Robust Predictable Control (RPC) for Optimizing Fed-Batch Penicillin Production

Gary Simethy [*] **Margret Bauer** [**] **Ruomu Tan** [***] **Fabian Buelow** [****]

[*] *Process Engineering Dept. HAW Hamburg Hamburg, Germany (email: Gary.Simethy@haw-hamburg.de)*
[**] *Process Engineering Dept. HAW Hamburg Hamburg, Germany (email: Margret.Bauer@haw-hamburg.de)*
[***] *Digitalization & Software Technologies Dept. ABB Corporate Research Germany Mannheim, Germany (email: Ruomu.Tan@de.abb.com)*
[****] *Digitalization & Software Technologies Dept. ABB Corporate Research Germany Mannheim, Germany (email: Fabian.Buelow@de.abb.com)*

Abstract:
Biochemical processes, characterized by nonlinear dynamics and uncertainties, pose significant optimization challenges. This work explores Robust Predictable Control (RPC) as a Reinforcement Learning (RL) algorithm to enhance a fed-batch penicillin production process utilizing the simulation model IndPenSim. Unlike some RL implementations that constrain exploration based on prior knowledge, the selected RPC approach allows the RL agent to explore freely and identify optimal control strategies by itself. We trained the RL agent under disturbance-free conditions and evaluated its performance against various unseen initial process conditions and disturbances. Results show that RPC significantly outperforms other process control methods, including other RL implementations, achieving higher yields with fewer necessary measurements as input for the RL agent. Analyzing two reward functions - penicillin concentration and yield - revealed that using concentration in the reward function improved agent training for maximizing yield, highlighting the importance of reward design in RL. Additionally, the trained RL agent effectively adapted to different action intervals, demonstrating robustness in dynamic environments without retraining. Our findings underscore RPC's potential for optimizing biochemical processes, especially in scenarios with few measurements, paving the way for AI-driven control systems in industrial applications.

Keywords: Robust Predictable Control (RPC), Reinforcement Learning (RL), Penicillin Production, Nonlinear Dynamics, Process Optimization, Reward Function

1. INTRODUCTION

Optimizing biochemical processes is inherently complex due to their nonlinear dynamics, stochastic behavior, and variable operating points. Advanced Process Control (APC) techniques, such as Model Predictive Control (MPC) and Nonlinear MPC (NMPC), have been widely applied to fermentation processes like penicillin production (Ashoori et al. (2008, 2009); Bolmanis et al. (2023)). These methods perform well when accurate models of the process and its uncertainties are available. For instance, Duran-Villalobos et al. (2019) applied Model Predictive Control (MPC) in the IndPenSim penicillin simulation (developed by Goldrick et al. (2015)) to minimize batch-to-batch variation by optimizing substrate feed. Similarly, Kager et al. (2019) implemented Nonlinear Model Predictive Control (NMPC) to enhance process yield by controlling substrate, precursor, and nitrogen flow rates, while also estimating unmeasured variables like oxygen uptake and carbon evolution rates.

In a recent study by Li et al. (2024), NMPC with seven control inputs was compared with reinforcement learning (RL) in the IndPenSim process. Despite assumptions eliminating the use of state estimation, NMPC required 3 to 6 minutes to compute a control action for each 12-minute step, while RL needed only 0.01 to 0.03 seconds. RL also showed fewer oscillations in control trajectories. By integrating historical operating data, RL agents, such as those using Soft Actor-Critic (SAC) and Deep Deterministic Policy Gradient (DDPG), improved sampling efficiency and stability. SAC outperformed DDPG and traditional PID controllers, improving yields by 14%, while NMPC achieved 15%.

RL methods have proven to be effective in optimizing fed-batch bioreactors, even under uncertain dynamics. A two-stage RL framework, which initially trains on an approximate model before fine-tuning with real data, outperformed NMPC while reducing process evaluations, enhancing both time and cost efficiency (Petsagkourakis et al.

(2019)). In a polymerization process, RL combined with Monte Carlo learning managed stochastic disturbances, improving robustness and control accuracy (Haeun et al. (2021)). Effective reward function design is crucial for ensuring safe and efficient operation in RL, as demonstrated in various control processes, including those with abrupt phase transitions in which phase segmentation improved performance (Haeun et al. (2021)).

In this work, we apply Robust Predictable Control (RPC), an RL algorithm by Eysenbach et al. (2021), to the IndPenSim process. Unlike prior approaches that limit exploration by incorporating prior knowledge, our RL agent explores freely to discover optimal strategies. We trained the agent under disturbance-free conditions and tested it against 26 unseen scenarios, including process disturbances, batch-to-batch variations, and faults. We also evaluate the impact of different reward functions. Our results demonstrate that RPC not only achieves higher yields under disturbances compared to SAC, DDPG, and NMPC (implemented by Li et al. (2024)), but also requires fewer measurements, highlighting its robustness and potential for a broader application in biochemical process optimization.

2. ROBUST PREDICTABLE CONTROL

Reinforcement Learning (RL) is a closed-loop framework where an agent learns to make sequential decisions by interacting with its environment, in order to maximize long-term rewards (Sutton and Barto (1998)). When the environment can be simulated with high fidelity, the agent can learn optimal strategies in a controlled setting, reducing the risk of negatively impacting the real process (Haeun et al. (2021)).

Robust Predictable Control (RPC) builds upon actor-critic algorithms like SAC, emphasizing robustness, generalization, and computational efficiency. It integrates concepts from information bottlenecks, model-based RL, and bits-back coding to learn a compressed policy in a latent space.

The information bottleneck principle limits the amount of information the RL agent can use, which helps prevent overfitting to the training environment. This constraint changes the agent's behavior, driving it to states with predictable dynamics by minimizing the information or 'bits' needed (Eysenbach et al. (2021)).

RPC also focuses on temporally extended policies, where information from one time step is used to predict subsequent steps. When these predictions are accurate, the agent relies on them instead of frequently querying the environment. Furthermore, the distribution from which actions are sampled can be adjusted to increase the likelihood of visiting states that are easier to compress, improving learning efficiency (Eysenbach et al. (2021)).

The architecture of RPC involves training an encoder to create a compact representation of the current state and a high-level policy to decode this representation into actions. The objective is to maximize rewards while minimizing the number of bits used. Variational Information Bottleneck (VIB) is applied to compress the sequence of states, encouraging the agent to prioritize states where its learned model can accurately predict future states. Results indicate that RPC can develop policies that are more robust to missing observations and noise, while achieving comparable or higher rewards than other actor-critic algorithms (Eysenbach et al. (2021)).

In this work, we experiment with different levels of compression by selecting the number of bits (a measure of how much information from the observations is preserved after compression) as a hyperparameter. We explored 1-bit, 10-bit, and 15-bit configurations, while acknowledging that other configurations could also be explored. Other hyperparameters, such as the number of neurons for the different neural networks comprising the agent, batch sizes, and learning rates were kept consistent with the experiments in Eysenbach et al. (2021).

3. SIMULATION SETUP AND ENVIRONMENT CONFIGURATION

The RL environment selected for this study is IndPenSim, a validated simulation of an industrial-scale Penicillium chrysogenum fermentation process, developed by Paul and Thomas (1996). IndPenSim simulates fermenter volumes up to 100,000 L and incorporates a structured model accounting for key factors such as biomass growth, morphology, and metabolic production/degeneration, providing a comprehensive representation of the fermentation process. Originally implemented in MATLAB R2013B and updated to MATLAB R2018B, the model was validated with historical data, showing good agreement with the real process (Goldrick et al. (2015)). Additionally, a standalone dataset of 100 batches, including a Raman spectroscopy device, was made available by Goldrick et al. (2019), further enhancing simulation capabilities with options for control strategies, concentration disturbances and inhibition effects. IndPenSim also simulates process faults such as agitation, aeration issues, and sensor malfunctions.

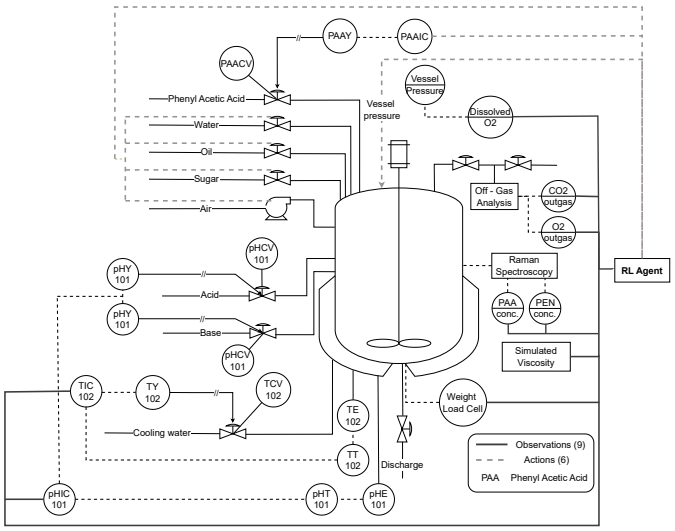

Fig. 1. Piping and Instrumentation Diagram (P&ID) of the IndPenSim process, featuring a schematic representation of an RL agent, its observations, and actions.

To configure IndPenSim for RL, the simulation was adapted to the OpenAI Gym format, a common standard

for RL environments (Brockman et al. (2016)). Figure 1 illustrates the closed-loop control of IndPenSim, with the RL agent executing control actions (represented by grey dashed lines). The six control actions selected include manipulation of substrate (sugar), oil, water, and air flowrates, along with the setpoint for the PI controller of Phenyl Acetic Acid (PAA) and vessel pressure.

The environment outputs 35 observations, but only 9 are provided to the RL agent (represented by black solid lines): temperature, dissolved O_2, O_2 outgas, CO_2 outgas, PAA concentration, viscosity, vessel weight, penicillin (PEN) concentration, and pH. These observations are critical for process control and can be measured in real-world applications.

Although a Python implementation of IndPenSim exists, testing showed it deviated from the original MATLAB results. Therefore, we opted to use the MATLAB version of the simulator while programming the RL agents in Python. Communication between the MATLAB simulator and Python agents was achieved using the MATLAB engine API, enabling the execution of MATLAB scripts within the Python environment. This approach increased training times, but the improved accuracy of results justified the trade-off.

4. TRAINING PROCEDURE

Training an RL agent directly on a real chemical plant poses risks, especially during exploration. Instead, the agent can be trained using either plant data (offline RL) or a simulated environment. Offline RL, which relies on past experiences, limits exploration. Simulations, however, allow for a wider variety of experiences through random actions, enabling greater exploration while avoiding real-world dangers. Using process simulators provides a safe and realistic environment for training. However, a simulation may not always accurately represent the real process.

Each gym environment used in training requires an action and observation space. In our setup, the action space consists of six actions, and the observation space includes nine observations, both initialized as continuous due to the nature of the simulation.

In our experiments, each batch run lasts 230 hours. For the first 70 hours, the biomass remains in its rapid growth phase, followed by 160 hours where the RL agent controls the process, taking actions every hour (160 steps per episode). While shorter action intervals could be used, as in Li et al. (2024), we opted for hourly actions to reduce training time. We also evaluated agent performance with actions taken every two hours.

Figure 2 illustrates the interaction between the agent, actions, observations, and rewards. The agent's policy is represented as a neural network that maps observations to actions, aiming to maximize rewards. Each episode begins with the environment running a predefined recipe for the first 70 hours, after which the agent takes control.

During each training step, the agent's action is sent to the MATLAB simulation via the MATLAB engine API, which serves new observations after one hour of simulated time. This loop continues until either all 160 steps are completed

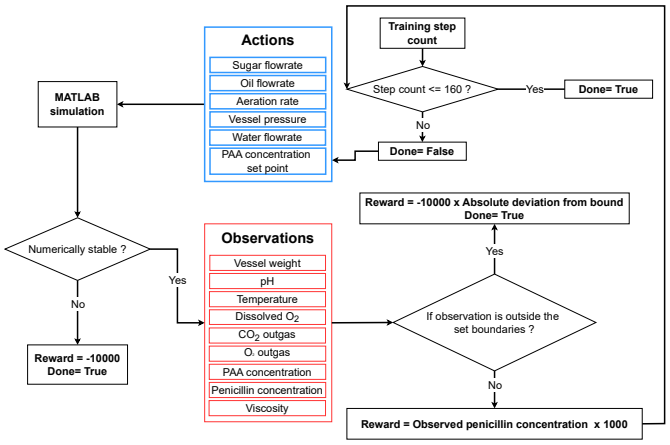

Fig. 2. Relationship between actions, observations, and rewards for RL agent training in this work.

or the episode terminates early due to constraint violations or simulation errors. The reward is based on the observations: positive rewards are proportional to penicillin concentration, while violations result in negative rewards and termination of the episode. Penicillin concentration is scaled by 1000 to ensure the agent recognizes incremental improvements, critical for the batch process. Alternative reward functions, such as yield, are discussed in Section 6.

5. RESULTS

As discussed in Section 2, the chosen RL algorithm uses information bottlenecks to train compressed policies, with the compression rate adjustable by the number of bits. Experiments were conducted with 1-bit, 10-bit, and 15-bit RPC agents to assess how bit selection affects training and outcomes. In addition, the impact of two different reward definitions was evaluated, one using penicillin yield and the other using penicillin concentration. The relation of these quantities can be seen in Equation 1,

$$\text{yield}_{\text{PEN,total}} = V_{\text{end}} \times C_{\text{end}} + \sum_{i=1}^{n} \overbrace{V_i \times C_i}^{\text{yield}_{\text{PEN},i}} . \qquad (1)$$

V_{end} and C_{end} represent the volume and penicillin concentration in the reactor at the end of a batch, respectively. For each reactor discharge during the batch, V_i and C_i denote the corresponding volume and penicillin concentration, where n represents the total number of discharges.

Table 1 highlights the effect of reward function selection and bit compression on total penicillin yield. In all cases, using penicillin concentration as the reward function resulted in better overall performance. Each agent was trained for 50,000 steps (approx. 312 episodes), with both the 1-bit and 10-bit RPC agents outperforming the 15-bit agent regardless of the reward function. Consequently, further experiments focused on the 1-bit and 10-bit agents.

The goal of this study is not only to maximize yield but also to minimize batch-to-batch variation by mitigating process disturbances. The RPC agents (1-bit and 10-bit) were trained under specific initial conditions and validated on 26 unseen conditions. Figure 3 shows the performance

Reward function	Penicillin yield at end of batch (Kg)		
	1 bit	10 bit	15 bit
Penicillin yield	4796 Kg	4940 Kg	3718 Kg
Penicillin concentration	4820 Kg	4961 Kg	4657 Kg

Table 1. Comparison of penicillin yield at end of batch for different reward functions and bit resolutions.

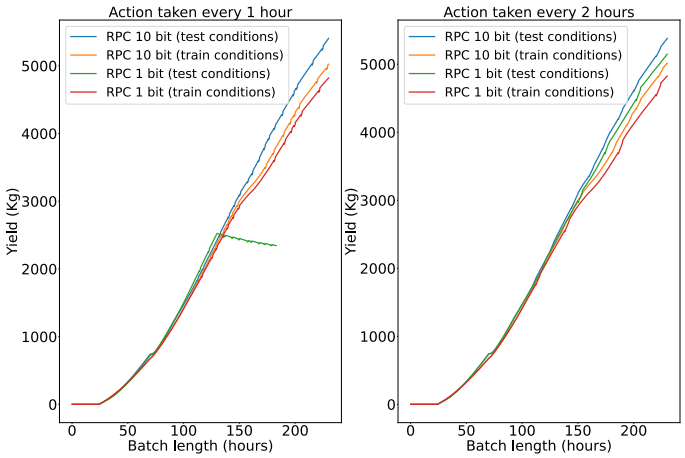

Fig. 3. Penicillin yield for 1-bit and 10-bit RPC agent at different action intervals.

of 1-bit and 10-bit RPC for the train conditions versus one of the 26 test conditions. The agents were trained to take actions every hour. Between the 1- and 10-bit RPC agent, the 1-bit did not violate any constraints and thus completed the required batch length of 230 hours. Under test conditions, the 1-bit RPC terminated the batch prematurely due to a violation of the viscosity constraint. Interestingly, when the same agent, trained to take actions every hour, was instructed to act every two hours during evaluation, it performed better under those conditions without requiring any retraining. The reasoning behind this behaviour is discussed in Section 6.

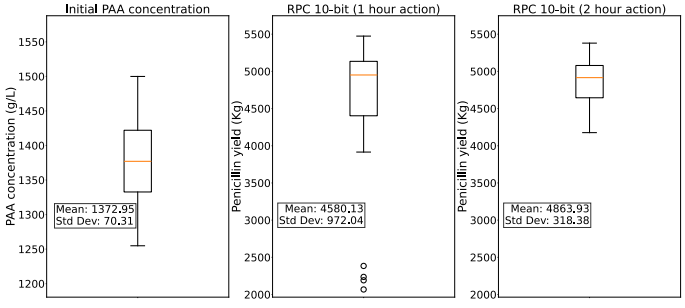

Fig. 4. Effect of different initial PAA concentrations on the penicillin yield for the trained 10-bit RPC agent during tests.

Based on the sensitivity analysis by Goldrick et al. (2019), PAA was identified as the most critical parameter in the process. Therefore, 26 different initial PAA concentrations were used to evaluate the performance of the 10-bit RPC with 1-hour and 2-hour action intervals. As shown in Figure 4, the impact of varying initial PAA concentrations on batch yield is clear. Although all PAA concentrations resulted in yields above the minimum yield of 2000 kg (Goldrick et al. (2019)), the 1-hour action intervals showed greater variability, with some batches deviating significantly from the mean. In contrast, the 2-hour action intervals produced superior performance with more consistent yields. The performance differences between the 10-bit RPC agent acting every hour versus every two hours are discussed in detail in Section 6.

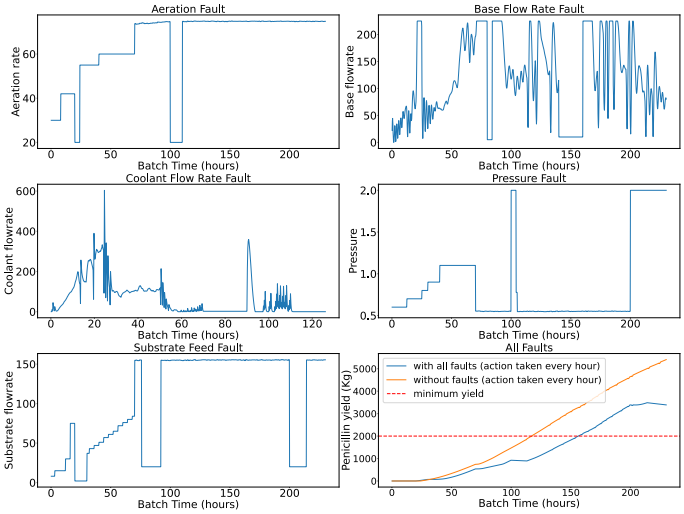

Fig. 5. Individual process faults and their impact on the performance of the 10-bit RPC agent taking actions every hour.

In a real process, disturbances to the process in the form of sensor noise and faults can affect the overall process. To see how a 10-bit RPC agent behaves under such conditions, it is tested against the faults (which are taken from a real process by Goldrick et al. (2019)) shown in Figure 5.

Figure 5 shows individual faults in the process and the performance of the 10-bit RPC agent when all faults are activated simultaneously. Despite the drop in the yield, it still meets the minimum yield requirement. Even with all the faults activated, the 10-bit RPC agent can complete the entire batch without any violation of the constraints and still achieve yields much higher than the minimum requirement.

6. DISCUSSION

Three versions of RPC agents—1-bit, 10-bit, and 15-bit—were trained using two types of reward functions. The choice of reward function is critical for training an RL agent. In this study, we utilized penicillin concentration and penicillin yield as reward functions, which are related as shown in Equation 1. The RL agent performed better when penicillin concentration was used as the reward function. To understand this, we analyzed the plots of penicillin yield and concentration. As illustrated in Figure 6, the penicillin yield profile exhibits small dips and appears noisier compared to penicillin concentration. These dips occur due to vessel discharges implemented to prevent overflow; while they do not affect penicillin concentration, they impact the mixture volume, which is related to penicillin yield (as shown in Equation 1). This results in a smoother reward function for penicillin concentration, enhancing training performance.

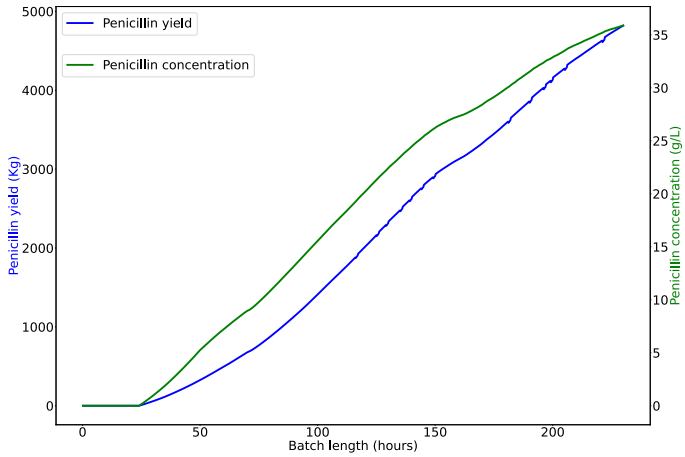

Fig. 6. Profile comparison of the penicillin yield and concentration for a 10-bit RPC agent taking hourly actions during a test batch.

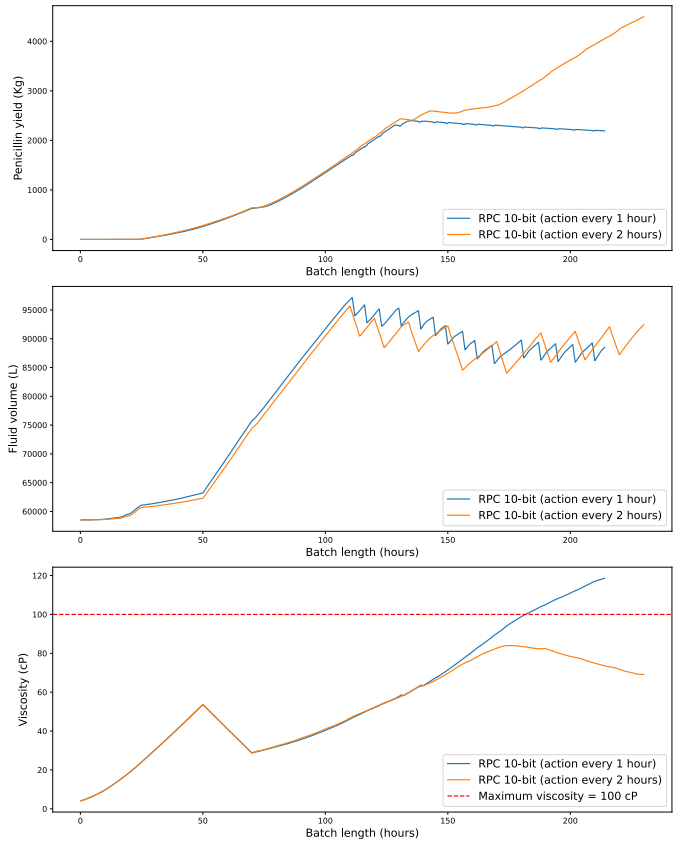

Fig. 7. Impact of the choice of action interval length for 10-bit RPC agent.

During tests under one set of initial conditions, the 10-bit RPC agent, trained to take actions every hour, exhibited inferior performance compared to when it was configured to take actions every two hours. This observation can be explained by analyzing Figure 7: When actions were taken every hour, the batch process failed to complete due to a violation of the viscosity constraint. In contrast, when actions were taken every two hours with the same agent and process conditions, significantly higher penicillin yields were achieved without violating any constraints.

The reason for this disparity can be understood by examining the discharge rates depicted in Figure 7. With longer action intervals, the vessel discharge was delayed for a longer period, allowing for more available volume for biomass growth and extending the residence time in the vessel. Given the slow dynamics of biochemical processes, it takes time for the effects of the agent's actions to become observable. With a longer residence time, these effects are more pronounced, enabling the agent to make better informed decisions based on the observed outcomes. When actions were taken every hour, the agent could not effectively control the viscosity, as subsequent actions were initiated before the full effects of the previous actions had manifested. As a result, the performance of the 10-bit RPC agent acting every 2 hours under varying input conditions achieved high penicillin yields with a small variation compared to the same agent acting every 1 hour (see Figure 4). We therefore strongly advise to adjust an RL agent's action interval to the observed process dynamics.

The discharge could have been treated as an additional action by the RL agent, but we chose it to be a discrete operation (either fully open or fully closed between the action intervals), depending on whether the vessel was at risk of overflowing. If discharge were to be treated as an additional action, due to the chosen RL algorithm, it would need to have continuous values just like the other actions in this study. For future work, an alternative approach could involve a multi-agent system, where one agent is solely responsible for controlling the discharge and dynamically adjusting the flow rate. However, if all discharge flow rates are non-zero, the process would no longer be a fed-batch process but would resemble more of a continuous process.

Furthermore, under different process faults (all active in the same batch) none of the batch yields were below target (or minimum) in this work. This shows the generalization capabilities of the 10-bit RPC agent to take actions at different intervals, reject disturbances, achieve high penicillin yields and operate safely (no violation of constraints) without the need for retraining, inclusion of disturbances or other prior knowledge in the agent training. Finally, Figure 8 compares the best RL agent from this work with other implementations. As reported in Goldrick et al. (2019), when operated manually or with an optimal predefined recipe using Sequential Batch Control (SBC), there were batches below target. With the help of Raman Spectroscopy for online measurements and PI controller to control the PAA concentration which is a critical parameter, the performance was improved as no batches were below target and the overall penicillin yield increased by 20% in comparison to SBC. Nonlinear MPC implemented by Li et al. (2024) was able to increase the yield by 37% (with respect to SBC). However, it assumed a full state feedback which means all 35 observations were recorded and used to get the next optimal control action. Soft Actor Critic was also implemented as an RL controller (Li et al. (2024)) and trained with 16 observations for 1 million steps. The SAC implementation was able to achieve yields 35% higher than SBC, resulting in performance comparable to that of NMPC. Finally, the 10-bit RPC implementation in this study was able to achieve the highest increase of 67% in penicillin yield compared to SBC while requiring the least

number of observations (9 observations) all of which are actually measurable in a process.

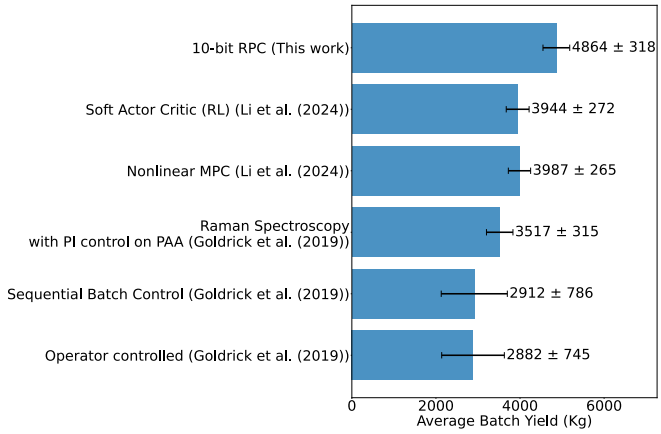

Fig. 8. Comparison of average penicillin yields for different control strategies on IndPenSim.

7. CONCLUSION

In this study, RPC was chosen as the algorithm for RL agents trained as high-level controllers in a penicillin process simulation, with the goals of optimizing penicillin yield, ensuring safe operation, and generalizing to unseen scenarios. Among all trained agents, the 10-bit RPC RL agent demonstrated superior performance, showing adaptability and robustness across unseen conditions, highlighting its real-world applicability. The agent's ability to generalize behaviors, including executing actions outside its training range, underscores its flexibility.

Even when encountering disturbances, the agent consistently met control objectives, maintaining yields above the minimum requirement. These findings suggest that the RL agent can adapt to system changes, a crucial feature for dynamic industrial settings. Achieving these results with fewer training steps and observations makes the approach practical for deployment, where data collection can be costly. Its strong performance with limited data inputs further streamlines AI-driven control in industrial applications.

These results align with the broader vision of increasing autonomy in industrial systems, as outlined by Gamer et al. (2020), where AI-driven methods like reinforcement learning can enhance the adaptability and efficiency of complex processes in dynamic environments.

REFERENCES

Ashoori, A., Ghods, A.H., Khaki-Sedigh, A., and Bakhtiari, M.R. (2008). Model predictive control of a nonlinear fed-batch fermentation process. In *2008 10th International Conference on Control, Automation, Robotics and Vision*, 1397–1400. doi: 10.1109/ICARCV.2008.4795727.

Ashoori, A., Moshiri, B., Khaki-Sedigh, A., and Bakhtiari, M.R. (2009). Optimal control of a nonlinear fed-batch fermentation process using model predictive approach. *Journal of Process Control*, 19(7), 1162–1173. doi: https://doi.org/10.1016/j.jprocont.2009.03.006.

Bolmanis, E., Dubencovs, K., Suleiko, A., and Vanags, J. (2023). Model predictive control—a stand out among competitors for fed-batch fermentation improvement. *Fermentation*, 9, 206. doi:10.3390/fermentation9030206.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *CoRR*, abs/1606.01540. URL `http://arxiv.org/abs/1606.01540`.

Duran-Villalobos, C., Goldrick, S., and Lennox, B. (2019). Multivariate statistical process control of an industrial-scale fed-batch simulator. *Computers & Chemical Engineering*, 132, 106620. doi: 10.1016/j.compchemeng.2019.106620.

Eysenbach, B., Salakhutdinov, R.R., and Levine, S. (2021). Robust predictable control. *Advances in Neural Information Processing Systems*, 34, 27813–27825.

Gamer, T., Hoernicke, M., Klöpper, B., Bauer, R., and Isaksson, A. (2020). The autonomous industrial plant – future of process engineering, operations and maintenance. *Journal of Process Control*, 88, 101–110. doi: 10.1016/j.jprocont.2020.01.012.

Goldrick, S., Duran-Villalobos, C., Jankauskas, K., Lovett, D., Farid, S., and Lennox, B. (2019). Modern day monitoring and control challenges outlined on an industrial-scale benchmark fermentation process. *Computers & Chemical Engineering*, 130. doi: 10.1016/j.compchemeng.2019.05.037.

Goldrick, S., Ştefan, A., Lovett, D., Montague, G., and Lennox, B. (2015). The development of an industrial-scale fed-batch fermentation simulation. *Journal of biotechnology*, 193, 70–82.

Haeun, Y., Kim, B., Kim, J., and Lee, J. (2021). Reinforcement learning based optimal control of batch processes using monte-carlo deep deterministic policy gradient with phase segmentation. *Computers & Chemical Engineering*, 144, 107133. doi: 10.1016/j.compchemeng.2020.107133.

Kager, J., Tuveri, A., Ulonska, S., Kroll, P., and Herwig, C. (2019). Experimental verification and comparison of model predictive, pid and model inversion control in a penicillium chrysogenum fed-batch process. *Process Biochemistry*, 90. doi:10.1016/j.procbio.2019.11.023.

Li, H., Qiu, T., and You, F. (2024). Ai-based optimal control of fed-batch biopharmaceutical process leveraging deep reinforcement learning. *Chemical Engineering Science*, 292, 119990.

Paul, G. and Thomas, C. (1996). A structured model for hyphal differentiation and penicillin production using penicillium chrysogenum. *Biotechnology and bioengineering*, 51(5), 558–572.

Petsagkourakis, P., Sandoval, I., Bradford, E., Zhang, D., and del Rio-Chanona, E. (2019). Reinforcement learning for batch-to-batch bioprocess optimisation. *Computer Aided Chemical Engineering*, 46, 919–924. doi: 10.1016/B978-0-12-818634-3.50154-5.

Sutton, R. and Barto, A. (1998). Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*, 9(5), 1054–1054. doi:10.1109/TNN.1998.712192. URL `https://ieeexplore.ieee.org/document/712192`.