-stage run-of-mine (ROM) ore milling circuit, as depicted in the process flow diagram in Fig. 2. The multivariable nonlinear plant model used in the simulations that follow was verified with industrial data as described in le Roux, Craig, Hulbert, and Hinde (2013). A concise overview of the process is provided here, with a more detailed discussion available in le Roux et al. (2013). The mill receives inputs of water u_{MIW}, steel balls u_{MFB}, and ore u_{MFO}. The volume fraction of the mill that is filled is denoted by y_{JT}. The slurry is discharged through an end-discharge grate from the mill and collected in a sump, with the fill volume represented by y_{SVOL}. The sump slurry is diluted with additional water u_{SFW} before being pumped to the cyclone at a feed rate of u_{CFF}. The cyclone separates the particles that meet the required specifications from those that do not. The particles within specification y_{PSE} overflow from the cyclone for further downstream processing, while the out-of-specification particles are returned to the mill via the cyclone underflow for additional milling (Coetzee, Craig, & Kerrigan, 2010; Craig & MacLeod, 1995; le Roux et al., 2013).

The manipulated and control variables, with their respective constraints and nominal values, are provided in Table 1. These nominal values are derived from data collected during a sampling campaign on an operational grinding mill under steady-state conditions, as documented by le Roux et al. (2013). Although steel balls are typically loaded in batches by an operator, they are maintained at a constant feed rate of $u_{MFB} = 6.43$ t/h for the purposes of this simulation. While the mill water feed could be adjusted to extend the range of y_{PSE}, as noted by Craig, Hulbert, Metzner, and Moult (1992), it is maintained at a ratio of $u_{MIW} = 0.06\,u_{MFO}$ in this paper.

The nonlinear model of the ore milling circuit, first introduced by Coetzee et al. (2010), was developed specifically for controller design and simulation purposes, and is utilised in this paper. A detailed explanation of the functionality of the model, along with its validation using data from a sampling campaign at a full-scale plant, is provided by le Roux et al. (2013) and le Roux and Steyn (2022). Further refinement of the symbology used in the milling circuit model is presented by Noome, le Roux, and Padhi (2023).

Table 1
Process variable descriptions, constraints and nominal operating values.

Manipulated variables		Unit	Min	Max	Nominal
u_{MIW}	Flow-rate of water to the mill	m^3/h	–	–	4.17
u_{MFO}	Flow-rate of ore to the mill	t/h	0	100	66.9
u_{MFB}	Flow-rate 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

Table 2
Hyperparameters values selected for pre and final training.

Para	Description	Pre-training	Final
γ	Discount rate	0	No change
w	Entropy loss weight	0.02	0.0
N	Trajectory steps	1	No change
M	Mini-batch size	1	No change
ϵ	Clip factor	0.2	No change
K_e	Number of epochs	10	No change
α_{actor}	Actor learn rate	5×10^{-5}	No change
α_{critic}	Critic learn rate	5×10^{-4}	No change
E	Training episodes	10000	500

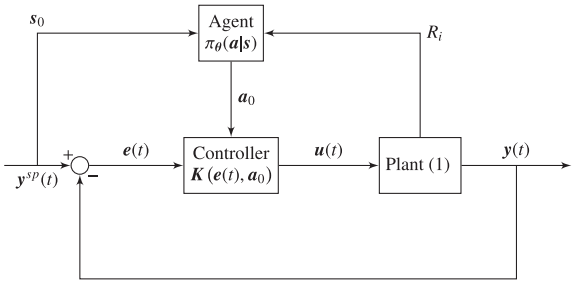

Fig. 3. RL PPO framework depicting the interactions between the agent, controller and plant.

4.3. Framework

The RL PPO framework illustrated in Fig. 3 highlights the interactions between the PPO agent, represented by the policy $\pi_\theta(a|s)$, the setpoints of the controlled variables $\boldsymbol{y}^{sp}$, the controller $\boldsymbol{K}$ configured in a unity feedback loop, and the plant, which corresponds to the RL environment. The objective of the framework is to enable the automatic tuning of the controller to achieve optimal performance across multiple operating points while minimising the disruption caused by online agent training on the production of the ore milling circuit.

To minimise the duration and risks associated with online training, the agent is pre-trained offline using multiple linear models obtained by linearising the nonlinear plant model at various operating points. These operating points correspond to 95%, 97.5%, and 105% of the nominal values for y_{PSE} and y_{JT}, as outlined in Table 1. The resulting plant models are represented as

$$\boldsymbol{y} = \boldsymbol{G}_{nom}(s)\boldsymbol{u} \tag{20a}$$

$$\boldsymbol{y} = \boldsymbol{G}_{95\%}(s)\boldsymbol{u} \tag{20b}$$

$$\boldsymbol{y} = \boldsymbol{G}_{97.5\%}(s)\boldsymbol{u} \tag{20c}$$

$$\boldsymbol{y} = \boldsymbol{G}_{105\%}(s)\boldsymbol{u} \tag{20d}$$

where $\boldsymbol{G}_{nom}(s)$ represents the nominal plant model, while $\boldsymbol{G}_{95\%}(s)$, $\boldsymbol{G}_{97.5\%}(s)$, and $\boldsymbol{G}_{105\%}(s)$ denote the linear plant models at their respective operating points. $\boldsymbol{G}_{95\%}(s)$ represents a plant where y_{PSE} and y_{JT} are at 95% of their respective values in the nominal plant $\boldsymbol{G}_{nom}$, while

y_{SVOL} remains unchanged at the nominal operating point. In this paper, these models are derived from the nonlinear plant model; however, in practice, they could be obtained through step tests conducted on the actual process. The output vector $\boldsymbol{y}$ and the input vector $\boldsymbol{u}$ are defined as follows

$$\boldsymbol{y} = \left[y_{PSE}, y_{SVOL}, y_{JT}\right]^T \tag{21}$$

$$\boldsymbol{u} = \left[u_{CFF}, u_{SFW}, u_{MFO}\right]^T. \tag{22}$$

The selected operating points lie within the permissible tolerances of the controlled variables and are designed to minimise the occurrence of manipulated variable saturation during closed-loop step training. Since the plant model is nonlinear, the linear models differ from one another, allowing the agent to learn optimal tuning parameters by maximising the cumulative reward at each operating point. This formulation ensures that the agent is trained across a representative range of operating points, facilitating better generalisation to the nonlinear plant during subsequent online training. The operating point for y_{SVOL} remains unchanged, as sump volume control primarily addresses disturbance rejection and does not require setpoint changes tailored to production scenarios.

During pre-training, training begins with the linear plant model, $\boldsymbol{G}_{nom}$, operating at steady state. The training procedure follows these steps:

1. At the start of each training episode, the setpoints $\boldsymbol{y}^{sp}$ are randomly stepped to 95%, 97.5%, or 105% of their nominal values for y_{PSE} or y_{JT}.
2. The agent observes this setpoint vector change at time t_0, forming the initial state $\boldsymbol{s}_0$.
3. Based on its current policy, the agent maps $\boldsymbol{s}_0$ to an action $\boldsymbol{a}_0$, which determines the tuning parameters of the controller $\boldsymbol{K}$.
4. The plant is represented by the linear model corresponding to the chosen setpoint.
5. The controller applies the setpoint vector $\boldsymbol{y}^{sp}$ and adjusts the manipulated variables to drive the controlled variables towards the setpoints.
6. The plant dynamics are allowed to reach steady state after a setpoint change.
7. The plant evaluates the step response and provides a single episodic reward R_i to the agent.
8. The required number of pre-training episodes, specified in Table 2, ensures that the agent converges to the maximum episodic reward.

Following pre-training in the linear environment, the agent undergoes final training on the nonlinear plant model to mitigate the mismatch between the linear and nonlinear plant models. The training approach mirrors pre-training: $\boldsymbol{y}^{sp}$ is randomly stepped, the agent observes the step change and adjusts the tuning parameters before receiving the episodic reward. The number of online training episodes required is detailed in Table 2. Unlike pre-training, the number of online training episodes are constrained to minimise its impact on production.

Validation of the RL-tuned controller is inherent in the procedure, as the final tuning stage is performed on the nonlinear model. While the ultimate standard for validation is implementation on a physical ore milling circuit, in the absence of a real plant, a surrogate nonlinear model, validated against sampled process data (le Roux et al., 2013), is used.

Since each episode consists of a single step from $\boldsymbol{s}_0$ to the terminal state, the return G_{ti} is equal to the episodic reward R_i. This framework is synonymous to a contextual multi-armed bandit problem, where the initial state $\boldsymbol{s}_0$ provides the context, and the multi-armed action represents the tuning parameter vector of the controller. This approach

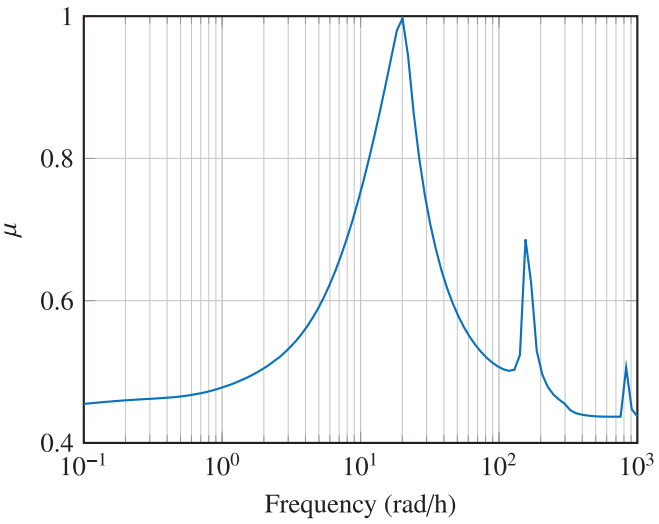

Fig. 4. Robust stability structured singular value μ-plot with the RL agent actions restricted to the safe search space $\mathcal{A}_{\text{safe}}$ as defined in (28).

reduces the learning effort for the agent by eliminating the overhead of dealing with a multi-step MDP.

Table 2 lists the hyperparameter values selected for both pre and final training. The discount rate is set to $\gamma = 0$ because only the immediate reward is considered. The number of steps in the trajectory is $N = 1$, and the mini-batch size is $M = 1$, since there is only a single state–action-reward step in each episode. The clip factor is $\epsilon = 0.2$ as suggested by Schulman et al. (2017). The values of the entropy loss weight w, the actor learning rate α_{actor}, and the critic learning rate α_{critic} were determined empirically. The entropy loss weight is reduced to zero to discourage exploration during final training.

4.4. Controller

The nominal linear plant model $\boldsymbol{G}_{nom}(s)$ is used to determine the controller structure, identify the appropriate input–output pairings, and determine the safe parameter search space $\mathcal{A}_{\text{safe}}$. The transfer function matrix of the nominal linear plant model, $\boldsymbol{G}_{nom}(s)$, is given by

$$\boldsymbol{G}_{nom}(s) = \begin{bmatrix} g_{p11} & g_{p12} & g_{p13} \\ g_{p21} & g_{p22} & g_{p23} \\ g_{p31} & g_{p32} & g_{p33} \end{bmatrix} \tag{23}$$

with the individual transfer functions specified as

$$g_{p11} = \frac{0.654s + 0.626}{s^2 + 439.7s + 552} e^{-0.00833s} \tag{24a}$$

$$g_{p12} = \frac{0.05262s + 0.08517}{s^2 + 20.24s + 60.98} e^{-0.00833s} \tag{24b}$$

$$g_{p13} = \frac{-0.008287}{s + 3.601} e^{-0.00833s} \tag{24c}$$

$$g_{p21} = \frac{-0.4795}{s} \tag{24d}$$

$$g_{p22} = \frac{1.045}{s} \tag{24e}$$

$$g_{p23} = \frac{0.00508s + 1.436}{s^2 + 3.424s + 2.605 \times 10^{-10}} \tag{24f}$$

$$g_{p31} = \frac{0.004636}{s + 5.597} e^{-0.0111s} \tag{24g}$$

$$g_{p32} = \frac{-0.005685}{s + 4.452} e^{-0.0111s} \tag{24h}$$

$$g_{p33} = \frac{0.00577}{s}. \tag{24i}$$

In these equations, s denotes the complex frequency variable in the Laplace transform, with time expressed in hours.

The relative gain array (RGA) (Bristol, 1966) of $\boldsymbol{G}_{nom}(s)$ suggest the input–output pairings of u_{CFF}-y_{SVOL}, u_{SFW}-y_{PSE}, and u_{MFO}-y_{JT} for implementing a diagonal controller, albeit with significant interactions. Chen, Zhai, Li, and Li (2007) reports that the pairing of u_{CFF}-y_{SVOL} and u_{SFW}-y_{PSE} can lead to challenges in industrial plants, particularly with sump level control, which is adversely affected by variations in ore hardness, feed rate, and particle size. In contrast, the alternative pairing of u_{CFF}-y_{PSE} and u_{SFW}-y_{SVOL} demonstrates significantly improved robustness to feed disturbances, provided actuator limitations are considered (Coetzee, 2009). The pairing of u_{CFF}-y_{PSE} and u_{SFW}-y_{SVOL} are used in this paper.

The tuning parameters of a reference controller are essential for determining $\mathcal{A}_{\text{safe}}$ and evaluating the performance of the RL-trained controller. The SIMC rules are employed to compute the tuning parameters for the diagonal controller $\boldsymbol{K}$, which is defined as

$$\boldsymbol{K} = \begin{bmatrix} k_{11} & 0 & 0 \\ 0 & k_{22} & 0 \\ 0 & 0 & k_{33} \end{bmatrix} \tag{25}$$

where each diagonal element k_{jj} corresponds to a PI controller in the Laplace domain, expressed as

$$k_{jj} = k_{Pjj}\left(1 + \frac{1}{\tau_{Ijj}s}\right),\ j = 1, 2, 3. \tag{26}$$

In this expression, k_P represents the proportional gain, and τ_I (hours) is the integral time constant. The calculated tuning parameters based on $\boldsymbol{G}_{nom}(s)$ are summarised as follows

$$\boldsymbol{\eta} = [k_{P11}, \tau_{I11}, k_{P22}, \tau_{I22}, k_{P33}, \tau_{I33}]^T \tag{27a}$$

$$= [16.37, 0.0023, 34.18, 0.11, 6189, 0.11]^T. \tag{27b}$$

4.5. Constraints

To avoid closed-loop instability during the training of the RL agent, it is crucial to restrict the actions of the agent (tuning parameters) to a safe search space, $\mathcal{A}_{\text{safe}}$, as defined in (3). The safe parameter search space should be carefully selected to exclude parameters that could lead to instability while still encompassing the global optimum. To extend the search space beyond the initial tuning parameters $\boldsymbol{\eta}$, a robust stability analysis is performed using the MATLAB Robust Control Toolbox, following the μ-analysis approach (MATLAB, 2023; Skogestad & Postlethwaite, 2007), while keeping the plant parameters constant. This robust stability analysis determines the maximum level of parameter uncertainty that can be tolerated before the system becomes unstable under the worst-case conditions. This restriction of the search space is categorised as Safe RL by applying prior knowledge, according to Garcia and Fernández (2015).

The μ-analysis approach is applied to the nominal linear plant model, $\boldsymbol{G}_{nom}$, to derive $\mathcal{A}_{\text{safe}}$. The constraints defined by $\mathcal{A}_{\text{safe}}$ remain valid for the nonlinear model, provided that the step tests conducted during final training stay within the approximately linear region around the operating point used to derive $\boldsymbol{G}_{nom}$. This is achieved by applying setpoint step changes that are large enough to elicit a measurable response from the process variables, sufficiently above the measurement noise, but small enough to remain within the approximately linear operating region. In practical applications, the extent of this linear region can be established by conducting step tests of varying magnitudes to determine the range over which the process behaves approximately linearly, thereby allowing appropriate bounds to be placed on setpoint step changes.

The resulting maximum allowable uncertainty defines the constraints of the safe search space, which are expressed as follows

$$\begin{aligned} \mathcal{A}_{\text{safe}} = \{&(k_{P11}, \tau_{I11}, k_{P22}, \tau_{I22}, k_{P33}, \tau_{I33}) \mid \\ &k_{P11} \in [6.48, 41.35], \\ &\tau_{I11} \in [0.0009, 0.0058], \\ &k_{P22} \in [13.53, 86.29], \\ &\tau_{I22} \in [0.0444, 0.2828], \\ &k_{P33} \in [2451, 15629], \\ &\tau_{I33} \in [0.0444, 0.2828]\}. \end{aligned} \tag{28}$$

Expanding the search space to the instability threshold, as defined in (28), enhances the probability of identifying the optimal tuning parameters corresponding to the global maximum of the reward. Although the presence of the global maximum within this space cannot be definitively guaranteed, given that the robust stability analysis is conducted on a linear model, it is likely that the global maximum is within the search domain. By excluding parameters that cause instability, online training of the RL agent can proceed with reduced risk of production losses due to unstable tuning parameters.

The robust stability structured singular value μ-plot is presented in Fig. 4. Stability is assured as long as $\mu < 1$ across all frequencies. Fig. 4 confirms the closed-loop stability of the plant $\boldsymbol{G}_{nom}(s)$ under control by $\boldsymbol{K}$, provided the parameters are constrained to the safe search space $\mathcal{A}_{\text{safe}}$ as defined in (28). It is evident from Fig. 4 that the μ-plot reaches its peak at the crossover frequency of the closed-loop system, which is expected given that the system is operating near the instability threshold.

In the context of Safe RL as described by Gu et al. (2024), the approach followed in this paper is categorised as model-based Safe RL and, as such, requires a linear model of the plant. The same linear model used for pre-training is also employed to derive $\mathcal{A}_{\text{safe}}$, and can be readily obtained by performing step tests on the actual process in the absence of a nonlinear model, as applied in this study. Given this linear model, initial tuning parameters can be determined using, for example, the SIMC tuning rules, as demonstrated in this paper. These initial parameters are then treated as uncertain parameters in the structured singular value (μ) analysis to derive the boundaries of $\mathcal{A}_{\text{safe}}$.

4.6. Rewards

The reward function is fundamental to the concept of RL because the primary goal of the agent is to determine the optimal policy that maximises the long-term cumulative reward through interactions with the environment (Sutton & Barto, 2018). For the reward function to be effective, it must assess the performance of the tuning parameters against the specified criteria of the system and increase the rewards as the performance nears these specifications. The RL framework used in this article is a contextual bandit problem, where the agent observes the context $\boldsymbol{s}_0$ and takes a single action $\boldsymbol{a}_0$ to maximise the immediate episodic reward R_i. In this framework, the step change to the setpoint vector is mapped to the bandit context, denoted as $\boldsymbol{y}_{sp} \mapsto \boldsymbol{s}_0$, and the action taken by the agent is mapped to the tuning parameters, expressed as $\boldsymbol{a}_0 \mapsto \boldsymbol{\eta}$. The episodic reward is generated after a specified time period to allow the controlled variables to settle at $\boldsymbol{y}_{sp}$. The reward should evaluate the complete trajectory of the controlled variables in response to the step changes, rather than focusing solely on individual metrics such as settling time, peak overshoot, or rise time. For instance, while the settling time might be satisfactory, the system could exhibit significant oscillations before settling, which should not be encouraged by the reward.

In their work, Lawrence et al. (2020b) introduces a reward function that compares the discretised closed-loop step response of the perturbed controller to the target or desired step response. The response comprises n samples evenly distributed over the duration of the episode, where

the perturbed response of the controller is represented by the state vector $\boldsymbol{s} = [y(0), \ldots, y(t_{n-1})]$. Similarly, the sample data $\tilde{y}(t_i)$ of the target response is contained in the target vector $\tilde{\boldsymbol{s}}$. The user-defined target step response can be intuitively selected to incorporate performance specifications into the reward function. The reward function is motivated by the IAE and ISE of the state and target vectors, expressed as

$$R_i(\boldsymbol{s}, \boldsymbol{a}) = -\frac{1}{n}\|\boldsymbol{s} - \tilde{\boldsymbol{s}}\|_q^q \tag{29}$$

where R_i denotes the episodic reward received after the ith selection of tuning parameters and $q \in \{1, 2\}$ is fixed for the duration of the episode.

For the task of tuning a PI controller according to Dogru et al. (2022), the objective is to search for optimal PI parameters by maximising the reward, as defined by

$$R_i(\boldsymbol{s}, \boldsymbol{a}) = -\sum_{t=0}^{T}\left(e_t^2 + \boldsymbol{W}\boldsymbol{C}_t^T \Lambda\right). \tag{30}$$

Here, T represents the episode duration, e_t is the tracking error at time step t, $\boldsymbol{W}$ is a vector of weights that emphasises the relative importance of the constraints $\boldsymbol{C}$ imposed on the tuning parameters, controlled variable, and manipulated variable, and Λ represents the Lagrangian penalty. The first term in the equation is the ISE of the tracking error sampled at fixed intervals during the episode. Throughout the training process, exploration beyond constraints is discouraged by introducing the Lagrangian penalty, consisting of a Lagrangian coefficient that begins at zero and progressively learns during training.

The reward is the sum of the ISE and the product of the constraint violation penalty and the Lagrangian coefficient. The constraints guide the selection of PI parameters with the objective of keeping the manipulated and controlled variables within acceptable operational limits. Deviations from these constraints result in reduced rewards. When all constraints are adhered to, the reward consists exclusively of the ISE.

Approaches to reward modification, like Lagrangian relaxation, incorporate a gentle penalty for constraint violations or unstable responses, aiming to discourage such explorations. Nevertheless, they lack the capability to guarantee rigorously that the tuning parameters suggested by the RL agent for subsequent episodes consistently stay within the stable region (Lakhani et al., 2022).

The study by Lakhani et al. (2022) introduces an innovative approach to the utilisation of the reward function. This approach not only provides the scalar reward for an action but also evaluates the stability of the actions of the agent, which represent the tuning parameters of the PID controller. The stability assessment is accomplished by monitoring the reward of the controller perturbed by the agent in comparison to the reference reward of a conservative and stabilising controller. If the reward of the perturbed controller is less than that of the reference controller by more than a preset factor, the perturbed controller is deemed potentially unstable, with poles in the right half plane, and is replaced by the conservative reference controller. The reward employs trajectory-based metrics such as ISE, as opposed to point-based metrics like overshoot or settling time, and is defined as

$$R_i(\boldsymbol{s}, \boldsymbol{a}) = -\sum_{t=0}^{T} e_t^2. \tag{31}$$

The trajectory of the ISE enables the evaluation of the perturbed performance of the controller throughout the episode, allowing for its replacement with the reference controller whenever instability is detected during the episode.

In the work of Lawrence et al. (2022, 2020a), the authors explore the performance of RL-controllers in SISO processes, using a reward function that depends on the tracking error and changes in the manipulated variable, given by

$$R_t(\boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1}) = -\left(|e_t|^p + \lambda\,|\Delta u_t|^q\right). \tag{32}$$

Here, p and q are chosen from the set $1, 2$, and $\lambda \geq 0$ remains fixed throughout training. The user can select the reward tuning parameter λ, with the authors recommending

$$\lambda = \frac{1}{|\Delta u|_{max}} \tag{33}$$

where $|\Delta u|_{max}$ is the maximum observed change to the actuator. Notably, larger changes to the manipulated variable lead to lower rewards, encouraging a more subtle use of the actuator. This approach can be extended to MIMO systems by applying the ℓ_1 or ℓ_2 norms to (32) and (33).

Setting $\lambda = 0$ yields a reward based on either the absolute or squared tracking error, depending on the choice of p. The absolute error prioritises small errors, while the squared error penalises larger errors more harshly. Opting for the absolute error results in slower responses but improved attenuation of overshoots. Conversely, the squared error leads to faster response times but greater overshoots.

The reward function in (32) provides an immediate reward for the action executed at time step t. To adapt it for use as an episodic reward in the ore milling circuit, it is modified as follows

$$\begin{aligned} R_i(\boldsymbol{s}, \boldsymbol{a}) = &-\sum_{t=0}^{T}\left(\boldsymbol{\omega}_e\,\mathrm{diag}\left(\boldsymbol{\beta}_e\right)^{-1}\boldsymbol{\phi}_{et}\right) - \\ &\sum_{t=0}^{T}\left(\boldsymbol{\omega}_u\,\mathrm{diag}\left(\boldsymbol{\beta}_u\right)^{-1}\boldsymbol{\phi}_{ut}\right) \end{aligned} \tag{34}$$

where t is sampled at 10-s intervals, which is sufficient to capture the system dynamics, and $T = 2$ h to allow the system to stabilise following a step change. The open-loop step test of u_{CFF} shows that y_{PSE} takes approximately 1 h to reach its steady-state value, therefore, setting $T = 2$ h provides sufficient time for poorly tuned controller dynamics to settle. Stability is not a prerequisite for the correct functioning of the reward; rather, the duration of T is selected to enable effective monitoring of the response of the tuning parameters. The row vectors $\boldsymbol{\omega}_e$ and $\boldsymbol{\omega}_u$ represent performance weights, $\boldsymbol{\beta}_e$ and $\boldsymbol{\beta}_u$ are factors on the diagonal of the scaling matrix, and $\boldsymbol{\phi}_{et}$ and $\boldsymbol{\phi}_{ut}$ are column vectors of the error tracking and control effort performance indices at time step t. The reward function (34) is based on principles from model predictive control (MPC), balancing dynamic performance, robustness, and practical actuator utilisation.

The error tracking performance index column vector for the milling circuit controller is defined as

$$\boldsymbol{\phi}_e = [e_{PSE}, e_{PSE_{SVOL}}, e_{PSE_{JT}}, e_{JT_{PSE}}, e_{JT_{SVOL}}, e_{JT}]^T \tag{35}$$

where

$$\begin{aligned} e_{PSE} &= \left(y_{PSE}^{sp} - y_{PSE}\right)^2 \\ e_{PSE_{SVOL}} &= \left(y_{SVOL}^{nom} - y_{SVOL}\right)^2 \\ e_{PSE_{JT}} &= \left(y_{JT}^{nom} - y_{JT}\right)^2 \\ e_{JT_{PSE}} &= \left(y_{PSE}^{nom} - y_{PSE}\right)^2 \\ e_{JT_{SVOL}} &= \left(y_{JT}^{nom} - y_{JT}\right)^2 \\ e_{JT} &= \left(y_{JT}^{sp} - y_{JT}\right)^2. \end{aligned} \tag{36}$$

Here e_{PSE} represents the tracking error for y_{PSE}, while $e_{PSE_{SVOL}}$ and $e_{PSE_{JT}}$ capture disturbance errors of y_{SVOL} and y_{JT} caused by interaction during the stepping of y_{PSE}. Similarly, $e_{JT_{PSE}}$, $e_{JT_{SVOL}}$, and e_{JT} represent disturbance and tracking errors related to the stepping of y_{JT}. The summation of $\boldsymbol{\phi}_e$ over the episode duration is equivalent to the ISE of the tracking error trajectory.

The scaling vector for the tracking performance index is

$$\begin{aligned} \boldsymbol{\beta}_e = \Big[&\min\left(\left|y_{PSE}^{nom} - y_{PSE}^{max}\right|, \left|y_{PSE}^{nom} - y_{PSE}^{min}\right|\right), \\ &\min\left(\left|y_{SVOL}^{nom} - y_{SVOL}^{max}\right|, \left|y_{SVOL}^{nom} - y_{SVOL}^{min}\right|\right), \\ &\min\left(\left|y_{JT}^{nom} - y_{JT}^{max}\right|, \left|y_{JT}^{nom} - y_{JT}^{min}\right|\right), \end{aligned}$$

$$\min\left(\left|y_{PSE}^{nom}-y_{PSE}^{max}\right|,\left|y_{PSE}^{nom}-y_{PSE}^{min}\right|\right),$$
$$\min\left(\left|y_{SVOL}^{nom}-y_{SVOL}^{max}\right|,\left|y_{SVOL}^{nom}-y_{SVOL}^{min}\right|\right),$$
$$\min\left(\left|y_{JT}^{nom}-y_{JT}^{max}\right|,\left|y_{JT}^{nom}-y_{JT}^{min}\right|\right)\Big] \tag{37}$$

where the elements of the vector represent the maximum permitted value of the controlled variable error. The number of scaling elements are determined by the size error tracking performance index $\boldsymbol{\phi}_e$ defined in (35).

The weight row vector to assign relative importance to the tracking indices is

$$\boldsymbol{\omega}_e = [1, 1, 10, 1, 1, 10] \tag{38}$$

where the weights of 10 penalise poor disturbance rejection and set-point tracking of y_{JT}.

The control effort index column vector is expressed as

$$\boldsymbol{\phi}_u = [\Delta u_{CFF_{PSE}}^2, \Delta u_{SFW_{PSE}}^2, \Delta u_{MFO_{PSE}}^2, \Delta u_{CFF_{JT}}^2, \Delta u_{SFW_{JT}}^2, \Delta u_{MFO_{JT}}^2]^T \tag{39}$$

where $\Delta u_{CFF_{PSE}}, \Delta u_{SFW_{PSE}}$ and $\Delta u_{MFO_{PSE}}$ denote the change in control effort for u_{CFF}, u_{SFW} and u_{MFO} during the stepping off y_{PSE} respectively. Similarly, $\Delta u_{CFF_{JT}}, \Delta u_{SFW_{JT}}$ and $\Delta u_{MFO_{JT}}$ refer to control effort changes during the stepping of y_{JT}. The squared form of the control effort aims to penalise significant changes made to the actuators, enhancing utility efficiency, and reducing mechanical wear.

The scaling vector for control effort indices is

$$\begin{aligned}\boldsymbol{\beta}_u = \Big[&\min\left(\left|u_{CFF}^{nom}-u_{CFF}^{max}\right|,\left|u_{CFF}^{nom}-u_{CFF}^{min}\right|\right),\\ &\min\left(\left|u_{SFW}^{nom}-u_{SFW}^{max}\right|,\left|u_{SFW}^{nom}-u_{SFW}^{min}\right|\right),\\ &\min\left(\left|u_{MFO}^{nom}-u_{MFO}^{max}\right|,\left|u_{MFO}^{nom}-u_{MFO}^{min}\right|\right),\\ &\min\left(\left|u_{CFF}^{nom}-u_{CFF}^{max}\right|,\left|u_{CFF}^{nom}-u_{CFF}^{min}\right|\right),\\ &\min\left(\left|u_{SFW}^{nom}-u_{SFW}^{max}\right|,\left|u_{SFW}^{nom}-u_{SFW}^{min}\right|\right),\\ &\min\left(\left|u_{MFO}^{nom}-u_{MFO}^{max}\right|,\left|u_{MFO}^{nom}-u_{MFO}^{min}\right|\right)\Big]\end{aligned} \tag{40}$$

where the elements of the vector represent the maximum permitted actuator movement.

The weight row vector for the control effort indices is

$$\boldsymbol{\omega}_u = [1, 10000, 1, 10000, 1, 1]. \tag{41}$$

where the weights of 10 000 penalise the control effort of u_{SFW}. Aggressive control of y_{SVOL} by manipulating u_{SFW} strongly interacts with y_{PSE}. This interaction is mitigated by heavily penalising the movements of u_{SFW}.

4.7. Simulation platform

The simulations were run using the RL toolbox of MATLAB on a 1.8 GHz dual-core Intel Core i5 processor with 8 GB of RAM, operating on macOS Monterey.

5. Results

5.1. Pre-training

During the pre-training phase, training is conducted by stepping to three different operating points, namely 95%, 97.5%, and 105% of the nominal plant reference values for the particle size estimate and mill load. The maximum reward differs for each of these operating conditions: −16.09 for the 95% case, −4.06 for the 97.5% case, and −20.29 for the 105% case. These differences in reward values are due to the varying error and actuator usage integrals accumulated at the three operating points over the training period. Fig. 5 illustrates the pre-training progress of the PPO reinforcement learning agent, with the normalised episodic reward plotted against the episode number. Normalisation of the episodic reward is necessary due to differences in the maximum rewards, and was achieved by dividing the reward at each operating point by its corresponding maximum reward. This also explains the convergence towards the maximum reward, which is conveniently represented by a value of −1 in Fig. 5.

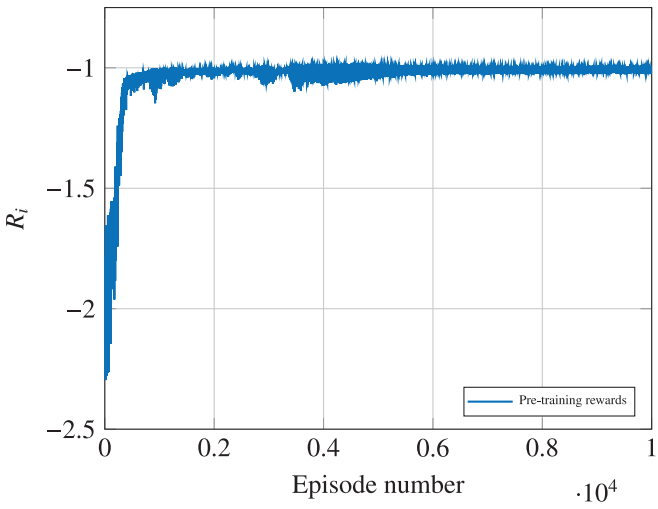

Fig. 5. PPO agent pre-training progress on the linear plant model.

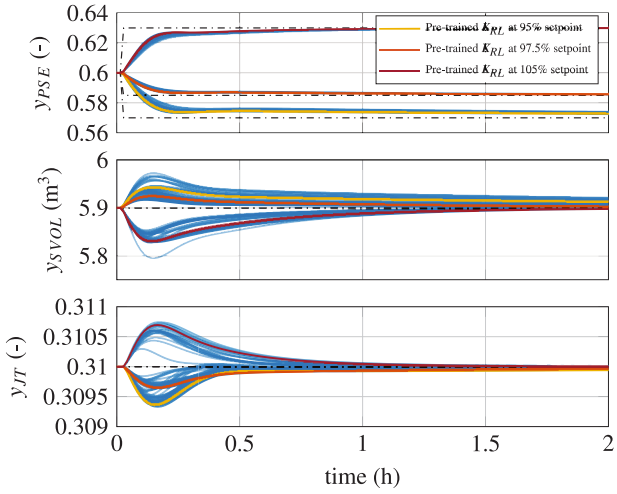

Fig. 6. Response of the controlled variables to a y_{PSE}^{sp} step change during pre-training. The training episodes are shown in blue, while the responses of the controller after completing pre-training, at setpoints of 95%, 97.5% and 105% of the nominal value of y_{PSE}, are shown in yellow, orange, and red, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

A steady increase in reward is observed over the first 1000 episodes, indicating effective learning. Between episodes 1000 and 4000, the reward exhibits some volatility, reflecting the ongoing exploration of the action space by the agent. After approximately episode 4000, the reward converges to a value near –1.0, suggesting that learning has stabilised with minimal further improvement. The absence of significant deviations from the stabilised reward range indicates that the safe search space ($\mathcal{A}_{\text{safe}}$), the learning rate and exploration parameters were appropriately selected, avoiding unstable iterations.

Figs. 6 and 7 depict the responses of the controlled and manipulated variables to various step changes applied to y_{PSE}^{sp} during pre-training on the linear grinding circuit model. The step changes, corresponding

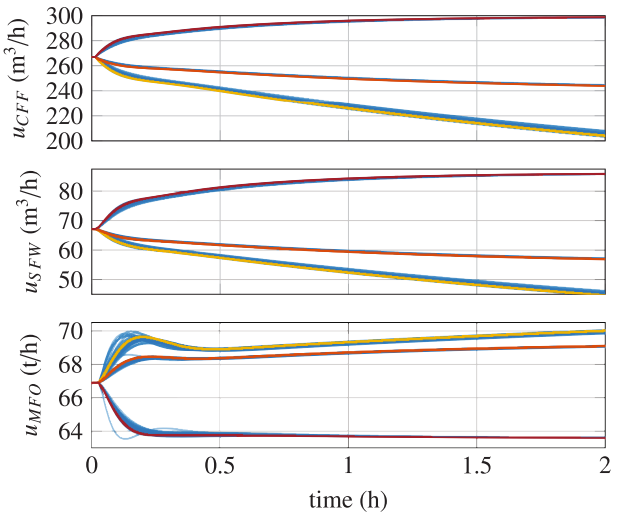

Fig. 7. Response of the manipulated variables to a y^{sp}_{PSE} step change during pre-training. The training episodes are shown in blue, while the responses of the controller after completing pre-training, at setpoints of 95%, 97.5% and 105% of the nominal value of y_{PSE}, are shown in yellow, orange, and red, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

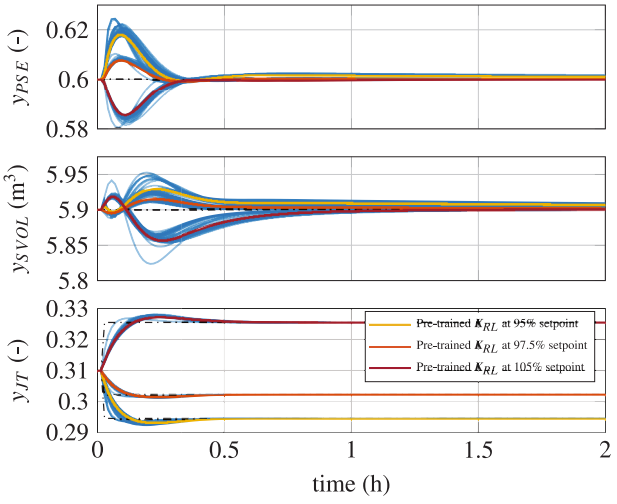

Fig. 8. Response of the controlled variables to a y^{sp}_{JT} step change during pre-training. The training episodes are shown in blue, while the responses of the controller after completing pre-training, at setpoints of 95%, 97.5% and 105% of the nominal value of y_{JT}, are shown in yellow, orange, and red, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

to 95%, 97.5%, and 105% of the nominal value of y_{PSE} are the fractions 0.57, 0.585, and 0.63, respectively. For clarity, only every 30th episode is displayed as well as the response of the controller on completion of pre-training. In reaction to an increased step change, u_{CFF} quickly adjusts to increase the fraction of particles within specification. This causes y_{SVOL} to drop, triggering an increase in u_{SFW} to prevent the sump from running dry. This sequence causes an increase in the hydrocyclone underflow, which results in a slight increase in y_{JT}, necessitating a reduction in u_{MFO} to bring y_{JT} back to the desired level.

Fig. 7 shows a gradual and sustained decrease in the values of u_{CFF} and u_{SFW}, as well as an increase in y_{JT}, to maintain the controlled variables at the setpoint following the 95% step change in y_{PSE}. Although not shown in the figure, the manipulated variables reach a steady state of approximately 150 m^3/h, 26 m^3/h and 72 t/h after 10 h for u_{CFF}, u_{SFW}, and y_{JT}, respectively. This slow adjustment of the manipulated variables to maintain the process at the setpoint does not manifest in the nonlinear plant model and results from the simplification required to identify a linear plant model.

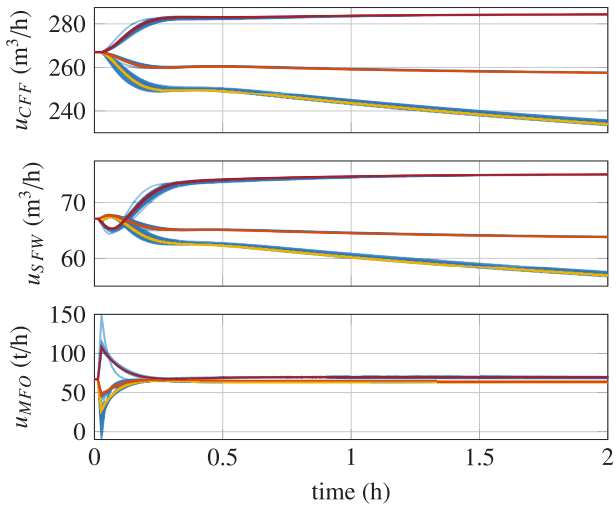

Fig. 9. Response of the manipulated variables to a y^{sp}_{JT} step change during pre-training. The training episodes are shown in blue, while the responses of the controller after completing pre-training, at setpoints of 95%, 97.5% and 105% of the nominal value of y_{JT}, are shown in yellow, orange, and red, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 3
Comparison of the tuning parameters for the pre-trained controller K_{RL} at varying operating setpoints. The setpoints are represented as percentages of the nominal operating conditions.

Parameter	95%	97.5%	105%
k_{P11}	21.568	21.541	21.46
τ_{I11}	0.00312	0.00312	0.00313
k_{P22}	85.119	85.165	85.291
τ_{I22}	0.1699	0.1681	0.1683
k_{P33}	3166	3150	3105
τ_{I33}	0.2684	0.2687	0.2696

Figs. 8 and 9 show the behaviour of the controlled and manipulated variables during pre-training as step changes are applied to the various operating points of y^{sp}_{JT}. The step changes correspond to fractions of 0.2945, 0.3023, and 0.3255. Following a rising step change in y^{sp}_{JT}, u_{MFO} promptly adjusts to increase the ore feed to the mill. Consequently, y_{PSE} drops below its setpoint, requiring u_{CFF} to be adjusted upwards to bring y_{PSE} back to its target value. As a result the sump level drops, which triggers an increase in u_{SFW} to prevent the sump from running dry.

To demonstrate the nonlinear capabilities of the pre-trained controller, Table 3 presents the slight variations in tuning parameters required for operation at 95%, 97.5%, and 105% of the nominal values of y_{PSE} and y_{JT}. The agent effectively functions as a look-up table, supplying optimal tuning parameters for each operating condition. While the milling circuit exhibits only mild nonlinearities, this approach would be particularly advantageous for processes with more pronounced nonlinear behaviour.

5.2. Final training

Given the slight variations in the tuning parameters presented in Table 3 for operation at 95%, 97.5%, and 105% of the nominal values of y_{PSE} and y_{JT}, further training of the agent at these specific operating points is unnecessary. The minimal differences in the tuning parameters

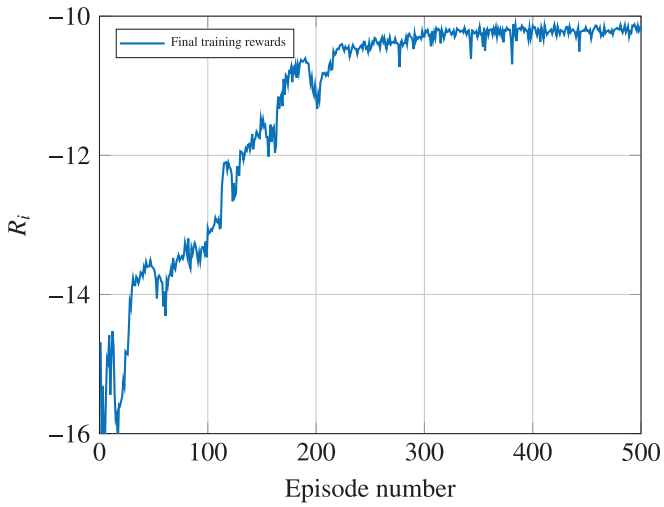

Fig. 10. PPO agent final training progress on the nonlinear plant model.

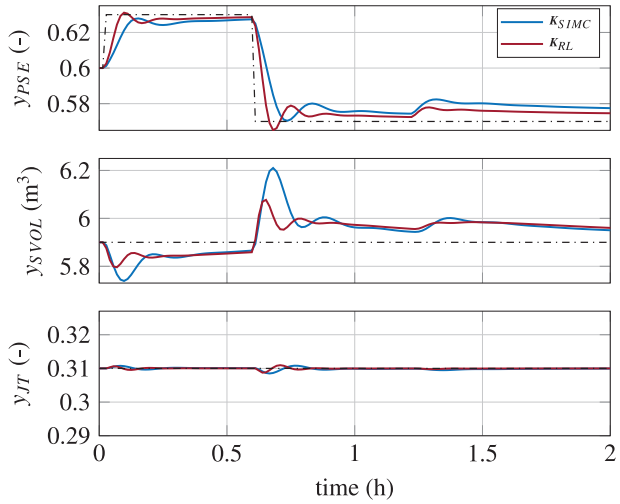

Fig. 11. Comparative performance of $\boldsymbol{K}_{SIMC}$ and $\boldsymbol{K}_{RL}$ to a y^{sp}_{PSE} step change.

indicate that training at a single operating point is adequate. Accordingly, for the final training phase, the operating point is set at 105% of the nominal values of y_{PSE} and y_{JT}.

Fig. 10 presents the final training phase of the PPO RL agent on the nonlinear process model. The plot shows a substantial increase in episodic rewards during the first 300 episodes, indicating effective adaptation to the nonlinear environment. Beyond episode 300, the curve flattens, suggesting convergence with minimal additional improvement. No unstable closed-loop responses were observed during the training episodes.

Stepping y^{sp}_{PSE} and y^{sp}_{JT} and observing the responses will require 4 h on an industrial grinding mill circuit. Therefore, completing 300 training episodes would take a minimum of 1200 h (50 days). To make such a long training period feasible for industry, training must occur during production without disrupting quality or throughput. In the case of the milling circuit, this can be achieved through small setpoint adjustments around a nominal setpoint value. Over time, these setpoint changes will average out to the nominal values required to maintain throughput. Training during production is feasible provided that the setpoint changes are large enough to rise above process noise but small enough to avoid disrupting quality.

Fig. 11 compares the performance of the reference controller $\boldsymbol{K}_{SIMC}$, as defined in (25) and (27), with the RL-tuned controller $\boldsymbol{K}_{RL}$

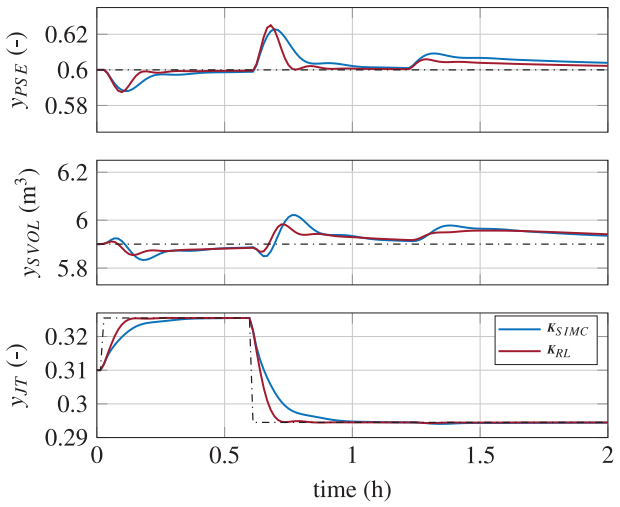

Fig. 12. Comparative performance of $\boldsymbol{K}_{SIMC}$ and $\boldsymbol{K}_{RL}$ to a y^{sp}_{JT} step change.

Table 4
Quantitative comparison of controllers K_{SIMC} and K_{RL} using the ISE metric.

ISE	K_{SIMC}	K_{RL}	Improvement (%)
Tracking of y^{sp}_{PSE}			
y_{PSE} tracking	0.099	0.056	43.5
y_{SVOL} disturbance rejection	6.119	3.849	37.1
y_{JT} disturbance rejection	8.7×10^{-5}	4.1×10^{-5}	53.5
Tracking of y^{sp}_{JT}			
y_{PSE} disturbance rejection	0.034	0.021	39.3
y_{SVOL} disturbance rejection	1.669	1.133	32.1
y_{JT} tracking	0.018	0.013	29.1

in response to step changes of 105% and 95% of the nominal value in y^{sp}_{PSE}. The first step to 105% occurs at 60 s, followed by a second step to 95% at 0.6 h. A 10% reduction in ore hardness is introduced as a disturbance at the 1.2-h mark. The top subplot illustrates the step changes applied to y^{sp}_{PSE}, while the middle and bottom subplots depict how the controllers reject the disturbances in y_{SVOL} and y_{JT} caused by controller interactions and the disturbance. Although there is potential to further improve the performance of $\boldsymbol{K}_{RL}$, this is impractical due to the increased actuator demand and wear. As a result, the reward function used for RL-tuning $\boldsymbol{K}_{RL}$ prioritises reduced control effort over maximum performance.

Fig. 12 presents a similar comparison between the reference controller $\boldsymbol{K}_{SIMC}$ and the RL-tuned controller $\boldsymbol{K}_{RL}$, but with a step changes applied to y^{sp}_{JT}. In this case, the bottom subplot shows the step changes in y^{sp}_{JT}, while the top and middle subplots illustrate the interactions and disturbance on y_{PSE} and y_{SVOL}, respectively.

As a quantitative comparison, Table 4 compares the ISE values of the controllers $\boldsymbol{K}_{SIMC}$ and $\boldsymbol{K}_{RL}$ in response to step changes in y^{sp}_{PSE} and y^{sp}_{JT}. The results demonstrate that the RL-tuned MIMO controller $\boldsymbol{K}_{RL}$ is not only successfully tuned but also achieves performance comparable to the reference controller $\boldsymbol{K}_{SIMC}$. Notably, setpoint tracking for y_{PSE} and y_{JT} improves by 43.5% and 29.1%, respectively. The findings demonstrate that reinforcement learning can be effectively applied to automatically tune diagonal controllers in a MIMO loop of a multivariable plant, achieving performance comparable to that of a manually tuned controller.

6. Conclusion

In conclusion, this article demonstrates the efficacy of RL for the automatic tuning of PID controllers in complex industrial environments such as the multivariable grinding mill circuit. By leveraging the PPO

algorithm, the RL agent successfully optimised the controller parameters at different operating points, resulting in improved closed-loop performance.

The proposed method offers a promising approach for real-time controller tuning in industrial processes, potentially enhancing productivity and product quality while reducing the need for manual intervention. The research contributes to the field by establishing a robust framework for applying RL in process control, designing effective reward functions, and constraining the agent to a safe operational space.

The reliability of the proposed method is evaluated in terms of performance, robustness, and stability. Simulation results demonstrate that the RL-tuned controller outperforms the manually tuned counterpart, particularly with respect to setpoint tracking and disturbance rejection. Robustness is promoted by incorporating a penalty on control effort within the reward function, thereby discouraging excessive actuator usage in favour of more balanced performance. Stability is grounded in the structured singular value analysis of the linear model and further validated through training on the nonlinear model.

PID tuning often requires repeated intervention from control engineers to adjust controller settings as operating conditions evolve. In contrast, RL-based tuning shifts this burden to a once-off design and training phase, during which domain expertise is applied to define the reward structure and select appropriate learning parameters during pre-training. Once trained, the RL agent can autonomously re-tune the controller on demand, without requiring further expert involvement.

Future work will focus on comparing the results with more sample efficient learning methods such as Bayesian optimisation.

CRediT authorship contribution statement

J.A. van Niekerk: Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **J.D. le Roux:** Writing – review & editing, Supervision, Methodology, Conceptualization. **I.K. Craig:** Writing – review & editing, Supervision, Methodology, 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

Åström, K. J., & Hägglund, T. (1995). *PID controllers: Theory, design, and tuning* (2nd ed.). Instrumentation Society of America.

Åström, K. J., & Hägglund, T. (2004). Revisiting the Ziegler–Nichols step response method for PID control. *Journal of Process Control*, *14*(6), 635–650.

Bauer, M., Auret, L., Bacci di Capaci, R., Horch, A., & Thornhill, N. F. (2019). Industrial PID control loop data repository and comparison of fault detection methods. *Industrial & Engineering Chemistry Research*, *58*(26), 11430–11439.

Bennett, S. (2001). The past of PID controllers. *Annual Reviews in Control*, *25*, 43–53.

Boubertakh, H., Tadjine, M., Glorennec, P.-Y., & Labiod, S. (2010). Tuning fuzzy PD and PI controllers using reinforcement learning. *ISA Transactions*, *49*(4), 543–551.

Bristol, E. (1966). On a new measure of interaction for multivariable process control. *IEEE Transactions on Automatic Control*, *11*(1), 133–134.

Carlucho, I., De Paula, M., & Acosta, G. G. (2020). An adaptive deep reinforcement learning approach for MIMO PID control of mobile robots. *ISA Transactions*, *102*, 280–294.

Chen, X., Zhai, J., Li, S., & Li, Q. (2007). Application of model predictive control in ball mill grinding circuit. *Minerals Engineering*, *20*(11), 1099–1108.

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.

Craig, I. K., Hulbert, D. G., Metzner, G., & Moult, S. P. (1992). Extended particle-size control of an industrial run-of-mine milling circuit. *Powder Technology*, *73*(3), 203–210.

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.

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

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

Dogru, O., Wieczorek, N., Velswamy, K., Ibrahim, F., & Huang, B. (2021). Online reinforcement learning for a continuous space system with experimental validation. *Journal of Process Control*, *104*, 86–100.

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

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.

Garpinger, O., Hägglund, T., & Åström, K. J. (2014). Performance and robustness trade-offs in PID control. *Journal of Process Control*, *24*(5), 568–577.

Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., et al. (2024). A review of safe reinforcement learning: Methods, theories and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Lakhani, A. I., Chowdhury, M. A., & Lu, Q. (2022). Stability-preserving automatic tuning of PID control with reinforcement learning. *Complex Engineering Systems*, *2*(3), 1135–1149.

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*, Article 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., & Steyn, C. W. (2022). Validation of a dynamic non-linear grinding circuit model for process control. *Minerals Engineering*, *187*, Article 107780.

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.

Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on world wide web* (pp. 661–670).

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Ma, Y., Zhu, W., Benton, M. G., & Romagnoli, J. (2019). Continuous control of a polymerization system with deep reinforcement learning. *Journal of Process Control*, *75*, 40–47.

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*, Article 100131.

MATLAB (2023). *R2023b Update 7*. Natick, Massachusetts: The MathWorks Inc..

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

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*, Article 106886.

Noome, Z. M., le Roux, J. D., & Padhi, R. (2023). Optimal control of mineral processing plants using constrained model predictive static programming. *Journal of Process Control*, *129*, Article 103067.

Olivier, L. E., & Craig, I. K. (2017). A survey on the degree of automation in the mineral processing industry. In *2017 IEEE AFRICON* (pp. 404–409). IEEE.

OpenAI (2025). Proximal policy optimization. URL https://spinningup.openai.com/enlatest/algorithms/ppo.html. (Accessed 30 January 2025).

Pongfai, J., Su, X., Zhang, H., & Assawinchaichote, W. (2020). PID controller autotuning design by a deterministic Q-SLP algorithm. *IEEE Access*, *8*, 50010–50021.

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

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 preprint 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.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *International conference on machine learning* (pp. 387–395). Proceedings of Machine Learning Research.

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

Skogestad, S., & Postlethwaite, I. (2007). *Multivariable feedback control: Analysis and design* (2nd ed.). Wiley.

Spielberg, S., Tulsyan, A., Lawrence, N. P., Loewen, P. D., & Bhushan Gopaluni, R. (2019). Toward self-driving processes: A deep reinforcement learning approach to control. *AIChE Journal*, *65*(10), Article e16689.

Stewart, G. E. (2012). A pragmatic approach to robust gain scheduling. *IFAC Proceedings Volumes*, *45*(13), 355–362.

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*.

Thorndike, E. L. (1898). Animal intelligence: An experimental study of the associative processes in animals. *The Psychological Review: Monograph Supplements*, *2*(4).

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*, Article 103008.

Wang, X.-S., Cheng, Y.-H., & Wei, S. (2007). A proposal of adaptive PID controller based on reinforcement learning. *Journal of China University of Mining and Technology*, *17*(1), 40–44.

Wang, Y., He, H., & Tan, X. (2020). Truly proximal policy optimization. In R. P. Adams, & V. Gogate (Eds.), *Proceedings of machine learning research*: *vol. 115*, *Proceedings of the 35th uncertainty in artificial intelligence conference* (pp. 113–122). Proceedings of Machine Learning Research.

Xue, W., Fan, J., Lopez, V. G., Li, J., Jiang, Y., Chai, T., et al. (2019). New methods for optimal operational control of industrial processes using reinforcement learning on two time scales. *IEEE Transactions on Industrial Informatics*, *16*(5), 3085–3099.

Ziegler, J. G., & Nichols, N. B. (1942). Optimum setting for automatic controllers. *Transactions of the ASME*, *64*, 759–768.