2025 25th International Conference on Process Control (PC)
June 3–6, 2025, Štrbské Pleso, Slovakia

# Robust Parameter Identification via Reinforcement Learning

Ehsan Hatami
*Institute of Automation and Control*
*Graz University of Technology*
Graz, Austria
ehsan.hatami@tugraz.at

Martin Steinberger
*Institute of Automation and Control*
*Graz University of Technology*
Graz, Austria
martin.steinberger@tugraz.at

***Abstract*—Identifying the parameters of a dynamic model under real-time operating conditions is a challenging task, especially when dealing with uncertain disturbances. This study introduces a reinforcement learning-based approach for robust parameter estimation, enabling the determination of optimal model parameters directly from raw data. The proposed method operates as an online parameter identification algorithm, eliminating the need for labeled training datasets or observation windows. It demonstrates superior performance compared to classical methods, effectively rejecting unknown disturbances in dynamic processes and adapting seamlessly to new environments without requiring historical data. The algorithm's robustness and effectiveness are validated through simulations focused on acquiring accurate parameter values.**

***Index Terms*—reinforcement learning, proximal policy optimization, parameter estimation.**

## I. Introduction

Mathematical models are the building blocks of modern systems in the area of engineering, representing information and providing the foundation for decision-making and control. These models play a significant role in many applications, including chemical process control, manufacturing systems, and aerospace engineering, as well as smart buildings [1]. The increasing availability of computational resources has enabled the use of complex models in real-time decision-making, paving the way for autonomous systems. However, many of these systems exhibit nonlinear and time-varying behaviors, necessitating continuous adjustments to model parameters during operation. Addressing this challenge is vital for achieving high performance and adaptability in dynamic and uncertain environments. Traditional parameter estimation techniques often involve methods like gradient-based optimization [2], statistical modeling [3], and recursive prediction error minimization [4]. While these methods have been successfully applied to systems of manageable complexity, they face significant limitations in highly nonlinear, time-varying systems. Approaches such as extended Kalman filters (EKF) [5], unscented Kalman filters (UKF) [6], and particle filters [7] address some of these challenges but are computationally expensive or require simplifying assumptions about system dynamics. These limitations make them less suitable for modern, complex, and dynamic engineering applications. To overcome these challenges, researchers have turned to data-driven and machine learning approaches. Supervised learning methods offer solutions for parameter estimation but are constrained by the need for large datasets and struggle to generalize outside predefined conditions [8]. Furthermore, noisy measurements frequently lead to suboptimal performance in tracking model parameters. Reinforcement learning (RL) has emerged as a promising alternative, offering the ability to learn complex tasks with high sample efficiency [9]. Unlike supervised methods, RL enables parameter estimation policies that adapt dynamically to changes in system behavior without requiring extensive offline training datasets or intrusive system perturbations. The benefits of RL are particularly relevant for nonlinear and time-varying systems, where traditional methods fall short. By formulating parameter estimation as a sequential decision-making problem, RL provides a robust framework for exploring and exploiting parameter spaces. Existing works have demonstrated the application of RL in calibration tasks [10], parameter estimation for chemical process [11], and forging machines [12]. However, challenges remain in ensuring convergence, robustness, and scalability to broader contexts.

This paper focuses on leveraging Reinforcement Learning (RL) for online parameter estimation, a key challenge in dynamic systems, where real-time adaptability and robustness are essential. Specifically, we propose an automated RL-based approach utilizing the Proximal Policy Optimization (PPO) method to estimate parameters from limited and disturbed observations [13]. PPO, a policy-gradient algorithm, is particularly well-suited for this task due to its ability to balance exploration and exploitation while maintaining stability during training. The proposed method incorporates disturbance rejection mechanisms directly into the learning framework, enabling the RL agent to minimize the impact of disturbances on the parameter estimation process. This ensures a more robust and consistent estimation in scenarios with high variability or unpredictable disturbance. To evaluate its effectiveness, we test the approach on a rotary flexible joint, a representative system for dynamic and disturbance-prone mechanism.

The paper is organized as following: Section II describes the RL's procedure and releases the proposed approach. Section III presents the model of rotary flexible joint. In Section IV, the model parameters are elaborated by the proposed approach

979-8-3315-2530-9/25/$31.00 ©2025 IEEE

and comparisons are made with a classical method. Finally, conclusions are drawn in Section V.

## II. Proposed Method

This section presents the main algorithm and the key components of the RL problem.

### A. Proximal Policy Optimization

In reinforcement learning, policy gradient-based algorithms present an excellent framework for addressing problems in continuous action spaces. The primary objective of these algorithms is to maximize the cumulative expected rewards, represented as $L = \mathbb{E}\left[\sum_t r(s_t, a_t)\right]$ where r is the immediate reward at time $t$, $s_t$ is the observation at time $t$ and $a_t$ is the action at time $t$. The gradient of the objective function $L$, incorporating a baseline, is commonly expressed

$$\begin{aligned} \nabla_\phi L &= \int_S \mu(s) \int_{\mathcal{A}} \nabla_\phi \pi_\phi(a|s) A(s,a) \\ &= \mathbb{E}\left[\nabla_\phi \log \pi_\phi(a|s) A(s,a)\right] \end{aligned} \tag{1}$$

where $\phi$ represents the parameters of policy, $S$ denotes the set of all states and $\mathcal{A}$ denotes the set of all actions, respectively. $\mu(s)$ is an on-policy distribution over states. $\pi$ is a stochastic policy that maps state $s \in S$ to action $a \in \mathcal{A}$, and $A$ is an advantage function.

Trust region policy optimization (TRPO) [14] replaces the objective function $L$ with a surrogate objective and constrains the divergence between the new policy and the fixed, previous policy. This is achieved by solving the following optimization problem:

$$\begin{aligned} \underset{\phi}{\text{maximize}} \quad & \mathbb{E}\left[\frac{\pi_\phi(a|s)}{\pi_{\phi_{\text{old}}}(a|s)} A(s,a)\right] \\ \text{subject to} \quad & \mathbb{E}\left[\text{KL}\left(\pi_{\phi_{\text{old}}}(\cdot|s), \pi_\phi(\cdot|s)\right)\right] \leq \delta \end{aligned} \tag{2}$$

where, the previous policy, denoted as $\pi_{\phi_{\text{old}}}$, remains fixed during training and is responsible for generating actions. The goal is to optimize the current policy, $\pi_\phi$, while ensuring that updates are not overly aggressive. To achieve this, an upper bound $\delta$ is imposed to constrain the Kullback-Leibler divergence between $\pi_\phi$ and $\pi_{\phi_{\text{old}}}$.

Fig. 1 shows the overall process of an Actor–Critic PPO algorithm performed by an agent whenever every episode ends. First, the Actor–Critic PPO obtains a finite mini-batch of sequential samples (i.e., experience tuples) from the trajectory memory. In a mini-batch, the first data point is selected randomly, but all following data points must come in order, one after another, without skipping any steps. PPO can be treated as an approximated but much simpler version of TRPO. It roughly clips the ratio between $\pi_{\phi_{\text{old}}}$ and $\pi_\phi$ and changes the surrogate objective function into the form:

$$y_L^{\text{CLIP}}(\phi) = \mathbb{E}\left[\min\left(r(\phi)A,\ \text{clip}\left(r(\phi), 1-\epsilon, 1+\epsilon\right) A\right)\right] \tag{3}$$

where $r(\phi) = \frac{\pi_\phi(a|s)}{\pi_{\phi_{\text{old}}}(a|s)}$ and $\epsilon$ is the clipping bound. Note that PPO here is an actor-critic style algorithm, where the actor is the policy $\pi_\phi$ and the critic is the advantage function $A$.

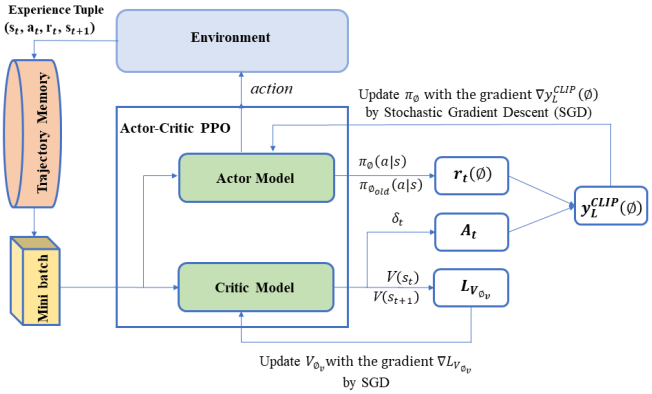


Fig. 1. Architecture of PPO model

The advantage function intuitively measures how good a state-action pair is relative to the average value of the current state. Specifically, $Q(s,a)$ represents the expected cumulative reward for taking action $a$ in state $s$ while $V(s)$ represents the expected return for being in state $s$. The advantage function is then defined as $A(s,a) = Q(s,a) - V(s)$. Generalized Advantage Estimation (GAE) [15] is the most commonly used technique for computing the advantage function. This approach is particularly suitable for PPO and other policy gradient algorithms and can be formulated as follows:

$$\begin{aligned} A_t(s_t, a_t) = r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} \\ + \gamma^{T-t} V(s_T) - V(s_t) \end{aligned} \tag{4}$$

where $T$ represents the predefined maximum trajectory length rather than the final time step of an episode and $\gamma$ denotes the discount factor. To prevent bootstrapping, the value function $V(s_T)$ is set to zero if the episode terminates. Under such conditions, the advantage function is defined as $A_t = G_t - V(s_t)$, where $G_t$ is the discounted return following time $t$, as established in [9]. Alternatively, a more generalized form of GAE can be utilized:

$$\begin{aligned} A_t(s_t, a_t) = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots \\ + (\gamma\lambda)^{T-t+1} \delta_{T-1} \end{aligned} \tag{5}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, which is known as the Temporal Difference (TD) error. Notably, setting $\lambda = 1$ reduces this equation to (4), introducing high variance, while choosing $\lambda = 0$ reduces it to $\delta_t$ which minimizes variance but introduces bias [9].

To approximate the advantage function, it suffices to estimate the state value function $\hat{V}_{\phi_v}(s)$ instead of $\hat{Q}(s,a)$, where $\hat{\delta}_t = r_t + \gamma \hat{V}_{\phi_v}(s_{t+1}) - \hat{V}_{\phi_v}(s_t)$. Instead of optimizing the surrogate objective, we aim to minimize the loss function.

$$\begin{aligned} L_{V_{\phi_v}} = \big(r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} \\ + \gamma^{T-t} \hat{V}_{\phi_v}^{\text{old}}(s_T) - \hat{V}_{\phi_v}(s_t)\big)^2 \end{aligned} \tag{6}$$

where $\hat{V}_{\phi_v}^{\text{old}}$ denotes the fixed value function utilized in computing $\hat{\delta}_t$.

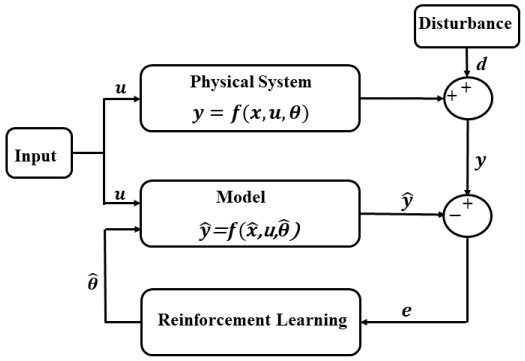


Fig. 2. Employing reinforcement learning for parameter estimation

### B. RL for Parameter Estimation

This section formulates the RL approach to estimate the parameters $\theta$ of a dynamic system model by minimizing the error between the physical system output and the model output. The physical system output $y$ and model output $\hat{y}$ are represented as

$$y = f(\theta, x, u) + d \tag{7}$$

$$\hat{y} = f(\hat{\theta}, \hat{x}, u) \tag{8}$$

with real valued parameters $\theta$, states $x$, input $u$ of the system and an external disturbance $d$. $\hat{y}$ denotes the output responses from the simulator with the estimated parameter values of $\hat{\theta}$ and estimated states $\hat{x}$. The estimation policy $\pi$ aims to learn a strategy that determines how parameter adjustments can minimize the model error $e = y - \hat{y}$. Fig. 2 illustrates the proposed parameter estimation method.

Optimizing parameter values $\hat{\theta}$ to minimize the model error $e$ toward zero can be framed as a sequential decision-making problem. In this context, each sequence of parameter adjustments leads to a new state, characterized by the model's deviations. The choice of the next parameter estimate relies solely on the current information available at each step, making the process approximately Markovian—an essential property for applying Reinforcement Learning (RL) methods. This Markovian property is valid when observations $o$ of the environment are sufficiently informative to guide the decision for the next action $\hat{\theta}$.

In RL, the environment is defined at discrete time steps $t$, where the agent receives observations $o_t$ and selects actions $\hat{\theta}_t$. These actions lead to transitions to new states $s_{t+1}$ accompanied by scalar rewards $r_t$ that provide feedback on the agent's performance.

Note that, in this study, our strategy to guarantee that the parameter values are within physically possible ranges is to clip the action values when the estimated parameter values are out-of-bound. This clipping in turn ensures the stability and reliability of our model estimation. The clipping function is formulated as:

$$\hat{\theta}_{\text{clip}} = \min\left(\max(\hat{\theta}, \hat{\theta}_{\text{min}}), \hat{\theta}_{\text{max}}\right) \tag{9}$$

### C. Agent Observations

The proposed RL agent aims to minimize the model's prediction error by incorporating information from the physical system output $y$ and the model output $\hat{y}$. For this purpose, the model error $e$ is selected as the primary observation. Additionally, it has been observed that including the integral and derivative of the prediction error in the observation space provide the RL agent with richer information about the system's dynamic behavior.

Note that, state transitions are dictated by the physical system, which is presumed to be unknown to the RL agent. The agent has access only to the observed output of the system. In this study, the physical system is simulated using a "ground truth" model.

### D. Robust Disturbance Rejection Reward

The reward function plays a critical role in RL as it serves as the guiding principle for the agent to learn and improve its behavior. By quantifying the desirability of specific actions or outcomes, the reward function shapes the agent's decision-making process, ensuring it aligns with the task's objectives. In this work, the designed reward function is formulated as

$$R = -\alpha_1 e^2 - \sum_i \beta_i (\hat{\theta}^i - \hat{\theta}^i_{\text{clip}})^2 - \sum_i \mu_i (\hat{\theta}^i_t - \hat{\theta}^i_{t-1})^2 - \alpha_2 \sigma_e^2 \tag{10}$$

where $i$ is the number of estimated parameters, and $\alpha_1$,$\beta_i$, $\mu_i$ and $\alpha_2$ are weighting factors. The first term penalizes the squared error $e^2$ ensuring the agent focuses on minimizing the discrepancy between system output and model output. The second term penalizes the squared difference between $\hat{\theta}$ and a predefined clipping threshold $\hat{\theta}_{\text{clip}}$. This prevents the agent from taking actions that result in parameters outside the allowed range. The third term penalizes the squared difference between the current parameter estimate $\hat{\theta}_t$ and $\hat{\theta}_{t-1}$. This discourages the agent from taking abrupt parameter updates, promoting smoother transitions in the parameter estimation process. The fourth term penalizes the variance of error signal, $\sigma_e^2$ which reflects the impact of disturbances on the system. Penalizing the variance in the reward function allows the RL agent to minimize the effects of disturbances. The variance captures the degree of fluctuation in the error signal, with high variance indicating significant disturbances and low variance reflecting consistency.

To compute an approximation of the error variance, we use a sliding window approach. A window of size $N$ is defined to capture the last $N$ error values during the agent's interaction with the system. The mean error $\mu_e$ is calculated over a $N$-value buffer. Using this, the variance of the error signal is approximated by

$$\sigma_e^2 = \frac{1}{N} \sum_{i=1}^{N} (e_i - \mu_e)^2 \tag{11}$$

which quantifies how much the error fluctuates over time. By incorporating this term into reward function, the agent learns

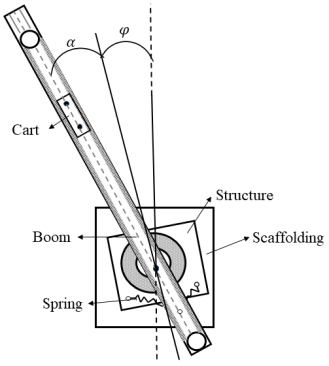


Fig. 3. Rotary flexible joint platform.

to minimize both the error magnitude ant its variability, promoting disturbance rejection and higher system performance.

The weighting factors should be carefully tuned to avoid disproportionately emphasizing any single term. Proper tuning ensures that the main part of the reward function $e^2$ remains dominant while still allowing the additional terms to effectively capture their respective impacts on estimation accuracy.

## III. Case Study

### A. Rotary Flexible Joint

In Fig. 3, the physical model of the rotary flexible joint is shown. The basis of the model is the scaffolding that supports the rotating structure. An electric drive, which is designed to rotate the structure, is attached to the scaffolding. The structure is mounted on the gear shaft of this drive. The rotatably mounted boom, which is additionally stabilized by two springs, is connected to it. Furthermore, an incremental angle encoder for measuring the boom angle and a structure for attaching the springs to the boom are also installed. Since the boom is rotatably mounted, it can be deflected by an angle $\alpha$ relative to the structure. The two springs generate a restoring moment that brings the boom back to its rest position. The angle between the scaffolding and the superstructure is referred to as $\varphi$. The boom is formed by guide rail, along which the cart can be positioned.

### B. Mathematical Model

Based on the angular momentum theorem, the corresponding differential equation of motion can be set up for both the structure and the boom for an external driving torque $\mathrm{M}_A$

$$\frac{d}{dt}\left(J_A(\dot{\alpha}+\dot{\varphi})\right)+k_s\alpha=0 \tag{12}$$

$$\frac{d}{dt}\left(J_A(\dot{\alpha}+\dot{\varphi})+J_\mathrm{S}\dot{\varphi}\right)+\beta\dot{\varphi}=M_A \tag{13}$$

where $J_A$ is the moment of inertia of the boom, $J_\mathrm{S}$ is the moment of inertia of the structure, $k_s$ is the torsional spring stiffness, $\alpha$ is the deflection angle of the boom, $\varphi$ is the angular displacement, $\beta$ is the damping coefficient, and $M_A$ is the applied torque.

The moment of inertia of the boom $J_A$ can be adjusted by varying the cart position and is defined as $J_A = J_B + m_W l^2$ , where $J_B$ is a constant component, $m_W$ is the mass of cart, and $l$ represents its position on the boom. The time derivative of the moment of inertia is given by $\dot{J}_A = 2m_W\dot{l}$.

The external moment $M_A$, which acts on the structure, is proportional to the armature current of the electric drive. This proportionality is described by the torque constant $k_m$. Since a gearbox is used, the gear ratio $k_g$ must also be taken into account. The direction of the angles $\alpha$ and $\varphi$ was chosen to be opposite to the direction of rotation of the drive. This is taken into account here by the negative sign: $M_A = -ik_gk_m$.

Neglecting the motor inductance, the armature current is $i = \frac{u}{R_m} + \frac{k_gk_m}{R_m}\dot{\varphi}$.

This results in the torque: $M_A = -\frac{k_gk_m}{R_m}u - \frac{k_g^2k_m^2}{R_m}\dot{\varphi}$, where $R_m$ is the motor resistance.

This results in the following differential equations of motion:

$$\ddot{\varphi} = \frac{k_s}{J_S}\alpha + \left(-\frac{k_m^2k_g^2}{R_mJ_S} - \frac{\beta}{J_S}\right)\dot{\varphi} - \frac{k_mk_g}{R_mJ_S}u \tag{14}$$

$$\ddot{\alpha} = -\left(\frac{k_s}{J_S}+\frac{k_s}{J_A}\right)\alpha + \left(\frac{k_m^2k_g^2}{R_mJ_S}+\frac{\beta}{J_S}-\frac{J_A}{J_S}\right)\dot{\varphi} - \frac{\dot{J}_A}{J_A}\dot{\alpha} + \frac{k_mk_g}{R_mJ_S}u \tag{15}$$

After introducing a state vector of the form $\mathbf{x} = (\varphi \quad \alpha \quad \dot{\varphi} \quad \dot{\alpha})^T$, we can derive the fourth order state space model of system. In this model, $\varphi$ and $\alpha$ are system outputs.

The used model parameters and data are summarized in Table I.

TABLE I
MODEL PARAMETERS

| system parameter | value |
|---|---|
| $k_s$ | 2.88 Nm/rad |
| $J_\mathrm{S}$ | $94386 \times 10^{-6}\ kgm^2$ |
| $J_B$ | $43789 \times 10^{-6}\ kgm^2$ |
| $K_m$ | 0.0209 Nm/A |
| $K_g$ | 134 |
| $R_m$ | 1.34 Ω |
| $\beta$ | 0.45 Nms/rad |
| $m_W$ | 0.08 $kg$ |

Since many real-world plants have actuators which can only be updated at a fixed sampling frequency, we discretize the continuous-time model using Zero-Order Hold (ZOH) method. The parameter estimation is then performed on the discrete-time model, ensuring consistency between the estimated parameters and the system's sampled behavior.

## IV. Simulation and Results

In the discretized model, we conducted a sensitivity analysis to evaluate the influence of each parameter on the system's output. This analysis identified three parameters as significantly more impactful than the others, making them the

primary candidates for estimation. However, due to the nature of this discretized system dynamics, two of these parameters consistently maintain the same value. As a result, a single estimation value is used to represent both parameters, simplifying the estimation process without compromising accuracy. This approach reduces redundancy while maintaining consistency with the system's inherent dynamics. Furthermore, although the system has two outputs, $\varphi$ is not sufficiently sensitive to the system parameters, and thus only $\alpha$ is considered for estimation. Furthermore, for parameter estimation, we used a pseudo-random signal as control input to excite the system dynamics effectively.

The PPO network consists of two hidden layers, each containing 64 neurons with tanh activations for the learned parameters. Regularization is applied to prevent overfitting. The agent's parameters are optimized using the GAE advantage function with a learning rate of 0.0001. For each training episode, 100 rollouts of one timestep (0.1 s) each is collected. Key hyperparameters include a discount factor $\gamma$ of 0.99 and a batch size of 64. The trained agent's performance is validated using a separate dataset distinct from the training data. This is ensured by generating random sets of model parameters with seeds different from those used in training.

We conducted experiments to evaluate the performance of our proposed model under two distinct scenarios to analyze its robustness and disturbance rejection capability. In the first case, we estimated the parameters without introducing any disturbances to the physical system output, providing a baseline for comparison. In the second case, we introduced a correlated disturbance to the system output and tested the performance of our method incorporating disturbance rejection term into the reward function. This scenario demonstrated the full potential of our approach in handling disturbances while maintaining high parameter estimation accuracy.

### A. Case A

We first evaluated our approach in the absence of disturbances to the physical system output. The agent was trained for 350 episodes to optimize estimated parameters. We observed that incorporating the second and third terms in the reward function significantly reduced the number of training episodes required for convergence. Fig. 4 illustrates the training process and the reward signal progression, highlighting the agent's learning dynamics and convergence. For comparison, we employed the Prediction Error Minimization (PEM) [16] approach, a widely used method in system identification. The PEM method estimates model parameters by minimizing discrepancy between the physical system output and predicted output based on the model. By iteratively optimizing the parameter set, PEM ensures that the identified model best fits the observed data, providing a benchmark to evaluate the performance of our proposed estimation approach. Fig. 5 demonstrates that both PPO and PEM achieved high accuracy, with estimated parameter values closely following their reference values.

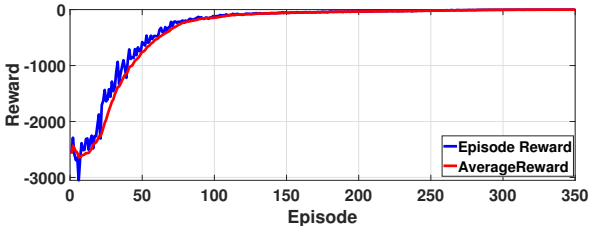


Fig. 4. Training process with PPO

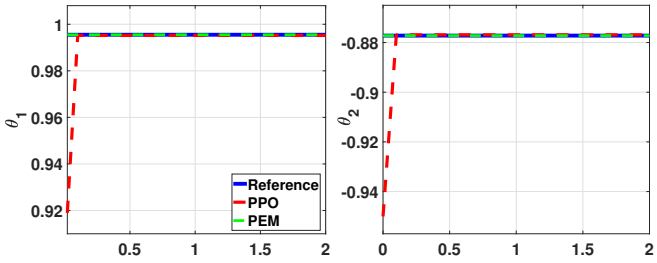


Fig. 5. Parameter estimation results in the case without disturbance

### B. Case B

In the second scenario, we evaluated our method in the presence of an external disturbance affecting the system output. Unlike uncorrelated disturbances, which can often be mitigated more easily, we considered a correlated disturbance—one of the most challenging forms of disturbances in system identification. Correlated disturbances introduce dependencies over time, making parameter estimation more difficult by masking the true system dynamics. The applied disturbance is modeled as a combination of white noise, a sinusoidal signal, and a constant offset, given by

$$d(t) = w(t) + A\sin(2\pi ft) + C \tag{16}$$

where $w(t)$ represents white noise, $A\sin(2\pi ft)$ is a sinusoidal disturbance with amplitude $A$ and frequency $f$ and $C$ is a constant offset.

To evaluate the algorithm's performance, we first train the agent for 600 episodes in the presence of disturbances without incorporating disturbance rejection term in the reward function. As shown in Fig. 6, the estimated value of the first parameter ($\theta_1$) exhibits small discrepancies and fluctuations due to disturbances, while the second parameter's estimation ($\theta_2$) is significantly worse, deviating further from the reference value. To enhance accuracy, we introduce a disturbance rejection term in the reward function and train the agent using two different window sizes—one small $N = 5$ and one large $N = 50$—to assess their impact. With $N = 5$, the agent tracks reference values more accurately for both $\theta_1$ and $\theta_2$, albeit with higher fluctuations. In contrast, for $N = 50$ the estimation is smoother but less accurate. The reason behind is that for the small window, the agent reacts quickly to the changes, capturing disturbances and rapid variations, leading to higher accuracy but increased fluctuations. In contrast, a larger win-

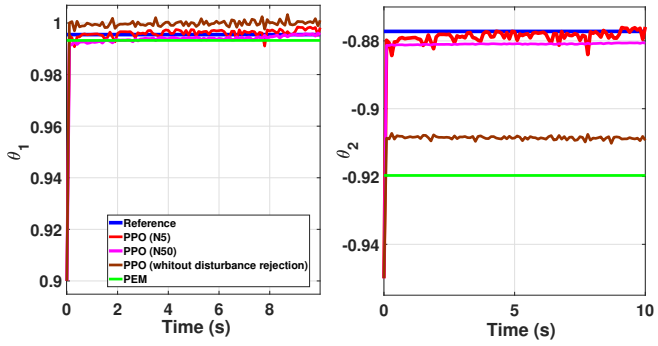


Fig. 6. Parameter estimation results in presence of external disturbance

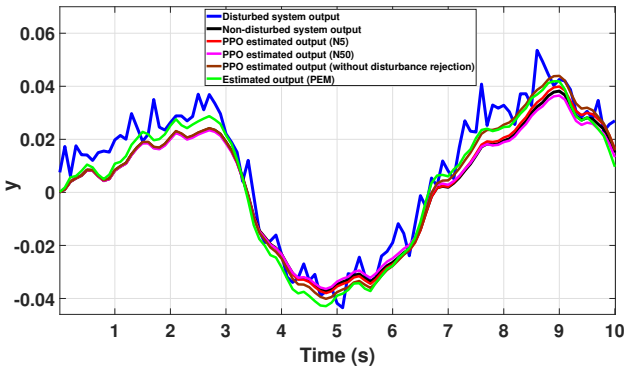


Fig. 7. Comparison of estimated outputs: effectiveness of disturbance rejection in parameter estimation

dow averages out disturbances over a longer period, resulting in smoother behavior but lower accuracy due to lag in adapting to parameter changes. Finally, we compare our approach with the Prediction Error Method (PEM). While PEM struggles to estimate $\theta_1$ accurately and performs even worse for $\theta_2$ our method successfully estimates both parameters dynamically during training.

It should be noted that, as previously mentioned, due to the nature of the discretized system dynamics, two of the three selected parameters for estimation consistently maintain the same value. In our model, we estimated only one of these two similar parameters and used the same value for both. However, PEM does not allow this approach; it requires either estimating both parameters separately or fixing one and estimating the other. In our study, we opted for the latter approach, while estimating both parameters simultaneously led to worse performance.

Finally, we compared the disturbed and non-disturbed outputs of physical system with estimated outputs across the considered parameter estimation approaches, see Fig. 7. Among these approaches, the PEM method exhibits the weakest performance, as its estimated output closely follows the disturbed output, indicating poor disturbance mitigation. In contrast, our disturbance rejection framework, implemented with both $N = 5$ and $N = 50$ demonstrates superior performance. The estimated outputs in these cases deviate significantly from the disturbed output and closely align with the non-disturbed output, highlighting the effectiveness of our method in attenuating disturbances and improving parameter estimation accuracy.

## V. CONCLUSION

This work proposes an RL-based method for estimating parameter sets and real-time system tracking using a disturbance rejection PPO algorithm. The reward function includes a disturbance rejection term to minimize error variance. This enables the agent to reduce disturbances' impact on parameter estimation. Results show the learned strategy accurately estimates rotary flexible joint parameters. It remains effective even with correlated disturbances. The approach outperforms the traditional Prediction Error Method (PEM).

## REFERENCES


[1] F. Lamnabhi-Lagarrigue, A. Annaswamy, S. Engell, A. Isaksson, P. Khargonekar, R. M. Murray, H. Nijmeijer, T. Samad, D. Tilbury, and P. Van den Hof, "Systems & control for the future of humanity, research agenda: Current and future roles, impact and grand challenges," *Annual Reviews in Control*, vol. 43, pp. 1–64, 2017.

[2] S. Decker, J. Richter, and M. Braun, "Predictive current control and online parameter identification of interior permanent magnet synchronous machines," in *2016 18th European Conference on Power Electronics and Applications (EPE'16 ECCE Europe)*, pp. 1–10, IEEE, 2016.

[3] N. Tietze, "Model-based calibration of engine control units using gaussian process regression," 2015.

[4] L. Ljung and T. Söderström, *Theory and practice of recursive identification*. MIT press, 1983.

[5] E. Hatami, H. Salarieh, and N. Vosoughi, "Design of a fault tolerated intelligent control system for a nuclear reactor power control: Using extended kalman filter," *Journal of Process Control*, vol. 24, no. 7, pp. 1076–1084, 2014.

[6] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 adaptive systems for signal processing, communications, and control symposium (Cat. No. 00EX373)*, pp. 153–158, Ieee, 2000.

[7] P. D. Moral, "Nonlinear filtering: Interacting particle resolution," *Comptes Rendus de l'Academie des Sciences-Serie I-Mathematique*, vol. 325, no. 6, pp. 653–658, 1997.

[8] A. Hefny, C. Downey, and G. J. Gordon, "Supervised learning for dynamical system learning," *Advances in neural information processing systems*, vol. 28, 2015.

[9] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

[10] S. Yatawatta and I. M. Avruch, "Deep reinforcement learning for smart calibration of radio telescopes," *Monthly Notices of the Royal Astronomical Society*, vol. 505, no. 2, pp. 2141–2150, 2021.

[11] K. Alhazmi and S. M. Sarathy, "Nonintrusive parameter adaptation of chemical process models with reinforcement learning," *Journal of Process Control*, vol. 123, pp. 87–95, 2023.

[12] D. Zhang, L. Du, and Z. Gao, "Real-time parameter identification for forging machine using reinforcement learning," *Processes*, vol. 9, no. 10, p. 1848, 2021.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[14] J. Schulman, "Trust region policy optimization," *arXiv preprint arXiv:1502.05477*, 2015.

[15] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[16] L. Ljung, "System identification," in *Signal analysis and prediction*, pp. 163–173, Springer, 1998.