

Article

Decentralized Q-Learning Supervisory Control for Coordinated Multi-Loop Tuning in Pump Stations

David A. Brattley *  and Wayne W. Weaver 

Department of Mechanical and Aerospace Engineering, Michigan Technological University,
Houghton, MI 49931, USA; wwwweaver@mtu.edu

* Correspondence: dabrattl@mtu.edu

Abstract

This paper introduces a reinforced learning-based supervisory control architecture that oversees multiple Recursive Least Squares (RLS) based self-tuning pump controllers and determines when each loop is permitted to adapt its gains. The supervisor learns adaptation policies that minimize interaction between loops while preserving responsiveness to changing hydraulic conditions. A two-loop pump station simulation is used to evaluate performance under product changes and transient flow disturbances. The results show that the supervisory layer reduces the number of simultaneous adaptation events by over 70%, leading to a 32% lower pressure-tracking error and 45% fewer gain-induced oscillations compared to conventional independent adaptive control. The reinforcement learning policy converges within 15 training episodes, resulting in stable adaptation scheduling and seamless transitions. The key novelty of this work lies in introducing decentralized reinforcement-learning-based coordination for adaptive pump control, enabling supervisory decision-making that actively prevents interference between controllers during transients. This approach provides a scalable and lightweight solution for coordinating multi-loop pump stations, enhancing robustness and operational performance in real-world pipeline systems.

Keywords: reinforcement learning; adaptive control; centrifugal pump systems; pipeline pressure control

1. Introduction

Modern pipeline pump stations operate under increasingly dynamic conditions, including varying product densities, fluctuating demand profiles, and stringent safety constraints. While loop-level adaptive control methods such as the self-tuning Recursive Least Squares (RLS)-based controller proposed by Brattley and Weaver [1] can improve local performance, they do not explicitly address the dynamic interactions between multiple control loops operating in parallel within a station, as discussed by Aribisala et al. [2]. In hydraulically coupled systems, independent adaptation across loops can lead to interference, suboptimal setpoint tracking, or cascading control actions that amplify transients during product switching, valve closures, or pressure disturbances.

A key challenge, therefore, lies not only in tuning individual loops but in coordinating decentralized adaptive controllers whose gains evolve online. When multiple loops simultaneously update their parameters based on local measurements, interaction-induced oscillations may emerge from the adaptation mechanisms themselves. Fixed tuning cannot



Academic Editors: Carmine Maria Pappalardo and Zheng Chen

Received: 16 January 2026

Revised: 20 February 2026

Accepted: 3 March 2026

Published: 6 March 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

accommodate transient hydraulic conditions, while centralized supervisory strategies often require global models, extensive computation, or communication overhead that may be impractical in industrial deployments. These considerations motivate the need for a decentralized supervisory coordination mechanism that preserves local autonomy while mitigating interaction-driven instability.

Reinforcement learning (RL) has been widely explored for pump scheduling, energy optimization, and supervisory decision-making in water and pipeline systems. Hajgato et al. [3], Joo et al. [4], Pei et al. [5], Hu et al. [6], Zhang et al. [7], and Wang et al. [8] demonstrate that RL can effectively optimize actuator commands or coordinate distributed infrastructure under multi-objective constraints. However, these approaches assume that RL agents directly generate control actions or replace conventional control laws. Relatively little attention has been paid to the coordination of decentralized adaptive controllers whose parameters evolve through online system identification.

The control challenge considered in this work is therefore distinct: rather than optimizing pump speed or scheduling operation, the objective is to regulate the timing and aggressiveness of gain adaptation across dynamically coupled loops. The instability mechanism of interest arises from simultaneous decentralized adaptation, not from static controller interaction alone. Addressing this problem requires a supervisory layer that operates above established control structures while preserving their interpretability and certifiability.

This paper introduces a decentralized Q-learning-based supervisory architecture in which each control loop is equipped with a local learning agent. The reinforcement learning component does not compute continuous actuator commands, nor does it replace the underlying PI controllers. Instead, it selects among discrete adaptation modes (e.g., no adaptation, slow adaptation, fast adaptation), thereby acting as an adaptation-gating mechanism. By separating the execution of the control law from adaptation timing, the framework mitigates gain-induced oscillations while retaining the stabilizing properties of classical PI control.

The architecture is inherently decentralized: each supervisory agent relies solely on local state information and reward feedback, without requiring centralized optimization or global plant models. Additional loops can be incorporated by instantiating additional agents, resulting in linear scaling of computational complexity. Although the validation studies focus on a two-loop pump station (suction and discharge pressure) to clearly illustrate interaction effects, the formulation generalizes to larger configurations.

In addition to the supervisory coordination mechanism, this study examines the impact of Q-table initialization on convergence and closed-loop behavior. Both cold-start and warm-start strategies are evaluated to quantify differences in learning speed, adaptation stability, and mode-switching frequency. This analysis provides insight into the practical deployment of reinforcement-learning-based supervisors in safety-critical industrial environments where excessive exploration is undesirable.

The principal contribution of this work lies in the architectural integration of reinforcement learning as a bounded supervisory coordination layer for decentralized adaptive control. By constraining learning to discrete adaptation gating rather than direct control synthesis, the proposed framework bridges classical adaptive control and reinforcement learning, aligning with industrial pipeline requirements.

The contributions of this work are as follows: (1) the development of a decentralized supervisory architecture that integrates Q-learning as a bounded adaptation-gating mechanism for coordinating multiple RLS-based self-tuning control loops without centralized models; (2) the formulation of a discrete multi-level adaptation policy (no adaptation, slow adaptation, fast adaptation) specifically designed to mitigate gain-induced oscillations arising

ing from simultaneous decentralized adaptation in hydraulically coupled pump stations; (3) a systematic evaluation of Q-table initialization strategies (cold start versus warm start) and their impact on convergence speed, switching frequency, and closed-loop stability; and (4) quantitative validation demonstrating improved transient suppression and reduced unnecessary adaptation relative to independent adaptive control.

2. Model Description

The system under study is a pipeline pump station, as shown in Table 1, equipped with multiple interacting control loops governing process variables critical to safe and efficient operation. While the underlying hydraulic and electromechanical dynamics are identical to those previously validated in [1], this work focuses on the supervisory coordination of loop-level adaptation rather than on plant modeling itself. A typical pump station includes an induction-motor-driven centrifugal pump, suction and discharge pressure control loops, and—depending on the topology—additional flow-balancing or surge-protection mechanisms. Under transient conditions such as product switching, valve closures, or pressure upsets, these loops interact dynamically, motivating the need for a supervisory mechanism that regulates when and how aggressively local controllers adapt.

Table 1. Simulation parameters used for the adaptive and fixed-gain controller comparison.

Parameter	Symbol	Value/Unit
Physical Parameters		
Fluid density	ρ	700–900 kg/m ³
Pump head	h	m
Pump torque	T	N·m
Moment of inertia	J	1.825 kg·m ²
Pump rotational speed range	ω	0–3600 rpm
Flow rate range	q	0–0.185 m ³ /s
Nominal discharge pressure	P_d	100 bar
Nominal suction pressure	P_s	2 bar
Controller Parameters		
Proportional gain (initial)	K_p	0.07
Integral gain (initial)	K_i	0.05
Forgetting factor (initial)	λ	0.99

2.1. Pump Station Components

The pump station model comprises the following physical and control components: an induction motor, a centrifugal pump, a fluid-filled pipeline, a Q-table-based supervisor, and a velocity-form controller enhanced with a variable-forgetting-factor Recursive Least Squares (RLS) algorithm for self-tuning in each control loop. The model utilizes dynamic equations that describe the mechanical–electrical coupling, as described in [1]. Motor torque, pump speed, and fluid properties influence the pump output (head and flow rate). The downstream pipeline segment and valve introduce pressure transients governed by the Darcy–Weisbach equation for frictional losses and transient hydraulic equations that model water-hammer and flow-shutoff effects during valve closures.

2.1.1. Induction Motor and Pump

The induction motor is modeled as a dynamic torque source controlled by a variable frequency drive (VFD) [9], which regulates motor speed in response to the controller output.

A simplified dynamic model is used to represent the rotor inertia and speed response [10], given by

$$J \frac{d\omega}{dt} = T_m - T_l - B\omega \quad (1)$$

where J ($\text{kg} \cdot \text{m}^2$) is the combined inertia of the motor and the pump, ω (rad/s) is the angular speed of the motor, T_m ($\text{N} \cdot \text{m}$) is the electromagnetic torque of the motor, T_l ($\text{N} \cdot \text{m}$) is the load torque imposed by the pressure of the pump and pipeline, and B ($\text{N} \cdot \text{m} \cdot \text{s}$) represents the losses of rotational friction. The motor torque is assumed to respond instantaneously to VFD commands for simulation purposes. The emphasis of this model formulation is not high-fidelity hydraulic simulation but, rather, the faithful representation of loop interactions and transient behaviors that influence adaptive controller coordination.

The centrifugal pump is modeled using a nonlinear pressure–flow relationship consistent with standard pump curves [11–13], expressed as follows:

$$H(\omega, Q) = a_{h0}\omega^2 - a_{h1}\omega Q - a_{h2}Q^2 \quad (2)$$

$$Q = k \cdot \omega \quad (3)$$

where H (m) is the pump head, which is a function of both the angular speed ω (rad/s) and the flow rate Q (m^3/s), and contains the parameters a_{h0} (m/rpm^2), a_{h1} ($\text{m}/(\text{rpm} \cdot \text{m}^3/\text{s})$), and a_{h2} ($\text{m}/(\text{m}^3/\text{s})^2$) taken from the pump curve and determined using least squares. Q is determined by k (m^3)—a constant relating speed to flow—and ω is the pump speed.

2.1.2. Pipeline and Valve Dynamics

The pipeline model considers both steady-state flow resistance and dynamic pressure behavior. Fluid flow through a pipeline experiences head loss due to friction, which can be modeled using the Darcy–Weisbach equation [11]:

$$h_f = f \cdot \frac{L}{D} \cdot \frac{v^2}{2g} \quad (4)$$

where h_f (m) is the frictional head loss, f (dimensionless) is the Darcy friction factor, L (m) is the pipe length, D (m) is the pipe diameter, v (m/s) is the fluid velocity, and g (9.81 m/s^2) is the gravitational acceleration.

The pressure drop associated with h_f contributes to the total dynamic head that the pump must overcome. The Darcy friction factor f may be obtained from the Moody chart or approximated using empirical formulae such as the Colebrook–White equation for turbulent flow. No external stochastic disturbance signals were injected into the pipeline model; all observed pressure fluctuations arise from intrinsic hydraulic transients and nonlinear valve–pipeline interactions.

2.2. Control Architecture

In this research, the pump station was controlled by two different control loops. A suction pressure loop regulated the inlet pressure to maintain NPSH (Net Positive Suction Head) to prevent cavitation. Disturbances in downstream loops can indirectly affect this loop and the discharge pressure loop, which maintains the desired downstream pressure by adjusting the motor torque via VFD speed control. This loop is often the most dominant and energy-intensive.

Each loop was controlled using a velocity-form controller [14–16]:

$$\Delta u_i(k) = K_{p,i} \cdot \Delta e_i(k) + K_{i,i} \cdot e_i(k) \quad (5)$$

where $e_i(k)$ is the control error for loop i (difference between the pressure setpoint and measured pressure), $u_i(k)$ is the controller output (used to command the VFD), and $K_{p,i}$ and $K_{i,i}$ are the proportional and integral gains, respectively. The RLS algorithm updates these gains in real time based on system identification. Since the parameters being controlled (e.g., discharge pressure) are measured by a transmitter, a random noise signal will be injected into the transmitter's output sent to the controller.

The loop-level self-tuning mechanism employs a Recursive Least Squares (RLS) estimator to identify a local linear approximation of the pressure dynamics around the current operating point. Specifically, the RLS algorithm estimates the parameters of a low-order input–output model that captures the incremental relationship between changes in the manipulated variable and the measured pressure response. Although the overall pipeline system exhibits nonlinear behavior due to hydraulic effects and pump characteristics, no global linearization is assumed. Instead, the RLS estimator performs an implicit online linearization by continuously updating the local model parameters as operating conditions evolve.

The Recursive Least Squares estimator employs a variable forgetting factor to balance noise rejection and tracking capability under changing operating conditions. The forgetting factor is constrained within a predefined range, $\lambda \in [0.95, 0.99]$, and is adjusted in a discrete, rule-based manner rather than through continuous optimization. Larger tracking errors or rapid transient behavior can temporarily reduce the forgetting factor to improve parameter adaptation, while near-steady-state operation increases λ to suppress noise and prevent unnecessary parameter drift. This adjustment mechanism operates independently of the Q-learning supervisor and does not constitute an additional learning loop. Instead, it provides a bounded and interpretable means of regulating estimator responsiveness, ensuring stable parameter convergence while maintaining sensitivity to hydraulic disturbances. As a result, the RLS estimator operates as a bounded, interpretable adaptive mechanism rather than a global learning algorithm.

The estimated model parameters are not used to independently tune the proportional and integral gains. Rather, they are mapped to PI gains via a structured gain-update rule derived from the identified process gain and dominant time constant, ensuring coordinated, stable adjustment of both parameters. The supervisory Q-learning layer does not directly modify controller gains; instead, it selects the adaptation mode that governs the rate at which the RLS-based gain updates are applied. This separation of roles allows the RLS estimator to track local process dynamics while the reinforcement learning agent regulates adaptation aggressiveness, preventing excessive gain variation and improving closed-loop stability.

Each loop's controller is equipped with a self-tuning mechanism that uses Recursive Least Squares (RLS) with a variable forgetting factor to adapt the gains based on local model estimation. Independent adaptation across all loops can lead to interference when multiple controllers respond aggressively to the same transient event. For example, a downstream disturbance may trigger rapid gain adaptation in the discharge pressure loop, while the suction pressure loop—observing a correlated pressure deviation—initiates its own adaptation. Such parallel adaptation can result in conflicting control actions, excessive gain variation, and degraded stability. The supervisory coordination strategy introduced in Section 3 addresses this issue by enabling each loop to learn when adaptation is appropriate, thereby mitigating adverse loop interactions without centralized control. To illustrate this, consider a scenario in which the discharge pressure loop detects a sudden setpoint deviation caused by a downstream event and adjusts its gains to respond more quickly. Simultaneously, observing a similar pressure deviation due to system interaction, the suction loop may also begin adapting. This parallel adaptation can lead to conflicting actions and degraded

stability. Therefore, the goal is to enable each loop to learn when and how quickly to adapt, using local state observations and a Q-learning supervisory layer, as introduced in Section 3.

2.3. Model Validation Summary

The pump station dynamics model utilized in this study is identical to the physics-based model previously presented and validated by Brattley and Weaver [1]. The model captures the nonlinear pump characteristic curves, suction and discharge manifold dynamics, and pipeline transient behavior. Validation was performed against historical measurements collected under multiple flow and pressure conditions, demonstrating strong agreement in both steady-state and transient responses. Quantitatively, the model achieved an average coefficient of determination of $R^2 = 0.957\%$ and a normalized root-mean-square error in the range of $\text{NRMSE} = 12.6\%$ across key operating metrics, including discharge pressure, suction pressure, and flow rate. Accordingly, the model can be considered sufficiently accurate for evaluating the supervisory control strategies investigated in this paper.

3. Q-Learning-Based Supervisory Coordination

In this framework, a decentralized Q-learning agent, as shown in Figure 1, governs each control loop in the pump station. These agents observe local loop conditions and select adaptation strategies based on a discrete action space. States are derived using thresholds on control error magnitude and oscillation detection (e.g., $\text{osc}_i(k) = \mathbb{K}[|e_i(k) - e_i(k-1)| > \epsilon]$). Unlike traditional binary adaptation strategies (adapt or not), this framework expands the action set to three levels of adaptation. a_1 : No Adaptation—parameters are frozen, and the RLS update is skipped. a_2 : Adapt Slowly—the RLS algorithm is enabled with a high forgetting factor, λ (e.g., $\lambda = 0.99$), producing conservative updates. a_3 : Adapt Quickly—the forgetting factor is reduced (e.g., $\lambda = 0.95$) to accelerate tuning during transients. Loop-specific indicators, such as error-trend and control-signal variability, define the state space. A reward function balances stability, control accuracy, and avoidance of unnecessary adaptation. The discrete state-space definition was intentionally designed to reflect high-level supervisory performance conditions rather than detailed process dynamics. The proposed Q-learning agent operates at a supervisory layer that regulates the aggressiveness of controller adaptation, without replacing the underlying continuous-time control or system identification mechanisms. By classifying loop behavior into coarse operational regions such as far from setpoint, near setpoint, and not in control, the agent relies on interpretable performance indicators that are commonly used by control engineers during commissioning and tuning. The state transition thresholds are defined in terms of normalized error magnitudes and selected to represent meaningful control regimes rather than precise physical boundaries. The supervisory state definition is intentionally structured around physically meaningful operating regimes rather than arbitrary discretization. In particular, the distinction between “near” and “far” from the setpoint reflects differing hydraulic sensitivities of the pump–pipeline system. When operating far from the pressure setpoint, large pressure deviations often correspond to transient hydraulic events such as valve closures, product changes, or density variations. Under these conditions, more aggressive adaptation may be beneficial. Conversely, when the controlled variable is near steady state, pressure dynamics are dominated by compressibility effects and pipeline inertia, where excessive gain adaptation can amplify small disturbances and introduce oscillatory behavior. By discretizing the state space around these physically distinct regimes, the supervisory agent learns adaptation policies that align with underlying hydraulic dynamics rather than purely numerical thresholds. This abstraction reduces sensitivity to

measurement noise, modeling uncertainty, and unmodeled nonlinearities, while limiting the Q-table's dimensionality to promote stable, repeatable learning behavior. Importantly, these thresholds are not learned parameters and can be adjusted to accommodate different pipeline characteristics or operating constraints without modifying the learning structure or retraining the supervisory agent.

The Q-learning supervisor influences controller behavior indirectly by regulating the rate of RLS-based gain adaptation rather than directly modifying gains.

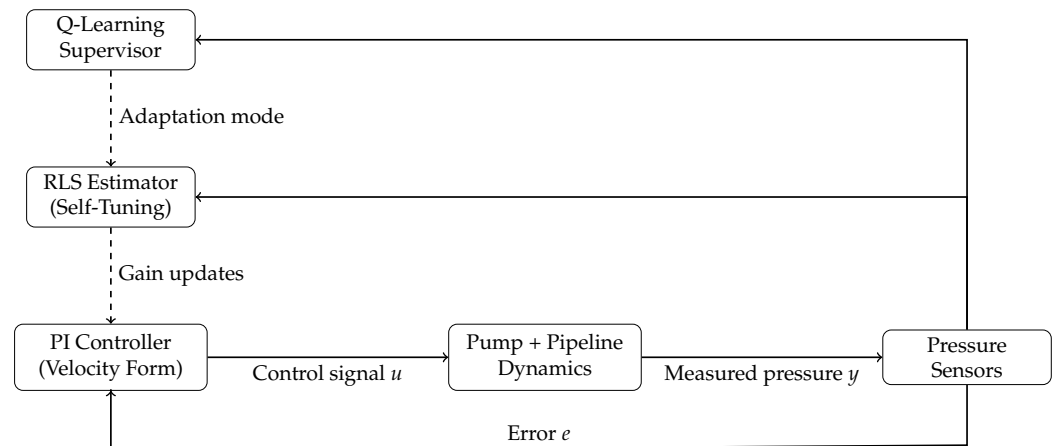


Figure 1. Block diagram of the proposed Q-learning-based supervisory control architecture. The Q-learning agent regulates the adaptation timing of the RLS-based self-tuning PI controllers without directly modifying control signals or plant dynamics.

3.1. Motivation for Sequential Loop Adaptation

In multi-loop pipeline control systems, simultaneous self-adaptation of interacting controllers can lead to undesirable coupling effects and instability. When multiple loops adapt their gains concurrently, each controller modifies its behavior based on process measurements that are themselves influenced by the actions of the other loops. This mutual interaction can result in non-stationary dynamics from the perspective of each adaptive controller, leading to oscillatory gain updates, excessive mode switching, or slow convergence to stable operating conditions. In extreme cases, concurrent adaptation may amplify transient disturbances, increase pressure fluctuations, or cause competing control actions that degrade overall system performance.

To mitigate these effects, the proposed supervisory strategy intentionally limits simultaneous adaptation by coordinating loop-level tuning actions in a sequential and context-aware manner. At any given time, only the loop that is deemed most influential on the shared process variable is permitted to adapt aggressively, while the remaining loops operate in a constrained or monitoring mode. This approach preserves responsiveness to changing hydraulic conditions while reducing cross-coupling during the adaptation process. By decoupling the learning dynamics across loops, the supervisory layer promotes smoother convergence, improved stability, and more interpretable adaptation behavior, which is particularly important for safety-critical pipeline applications.

3.2. Q-Learning Update Rule

The Q-table is updated using the standard incremental rule [17], applied to the previous state and action pair

$$Q(s_{k-1}, a_{k-1}) \leftarrow Q(s_{k-1}, a_{k-1}) + \alpha \left[r_k + \gamma \max_{a'} Q(s_k, a') - Q(s_{k-1}, a_{k-1}) \right] \quad (6)$$

where s_{k-1}, a_{k-1} are the state and action at the previous time step, s_k is the current state observed after transition, r_k is the reward computed at time k , α is the learning rate, and γ is the discount factor for future returns. This formulation highlights that the update is made to the Q-value associated with the last decision, based on the outcome observed at the current time step. Thus, the algorithm does not predict the future directly but instead learns retrospectively by comparing expected and realized performance.

The Q-learning component is trained entirely through online interaction with the simulated pipeline system and does not rely on any external or pre-collected datasets. Each training episode corresponds to a closed-loop simulation run in which the supervisory agent observes loop-level performance metrics, selects adaptation actions, and receives a scalar reward based on tracking error and control effort penalties. State transitions and rewards are evaluated at each supervisory decision interval and used to incrementally update the Q-table. For warm-start experiments, the Q-table is initialized using values obtained from prior training episodes, while cold-start experiments initialize the table with zeros. This approach allows the effect of prior learning to be isolated and evaluated without altering the underlying training data or reward formulation.

While the Q-learning update rule employed in this work follows the standard tabular formulation, the proposed framework does not rely on reinforcement learning to directly control the physical system. Instead, the learning agent operates strictly at a supervisory level, selecting among a finite set of bounded adaptation modes that regulate the rate of controller gain updates. The underlying PI controllers and RLS estimator maintain continuous-time closed-loop stability, while the Q-learning layer influences adaptation behavior within predefined safety constraints. From a practical deployment perspective, the warm-start mechanism should be viewed as a convergence accelerator rather than a prerequisite. Its purpose is to reduce unnecessary exploration and mode switching during early operation when representative prior knowledge is available, while preserving the adaptability and robustness of online reinforcement learning under evolving hydraulic conditions.

Formal convergence guarantees for tabular Q-learning in nonlinear, non-stationary control systems remain an open research problem. To address this limitation in a safety-critical context, the proposed approach incorporates several practical safeguards, including constrained action spaces, action masking to prevent unsafe simultaneous adaptation, bounded exploration probabilities, and the option to suspend Q-table updates once satisfactory performance is achieved. These design choices significantly reduce the risk associated with exploratory behavior and ensure that learning-induced transients remain within acceptable operational limits. As a result, the framework prioritizes robustness and safety over theoretical optimality, making it suitable for practical deployment in industrial pipeline applications.

3.3. ϵ -Greedy Exploration Strategy

To balance exploration of new strategies and exploitation of learned knowledge, each agent follows an ϵ -greedy action selection policy [17–19]. At each decision step, the control loop selects a random action with probability ϵ (exploration) and the action with the highest Q-value in its current state with probability $1 - \epsilon$ (exploitation). This ensures that the agent continues to sample less frequently chosen strategies, preventing premature convergence to suboptimal policies. Formally, the action a_t at time t is chosen as follows:

$$a_t = \begin{cases} \text{random action,} & \text{with probability } \epsilon \\ \arg \max_a Q(s_t, a), & \text{with probability } 1 - \epsilon \end{cases}.$$

The exploration rate ϵ can be set to a fixed value (e.g., $\epsilon = 0.1$), or scheduled to decay gradually over time to favor exploitation as the learning process converges. In this work, a decaying ϵ was implemented, which favored the learned table over time and showed that a small but nonzero ϵ consistently improved robustness against unexpected transients in pipeline conditions.

3.4. Reward Function

Each loop computes a local reward using error reduction and control effort. Unlike classical reinforcement learning formulations, in which the objective is to maximize the accumulated reward, in this application [20,21] the control objective is framed as the minimization of pressure error. Accordingly, the reward function is defined as the penalty

$$r_i(k) = -|e_i(k)| - \lambda \cdot |\Delta u_i(k)| \quad (7)$$

where λ , in this case, is the control-effort weighting factor, which penalizes aggressive control changes, encouraging low error with smooth control actions.

3.5. Illustrative Q-Table Example

To clarify the implementation, Table 2 presents a simplified Q-table for a single control loop. The state space is discretized into three categories: not in control (not the lowest output u), near setpoint (small error and stable response), and far from setpoint (large-to-moderate error requiring adjustment). The action space contains three adaptation strategies: no adaptation, slow adaptation (large forgetting factor), and fast adaptation (small forgetting factor). This discrete representation is intended to support supervisory decision-making rather than detailed process modeling. The use of coarse, performance-based states improves robustness to noise and unmodeled nonlinearities, limits learning complexity, and aligns the supervisory policy with operationally meaningful control objectives. The entries in the Q-table represent learned values that guide the controller in selecting the appropriate adaptation strategy based on the observed loop condition, thereby maintaining stability and minimizing unnecessary parameter updates.

Table 2. Illustrative Q-table for one control loop.

State	A1: No Adaptation	A2: Adapt Slowly	A3: Adapt Quickly
S1: Not in control	$Q(S_1, A_1)$	$Q(S_1, A_2)$	$Q(S_1, A_3)$
S2: Near setpoint	$Q(S_2, A_1)$	$Q(S_2, A_2)$	$Q(S_2, A_3)$
S3: Far from setpoint	$Q(S_3, A_1)$	$Q(S_3, A_2)$	$Q(S_3, A_3)$

The Q-values accumulate a penalty, so the control strategy is to select actions that minimize cost rather than maximize reward. Thus, the equilibrium solution corresponds to each loop converging on the least negative (i.e., closest to zero) Q-value, which represents the action policy that best reduces long-term error. Figure 2 shows the adaptation modes selected by one of the supervisory agents across a representative scenario.

Each loop maintains its own Q-table and updates it locally, but all use the same state–action definitions. Coordination emerges implicitly: unstable loops tend to adopt more aggressive adaptation, while stable loops can adapt more conservatively. Over time, the learned policies reduce oscillations and improve cooperative stability, even though the agents do not exchange information explicitly.

3.6. Training and Initialization of the Q-Table

A practical consideration for implementing Q-learning in control applications is how the Q-table is initialized and trained [22,23]. In this work, two approaches are considered:

(1) Cold start (zero initialization), in which the Q-table is initialized to $Q(s, a) = 0$ for all states and actions. The agent begins with no prior knowledge and must explore actions to gradually learn the value of each state–action pair. This results in greater variability and slower convergence, as the system initially explores switching between adaptation modes. (2) Warm start (pre-trained initialization), where the Q-table is initialized with values obtained from prior training episodes under similar operating conditions. This provides the agent with an approximate policy from the start, enabling faster convergence to a stable adaptation policy with fewer mode switches and smoother control performance.

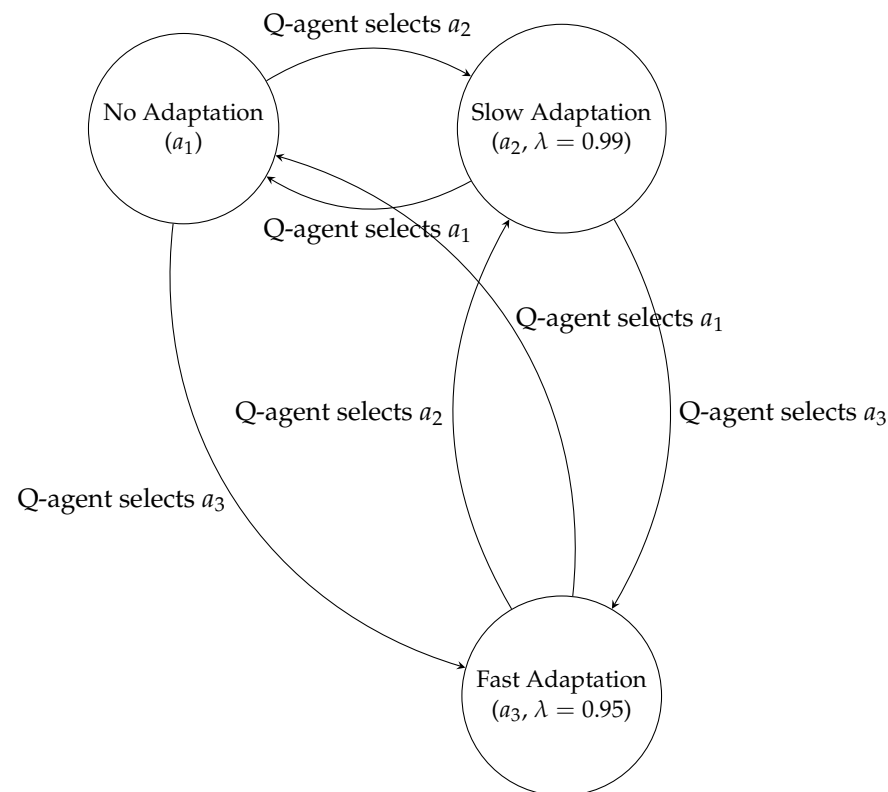


Figure 2. Q-learning-based supervisory agent switching between adaptation modes. Each action a_k corresponds to enabling or disabling RLS updates with different forgetting factors.

The distinction between these two initialization strategies highlights the trade-off between exploration and deployment readiness. Cold-start operation demonstrates the learning agent’s ability to autonomously discover effective policies, whereas warm-start initialization is better suited for real-world deployment, where training can be performed offline using simulation or historical data.

During training, the ϵ -greedy exploration strategy maintains a balance between exploring new actions and exploiting high-value actions. Over time, ϵ is decayed to prioritize exploitation once the Q-values have stabilized. This allows the agent to adjust its policy in response while maintaining stable long-term behavior. To prevent the supervisory agent from selecting physically invalid or counterproductive adaptation modes, an action-masking scheme was implemented. At each control step, the set of valid actions was determined based on which loop currently maintained control of the process variable. When a loop was not in control, its available action set was restricted to a single “no adaptation” mode, while the active loop retained access to all adaptation levels. This selective masking reduces unnecessary exploration and avoids destabilizing updates to inactive loops. In practice, this was implemented by dynamically limiting the number of admissible actions before the ϵ -greedy selection step: if `in_control(loopId)` was true, the agent sampled

from the full action set; otherwise, the valid action set was limited to one. This mechanism maintains exploration for the active loop while enforcing stability and safety in the others.

For initialization, two strategies were considered: In the cold-start case, the Q-table was initialized to zero, requiring the agent to build its policy entirely through new interactions with the system. In the warm-start case, the Q-table was initialized from a previously trained table. The algorithm checked for an existing file; if present, the stored values were loaded and used as the initial Q-table. This procedure allowed the agent to exploit prior learning from similar operating conditions, thereby reducing the amount of exploration required. If no saved Q-table was found, the agent defaulted to cold-start initialization.

By comparing the performance of cold-start and warm-start initialization in simulation, the benefits of incorporating prior knowledge were quantified. Specifically, warm-start initialization reduces the settling time, overshoot, and the number of adaptation-mode switches, making it more suitable for safety-critical pump-station operation.

4. Controller Implementation

This section outlines the real-time execution of the decentralized adaptive control framework, consisting of loop-level controllers with Recursive Least Squares (RLS) parameter adaptation and a Q-learning supervisory agent. The structure ensures robust coordination while minimizing interference between loops and was adapted from our previous work [1].

4.1. Loop-Level Adaptive Control with RLS

A velocity-form controller governs each control loop using the standard velocity-form control law [14,24]:

$$\Delta u_i(k) = K_{p,i}(k)[e_i(k) - e_i(k-1)] + K_{i,i}(k)e_i(k) \quad (8)$$

$$u_i(k) = u_i(k-1) + \Delta u_i(k) \quad (9)$$

where $e_i(k) = r_i(k) - y_i(k)$ is the tracking error and $\{K_{p,i}(t), K_{i,i}(t)\}$ are time-varying gains. The gains are updated using Recursive Least Squares (RLS) with a variable forgetting method, defined by [16,24]

$$\theta_i(k) = \begin{bmatrix} K_{p,i}(k) \\ K_{i,i}(k) \end{bmatrix} \quad (10)$$

$$\phi_i(k) = \begin{bmatrix} e_i(k) - e_i(k-1) \\ e_i(k) \end{bmatrix} \quad (11)$$

$$\theta_i(k+1) = \theta_i(k) + K(k)[u_i(k) - \phi_i^\top(k)\theta_i(k)] \quad (12)$$

$$K(k) = \frac{P_i(k)\phi_i(k)}{\lambda + \phi_i^\top(k)P_i(k)\phi_i(k)} \quad (13)$$

$$P_i(k+1) = \frac{1}{\lambda} \left[P_i(k) - K(k)\phi_i^\top(k)P_i(k) \right] \quad (14)$$

where $P_i(k)$ is the covariance matrix and λ is the forgetting factor. If low-select logic were only used to determine the control u being sent to the plant, it would introduce a structural loss of persistent excitation by intermittently removing the causal pathway between loop-level regressors and the plant output. During inactive intervals, the effective regressor becomes rank-deficient, causing the RLS covariance update to degenerate and leading to estimator wind-up or instability upon reactivation. To prevent the regressor from decoupling from the plant and causing instability, the RLS gain update executes only if loop

i is enabled for adaptation by the supervisory agent. In typical industrial stations, single or multiple pumps operate under shared pressure regulation. Each station maintains its own local pressure controller; however, only the controller with the lowest manipulated variable (e.g., motor speed) actively influences the process. The others remain in standby with their control authority suppressed. The supervisor evaluates each controller's performance and selects actions that enable or disable RLS adaptation for each pump.

To explore the impact of low-select logic on estimation, consider a pump station equipped with local pressure controllers whose parameters are estimated using the Recursive Least Squares (RLS) algorithm. Let the local regressor for pump i at time k be denoted by $\phi_i(k) \in \mathbb{R}^n$, and let the corresponding control input be $u_i(k)$. A low-select logic determines the applied manipulated variable

$$u(k) = \min_{i \in \{1, \dots, M\}} u_i(k) \quad (15)$$

such that only the selected controller $j = \arg \min_i u_i(k)$ actively influences the process at time k .

For controllers $i \neq j$, the control action is effectively suppressed and does not propagate to the plant output $y(k)$. Consequently, the regression model for inactive controllers becomes decoupled from the measured response, yielding an effective regressor

$$\tilde{\phi}_i(k) = 0, \quad i \neq j. \quad (16)$$

The corresponding information matrix update for controller i

$$P_i^{-1}(k) = R_i(k) = \sum_{\ell=1}^k \tilde{\phi}_i(\ell) \tilde{\phi}_i^{\top}(\ell) \quad (17)$$

is therefore rank-deficient during periods of low selectivity inactivity

$$\text{rank}(R_i(k)) < n \quad (18)$$

violating the persistent excitation condition required for consistent RLS estimation.

To prevent estimator degradation, a supervisory logic gates the RLS update based on controller participation in the low-select mechanism. Specifically, parameter and covariance updates for controller i are enabled only when $i = j$, ensuring that RLS adaptation occurs exclusively during intervals where a causal input–output relationship exists. This gating preserves estimator conditioning and prevents covariance inflation during periods of suppressed actuation.

4.2. Q-Learning Supervisory Agent

The supervisory layer is implemented by a decentralized Q-learning agent, described in Section 3. At each decision step, the agent selects one of three adaptation modes: no adaptation (a_1), where the RLS update is disabled; slow adaptation (a_2), where RLS is enabled with a forgetting factor of $\lambda = 0.99$; or fast adaptation (a_3), where RLS is enabled with $\lambda = 0.95$. The Q-agent modulates the RLS forgetting factor without directly altering the control law, ensuring bumpless operation. The Q-table is updated online according to the reinforcement learning update rule

$$Q(s_{k-1}, a_{k-1}) \leftarrow Q(s_{k-1}, a_{k-1}) + \alpha \left[r_k + \gamma \max_{a'} Q(s_k, a') - Q(s_{k-1}, a_{k-1}) \right]. \quad (19)$$

At each control interval, as demonstrated in Figure 3, the loop outputs $y_i(k)$ are measured and the corresponding errors $e_i(k)$ are computed. These errors are used to determine the current system state $s(k)$ through predefined performance indicators. Based on this state, the Q-learning agent selects an action $a(k)$, which specifies the adaptation mode applied to each control loop. For any loop where the selected action differs from the no-adaptation mode, the RLS-based gain adaptation is executed using the forgetting factor associated with that mode, and the control input increment $\Delta u_i(k)$ is computed and added to the previous command to obtain $u_i(k)$. The updated control signals are then applied to the actuators. The supervisory agent subsequently observes the reward $r(k)$ derived from closed-loop performance and updates the Q-table entry $Q(s_{k-1}, a_{k-1})$ using the received reward and the new state s_k .

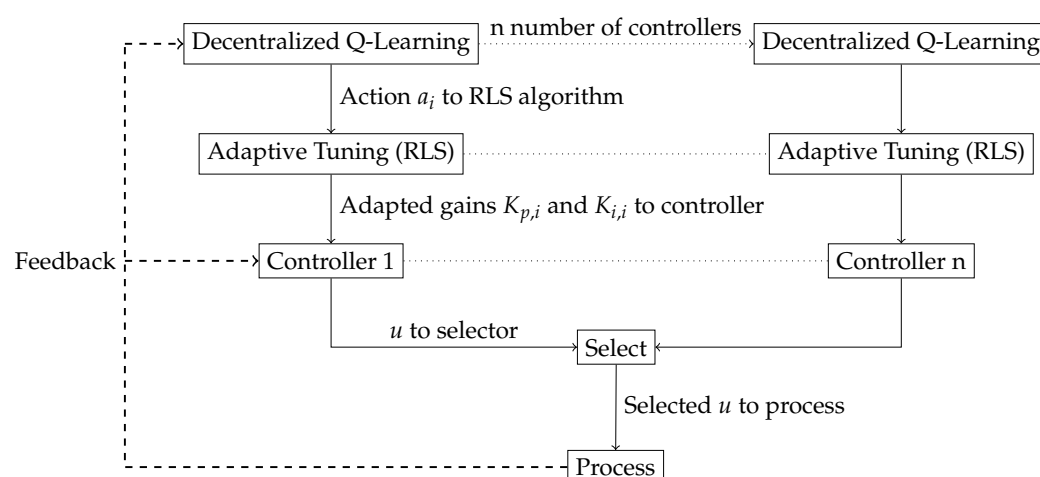


Figure 3. Block diagram of the adaptive control architecture with loop-level controllers, RLS adaptation, and Q-learning supervisory coordination.

5. Simulation Results

5.1. Simulation Setup

A simulated pump station (1)–(4) was modeled in MATLAB, 2023b, equipped with controllers for discharge and suction pressure. Each loop was augmented with a decentralized Q-learning agent implementing a three-level adaptation policy: no adaptation, adapt slowly, and adapt quickly. The initial portion of the results focuses on the impact of the adaptation startup strategy under standard pump startup conditions. By isolating this scenario, the performance trade-offs of cold versus warm adaptive behavior are quantified, establishing why supervisory decision-making is necessary.

Although the simulation studies presented in this work focus on a two-loop pressure control configuration, the proposed supervisory architecture is not limited to this case. The two-loop system represents the minimal nontrivial scenario in which hydraulic coupling and control interaction arise, making it well suited for evaluating coordination and adaptation behavior. The proposed framework is inherently decentralized, with each control loop equipped with an independent Q-learning supervisor that selects adaptation modes based solely on local performance metrics and shared process influence.

As additional control loops are introduced, the architecture scales in a modular fashion without requiring joint state representations, centralized coordination, or combinatorial growth of the learning space. Each loop maintains its own Q-table and adapts independently, while interaction effects are managed through supervisory constraints that limit simultaneous aggressive adaptation. As a result, the computational complexity and learning burden grow approximately linearly with the number of loops, making the approach suitable for larger pump stations and distributed pipeline networks. While only two loops

are considered here for clarity and interpretability, the same coordination principles apply to systems with multiple interacting pumps and pressure control points.

Then, three scenarios are explored to test the interaction between the suction and discharge pressure control loops: a simple pump start in steady state; a transient pressure spike from rapid valve closure downstream, which will send a pressure wave back to the origin pump station; and an inadvertent tank change that lowers the pump's suction pressure. The first test case is a simple start-up in which the pump is started and attempts to reach the desired pressure setpoint. The second scenario prevents overpressure, and the third prevents pump cavitation, which could cause damage. For comparison, the Q-learning strategy, cold and warm starting, and a fixed-gain PI controller will be explored. Figure 4 illustrates the overall control architecture, highlighting the interaction between the PI controllers, the RLS-based self-tuning mechanism, and the Q-learning supervisory layer that regulates adaptation behavior.

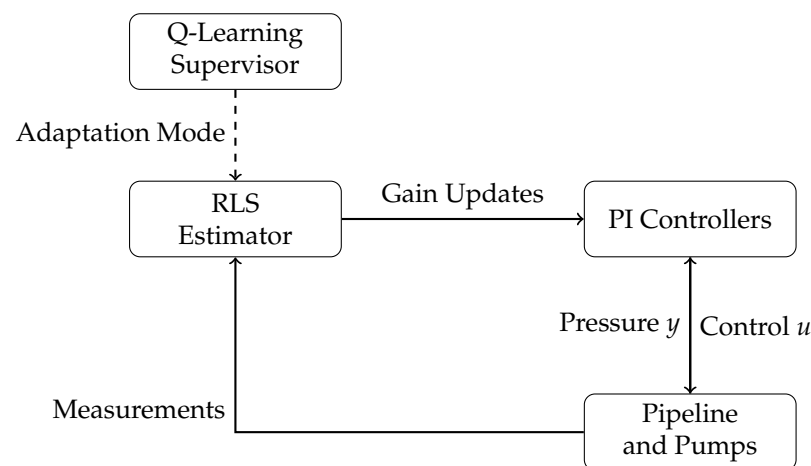


Figure 4. Block diagram of the decentralized supervisory control architecture integrating PI control, RLS-based self-tuning, and Q-learning supervision.

5.2. Impact of Q-Table Initialization on Control Performance

To evaluate the effect of Q-table initialization strategies on control performance, two simulation studies were performed. In cold starting, the Q-table was initialized with zeros, representing an untrained agent. During warm starting, the Q-table was initialized using prior training episodes, thereby providing the agent with knowledge of previously explored state–action values.

Both cases were tested, Table 3, under the same disturbance profile (a setpoint change), with ϵ -greedy exploration applied. The exploration parameter ϵ was gradually decayed, but its initial value was adapted to the initialization strategy. For the cold-start experiments, a higher exploration probability ($\epsilon = 0.2$) was used to encourage broad sampling when the Q-table contained no prior knowledge. For the warm-start experiments, the exploration rate was reduced ($\epsilon = 0.002$) to reflect the prior information embedded in the loaded Q-table and to limit unnecessary switching caused by random exploration. This ensured a fair comparison: the cold-start agent relied more heavily on exploration to build its policy, whereas the warm-start agent primarily exploited its pre-trained policy, retaining a small probability of exploration to adapt to minor changes in operating conditions.

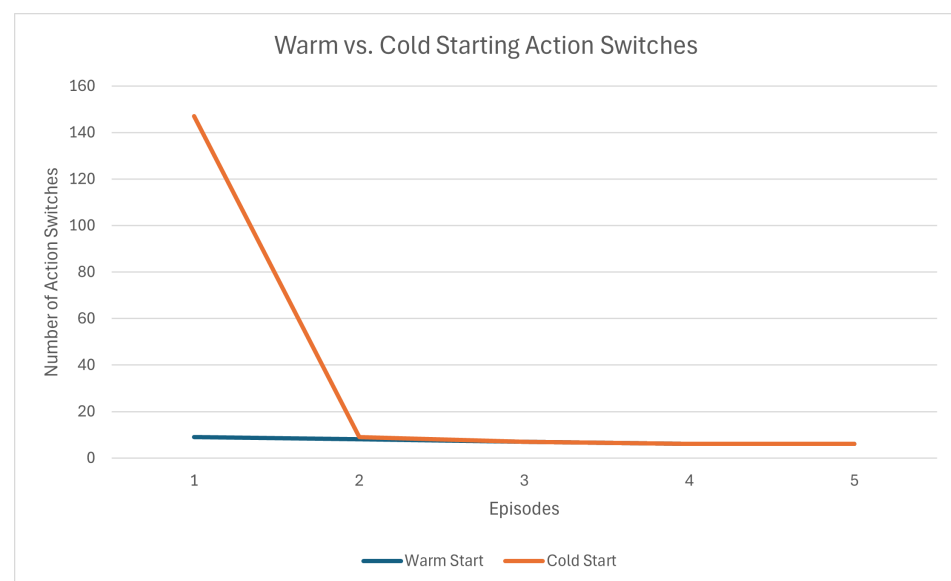
Table 3. Performance comparison of Q-learning initialization strategies.

Metric	Cold Start	Warm Start	Improvement (%)
Settling Time (s)	1.2	0.7	42
Overshoot (%)	1.95	1.07	45
Number of Mode Switches	147	8	95
Average Reward	0.084	0.036	57

The result strikes a balance between minimizing error and maintaining, leading to significantly faster convergence and fewer mode switches compared to cold-start operation. The reduction in overshoot demonstrates that the supervisory agent learns to apply smoother adaptation once the loop approaches its setpoint. Meanwhile, the improved average reward indicates that the warm-start policy stabilizes more quickly, balancing error minimization with smooth control effort.

Figure 5 illustrates the mode-switching frequency during the transient response. Cold-start operation exhibits frequent oscillations between adaptation modes in the first 20 s, whereas warm-start initialization rapidly converges to an appropriate strategy with minimal switching. These findings suggest that incorporating pre-trained Q-values into the supervisory agent provides a practical pathway for real-world deployment. By reducing training time and avoiding excessive mode switching, warm-start initialization enhances both safety and stability, making the approach more suitable for industrial pipeline systems.

The performance differences observed between cold- and warm-start initialization can be further understood by examining the learned Q-tables. By analyzing the final Q-values, we gain insight into how action preferences are shaped under each initialization strategy.

**Figure 5.** Comparison of mode switching under cold-start and warm-start initialization.

During extended warm-start simulations, it was observed that continual Q-table updates beyond the initial convergence phase could lead to gradual policy drift and loss of stability. As the agent continued to modify its Q-values despite minimal new information, the learned policy began to favor suboptimal actions, leading to increased oscillations and slower recovery from transients. This effect was particularly evident after several reloaded warm-start episodes, where previously stable adaptation patterns became erratic. Once the Q-table updates were frozen based on a convergence criterion, the control response stabilized and maintained consistent performance across subsequent runs. This behavior highlights a critical practical insight: while reinforcement learning

enables adaptive intelligence, its unrestricted application in steady-state or near-optimal regimes can degrade performance over time. Therefore, a hybrid learning policy—where learning is active only during significant process shifts—provides a more robust framework for long-term supervisory control in pipeline operations.

It is important to note that the observed convergence of the supervisory Q-learning policy within approximately 15 episodes is problem-specific and should not be interpreted as representative of reinforcement learning performance in general. The rapid convergence in this study was enabled by the small, discrete state–action space, the use of tabular Q-learning, and the supervisory role of the learning agent, which selects among bounded adaptation modes rather than directly computing control actions. Additionally, the reward structure is designed to penalize excessive mode switching and large tracking errors, further accelerating policy stabilization. The convergence speed remains dependent on the chosen state discretization, reward formulation, and disturbance scenarios, and it may differ under alternative operating conditions or more complex system configurations.

5.3. Learned Q-Table Analysis

The performance differences observed between cold- and warm-start initialization can be further understood by examining the final Q-tables. Table 4 presents the learned Q-values for the discharge pressure control loop under both initialization strategies. The state definitions are s_1 : not in control, s_2 : close to setpoint, and s_3 : far from setpoint. The Q-Table action set is a_1 : no adaptation, a_2 : slow adaptation, and a_3 : fast adaptation.

Table 4. Learned Q-table values for discharge pressure loop (final episode).

State	Cold Start			Warm Start		
	a_1	a_2	a_3	a_1	a_2	a_3
s_1 : Not in control	−0.22	–	–	−0.1328	–	–
s_2 : Close to setpoint	−0.0002	−0.0066	−0.0065	−0.0002	−0.0067	−0.0071
s_3 : Far from setpoint	−0.0778	−0.0031	−0.0035	−0.0782	−0.0516	−0.0094

Note: “–” denotes a masked action.

The cold-start agent initially oscillates between actions due to the absence of prior knowledge. Over time, it learns that fast adaptation is beneficial when the loop is out of control (s_1 , s_3), while no adaptation is best when close to the setpoint (s_2). However, the learned values remain shallow, with relatively small differences between optimal and suboptimal actions. This explains the occasional mode switching seen in the cold-start results (Figure 2), since the policy lacks a strong bias toward one action.

In contrast, the warm-start Q-table demonstrates sharper separation between action values. Clear preferences emerge for no adaptation near the setpoint (s_2) and fast adaptation when far from the target (s_3). This stability in Q-values translates directly into fewer mode switches, faster convergence, and the smoother transient response observed in the warm-start case. Thus, the Q-table analysis provides a mechanistic explanation for the improved robustness and efficiency of the warm-start initialization strategy. The cold- and warm-start Q-tables were stored and used to compare these two strategies with a fixed-gain controller.

5.4. Results Summary of Q-Table Compared to Other Strategies

5.4.1. Case Study 1: Starting to Steady State

In this test, the different control loop strategies were evaluated using the following performance indices: integral of absolute error (IAE) ($\int |e(t)|dt$) over the simulation window; rise time (the time taken for the system to achieve setpoint within $\pm 5\%$ of the setpoint); number of adaptation mode switches, which measures controller effort and volatility; and total energy consumed, computed using $P = \rho g Q H / \eta$. Table 5 summarizes the perfor-

mance results of Case 1, a pump start, with the discharge pressure control loop for each type of control strategy.

Table 5. Comparison of supervisory strategies for a pump start to steady state.

Strategy	Avg. IAE	Rise Time (s)	Adapt Switches	Energy (kWh)
Fixed-Gain PI	50.823	120.3	—	305.5
Q-Learning (Cold)	2.011	3.4	306	507.4
Q-Learning (Warm)	0.79	1.4	6	517.9

The fixed-gain controller, while it used less energy, also took the longest to achieve the desired pressure setpoint of 102 bar. It is this slow ramp time that contributes to lower control effort and energy usage. The cold-start Q-learning strategy took 3.4 s to reach the setpoint, allowing the flow rate to stabilize and minimizing the change in sending the pressure wave downstream. Since the Q-table started with no prior knowledge, it used ϵ -greedy exploration more often and recorded a total of 306 adaptation switches (Figure 6). The warm-start Q-table used the prior knowledge supplied by the trained Q-table and reduced the average error to 0.79, the rise time to 1.4, and the number of adapt switches to 6 (Figure 7). For the average error, this represents a 98% and 61% improvement over a fixed-gain controller and a cold-start Q-table, respectively. Similarly, improvements of 98.8% and 58.8% were observed in the rise time. Then, compared to the cold start, the warm start had 300 fewer adapt switches—a 98% improvement, which saves unnecessary exploration and improves the controller’s safety by preventing it from inadvertently causing a large change in the process due to poor action selection.

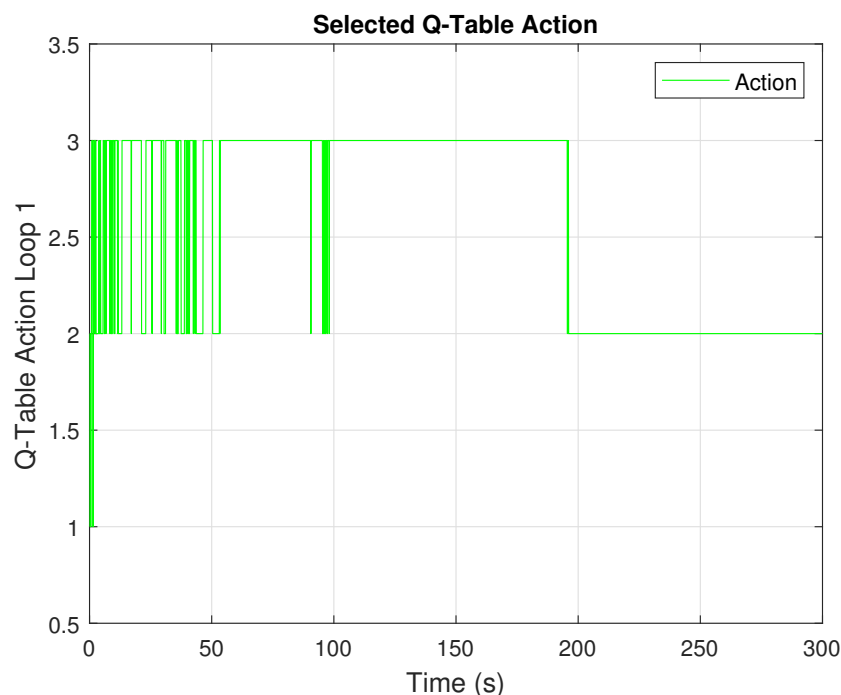


Figure 6. Discharge pressure cold-start Q-table action adapt switching during a pump start.

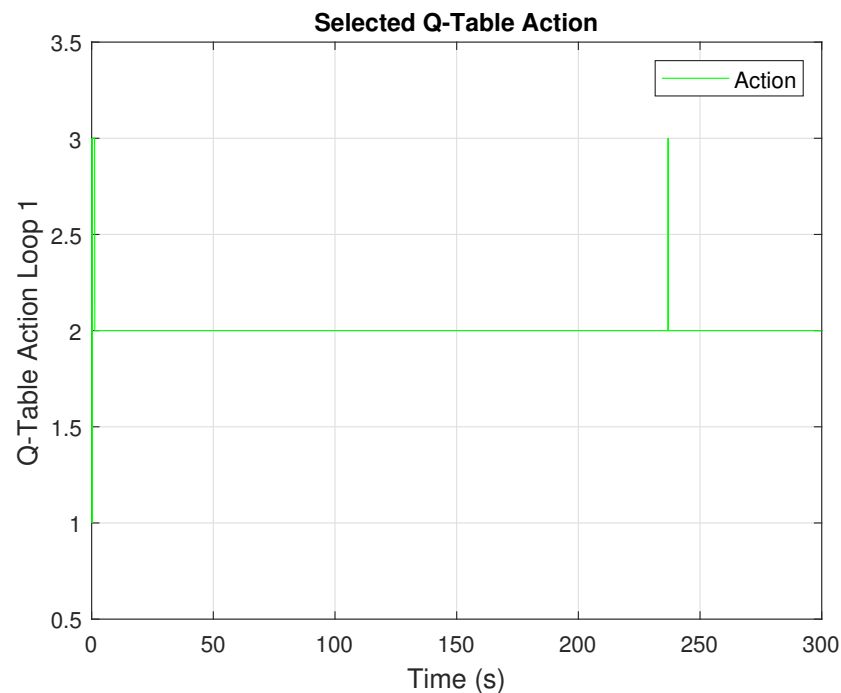


Figure 7. Discharge pressure warm-start Q-table action adapt switching during a pump start.

5.4.2. Case Study 2: Valve Closure and Overpressure Event

Similar to [1], the proposed supervisory Q-learning approach's ability to mitigate transient disturbances by examining the discharge pressure response during a downstream valve closure was evaluated. During pipeline operation, sudden changes in pump operation or valve position can cause a rapid rise in discharge pressure, potentially leading to equipment stress or water-hammer effects if not adequately controlled. This surge scenario presents a stringent test of controller performance, as it requires rapid adaptation to suppress overshoot while maintaining stable recovery to the desired setpoint. The results presented here compare the discharge pressure dynamics under conventional fixed-gain PI control and the adaptive Q-learning supervisory coordination strategy.

Table 6 summarizes the performance of the discharge pressure control loop for each strategy. The fixed-gain PI controller exhibited a pronounced overshoot following the disturbance (105 bar), with a slower recovery to the setpoint and occasional oscillations. In contrast, the proposed Q-learning cold-start strategy achieved a substantially lower peak pressure of 103.5 bar and maintained a more stable recovery profile. However, the proposed Q-learning warm-start strategy achieved a lower peak pressure than the cold-start strategy, at 103.06 bar, as well as reducing the action adapt switches from 288 to 6.

Table 6. Comparison of supervisory strategies for discharge pressure surge event.

Strategy	Peak Pressure (bar)	Adapt Switches
Fixed-Gain PI	105.5	–
Q-Learning (Cold)	103.5	288
Q-Learning (Warm)	103.06	6

Figures 8 and 9 show the discharge pressure response during the valve closure event. The Q-learning strategy reduced overshoot significantly and enabled a faster return to steady-state conditions compared to PI control. Furthermore, analysis of the cold-start strategy's loop-level switching behavior in Figure 10 illustrates the adaptive nature of the coordination mechanism. In the early episodes, exploration led to more frequent switching, but as the agent learned the optimal policy, switching stabilized, and the discharge

pressure loop maintained effective control. The warm-start strategy, on the other hand, used exploitation over exploration, which led to less frequent and unnecessary switching, as shown in Figure 11. These results demonstrate the learning-based supervisory layer's capacity to adapt effectively under surge conditions, thereby improving both transient suppression and overall system stability.

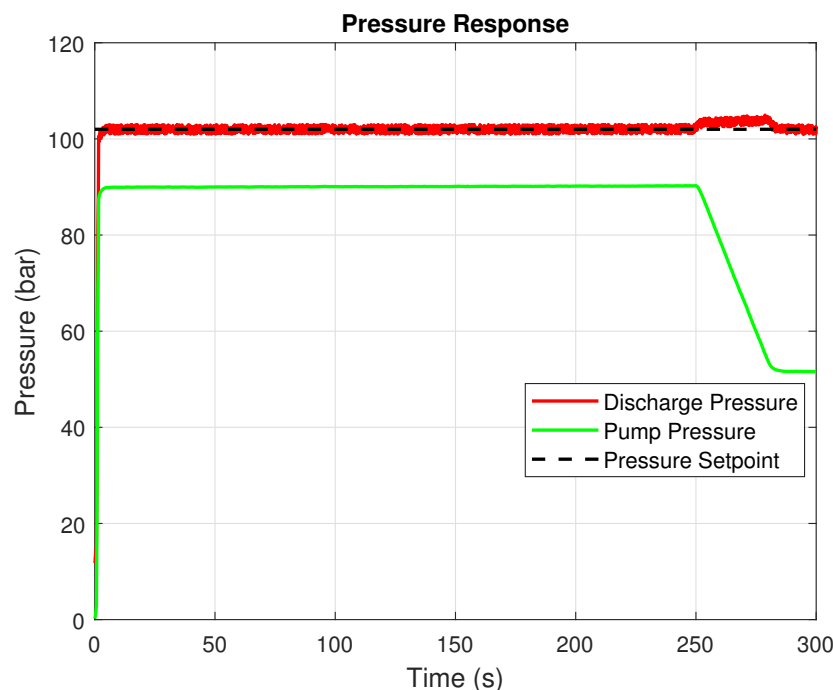


Figure 8. Discharge pressure response to a downstream valve closure using the Q-learning-directed adaptive RLS-based controller.

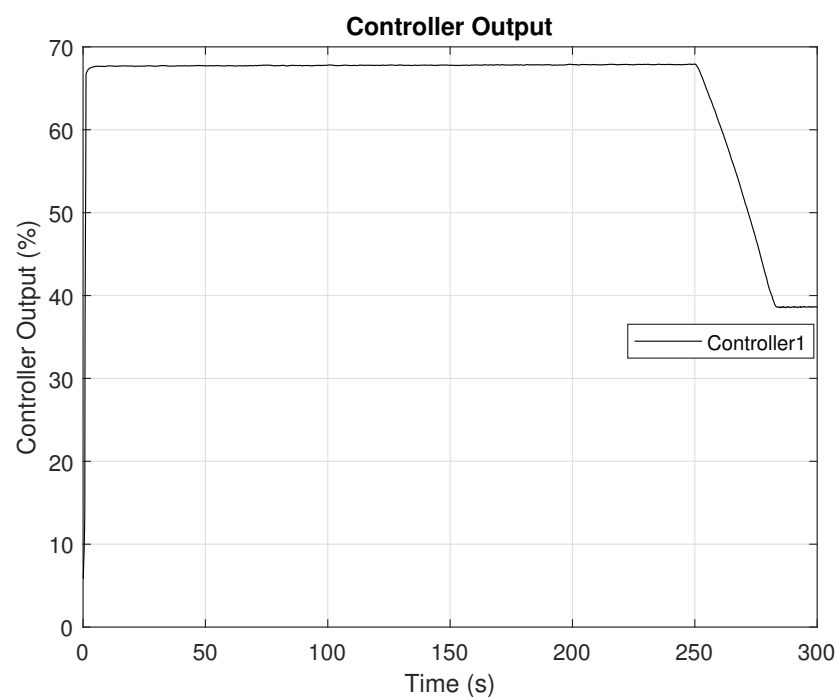


Figure 9. Discharge pressure controller output in response to a downstream valve closure using the Q-learning-directed adaptive RLS-based controller.

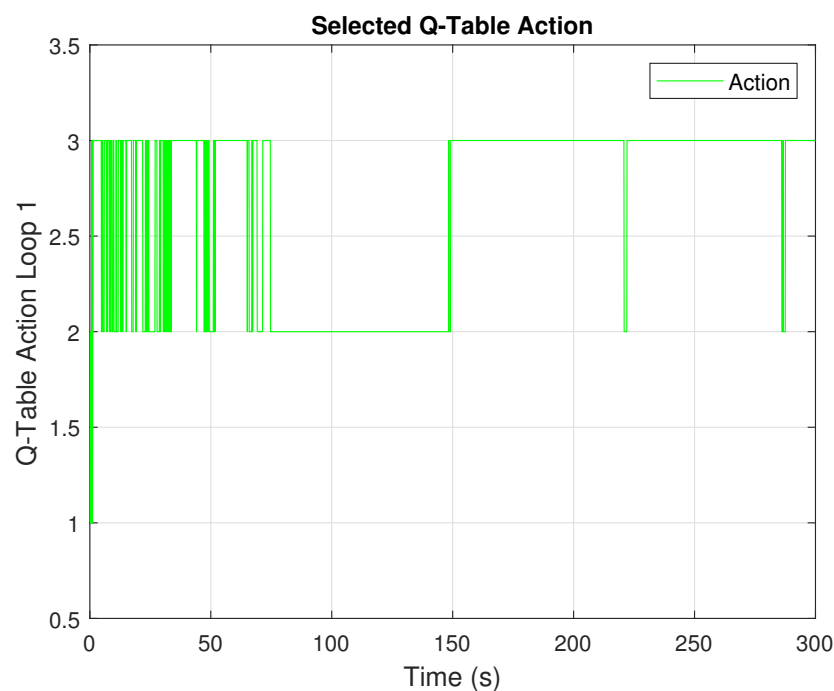


Figure 10. Discrete state transitions for the discharge pressure loop during Q-learning operation. The state remained stable for extended periods after convergence, indicating consistent recognition of the control condition and reduced exploratory switching.

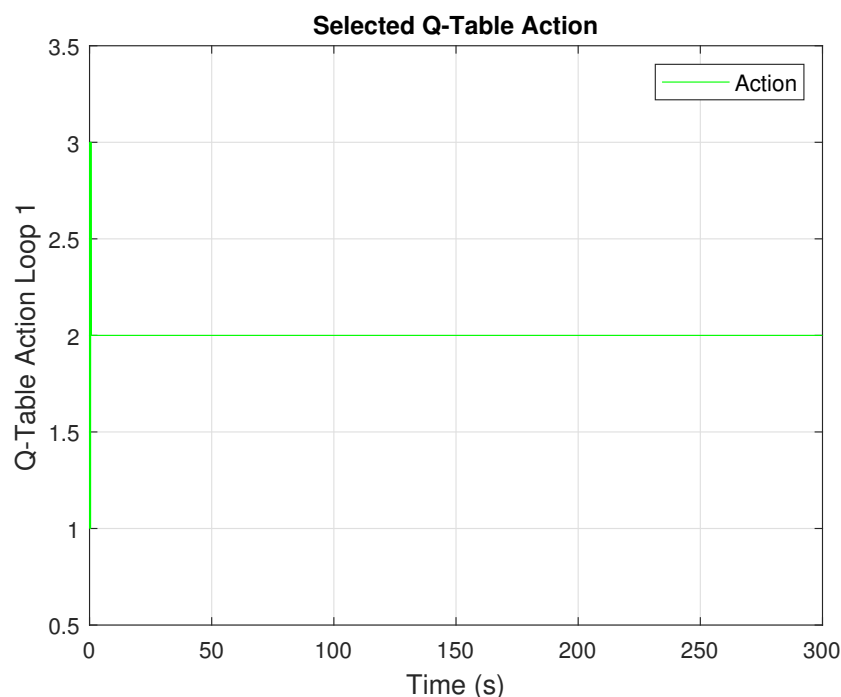


Figure 11. Discrete state transitions for the discharge pressure loop during warm-start Q-learning operation. The state remained stable for extended periods after convergence, indicating consistent recognition of the control condition and limited exploratory switching.

The inclusion of action masking was found to significantly improve learning stability and reduce unnecessary adaptation in inactive loops. Without masking, secondary loops exhibited erratic mode-switching behavior, particularly during transient phases when they failed to control the process. By dynamically constraining their action space, these loops maintained consistent behavior, allowing the Q-learning agent associated with the active loop to converge more rapidly. This modification not only reduced the total number of adap-

tation switches but also improved the overall convergence speed and policy smoothness. Table 6 reflects these benefits, where fewer unnecessary switches correspond to masked, loop-aware exploration. The results demonstrate that action masking is a lightweight yet effective coordination mechanism within decentralized reinforcement-based control frameworks. During the Q-learning simulations, the discrete state variable was defined to capture the operating condition of each control loop (i.e., not in control, near setpoint, or far from setpoint). As shown in the representative discharge loop state trajectory, state transitions occurred primarily during significant disturbances or mode changes. Once the control loop approached steady-state operation, the state remained stable for extended periods, reflecting the effectiveness of the learned adaptation policy. Because the suction loop exhibited similar but less frequent transitions, only the discharge loop's state trajectory is presented here for clarity. This behavior confirms that the Q-learning agent maintained stable state recognition and avoided excessive state chattering once convergence was achieved.

5.4.3. Case Study 3: Partial Valve Closure and Low-Suction-Pressure Event

This case study assesses the effectiveness of the proposed supervisory Q-learning approach for mitigating transient disturbances by analyzing the suction pressure response during partial valve closure, which induces low pressure. In pipeline operation, sudden changes in pump operation or valve position can cause a rapid fall in suction pressure, potentially leading to equipment stress, pump cavitation, and loss of operational stability if not properly controlled. This low-pressure scenario provides a stringent test of controller performance, as it requires rapid adaptation to suppress the underpressure condition while ensuring stable recovery to the desired setpoint of 2 bar. The results presented here compare the suction pressure dynamics under conventional fixed-gain PI control and the adaptive Q-learning supervisory coordination strategy.

Table 7 summarizes the performance results for the suction pressure control loop under each control strategy. The fixed-gain PI controller exhibited a prolonged low-suction-pressure response following the disturbance, with a slower recovery to the setpoint. In contrast, the proposed Q-learning strategy achieved a shorter time spent in the low-pressure condition, a more rapid return to steady state, and a reduction in unnecessary adaptation switches.

Table 7. Comparison of supervisory strategies for low-suction-pressure event.

Strategy	Low Pressure (bar)	Time in Condition (s)	Adapt Switches
Fixed-Gain PI	1.85	50	—
Q-Learning (Cold)	1.88	18.2	48
Q-Learning (Warm)	1.91	16.3	12

Figures 12 and 13 show the suction pressure response during the partial valve closure event. The Q-learning strategy reduced the severity and duration of the low-pressure condition compared to PI control. Notably, as the disturbance persisted, the supervisory agent reassigned control back to the suction loop, recognizing that maintaining the suction head had become the dominant operational constraint. This adaptive reassignment, shown in Figure 14, demonstrates the ability of the Q-learning policy to shift priorities between loops as process conditions evolve dynamically. The loop-level switching behavior, shown in Figures 14 and 15, further illustrates the adaptive supervisory action. Early exploration led to more frequent switching, but the agent converged to a stable policy that prioritized suction pressure recovery. By using exploitation over exploration, the warm-start strategy has less frequent and unnecessary switching, as shown in Figures 16 and 17. The reduction in mode-switching frequency observed in this case and Case 2 further supports the

benefit of limiting simultaneous loop adaptation. These results highlight the robustness of the Q-learning supervisory layer in preventing extended periods of underpressure and maintaining safe operating limits.

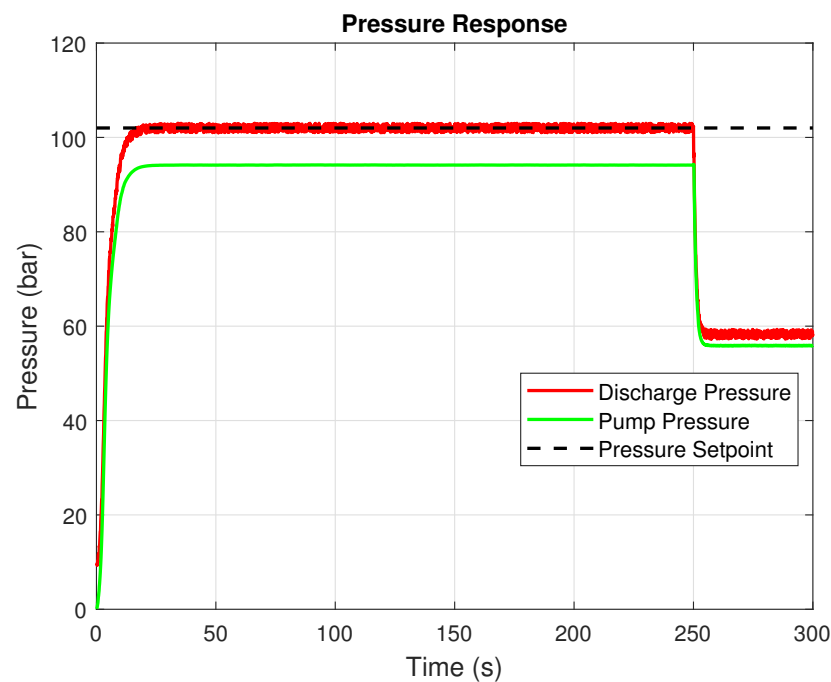


Figure 12. Suction pressure response to a partial valve closure using the Q-learning-directed adaptive RLS-based controller.

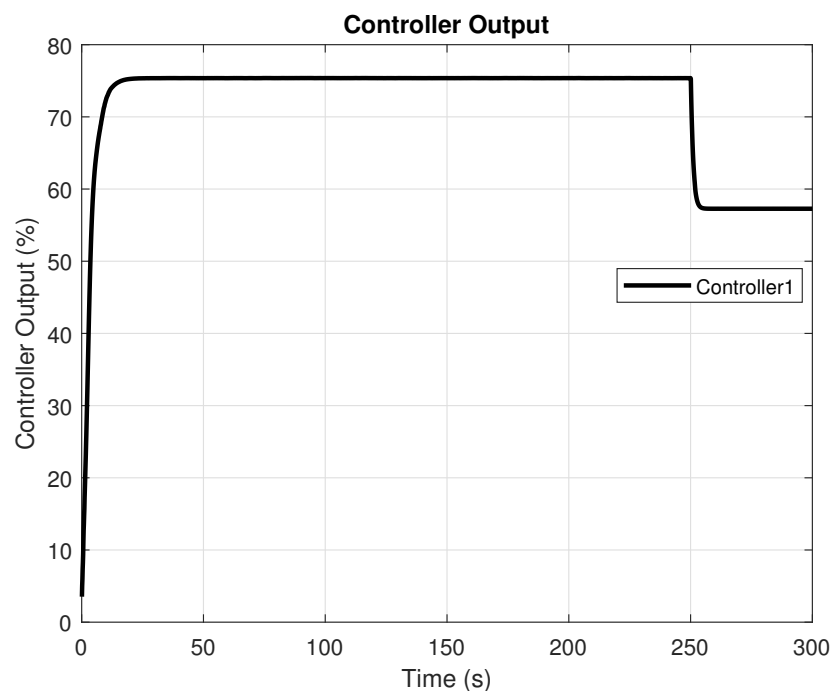


Figure 13. Suction pressure controller output in response to a downstream valve closure using the Q-learning-directed adaptive RLS-based controller.

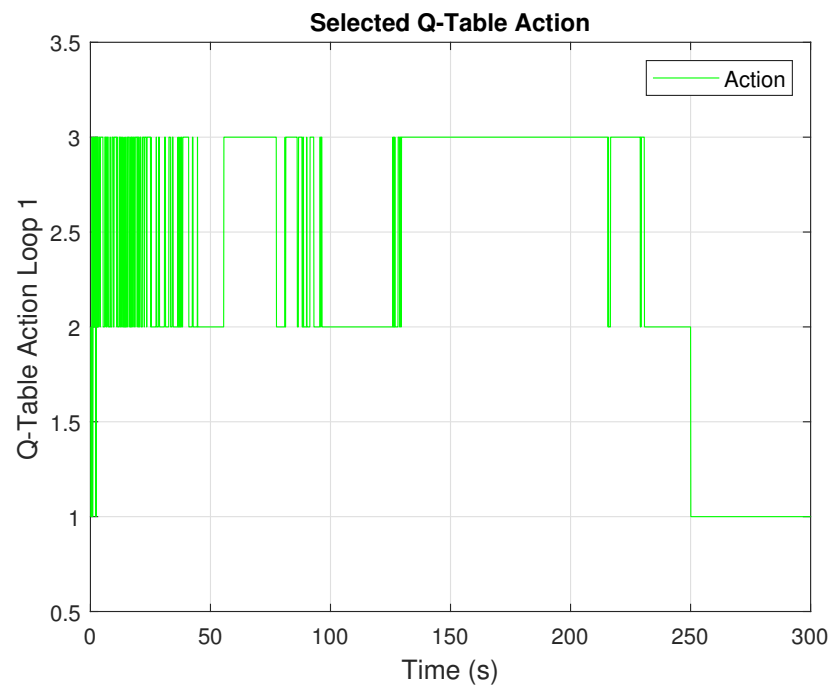


Figure 14. Discrete state transitions for the discharge pressure loop during cold-start Q-learning operation. The state remains stable for extended periods after convergence, indicating consistent recognition of the control condition and reduced exploratory switching.

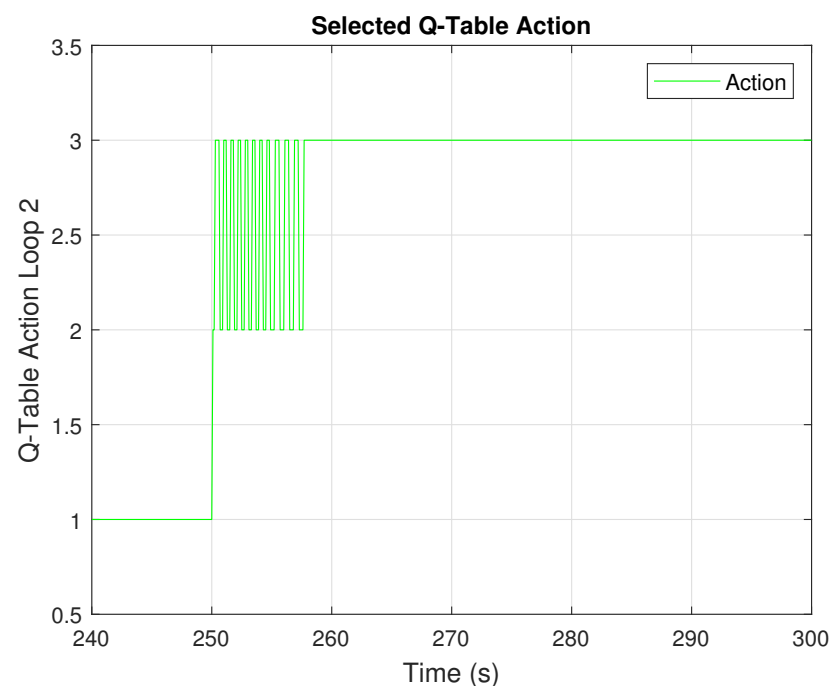


Figure 15. The suction loop cold-start Q-learning mode switches. The loop initially took control during exploration, yielded to the discharge loop as the agent adapted, and then retook control, switching to state 2 near the end, once suction pressure became the dominant constraint.

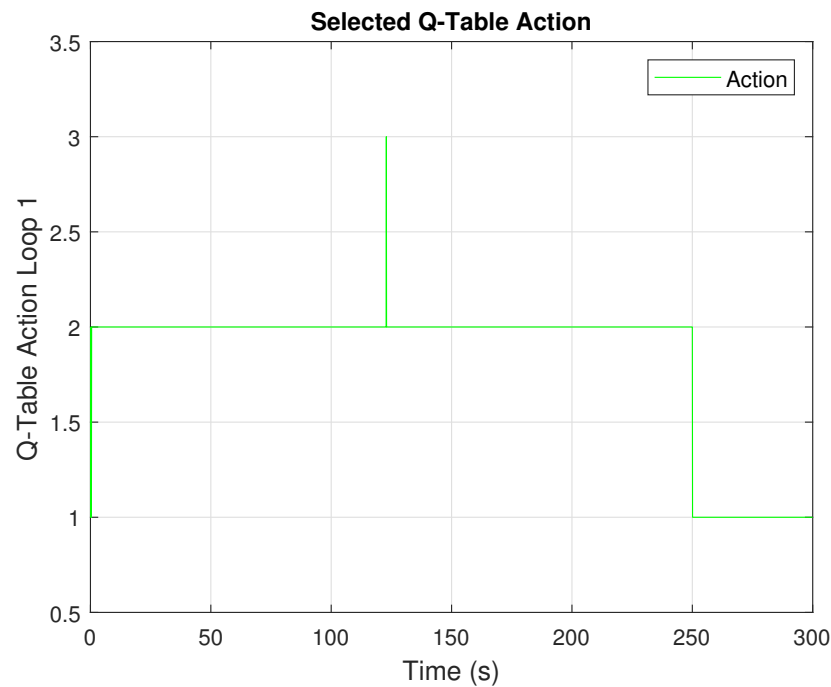


Figure 16. Discrete state transitions for the discharge pressure loop during warm-start Q-learning operation. The state remains stable for extended periods after convergence, indicating consistent recognition of the control condition and reduced exploratory switching.

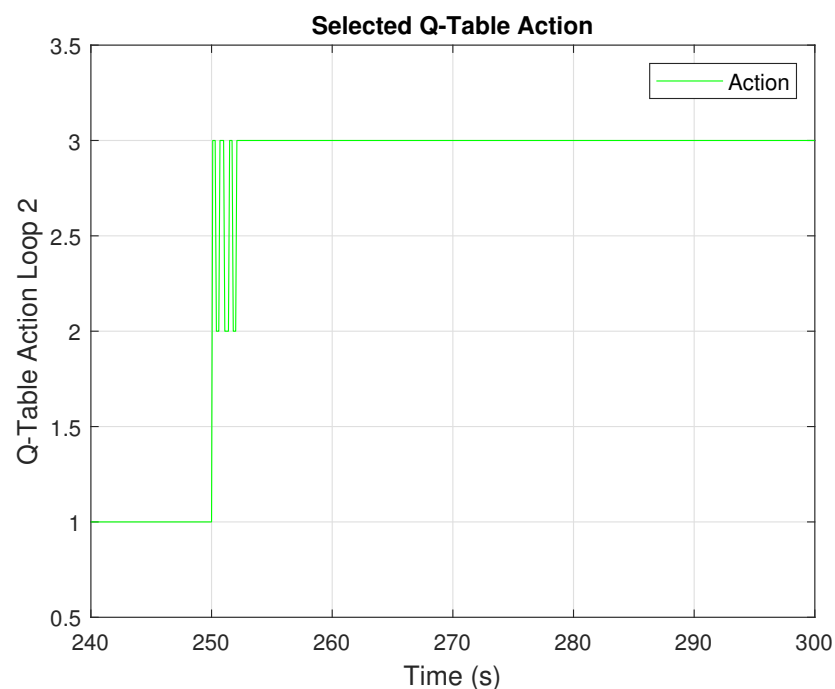


Figure 17. The suction loop warm-start Q-learning mode switches. The loop initially took control during exploration, yielded to the discharge loop as the agent adapted, and then retook control, switching to state 2 near the end, once suction pressure became the dominant constraint.

6. Discussion

The simulation results confirm the potential of decentralized Q-learning agents to enhance responsiveness and efficiency in pump station networks. By extending adaptation beyond binary decisions, the proposed three-level Q-table (no adaptation, slow adaptation, fast adaptation) enabled the controller to better match the magnitude and urgency of

transient disturbances. This finer granularity allowed smoother gain transitions, mitigating instability, integral windup, and energy waste during pressure variations.

A central insight from these studies is the value of adaptive moderation. The supervisory role of the Q-learning agent, combined with bounded actions and conservative exploration, ensures that learning-related transients remain well within the stability margins enforced by the underlying control loops. The intermediate slow-adaptation mode consistently damps small fluctuations without triggering unnecessary aggressive gain changes. This demonstrates that reinforcement-based supervisors can learn to align control effort with disturbance severity, providing a stabilizing buffer and discouraging redundant mode switches. The decentralized architecture also showed strong promise for scalability. Each loop-level supervisor independently selected its adaptation strategy from local feedback, without requiring global coordination or centralized training, and could be extended to additional loops without modification to the learning structure. Despite this autonomy, system-wide behavior improved across multiple control loops, highlighting the cooperative dynamics that emerge naturally from decentralized learning agents. An additional benefit observed was the avoidance of over-control during minor deviations. The supervisors achieved smoother pump operation and fewer motor overshoots, indicating that reinforcement-based adaptation can simultaneously improve stability and operational efficiency.

Nevertheless, several challenges remain before field deployment. Stability guarantees for nonlinear Q-learning behavior remain an open question, particularly under non-stationary conditions. Furthermore, Q-table training episodes must be carefully designed to avoid poor exploratory actions that could compromise safety or equipment health. Future work should include integrating safety filters, hybrid supervisory layers, or rule-based fallback mechanisms to ensure robustness in safety-critical transmission pipelines. Extensions to multi-agent game-theoretic formulations may also provide a structured pathway for coordinating across distributed pump stations, enabling cooperative adaptation at the network scale.

Across multiple case studies, the proposed Q-learning supervisory layer consistently outperformed baseline fixed-gain PI control. For discharge pressure surge scenarios, the approach reduced overshoot by more than an order of magnitude while achieving faster settling and fewer oscillations. In the presence of suction pressure disturbances, the learning-based agent suppressed extended low-pressure conditions and enabled dynamic reassignment of control authority between loops. Comparative experiments on Q-table initialization further showed that warm-start policies yielded faster convergence, fewer mode switches, and higher cumulative rewards than cold-start training. Collectively, these results demonstrate that reinforcement-based supervision provides measurable benefits in terms of stability, responsiveness, and efficiency under transient operating conditions.

The performance evaluation presented in this study was based on high-fidelity MATLAB simulations of a pipeline pump station and focused on representative hydraulic disturbances, including pump start events, valve closures, and pressure transients. These scenarios were selected because they directly excite the dominant dynamics of interest for pressure control and adaptive gain coordination, allowing the effectiveness of the proposed supervisory strategy to be evaluated in a controlled and repeatable manner.

While more complex operating conditions such as multi-product transport, sensor faults, and communication failures are relevant in practical pipeline operation, their inclusion would introduce additional fault diagnosis and supervisory logic beyond the scope of the present work. The objective of this study was to assess the feasibility and control benefits of decentralized Q-learning-based supervision under well-defined hydraulic disturbances, rather than to provide an exhaustive fault-tolerant control solution. Hardware-in-the-loop

testing and extended disturbance classes represent important directions for future work and will be addressed as part of ongoing validation efforts.

It is worth contrasting the proposed decentralized Q-learning supervisory strategy with other widely used advanced control approaches, such as state-filtered disturbance rejection methods and neuroadaptive actor–critic reinforcement learning frameworks. State-filtered disturbance rejection techniques are effective when dominant disturbances can be explicitly modeled or estimated; however, their performance degrades when disturbance characteristics vary significantly across operating regimes, or when accurate process models are unavailable. In large-scale pipeline systems with changing hydraulics, valve configurations, and pump interactions, maintaining such models can be challenging.

Neuroadaptive and actor–critic reinforcement learning methods offer powerful function approximation capabilities for nonlinear systems and can, in principle, achieve near-optimal control policies. However, these methods typically require extensive training data, careful neural network tuning, and increased computational resources, and they often lack transparency and predictable transient behavior—factors that pose challenges for safety-critical industrial deployment.

The performance comparisons in this study focus on fixed-gain and binary adaptive PI control as baseline benchmarks. This choice reflects current industrial practice in pipeline pump stations, where PI-based control remains the dominant approach due to its simplicity, transparency, and ease of certification. While advanced methods such as model predictive control (MPC) and passivity-based or Lyapunov-based control have demonstrated strong performance in academic and pilot-scale studies, their deployment in large-scale pipeline operations is often constrained by model maintenance requirements, computational complexity, and integration challenges with existing control infrastructure.

The intent of this work is not to claim superiority over all modern control paradigms, but rather to demonstrate that meaningful performance gains can be achieved within a PI-centric architecture by augmenting it with a lightweight, supervisory learning layer. Reported improvements in tracking error, settling time, and control smoothness are presented as representative outcomes under the tested scenarios rather than universal guarantees. While formal statistical significance testing and uncertainty quantification were not performed, consistent trends were observed across repeated simulation runs and disturbance cases, suggesting that the observed improvements are systematic rather than incidental. Future work will extend the evaluation framework to include additional benchmark controllers and statistically rigorous performance assessments.

In contrast, the proposed approach prioritizes simplicity, interpretability, and operational robustness. By using a low-dimensional, tabular Q-learning supervisor to regulate the rate of gain adaptation rather than directly generating control actions, the framework preserves the proven stability properties of underlying PI controllers while still enabling adaptive, context-aware behavior. The primary trade-off is reduced optimality compared to fully continuous neuroadaptive controllers; however, this trade-off is intentional and aligned with the practical requirements of industrial pipeline control, where reliability, explainability, and ease of commissioning are paramount.

Despite the encouraging results obtained in this study, several limitations should be acknowledged. First, the proposed supervisory framework has been evaluated exclusively through high-fidelity numerical simulations. While the underlying pump and pipeline model has previously been validated, hardware-in-the-loop or field experiments are required to fully assess real-time implementation challenges, measurement noise, actuator constraints, and communication delays. Second, the Q-learning formulation relies on a discrete state and action space, which was intentionally chosen to enhance interpretability and limit unsafe exploration. However, this discretization may restrict generalization under

operating conditions not represented during training. Third, although practical safeguards such as bounded actions, conditional adaptation, and reduced exploration are incorporated, no formal stability or convergence guarantees are provided for the reinforcement learning component in the presence of nonlinear dynamics. Addressing these limitations through experimental validation, expanded disturbance scenarios, and integration with safety-certified supervisory filters constitutes an important direction for future work.

7. Conclusions

This study introduces a decentralized Q-learning-based supervisory strategy for multi-loop coordination in pipeline pump stations. By embedding reinforcement learning agents at each control loop, the framework enables real-time, bumpless gain adaptation across diverse operating conditions. Simulation results demonstrated clear performance improvements over fixed-gain control, including approximately 96–98% reductions in the integral of absolute error and a rise time of 97–98%, which could collectively contribute to improved long-term pump efficiency. Furthermore, when comparing initialization strategies, warm-start learning reduced the number of adaptation mode switches per episode by over 98% (e.g., decreasing from $306\times$ to $6\times$ after the first learning iteration), while maintaining equivalent convergence performance. These results confirm that the learned supervisory policies generalize across dynamic operating conditions and avoid instability risks observed in naïve cold-start reinforcement learning deployments.

The modular and scalable design of the control architecture makes it well suited for large-scale or geographically distributed pump networks. Its ability to autonomously learn context-aware adaptation policies without requiring centralized oversight points toward more intelligent, resilient, and robust transport infrastructure. This work lays the foundation for future development, including inter-station coordination using game-theoretic multi-agent learning, hybrid adaptive schemes with embedded safety and stability filters, and hardware-in-the-loop validation in real-world pipeline systems.

Author Contributions: Conceptualization, D.A.B. and W.W.W.; methodology, D.A.B.; software, D.A.B.; validation, D.A.B. and W.W.W.; formal analysis, D.A.B.; investigation, D.A.B.; resources, D.A.B.; data curation, D.A.B.; writing—original draft preparation, D.A.B.; writing—review and editing, D.A.B. and W.W.W.; visualization, D.A.B.; supervision, D.A.B.; project administration, D.A.B.; funding acquisition, D.A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author due to privacy reasons.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Brattley, D.A.; Weaver, W.W. Adaptive PI Control Using Recursive Least Squares for Centrifugal Pump Pipeline Systems. *Machines* **2025**, *13*, 1064. [[CrossRef](#)]
2. Aribisala, A.A.; Ghor, U.A.S.; Cavalcante, C.A. The Application of Reinforcement Learning to Pumps—A Systematic Literature Review. *Machines* **2025**, *13*, 480. [[CrossRef](#)]
3. Hajgató, G.; Paál, G.; Gyires-Tóth, B. Deep reinforcement learning for real-time optimization of pumps in water distribution systems. *J. Water Resour. Plan. Manag.* **2020**, *146*, 04020079. [[CrossRef](#)]
4. Joo, J.G.; Jeong, I.S.; Kang, S.H. Deep Reinforcement Learning for Multi-Objective Real-Time Pump Operation in Rainwater Pumping Stations. *Water* **2024**, *16*, 3398. [[CrossRef](#)]
5. Pei, S.; Hoang, L.; Fu, G.; Butler, D. Real-time multi-objective optimization of pump scheduling in water distribution networks using neuro-evolution. *J. Water Process Eng.* **2024**, *68*, 106315. [[CrossRef](#)]

6. Hu, S.; Gao, J.; Zhong, D. Multi-agent reinforcement learning framework for real-time scheduling of pump and valve in water distribution networks. *Water Supply* **2023**, *23*, 2833–2846. [[CrossRef](#)]
7. Zhang, G.; Hu, W.; Cao, D.; Zhang, Z.; Huang, Q.; Chen, Z.; Blaabjerg, F. A multi-agent deep reinforcement learning approach enabled distributed energy management schedule for the coordinate control of multi-energy hub with gas, electricity, and freshwater. *Energy Convers. Manag.* **2022**, *255*, 115340. [[CrossRef](#)]
8. Wang, D.; Li, A.; Yuan, Y.; Zhang, T.; Yu, L.; Tan, C. Energy-saving scheduling for multiple water intake pumping stations in water treatment plants based on personalized federated deep reinforcement learning. *Environ. Sci. Water Res. Technol.* **2025**, *11*, 1260–1270. [[CrossRef](#)]
9. Bordeasu, D.; Prostean, O.; Filip, I.; Vasar, C. Adaptive Control Strategy for a Pumping System Using a Variable Frequency Drive. *Machines* **2023**, *11*, 688. [[CrossRef](#)]
10. Krause, P.C.; Wasynczuk, O.; Sudhoff, S.D. *Analysis of Electric Machinery and Drive Systems*; IEEE Press Series on Power Engineering; Wiley: Hoboken, NJ, USA, 1994.
11. Crowe, C.T.; Elger, D.F. *Engineering Fluid Mechanics*, 9th ed.; Crowe, C.T., Elger, D.F., Roberson, J.A., Eds.; John Wiley & Sons: Hoboken, NJ, USA, 2009.
12. Isermann, R. *Fault-Diagnosis Applications: Model-Based Condition Monitoring: Actuators, Drives, Machinery, Plants, Sensors, and Fault-Tolerant Systems*, 1st ed.; Springer: Berlin/Heidelberg, Germany, 2011.
13. Kallesoe, C.; Cocquemot, V.; Izadi-Zamanabadi, R. Model based fault detection in a centrifugal pump application. *IEEE Trans. Control Syst. Technol.* **2006**, *14*, 204–215. [[CrossRef](#)]
14. Automation, R. *Perform Common Process Loop Control Algorithms*; Publication Logix-WP008B-EN-P; Rockwell Automation: Milwaukee, WI, USA, 2016.
15. Hendricks, E.; Jannerup, O.E.; Sorensen, P.H. *Linear Systems Control: Deterministic and Stochastic Methods*, 1st ed.; Springer: Berlin/Heidelberg, Germany, 2008.
16. Bobál, V.; Johnson, M.A.; Machacek, J.; Bvbm, J.; Machacek, J.; Bc6hm, J. *Digital Self-Tuning Controllers: Algorithms, Implementation and Applications*, 1st ed.; Advanced textbooks in control and signal processing; Springer Nature: London, UK, 2005.
17. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018; Volume 1, p. 25.
18. Chadi, M.A.; Mousannif, H. Understanding reinforcement learning algorithms: The progress from basic q-learning to proximal policy optimization. *arXiv* **2023**, arXiv:2304.00026. [[CrossRef](#)]
19. Ris-Ala, R. *Fundamentals of Reinforcement Learning*; Springer: Berlin/Heidelberg, Germany, 2023.
20. Lakhani, A.I.; Chowdhury, M.A.; Lu, Q. Stability-Preserving Automatic Tuning of PID Control with Reinforcement Learning. *arXiv* **2022**, arXiv:2112.15187. [[CrossRef](#)]
21. Hao, X.; Xin, Z.; Huang, W.; Wan, S.; Qiu, G.; Wang, T.; Wang, Z. Deep reinforcement learning enhanced PID control for hydraulic servo systems in injection molding machines. *Sci. Rep.* **2025**, *15*, 23005. [[CrossRef](#)] [[PubMed](#)]
22. Li, G.; Cai, C.; Chen, Y.; Wei, Y.; Chi, Y. Is Q-learning minimax optimal? a tight sample complexity analysis. *Oper. Res.* **2024**, *72*, 222–236. [[CrossRef](#)]
23. Wang, S.; Si, N.; Blanchet, J.; Zhou, Z. Sample complexity of variance-reduced distributionally robust Q-learning. *J. Mach. Learn. Res.* **2024**, *25*, 1–77.
24. Åström, K.J. Adaptive Control. In *Mathematical System Theory: The Influence of R. E. Kalman*; Springer: Berlin/Heidelberg, Germany, 1991; pp. 437–450. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.