Imperial College London

Hierarchical Reinforcement Learning for Plant-wide Control

Maximilian Bloor
Supervised by Dr Antonio del Rio Chanona and Niki Kotecha

Department of Chemical Engineering, Imperial College London, UK

Abstract

Proportional-Integral-Derivative (PID) controllers provide a simple and effective control solution used widely across the chemical industry. Their gains require tuning with consideration of the entire system's dynamics to ensure adequate control. Traditional muli-loop PID tuning methods depend on dynamic models or heuristics to adjust these gains. In this paper, a hierarchical reinforcement learning algorithm designed for plant-wide control is presented, consisting of a high-level artificial neural network and low-level PID controllers using an evolutionary algorithm for policy optimisation. Three case studies including a reactor, reactor-separator and reactor-separator-recycle systems are used to test the algorithm's performance at setpoint tracking, noise and disturbance rejection. The hierarchical reinforcement learning algorithm is compared to derivative-free optimisation, multiloop relay tuning, and a nonlinear model predictive controller (NMPC) for each case study. Across all case studies, the hierarchical reinforcement learning algorithm has a lower integral square error than both PID tuning methodologies. However, the NMPC showed that manipulation of other units in the system could result in enhanced setpoint tracking performance that PID-based methods could not replicate. The robustness of all controllers is investigated with a parametric mismatch analysis, which replicates the fouling of the reactor cooling jacket and the degradation of the reactor catalyst. This showed that due to the hierarchical reinforcement learning algorithm's lack of dependence on an accurate model, it can outperform the NMPC when there is a plant-model mismatch.

Keywords: *Reinforcement Learning, Machine Learning, Process Control*

1 Introduction

Industrial chemical processes are crucial to the modern economy. With rising energy prices and the need for reduced greenhouse gas emissions, it is paramount that these industrial plants are operated efficiently and safely. Plant-wide control is a control philosophy in which the operation of multiple interconnected units is considered. This allows the control system to consider interactions between units, leading to improved overall performance. However, these interactions and non-linearities pose a challenging problem in the implementation of plant-wide control [1].

Advanced control methods such as model predictive control, can provide satisfactory results for plant-wide control problems. Still, they often suffer from the challenges of model identification and uncertain plant conditions [2]. Due to their simplicity and adequate control performance, Proportional-Integral-Derivative (PID) controllers are widely used across the chemical industry. Furthermore, PID controllers rely on only three parameters, which allow for reduced design space and ease of implementation. However, these parameters are unique to the dynamic system and it is necessary to carefully adjust them to ensure stable operation. The simplest method of tuning is via trial and error, however, this requires an experienced operator, and it can be difficult to achieve satisfactory control in multiloop control systems such as plant-wide control. There are more rigorous approaches that attempt to tune the

PID controllers in a system while considering the controller interactions. These methods include the detuning method [3], sequential loop tuning method [4], independent loop method [5] and multiloop relay auto-tuning [6]. The first three methods sequentially tune the controllers in the loop according to different metrics. The final method excites the process to an oscillatory state and uses this response sequentially along with the Ziegler-Nichols rules [7] to tune the controllers in the system. These methods are developed to find a stable set of PID gains for a particular system's dynamics and may not find the optimal set with respect to setpoint tracking. It should be noted that the optimal set of PID gains is unlikely to be stable. Furthermore, obtaining a stable set is important to ensure the safe operation of the process. Therefore, there is interest in investigating data-driven methods to create smart PID controller tuning methods.

Reinforcement learning has been shown to be successful in other fields, such as the board game Go [8] and quadcopter control [9]. Given the methodology's performance in these fields, there has been a recent focus on applying reinforcement learning to chemical and biological process control. Ma *et al.* [10] successfully used reinforcement learning to control a free radical polymerisation reactor. However, there are drawbacks to using reinforcement learning for process control, such as ensuring safe exploration and the large number of samples required by a reinforcement learning algorithm [11]. The sampling efficiency and scalability of reinforcement learning algorithms used for process control have been addressed by Zhu *et al.* [12], where a novel algorithm was developed that could achieve stable learning with an insufficient number of samples. A reinforcement learning algorithm developed with safety consideration has been achieved by Mowbray *et al.* [13], where a novel algorithm demonstrated safe operation on a bioprocess.

When a reinforcement learning algorithm is combined with PID controllers, a hierarchy of policies is created. The high-level policy tunes multiple PID controllers and is exposed to the states of several units. The low-level policy is the individual PID controller, which takes only one state as an input and outputs the control action. This algorithm structure is called "hierarchical reinforcement learning". The decomposition of the problem into two levels reduces the size of the policy space which should lead to faster convergence [14]. Significant research has been conducted on the use of reinforcement learning to tune PID controllers in a simulated process environment [15–17]. Dogru *et al.* [18] applied reinforcement learning to PID controller tuning and validated their method on a pilot scale tank system after an offline tuning step whilst considering safety constraints in the training of the algorithm.

Real industrial systems consist of multiple different units and recycle loops. Therefore, the objective of this paper is to develop a hierarchical reinforcement learning algorithm which is designed for plant-wide control. The proposed algorithm aims to improve the performance and robustness of plant-wide control systems. To evaluate the algorithm, three case studies are investigated and the performance of the algorithm is compared to other methods of PID tuning (derivative-free optimisation and multiloop relay tuning) and an advanced control technique (nonlinear model predictive control). Then the robustness of the algorithm is analysed by performing a parametric mismatch which will aim to simulate a plant-model mismatch.

The remainder of this paper is structured as follows: the theory behind the algorithm is provided in Section 2, a description of the proposed algorithm is given in Section 3, a discussion of the computational case studies in Section 4, and the conclusions are given in Section 5.

2 Background

2.1 PID Controllers

The PID controller used in this paper is in its discrete form, with a constant time step and can be written as Equation 1:

$$u_t = K_p\epsilon + K_i\sum_{t=0}^{t}\epsilon + K_d(\epsilon_t - \epsilon_{t-1}) \tag{1}$$

$$\text{with} \quad \epsilon = SP_t - CV_t \tag{2}$$

Where u_t is the manipulated variable, K_p is the proportional constant, K_i is the integral constant, K_d is the derivative constant, and ϵ_t represents the error between the setpoint SP_t and the controlled variable CV_t.

To assess the setpoint tracking of the PID controller, the integral square error (ISE) metric is used throughout this paper. Other measures of setpoint tracking performance include the integral absolute error, which does not penalise large errors to the same degree as ISE. The discretised form of the ISE is as follows.

$$ISE = \sum_{t=t_0}^{t_f}(x_t - SP_t)^2 \tag{3}$$

Here x_t represents a simulation state and SP_t represents the setpoint of the state.

2.2 Reinforcement Learning

Reinforcement learning formulates a problem as a sequential decision-making process in which an agent interacts with an environment to maximise its cumulative reward. Through these actions, the agent aims to learn an optimal policy which is the mapping between a state and the optimal action. This can be formalised as a Markov decision process (MDP) represented by the tuple:

$$\text{MDP:} \quad (x_t, u_t, R_t, P(\boldsymbol{x}_{t+1} \mid \boldsymbol{u}_t, \boldsymbol{x}_t))$$

Where the agent receives a state x from the environment and executes a control action u [19]. The state x and control action u exist in the state space $\mathcal{X}$ and control action space $\mathcal{U}$ respectively. The agent receives a reward $R_t \in \mathbb{R}$ when transitioning from x_t to x_{t+1}, which is the performance metric. $P(\boldsymbol{x}_{t+1} \mid \boldsymbol{u}_t, \boldsymbol{x}_t)$ represents the probability of transitioning from x_t to x_{t+1} given the action u_t. This probability will be approximated by a simulator, given the difficulty of explicitly representing this probability.

The objective of the reinforcement learning problem is to find an optimal policy π^* that maximises the cumulative reward, which can be represented as follows.

$$\pi^*(u_t \mid x_t) = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{t_f} R_t\right] \tag{4}$$

To find the optimum policy, algorithms can be divided into two groups depending on whether they utilise the policy function. If the policy function is not used, a value function or a state-action function will be used. Here, the value function (Equation 5) is the expected cumulative reward of state x_t following the policy π.

$$v_\pi(x_t) = \mathbb{E}_\pi\left[\sum_{t=0}^{t_f} R_t \;\middle|\; x_t\right] \tag{5}$$

The state-action function (or Q-function) is described by Equation 6.

$$q_\pi(x_t, u_t) = \mathbb{E}_\pi\left[\sum_{t=0}^{t_f} R_t \;\middle|\; x_t, u_t\right] \tag{6}$$

These functions describe the expected reward given a state or state-action pair with the most notable algorithm of this type being Q-learning [20]. However, if the policy is optimised directly, this comes with practical benefits over value function algorithms and theoretical convergence benefits given the continuous parameterisation of the policy [19]. When utilising the policy function, a gradient of the performance metric with respect to the policy parameterisation is required. This makes use of the policy gradient theorem which can be represented by Equation 7. This describes the proportionality between the performance metric $R(\theta)$ and the policy π that is used by policy gradient algorithms to iterate towards the optimal policy.

$$\nabla R(\theta) \propto \sum_x \mu(x_t) \sum_u q_\pi(x_t, u_t) \nabla \pi(u_t \mid x_t, \boldsymbol{\theta}) \tag{7}$$

The calculation of this policy gradient or value function can be computationally intensive since these functions will need to be approximated. However, evolutionary strategies have been shown to be more computationally efficient, since no gradient or value function is required with the trade-off of a larger data requirement [21]. Hence, this paper will make use of an evolutionary algorithm, particle swarm optimisation (PSO) [22], to optimise the policy.

2.3 Methods for Comparison

2.3.1 Multiloop Relay Tuning

The multiloop relay PID tuning method developed by Shen & Yu [23] is one of the PID tuning methods used to compare to the hierarchical reinforcement learning method. This method uses the relay tuning method to sequentially tune PID controllers in a multiloop system. This is achieved by initially setting all loops but one to manual whilst the other is tuned using the relay tuning method [24]. After the initial step, a sequence of setting all but one controller to automatic and tuning the remaining controller is executed. This sequence is continued for four iterations as suggested by Shen & Yu [23] or until the change in PID gains is sufficiently small. Note that this method of PID tuning finds a constant set of PID gains which are used throughout the episode.

2.3.2 Derivative-Free Optimisation

A second PID tuning method uses derivative-free optimisation (DFO) to directly minimise the ISE by manipulating the set of PID gains. Like the multiloop relay tuning method, the DFO method finds a constant set of PID gains that are used throughout the episode. The Powell DFO algorithm [25] is used with a direct stochastic search to find a suitable starting point for the DFO algorithm.

2.3.3 Nonlinear Model Predictive Control

A Nonlinear Model Predictive Controller (NMPC) is a state-of-the-art controller, which is compared to the hierarchical reinforcement learning algorithm.

The NMPC controller is implemented using the do-MPC [26] framework. The do-MPC framework uses CASADI [27] to create a symbolic representation of the model and perform automatic differentiation. CASADI also allows an interface with the IPOPT solver [28] which is used to solve the optimal control problem at each time step. The controller is set up with a time horizon of 5 time steps and uses the same model to predict and run the simulation as the other comparison and reinforcement learning methods. Hence, optimal control is expected from this method when a perfect model is provided.

3 Methodology

This section will describe the problem to be solved and the implemented hierarchical reinforcement learning algorithm.

3.1 Problem Statement

A Markov decision process is assumed to represent the system dynamics of the process control problem:

$$\boldsymbol{x}_{t+1} = P(\boldsymbol{x}_{t+1} \mid \boldsymbol{u}_t, \boldsymbol{x}_t) \tag{8}$$

The system dynamics can be further approximated as a discrete stochastic nonlinear system as follows:

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t, d_t) \quad \forall t \in [t_0, t_f] \tag{9}$$

Where $\boldsymbol{x}_t \in \mathbb{R}^{n_x}$ are the states of the system, $\boldsymbol{u}_t \in \mathbb{R}^{n_u}$ are the control actions, $d_t \in \mathbb{R}$ represents the disturbance to the system, and $f(\cdot)$ represents the nonlinear dynamics of the system which are dependent upon the case study and will be described in Section 4.

The hierarchical reinforcement learning method searches for an optimal policy to minimise the set-point error with a control input penalty over a specified time frame $(t_0 ... t_f)$ in a stochastic environment and under disturbances. The following Optimal Control Problem (OCP) represents this:

$$\begin{aligned} \min_{\boldsymbol{\pi_\theta}} \quad & \sum_{t=t_0}^{t_f} R(\boldsymbol{x}_t, \boldsymbol{u}_t) \\ \text{s.t.} \quad & \boldsymbol{x}_{t_0} = \boldsymbol{x(0)} \\ & \boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t, d_t) \\ & \boldsymbol{u}_t = \boldsymbol{\pi_\theta}(\boldsymbol{x}_t) \end{aligned} \tag{10}$$

The deterministic policy $(\boldsymbol{\pi})$ and its parameters $(\boldsymbol{\theta})$ in the OCP above represent all levels of the hierarchical reinforcement learning method. The reward function $R(\cdot)$ can be described by the following:

$$R(\boldsymbol{x}_t, \boldsymbol{u}_t) = |\boldsymbol{x_t} - \boldsymbol{x_{sp,t}}| + |\boldsymbol{u}_t - \boldsymbol{u}_{t-1}| + \left|\boldsymbol{u}_t - \boldsymbol{u^L}\right| \tag{11}$$

Where $\boldsymbol{x}_{sp}$ is a vector of the state setpoints and $\boldsymbol{u^L}$ is a vector of the control's lower bounds. The first term of the reward function rewards the setpoint tracking of the policy and the two other terms reward smooth control inputs. The problem can now be represented in a closed-loop form as shown in Figure 1.

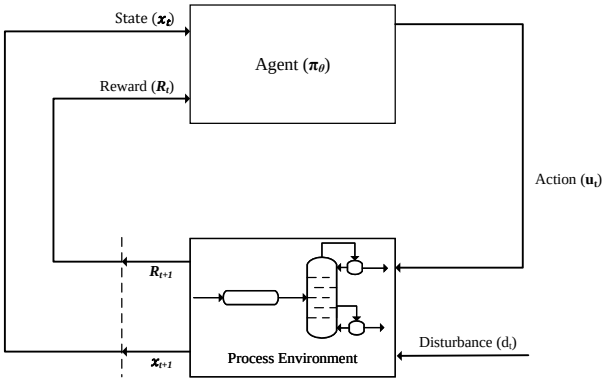

Figure 1: Closed-loop Representation of the Problem.

3.2 Hierarchical Reinforcement Learning

The agent in Figure 1 consists of two layers of policies arranged in a hierarchy. The top of the hierarchical structure is a policy formed by an Artificial Neural Network (ANN) that is parameterised by $\boldsymbol{\theta}_{ANN}$. This takes the current state $\boldsymbol{x}_t$, previous state $\boldsymbol{x}_{t-1}$ and the current setpoint $\boldsymbol{x_{sp,t}}$. Then outputs the parameters $(\boldsymbol{K_{p,t}}, \boldsymbol{K_{i,t}}, \boldsymbol{K_{d,t}})$ of the lower level of the hierarchy which is the PID policy. The PID policy takes these parameters along with the setpoint error and outputs the control action $\boldsymbol{u_t}$. The structure of the hierarchical reinforcement learning algorithm is represented in Figure 2.

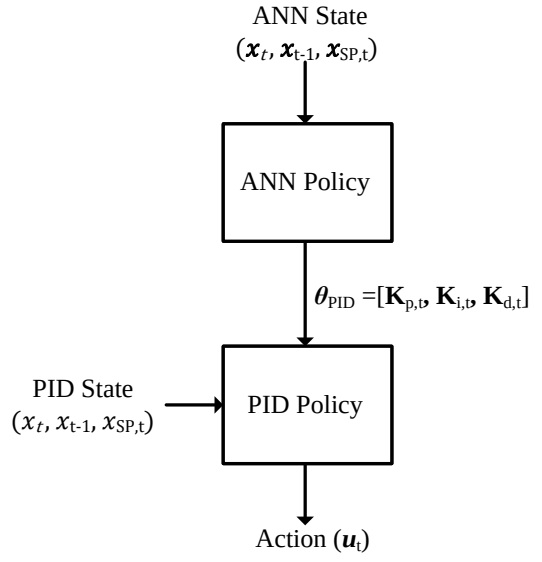

Figure 2: Hierarchical Reinforcement Learning Structure.

3.3 Policy Optimisation

To find the set of parameters $\boldsymbol{\theta}_{ANN}$ which solves the OCP (10) a combination of a direct stochastic search (Algorithm 1) and the evolutionary method PSO (Algorithm 2) [22] is used. The PSO algorithm is parameterised as follows: inertial weight ω which is set to 0.5, and the acceleration coefficients c_1 and c_2 which are set to 2.3 and 1.8, as suggested by Bratton & Kennedy [29]. The direct stochastic search finds an initial vector of parameters for the PSO algorithm. The policy optimisation algorithm is shown below in Algorithms 1 and 2.

Algorithm 1 Direct Stochastic Search Algorithm

Input:
N_R - Number of Iterations
D_θ - Parameter Dimensionality
Output:
$\boldsymbol{\theta}_{ANN}^{N_R \times D_\theta}$ - Vector of parameters
$\boldsymbol{R}^{N_R}$ - Vector of Rewards
Start:
for $i = 0$ to N_R **do**
  $\boldsymbol{\theta}_{ANN,i} \leftarrow U[5, -5]^{D_\theta}$ ▷ Uniform distribution
  $R_i \leftarrow \mathcal{P}(\theta^i_{ANN})$ ▷ $\mathcal{P}(\cdot)$ represents the OCP
end for

Algorithm 2 PSO Algorithm

Input:
D_θ - Parameter Dimensionality
T - Maximum Number of Iterations
N_S - Swarm Size
$\boldsymbol{\theta}_{ANN}^{N_R \times D_\theta}$ - Vector of parameters
$\boldsymbol{R}^{N_R}$ - Vector of Rewards
Output:
$\boldsymbol{\theta}_{ANN}^{Best}$ - Best ANN parameters
Start:
for $i = 0$ to N_s **do**
  $\boldsymbol{\theta}^0_{P,i} \leftarrow \boldsymbol{\theta}_{ANN}^{N_R \times D_\theta}[argmin(\boldsymbol{R}^{N_R})]$
end for
$\boldsymbol{\theta}_G \leftarrow \boldsymbol{\theta}_{ANN}^{N_R}[argmin(\boldsymbol{R}^{N_R})]$
$t \leftarrow 0$
while $t < T$ **do**
  for $i = 0$ to N_s **do**
    $\boldsymbol{r}_1, \boldsymbol{r}_2 \leftarrow U[0, 1]^{D_\theta}$
    $\boldsymbol{v}_i^{t+1} \leftarrow \omega v_i^t + c_1 \boldsymbol{r}_1(\boldsymbol{\theta}^t_{P,i} - \boldsymbol{\theta}^t_i) + c_2 r_2(\boldsymbol{\theta}^t_G - \boldsymbol{\theta}^t_i)$
    $\boldsymbol{\theta}_i^{t+1} \leftarrow \boldsymbol{\theta}_i^t + \boldsymbol{v}_i^{t+1}$
    if $\mathcal{P}(\theta_i^t) < \mathcal{P}(\theta^t_{P,i})$ **then**
      $\mathcal{P}(\theta^t_{P,i}) \leftarrow \mathcal{P}(\theta_i^t)$
    end if
  end for
  $\boldsymbol{\theta}_G \leftarrow \mathcal{P}(\min_{t,i}(\theta^t_{P,i})) \quad \forall t, i$
  $t \leftarrow t + 1$
end while

3.4 Training and Testing Methodology

To effectively train the reinforcement learning algorithm, exposure to a diverse set of training data is vital. This is achieved by training the algorithm on four different setpoint changes which vary in both magnitude and direction of the step. Along with the setpoint changes, there is a further training episode with a disturbance to the feed temperature and a constant setpoint. The rewards from all episodes are summed and then used to evaluate the performance of a policy. This example test episode is shown in Equation 12 where i is the training episode.

$$R_T = \sum_{i=1}^{5} R_i \tag{12}$$

An example training schedule for the controlled variable $x_{D,B}$ is presented in Table 1.

Table 1: Example Training Schedule

Training Episode	Setpoint Change [mol/mol]	Disturbance?
1	$0.9 \rightarrow 0.97$	False
2	$0.97 \rightarrow 0.90$	False
3	$0.97 \rightarrow 0.935$	False
4	$0.935 \rightarrow 0.97$	False
5	0.935	True

After the reinforcement learning algorithm has been trained, it is tested on an unseen episode with two setpoint changes and the inclusion of a disturbance at the start of the simulation. This is shown in Table 2

Table 2: Example Test Episode

Setpoint Change [mol/mol]	Disturbance?
$0.95 \rightarrow 0.98 \rightarrow 0.94$	True

4 Case Studies

Three case studies of increasing complexity are presented to test the effectiveness of the hierarchical reinforcement learning control strategy. In addition, they are compared against relay multiloop tuning, derivative-free optimisation tuning, and an NMPC controller. Furthermore, a parametric mismatch analysis is conducted to investigate control strategies under real-world scenarios.

4.1 Reactor

In the first case study, a multiloop single continuously stirred tank reactor (CSTR) system is investigated. The objective is to track the setpoint for the output concentration of A (C_A) and the reactor temperature (T) by manipulating the jacket temperature (T_J). There is also a penalty for control inputs to promote

smooth control. The process flow diagram of this case study is presented below in Figure 3 and the process parameters are in Table 3. The reactor is assumed to be isothermal and perfectly mixed, and the reaction is irreversible and exothermic. To model this case study, the following differential and auxiliary equations (adapted from Hedengren [30]) are used:

$$r_A = k_0 e^{-\frac{E}{RT}} C_A \tag{13}$$

$$\frac{dC_A}{dt} = \frac{F_{in}}{V(C_{A,in} - C_A)} - r_A + w_{C_A} \tag{14}$$

$$\begin{aligned}\frac{dT}{dt} =& \frac{F_{in}}{V_{CSTR}}(T_{in} - T) + \frac{\Delta H_{rxn}}{\rho_{A-B}} r_A \\ &+ \frac{UA}{V \rho_{A-B} C_{P,A-B}(T_J - T)} + W_T\end{aligned} \tag{15}$$

Where r_A [mol/s] represents the rate of consumption of component A, w_i is the additive process noise of state i, k_0 [1/s] is the pre-exponential factor, E [J/mol] is the activation energy, F_{in} [m^3/s] is the flowrate into the reactor, V [m^3]is the volume of the reactor, T_{in} [K] is the temperature of the inlet stream, ΔH_{rxn} [J/mol] is the heat of reaction, ρ_{A-B} [kg/m^3] is the density of the A-B mixture, U [W/m^2K] is the overall heat transfer coefficient, A [m^2] is the heat transfer area, and $C_{P,A-B}$ [J/mol] is the heat capacity of the A-B mixture.

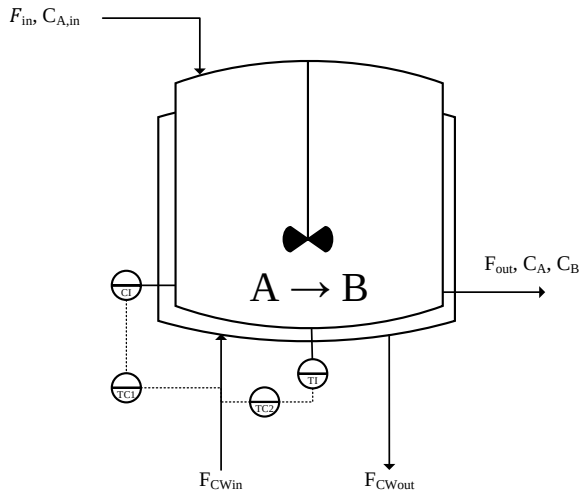

Figure 3: Process Flow Diagram of Case Study 1

4.1.1 Simulation and Comparison

The hierarchical reinforcement learning strategy is trained over 10 epochs with the direct stochastic search, then with 10 particles and 30 iterations with the PSO algorithm. The learning curve is shown in Figure 4. In each epoch, the case study was simulated over 25 minutes and divided into 120 time steps. The training schedule used 4 episodes with different

Table 3: Process Parameters for Case Study 1

Process Parameter	Value	Units
$C_{A,in}$	1	mol/m^3
T_{in}	350	K
F_{in}	100	m^3/s
V	100	m^3
ρ_{A-B}	1000	kg/m^3
$C_{P,(A-B)}$	0.239	J/kgK
ΔH_{rxn}	- 5 x 10^4	J/mol
E	7.2 x 10^4	J/mol
UA	5 x 10^4	W/K
k_0	7.2 x 10^{10}	s^{-1}

Table 4: Case Study 1 Training Schedule

Training Episode	Setpoint Change	Disturbance?
1	$C_A: 0.8 \rightarrow 0.9$ $T: 330 \rightarrow 320$	False
2	$C_A: 0.7 \rightarrow 0.9$ $T: 340 \rightarrow 320$	False
3	$C_A: 0.9 \rightarrow 0.8$ $T: 320 \rightarrow 330$	False
4	$C_A: 0.9 \rightarrow 0.7$ $T: 320 \rightarrow 340$	False
5	$C_A: 0.8$ $T: 330$	True

setpoint changes and one episode with a disturbance to the feed temperature (T_{in}) as shown in Table 4. The disturbance was defined by an increase to 360 K between 10 and 30 time steps. The ANN was implemented in Pytorch version 1.13.0.

The ANN control policy is designed to have an input layer, two fully connected hidden layers and an output layer. The first fully connected layer consists of 20 neurons and the second has 5 neurons. A sigmoid activation function is used to connect all layers, except for the output layer, which uses a ReLU6 activation function.

The trained hierarchical reinforcement learning algorithm, DFO, relay multiloop tuning, and NMPC is simulated in the Case Study 1 process environment. The setpoint changes used are described in Table 5 and a disturbance of the feed temperature (T_{in}) is introduced between 10 and 30 time steps. The simulation is repeated 10 times to investigate the variance in the control strategies. The resulting PID parameters of the relay multiloop tuning and DFO methods are constant with time and are shown in Table 6. The resulting simulation, control and PID parameter trajectories are shown in Figure 5 with the ISE shown in Table 7.

Table 5: Case Study 1 Test Schedule

Controlled Variable	Setpoint Change
C_A [mol/m^3]	$0.85 \rightarrow 0.9 \rightarrow 0.75$
T [K]	$330 \rightarrow 320 \rightarrow 335$

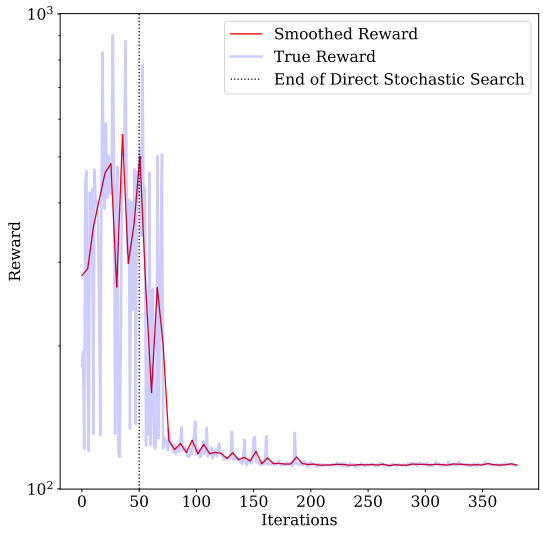

Figure 4: Case Study 1 Learning Curve. The smoothed learning curve is shown with the red solid line, the true learning curve by the blue solid line and the end of the direct stochastic search by a black dotted line.

Table 6: PID Gains for DFO and Relay Tuning Methods

	$C_A - loop$			$T - loop$		
	K_P	K_I	K_d	K_P	K_I	K_d
DFO	2.89	1.91	3.01	3.09	0.74	3.08
Relay	23.56	1.33	43.33	0.32	0.02	0.44

For both the trajectory of the reactor outlet concentration of A and reactor temperature, the hierarchical reinforcement learning algorithm, DFO and NMPC all follow similar trajectories. The relay multiloop tuning method shows a significant deviation at around 4 minutes in both trajectories which is reflected in the relay method exhibiting the largest ISE. This is expected since the multiloop relay tuning is designed for systems where each control loop controls a separate manipulated variable. The NMPC has the smallest ISE for both trajectories, which can be attributed to its close setpoint tracking, especially for the temperature setpoint.

For the hierarchical reinforcement learning method, the parameters of the C_A PID are significantly manipulated to maintain tracking of the setpoints. The C_A integral PID parameter increases initially to decrease the setpoint offset compared to the other PID tuning method which is apparent from 7-10 minutes. Then this parameter

Table 7: ISE for Case Study 1

Control Method	ISE	
	C_A [mol/m^3]	T [K]
HRL	0.151	530
DFO	0.154	583
NMPC	0.107	248
Relay	0.378	1643

Table 8: Range of Parameters (Case Study 1)

Parameter	Range	Units
k_0	1-7.2	(x10^{10}) [1/s]
UA	3-5	(x10^4) [W/K]

decreases to zero with the setpoint increase, since there is no significant setpoint offset. When the C_A setpoint decreases, the C_A-loop derivative and the proportional PID parameters decrease to ensure that there is no overshoot of the next setpoint. At the final setpoint, the C_A integral PID parameter is increased again to reduce the setpoint offset. While the T-loop PID parameters are static, except for a spike in the derivative when the temperature setpoint increases which prevents an overshoot of the temperature setpoint. This adaptive PID tuning results in the reinforcement learning method having the smallest ISE out of the PID tuning methods tested.

4.1.2 Parameteric Mismatch

The catalyst activity k_0 and reactor jacket heat transfer coefficient UA were varied in the range shown in Table 8. Then the resulting ISE is shown in Figures 6 and 7.

As the catalyst degrades (Figure 6), the ISE of all control methods increases, indicating a worsening ability to track the setpoint due to the parametric mismatch. For both the C_A and T ISE, the NMPC transitions from having the lowest ISE to the largest at around a k_0 of 6x10^{10} s^{-1}. Then as the cooling jacket fouls (Figure 7), all control methods follow similar trajectories with a sharp increase in both C_A and T ISE at around a UA of 42 kW/K.

Across both parameters, the relay tuning method consistently has the largest ISE out of the PID tuning methods indicating that this method has chosen the least robust set of PID parameters. Both the DFO and hierarchical reinforcement learning methods perform similarly and often exhibit a lower ISE than the NMPC, indicating they are the most robust in this parametric mismatch analysis. This is due to the PID controller's structure being less reliant on the model's structure compared to the NMPC.

4.2 Reactor & Separator

In the second case study, a distillation column is added. The objective function is then changed to track the distillate concentration of B ($x_{D,B}$) by manipulating the reflux ratio (R_R) and the reactor jacket temperature (T_j). There is also a penalty for control input to promote smooth control. The process flow diagram of this case study is presented

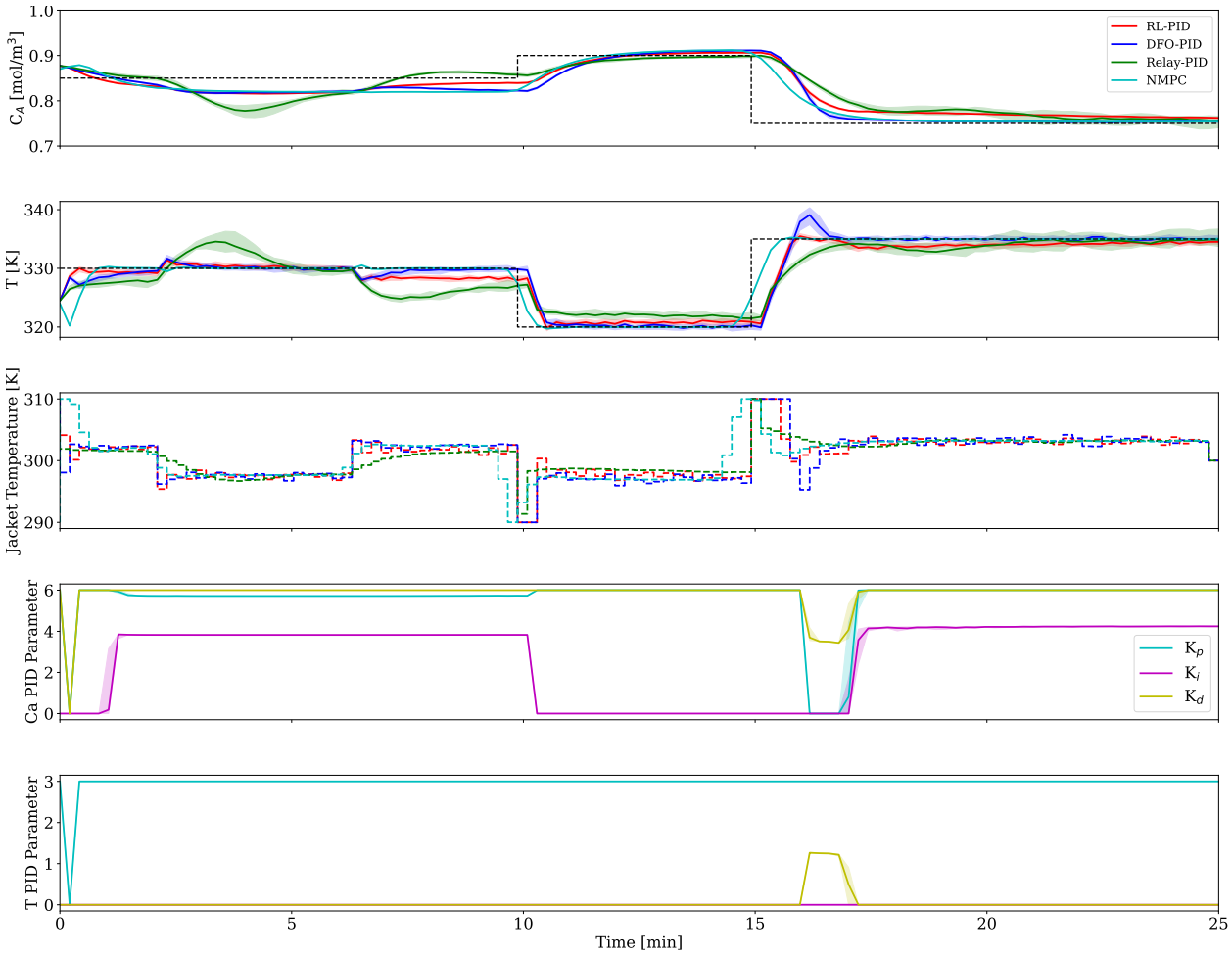

Figure 5: Case Study 1 Simulation. The red solid line represents the hierarchical reinforcement learning algorithm, the blue solid line is the DFO, the green solid line is the relay multiloop tuning method, and the solid cyan line is the NMPC. The lower two plots are the PID parameters determined by the reinforcement learning algorithm with proportional in cyan, integral in magenta, and derivative in yellow. The variance of 10 repetitions of the simulation is shown by a band of the same colour as the trajectory.

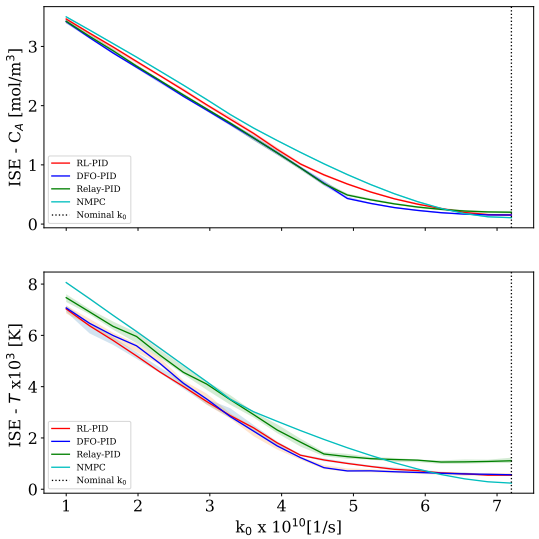

Figure 6: k_0 Parametric Mismatch for Case Study 1. C_A-ISE is shown in the upper plot and T-ISE in the lower. The black dotted line represents the nominal k_0, the red solid line represents the hierarchical reinforcement learning algorithm, the blue solid line is the DFO, the green solid line is the relay multiloop tuning method, and the solid cyan line is the NMPC. The variance of 10 repetitions of the simulation is shown by a band of the same colour as the trajectory.

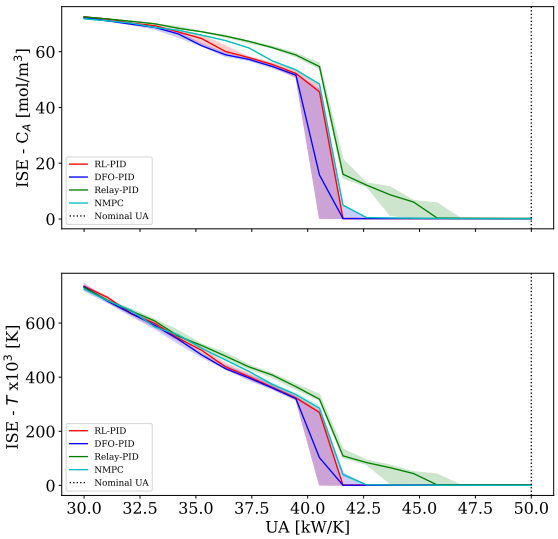

Figure 7: UA Parametric Mismatch for Case Study 1. C_A-ISE is shown in the upper plot and T-ISE in the lower. The black dotted line represents the nominal k_0, the red solid line represents the hierarchical reinforcement learning algorithm, the blue solid line is the DFO, the green solid line is the relay multiloop tuning method, and the solid cyan line is the NMPC. The variance of 10 repetitions of the simulation is shown by a band of the same colour as the trajectory.

below in Figure 8 and the process parameters in addition to those used in Case Study 1 (Table 3) are presented in Table 9.

The distillation column is assumed to be at a constant temperature. To model the addition of this distillation column (which is adapted from Hedengren [31]), the reactor outlet concentration was first converted to a mole fraction then for every stage the vapour mole fraction is given by the following algebraic equation:

$$y_{B,i} = x_{B,i}\frac{\alpha_{A,B}}{1+(\alpha_{A,B}-1)x_{B,i}} \tag{16}$$

Then the following differential equation is used to describe the liquid mole fraction in the condenser.

$$\frac{dx_{D,B}}{dt} = \frac{V(y_{B,1}-x_{D,B})}{M_{cond}} + w_{x_{D,B}} \tag{17}$$

There are 15 differential equations used to represent the liquid mole fraction of each stage in the rectifying section of the distillation column.

$$\frac{dx_{i,B}}{dt} = \frac{L_R(x_{i-1,B}-x_{i,B}) - V(y_{B,i}-y_{B,i+1})}{M_{tray}} \tag{18}$$

For the feed stage, the following differential equation is used.

$$\begin{aligned}\frac{dx_{16,B}}{dt} &= \frac{q_{df}x_{B,f}+L_Rx_{15,B})}{M_{tray}} \\ &- \frac{L_Sx_{16,B}+V(y_{16,B}-y_{17,B})}{M_{tray}}\end{aligned} \tag{19}$$

For the stripping section, there are 15 stages described with the following differential equation:

$$\frac{dx_{i,B}}{dt} = \frac{L_S(x_{i-1}-x_i-V(y_i-y_{i+1})}{M_{tray}} \tag{20}$$

Finally, for the reboiler the following differential equation is used.

$$\frac{dx_{B,B}}{dt} = \frac{L_Sx_{30}-(q_{df}-D)x_{31}-Vy_{31}}{M_{reb}} \tag{21}$$

To complete the model, the following set of algebraic equations is required which describe the distillate, bottoms, liquid, and vapour flowrates:

$$D = 0.5q_{rf} \tag{22}$$

$$B = q_{df} - D \tag{23}$$

$$L_R = R_RD \tag{24}$$

$$L_S = q_{df} + L_R \tag{25}$$

Where $y_{j,i}$ is the vapour molar fraction of component j in stage i, $x_{j,i}$ is the liquid molar fraction of component j in stage i, $W_{x_{D,B}}$ is the additive process noise for the distillate composition, $\alpha_{A,B}$ is the relative volatility of components A and B, M_i [mol] is the total molar holdup in stage i, V [mols^{-1}] is the vapour flowrate in the column, L_R and L_S [mols^{-1}] are the liquid flowrate in the rectifying and stripping sections, respectively, D [mols^{-1}] is the distillate flowrate, B [mols^{-1}] is the bottoms flowrate, and q_{df} [mols^{-1}] is the distillation feed flowrate.

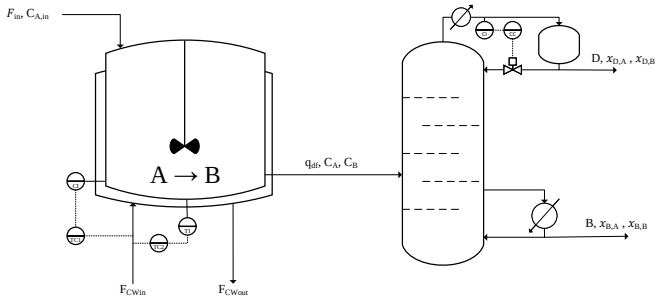

Figure 8: Process flow diagram of Case Study 2

Table 9: Process parameters for Case Study 2

Process Parameter	Value	Units
$\rho_{A-B,M}$	0.2	mol/m^3
$\alpha_{A,B}$	1.6	-
M_{cond}	1	mol
M_{tray}	3	mol
M_{reb}	0.5	mol
UA	7 x 10^5	kW/K
N_{tray}	32	Trays

4.2.1 Simulation and Comparison

The hierarchical reinforcement learning policy is trained over 10 epochs with the direct stochastic search and then with 30 particles and 20 iterations with the PSO algorithm. The simulation follows the same length and time steps as Case Study 1. The training schedule used is given in Table 4.

The ANN control policy is designed to have an input layer, two fully connected layers, and an output layer. The number of hidden neurons in each fully connected layer changes whether the ANN is outputting PID parameters for the reactor or the distillation column. For the reactor, there are 20 neurons in the first and 5 neuron in the second. For the distillation column, there are 20 in both fully connected layers. In between each of the layers, a Sigmoid activation function is used and for the output layer, the ReLU6 activation function is used. The resulting PID parameters of the relay multiloop tuning and DFO methods are constant over time and are shown in Table 10. The resulting simulation, control trajectories and PID parameters are shown in Figure 10 and corresponding ISEs are shown in Table 11.

During the first three minutes of the simulation, all three PID gains for the hierarchical reinforcement

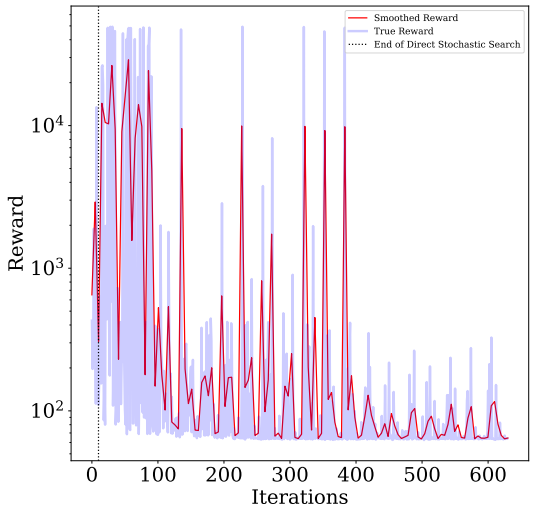

Figure 9: Case Study 2 Learning Curve. The smoothed learning curve is shown with the red solid line, the true learning curve by the blue solid line and the end of the direct stochastic search by the black dotted line.

Table 10: Case Study 2 Static PID Gains

Control Method	C_A-loop			T-loop			$x_{B,D}$-loop		
	K_p	K_i	K_d	K_p	K_i	K_d	K_p	K_i	K_d
Relay	18.39	5.11	6.90	0.27	0.07	0.10	159.38	35.33	74.88
DFO	22.52	11.80	30.15	0.14	0.13	0.3	76.39	133.40	74.77

learning algorithm are manipulated to dampen the oscillation when reaching the first setpoint from the initial state. This adaptive PID tuning allows the hierarchical reinforcement learning algorithm to track the setpoint closer in the initial stages of the simulation than the DFO and multiloop relay tuning methods. As the setpoint is increased, the derivative gain and integral gain are increased and the proportional gain is decreased. This is to reduce the overshoot when the setpoint is changed. As the setpoint is decreased all three PID gains are decreased to reduce overshoot. Then once the final setpoint is reached the derivative and proportional PID gains are increased to reduce static offset from the setpoint.

The adaptive PID tuning used by the hierarchical reinforcement learning algorithm results in it having a lower ISE than the other two PID tuning methods. However, the NMPC has a significantly lower ISE than all PID tuning methods. The NMPC control strategy is not influenced by the setpoint for C_A and T, therefore it follows a trajectory that decreases C_A by increasing the reactor jacket temperature when the setpoint of $x_{B,D}$ increases and vice versa instead of solely manipulating the reflux ratio.

4.2.2 Parametric Mismatch

The catalyst activity (k_0) and the reactor jacket heat transfer coefficient (UA) were varied in the range shown in Table 12. The resulting ISE in the range of

Table 11: Case Study 2 ISE

Control Method	ISE
	$x_{B,D}$ [mol/mol]
HRL	0.008
DFO	0.009
NMPC	0.002
Relay	0.009

Table 12: Range of parameters (Case Study 2)

Parameter	Range	Units
k_0	0.01-7.2	(x10^{10}) [1/s]
UA	0.1-7	(x10^4) [W/K]

parameters is shown in Figures 11 and 12.

As the catalyst degrades (Figure 11), all three PID tuning methods follow a similar stable trend until the catalyst degrades significantly where the ISE increases rapidly. The NMPC's ISE increases as the catalyst degrades, reaching a higher ISE than all PID tuning methods at a k_0 of 1.7 x 10^{10} s^{-1}. This is due to the NMPC's reliance on the reactor jacket temperature to control the distillate composition, which the PID tuning methods do not utilise. As the reactor jacket fouls (Figure 12), all four control methods follow a similar stable trend until the jacket fouls significantly.

4.3 Reactor, Separator & Recycle

In the third case study, a recycle is added to the process used in Case Study 2. The objective function and manipulated variables remain the same as in Case Study 2. The process flow diagram of this case study is presented below in Figure 13 and the process parameters in addition to those used in Case Study 2 (Table 9) are presented in Table 13.

The model of Case Study 3 includes Equations 13-15 and Equations 16-21 which describe the dynamics of the reactor and the distillation column, respectively. Then, in addition, a system of algebraic and differential equations which models the recycle stream is required and is shown below.

$$\frac{dF_R}{dt} = (D - P_R D)\rho_{A-B,M} \tag{26}$$

$$\frac{dC_{B,R}}{dt} = \frac{dx_{B,B}}{dt} B \tag{27}$$

$$\frac{dC_{A,R}}{dt} = 1 - \frac{dC_{B,R}}{dt} \tag{28}$$

$$F_{rf} = F_R + F_{in} \tag{29}$$

$$C_{A,in} = \frac{F_R C_{A,R} + F_{in} C_{A,pf}}{F_{in}} \tag{30}$$

Where F_k [m^3/s] is the flowrate of the stream k,

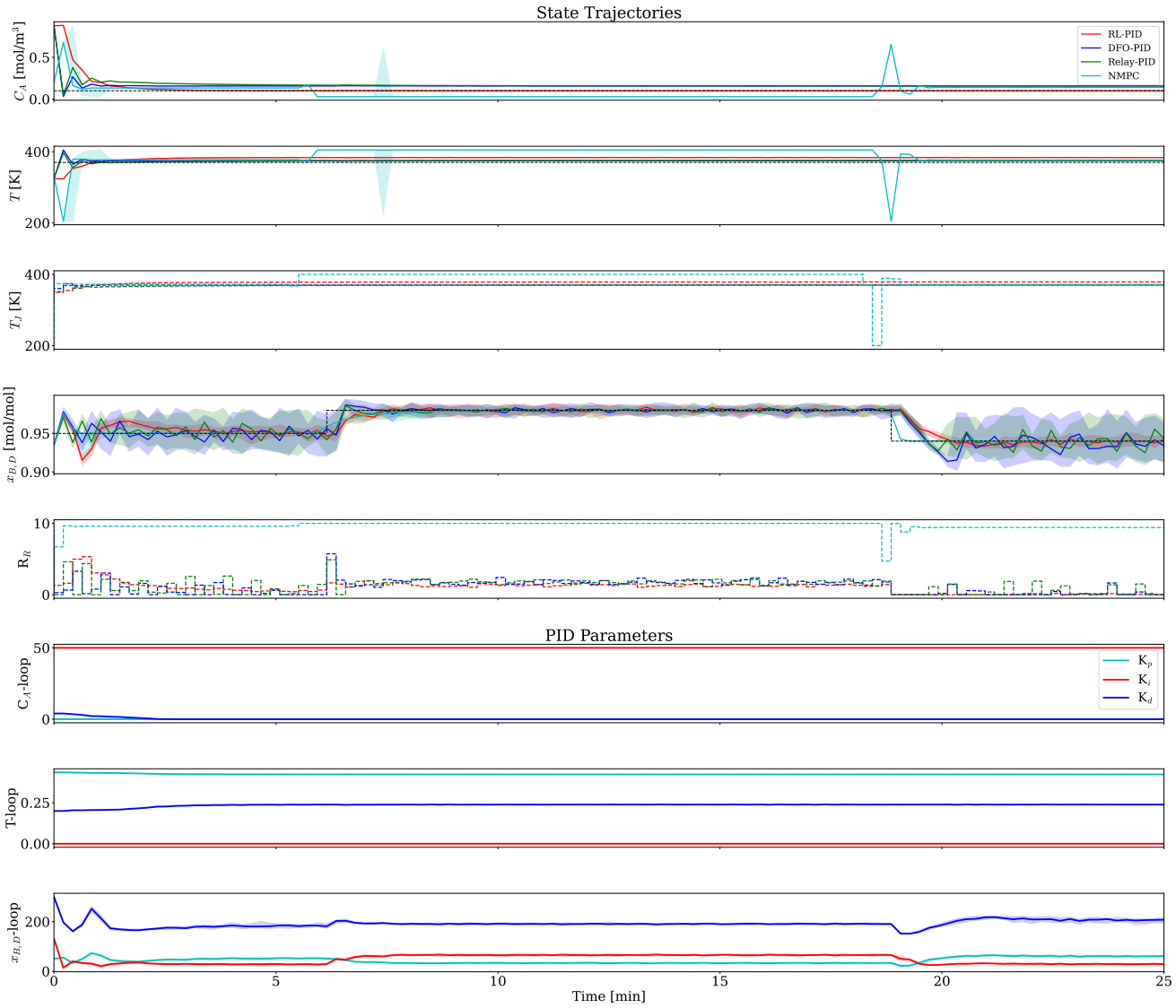

Figure 10: Case Study 2 Simulation. The red solid line represents the hierarchical reinforcement learning algorithm, the blue solid line is the DFO, the green solid line is the relay multiloop tuning method, and the solid cyan line is the NMPC. The two lower plots are the PID parameters determined by the reinforcement learning algorithm with proportional in cyan, integral in magenta, and derivative in yellow. The variance of 10 repetitions of the simulation is shown by a band of the same colour as the trajectory.

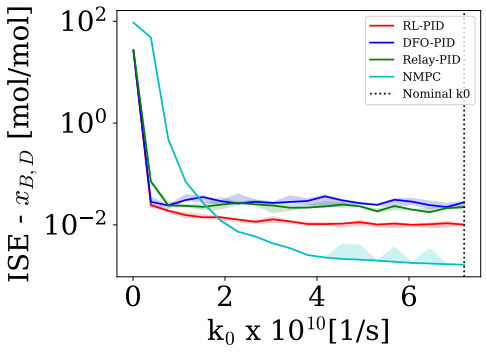

Figure 11: k$_0$ Parametric Mismatch for Case Study 2. C_A-ISE is shown in the upper plot and T-ISE in the lower. The black dotted line represents the nominal k$_0$, the red solid line represents the hierarchical reinforcement learning algorithm, the blue solid line is the DFO, the green solid line is the relay multiloop tuning method, and the solid cyan line is the NMPC. The variance of 10 repetitions of the simulation is shown by a band of the same colour as the trajectory

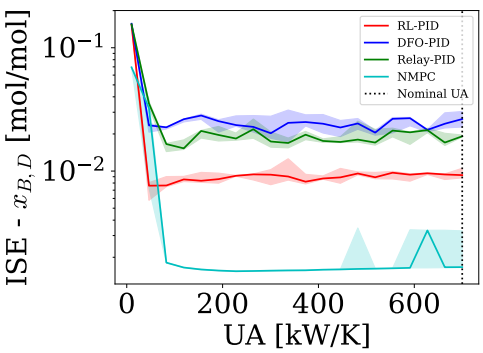

Figure 12: UA Parametric Mismatch for Case Study 2. C_A-ISE is shown in the upper plot and T-ISE in the lower. The black dotted line represents the nominal k$_0$, the red solid line represents the hierarchical reinforcement learning algorithm, the blue solid line is the DFO, the green solid line is the relay multiloop tuning method, and the solid cyan line is the NMPC. The variance of 10 repetitions of the simulation is shown by a band of the same colour as the trajectory.

P_R is the purge ratio, and $C_{j,k}$ [mol/m^3] is the concentration of component j in the stream k.

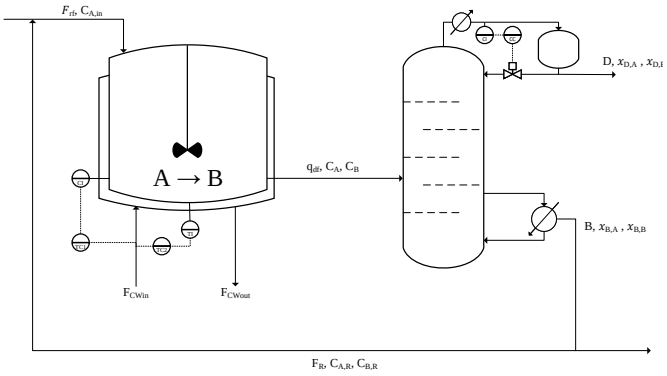

Figure 13: Process flow diagram of Case Study 3

Table 13: Process parameters for Case Study 3

Process Parameter	Value	Units
$C_{A,pf}$	1	mol/m^3
P_R	0.5	-

4.3.1 Simulation and Comparison

The hierarchical reinforcement learning policy is trained over 10 epochs with the direct stochastic search and then with 30 particles and 20 iterations with the PSO algorithm. The simulation follows the same length and time steps as Case Studies 1 and 2. The training schedule used is given in Table 4 and the learning curve is shown in Figure 14.

The ANN control policy is designed to have an input layer, two fully connected layers, and an output layer. For the reactor PID parameters, there are 20 neurons in the first and 5 neurons in the second. For the distillation column, there are 20 neurons in both the first and second. Between each of the layers, a Sigmoid activation function is used, and for the output layer, the ReLU6 activation function is used. The resulting simulation, control, and PID parameter trajectories are shown in Figure 15 and corresponding ISEs are shown in Table 11. The resulting PID parameters from the relay multiloop tuning and DFO methods are constant with time and are shown in Table 10.

The multiloop relay and hierarchical reinforcement learning tuning methods follow a similar trajectory. The DFO method has significant oscillations when reaching the setpoint from the initial state, this is due to the high integral PID gain compared to the other

Table 14: Case Study 3 Static PID Gains

Control Method	C_A-loop			T-loop			$x_{B,D}$-loop		
	K_p	K_i	K_d	K_p	K_i	K_d	K_p	K_i	K_d
Relay	17.16	5.13	5.97	0.23	0.06	0.08	261.50	32.95	222.90
DFO	5.73	5.75	9.27	0.32	0.38	0.21	175.10	556.80	458.40

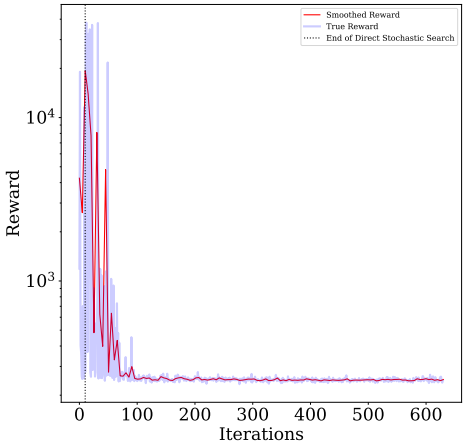

Figure 14: Case Study 3 Learning Curve. The smoothed learning curve is shown with the red solid line, the true learning curve by the blue solid line and the end of the direct stochastic search in a black dotted line.

Table 15: Case Study 3 ISE

Control Method	ISE
	$x_{B,D}$ [mol/mol]
HRL	0.033
DFO	0.044
NMPC	0.017
Relay	0.035

PID tuning methods. The hierarchical reinforcement learning algorithm reduces the proportional PID gain when the setpoint is increased to 0.98 mol/mol and decreased to 0.94 mol/mol. This reduces overshoot compared to the relay tuning method, resulting in a marginally lower ISE. The NMPC manipulates the jacket temperature as well as the reflux ratio to change the distillate composition.

4.3.2 Parametric Mismatch

The catalyst activity k_0 and the heat transfer coefficient of the reactor jacket UA are varied over the ranges shown in Table 16. The resulting ISE over this range of parameters is shown in Figures 16 and 17.

As the catalyst degrades (Figure 16), the PID-based control methods have a stable ISE across the range of parameters until the catalyst degrades significantly. However, the NMPC's ISE increases as the catalyst degrades indicating a worsened ability to track the setpoint due to the plant-model mismatch and the NMPC's reliance on the jacket temperature to control the distillate composition. As the reactor jacket fouls (Figure 17), all control methods except the DFO tuning method remain stable until significant fouling has occurred. The DFO tuning method exhibits a large variance across the parameters indicating that the setpoint tracking performance of the PID gains it has chosen is ineffective at noise rejection which is most likely due to the large integral gain.

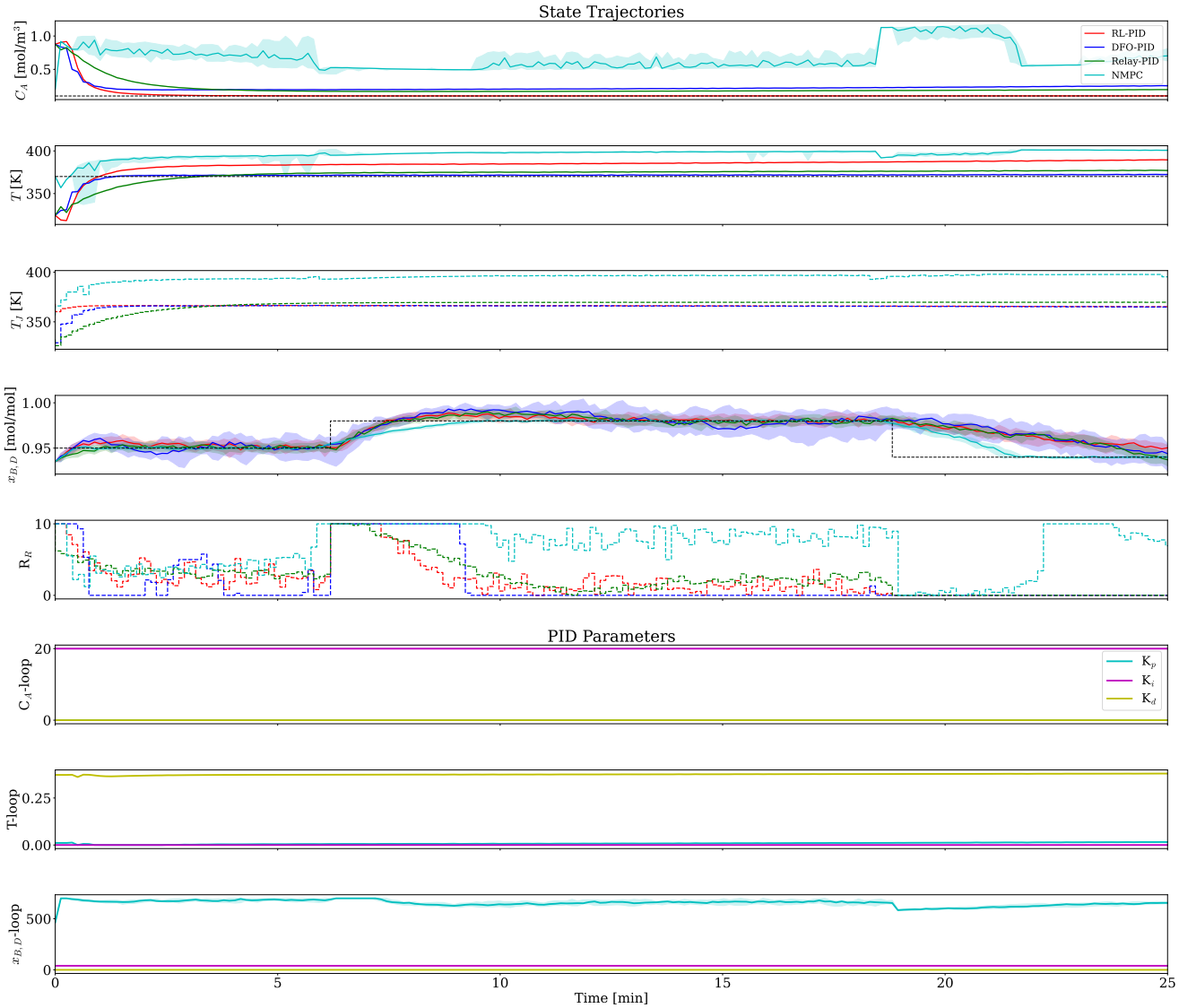

Figure 15: Case Study 3 Simulation. The red solid line represents the hierarchical reinforcement learning algorithm, the blue solid line is the DFO, the green solid line is the relay multiloop tuning method, and the solid cyan line is the NMPC. The two lower plots are the PID parameters determined by the reinforcement learning algorithm with proportional in cyan, integral in magenta, and derivative in yellow. The variance of 10 repetitions of the simulation is shown by a band of the same colour as the trajectory.

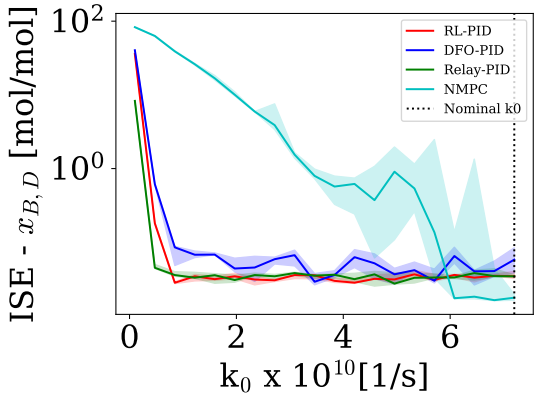

Figure 16: k$_0$ Parametric Mismatch for Case Study 3. C_A-ISE is shown in the upper plot and T-ISE in the lower. The black dotted line represents the nominal k$_0$, the red solid line represents the hierarchical reinforcement learning algorithm, the blue solid line is the DFO, the green solid line is the relay multiloop tuning method, and the solid cyan line is the NMPC. The variance of 10 repetitions of the simulation is shown by a band of the same colour as the trajectory

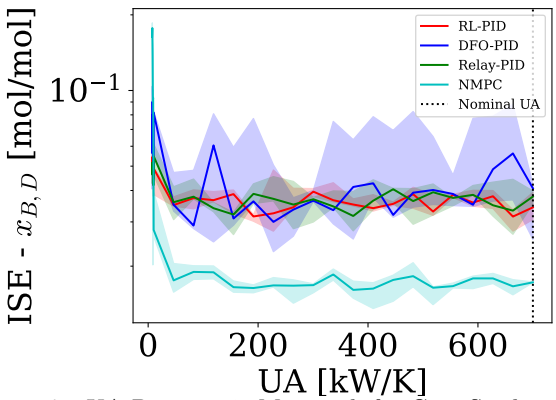

Figure 17: UA Parametric Mismatch for Case Study 3. C_A-ISE is shown in the upper plot and T-ISE in the lower. The black dotted line represents the nominal k$_0$, the red solid line represents the hierarchical reinforcement learning algorithm, the blue solid line is the DFO, the green solid line is the relay multiloop tuning method, and the solid cyan line is the NMPC. The variance of 10 repetitions of the simulation is shown by a band of the same colour as the trajectory.

Table 16: Range of parameters

Parameter	Range	Units
k_0	0.1-7.2	(x10^{10}) [1/s]
UA	0.1-7	(x10^5) [W/K]

4.4 Critical Analysis of Case Studies

In both Case Studies 2 and 3, the NMPC showed that changing the reactor jacket temperature to control the distillate composition significantly improved the ISE of this controller. The PID controllers with setpoints set by the operator are not able to do the same hence the significantly worse setpoint tracking performance over both case studies. However, the reliance on using the reactor cooling jacket to control the distillate composition led to a lessened ability to track the setpoint when the catalyst degraded as shown by the parametric mismatch analysis (Figure 11 and 16).

The hierarchical reinforcement learning algorithm adaptively controls the PID gains to reduce both static offset and overshoot of setpoint changes. As the case studies increase in complexity, the magnitude to which the PID gains are manipulated decreases to almost constant by Case Study 3. This is due to ANN not being able to fully capture the dynamics and interactions of the system due to the structure of the ANN. Since the learning curve shows a stable policy has been achieved in the case of the reactor-separator-recycle case study, additional training is unlikely to improve the policy significantly.

5 Conclusions and Future Work

This paper presents a computationally efficient hierarchical reinforcement learning algorithm that can tune PID controllers in a plant-wide control problem. This is achieved by utilising the PSO algorithm to optimise a hierarchical algorithm consisting of a high-level ANN and low-level PID controllers. Setpoint tracking, noise and disturbance rejection performance of the reinforcement learning algorithm were compared to an NMPC, DFO, and multiloop relay tuning across three case studies.

The reinforcement learning algorithm showed a lower ISE than the two PID tuning methods in all three case studies. However, the NMPC is able to manipulate the controlled variables of multiple units to reach the desired output setpoint which the PID-based controllers were not able to achieve, leading to a significantly lower ISE across all three case studies.

The robustness of each controller is investigated with a parametric mismatch analysis which indicated that the PID-tuning methods would result in a lower ISE compared to the NMPC when the catalyst degraded due to the NMPC's reliance on an accurate plant model.

In future works, the setpoint of the PID controller could be determined by the high-level ANN policy as well, or the PID policy may be replaced by another reinforcement learning agent, creating a multi-agent system to allow for the similar manipulation of controlled variables as the NMPC. Although ensuring the robustness of this method may prove difficult, as demonstrated by the NMPC's performance in the parametric mismatch.

Further testing on either the vinyl-acetate-monomer (VAM) or Tennesee Eastman process could provide further insight into the reinforcement learning algorithm's performance on more complex processes. To handle these more complex processes, different neural network structures, such as the transformer architectures, could be investigated.

6 Acknowledgements

I would like to acknowledge the Ph.D. student, Mr. Akhil Ahmed, for his support and guidance with the evaluation and analysis methods used in this paper.

7 Supplementary Information

The code used for the process environment and controllers as well as the model weights and raw data can be found here.

References

1. Rangaiah, G. P. & Kariwala, V. Plantwide control: Recent developments and applications (2012).
2. Qin, S. J. & Badgwell, T. A. *An overview of industrial model predictive control technology* in *AIche symposium series* **93** (1997), 232–256.
3. Luyben, W. L. Simple method for tuning SISO controllers in multivariable systems. *Industrial & Engineering Chemistry Process Design and Development* **25,** 654–660. https://api.semanticscholar.org/CorpusID:97803930 (1986).
4. Hovd, M. & Skogestad, S. Sequential design of decentralized controllers. *Automatica* **30,** 1601–1607. ISSN: 0005-1098. https://www.sciencedirect.com/science/article/pii/000510989490099X (1994).

5. Grosdidier, P. & Morari, M. A computer aided methodology for the design of decentralized controllers. *Computers Chemical Engineering* **11,** 423–433. ISSN: 0098-1354. https://www.sciencedirect.com/science/article/pii/0098135487850238 (1987).
6. Shen, S.-H. & Yu, C.-C. Use of relay-feedback test for automatic tuning of multivariable systems. *Aiche Journal* **40,** 627–646. https://api.semanticscholar.org/CorpusID:110568918 (1994).
7. Ziegler, J. G. & Nichols, N. B. Optimum Settings for Automatic Controllers. *Journal of Fluids Engineering.* https://api.semanticscholar.org/CorpusID:41336178 (1942).
8. Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529,** 484–489. ISSN: 14764687 (7587 Jan. 2016).
9. Lambert, N. O. *et al. Low-Level Control of a Quadrotor With Deep Model-Based Reinforcement Learning* 2019.
10. Ma, Y., Zhu, W., Benton, M. G. & Romagnoli, J. Continuous control of a polymerization system with deep reinforcement learning. *Journal of Process Control* **75,** 40–47. ISSN: 0959-1524. https://www.sciencedirect.com/science/article/pii/S0959152418304876 (2019).
11. Nian, R., Liu, J. & Huang, B. A review On reinforcement learning: Introduction and applications in industrial process control. *Computers Chemical Engineering* **139,** 106886. ISSN: 0098-1354. https://www.sciencedirect.com/science/article/pii/S0098135420300557 (2020).
12. Zhu, L., Cui, Y., Takami, G., Kanokogi, H. & Matsubara, T. Scalable reinforcement learning for plant-wide control of vinyl acetate monomer process. *Control Engineering Practice* **97,** 104331. ISSN: 0967-0661 (Apr. 2020).
13. Mowbray, M., Petsagkourakis, P., del Rio-Chanona, E. & Zhang, D. Safe chance constrained reinforcement learning for batch process control. *Computers Chemical Engineering* **157,** 107630. ISSN: 0098-1354. https://www.sciencedirect.com/science/article/pii/S0098135421004087 (2022).
14. Hengst, B. in *Encyclopedia of Machine Learning* (eds Sammut, C. & Webb, G. I.) 495–502 (Springer US, Boston, MA, 2010). ISBN: 978-0-387-30164-8. https://doi.org/10.1007/978-0-387-30164-8_363.
15. Kofinas, P. & Dounis, A. I. Online Tuning of a PID Controller with a Fuzzy Reinforcement Learning MAS for Flow Rate Control of a Desalination Unit. *Electronics* **8.** ISSN: 2079-9292. https://www.mdpi.com/2079-9292/8/2/231 (2019).
16. Sun, Q., Du, C., Duan, Y., Ren, H. & Li, H. Design and application of adaptive PID controller based on asynchronous advantage actor–critic learning method. *Wireless Networks* **27,** 3537–3547. ISSN: 15728196 (5 July 2021).
17. Jesawada, H., Yerudkar, A., Del Vecchio, C. & Singh, N. *A Model-Based Reinforcement Learning Approach for Robust PID Tuning* in *2022 IEEE 61st Conference on Decision and Control (CDC)* (2022), 1466–1471.
18. Dogru, O. *et al.* Reinforcement learning approach to autonomous PID tuning. *Computers Chemical Engineering* **161,** 107760. ISSN: 0098-1354. https://www.sciencedirect.com/science/article/pii/S0098135422001016 (2022).
19. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (MIT press, 2018).
20. Watkins, C. & Dayan, P. Q-learning. *Machine Learning* **8,** 279–292. https://api.semanticscholar.org/CorpusID:208910339 (1992).
21. Salimans, T., Ho, J., Chen, X. & Sutskever, I. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *ArXiv* **abs/1703.03864.** https://api.semanticscholar.org/CorpusID:11410889 (2017).
22. Eberhart, R. & Kennedy, J. *A new optimizer using particle swarm theory* in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science* (1995), 39–43.
23. Shen, S.-H. & Yu, C.-C. Use of relay-feedback test for automatic tuning of multivariable systems. *AIChE Journal* **40,** 627–646. eprint: https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.690400408. https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690400408 (1994).
24. Åström, K. & Hägglund, T. Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica* **20,** 645–651. ISSN: 0005-1098. https://www.sciencedirect.com/science/article/pii/0005109884900141 (1984).

25. Powell, M. J. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal* **7,** 155–162 (1964).

26. Lucia Tatulea-Codrean, S. & Engell. Rapid development of modular and sustainable nonlinear model predictive control solutions. *Control Engineering Practice* **60,** 51–62 (2017).

27. Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B. & Diehl, M. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation* (2018).

28. Wächter, A. & Biegler, L. T. On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming, *Mathematical Programming* **106(1),** 25–57 (2006).

29. Bratton, D. & Kennedy, J. *Defining a Standard for Particle Swarm Optimization* in *2007 IEEE Swarm Intelligence Symposium* (2007), 120–127.

30. Hedengren. *CSTR Modeling and Control Case Study* Accessed 01/04/2023. `http://www.apmonitor.com/che436/index.php/Main/CaseStudyCSTR`. 2014.

31. Hedengren. *Distillate Composition Control* Accessed 14/04/2023. `https://apmonitor.com/pdc/index.php/Main/DistillationControl`. 2020.