Received: 4 October 2020 | Revised: 23 April 2021 | Accepted: 3 May 2021

DOI: 10.1002/aic.17306

PROCESS SYSTEMS ENGINEERING

AIChE JOURNAL

Using process data to generate an optimal control policy via apprenticeship and reinforcement learning

Max Mowbray[1] | Robin Smith[1] | Ehecatl A. Del Rio-Chanona[2] | Dongda Zhang[1]

[1]Department of Chemical Engineering and Analytical Science, The University of Manchester, Manchester, UK

[2]Department of Chemical Engineering, Imperial College London, London, UK

Correspondence
Ehecatl A. Del Rio-Chanona, Department of Chemical Engineering, Imperial College London, South Kensington, London SW7 2AZ, UK.
Email: a.del-rio-chanona@imperial.ac.uk

Dongda Zhang, Department of Chemical Engineering and Analytical Science, The University of Manchester, Oxford Road, Manchester M1 3BU, UK.
Email: dongda.zhang@manchester.ac.uk

Abstract

Reinforcement learning (RL) is a data-driven approach to synthesizing an optimal control policy. A barrier to wide implementation of RL-based controllers is its data-hungry nature during online training and its inability to extract useful information from human operator and historical process operation data. Here, we present a two-step framework to resolve this challenge. First, we employ apprenticeship learning via inverse RL to analyze historical process data for synchronous identification of a reward function and parameterization of the control policy. This is conducted offline. Second, the parameterization is improved online efficiently under the ongoing process via RL within only a few iterations. Significant advantages of this framework include to allow for the hot-start of RL algorithms for process optimal control, and robust abstraction of existing controllers and control knowledge from data. The framework is demonstrated on three case studies, showing its potential for chemical process control.

KEYWORDS

apprenticeship learning, inverse reinforcement learning, machine learning, optimal control, reinforcement learning

1 | INTRODUCTION

Recent initiatives for efficiency improvements in industrial process operation has driven interest in the development of high performance, advanced process control (APC) schemes. Reinforcement learning (RL) has achieved impressive results on benchmark game-based control tasks,[1,2] providing an avenue for research in translation to APC. In spite of its high potential, RL has yet to produce any meaningful impact in the (bio)chemical process industry. This work presents a two-step approach to RL-based policy learning, which leverages process data to parameterize an existing control law and then improves the performance of such control further. Additionally, the approach promises to increase the learning efficiency of RL-based control policies, reducing computational and technical investment, as well as data demand.

RL constitutes a subfield of machine learning (ML), which aims to learn optimal control policies. Here, the control problem is formulated as a Markov decision process (MDP), which describes decision-making as a value maximization problem. MDPs construct a probabilistic framework for the discrete-time evolution of a stochastic decision process, with the cost (or value) associated with a control policy, and ultimately process trajectory, evaluated by a reward function. Explicitly, MDPs provide a mathematical basis for sequential decision-making in stochastic environments, which is a description common to process control.[3] Figure 1 details the interpretation of process control as an MDP. The structure of MDPs provides natural closed-loop feedback control.

Solution to an MDP provides a policy $\pi(\cdot)$, which minimizes the expected cost or equivalently maximizes the expected value associated with the evolution of process state. Such a policy satisfies the

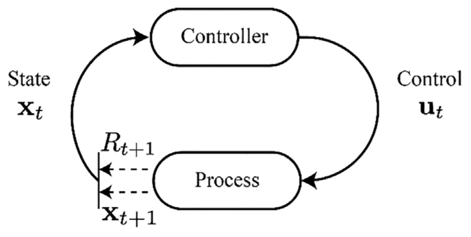

FIGURE 1 Translation of the framework provided by Markov decision process (MDPs) to process control, where the process is analogous to an environment, and the controller to an agent. $\boldsymbol{x}_t$ is representative of the true system state at discrete time t; $\boldsymbol{u}_t$ is the control action computed by the control law at discrete time t; and R_{t+1} is the scalar feedback signal (reward) indicative of the quality of process evolution at time $t+1$

Bellman optimality equation, which is a discrete-time analogue to the continuous-time Hamilton–Jacobi–Bellman equation.[3] Dynamic programming (DP) methods provide exact solution to the Bellman optimality equation. However, such an approach assumes knowledge of the exact process dynamics. DP becomes additionally impractical in the highly dimensional continuous state and action spaces often observed in the process industries.[4] In contrast, RL methods do not require knowledge of the exact process dynamics to learn a solution policy. Instead, RL learns from experience of the process, allowing for $\pi(\cdot)$ to be recalibrated as the process evolves through time via process data.[5] Furthermore, RL has shown significant industrial potential as demonstrated in a number of research works, which have explored application to the calibration of PID controllers;[6] set point tracking;[7] dynamic optimization of nonlinear, stochastic systems;[5,8,9] de novo drug[10] and protein design;[11] and in augmentation of the performance of various model predictive control (MPC) approaches.[12,13] Indeed, the potential use of RL draws discussion of its relation to MPC in the development of APC schemes. MPC schemes require periodic recalibration, which demands expense in technical expertise and often process downtime. The data-driven nature of RL could well mitigate this. Further, the framework provided by MDPs accounts for process stochasticity in a closed-loop manner, converse to MPC where decisions are based on open-loop simulation of the process model, with the loop only "closed" upon observation of the system state at the next discrete time index. Hence, inputs from an RL controller will account for disturbance whereas MPC may not. This provides a theoretical basis for the benefit of RL over MPC controllers.

One set of RL algorithms are known generally as policy optimization methods. Policy optimization methods aim to learn a policy by implicitly learning the value or cost over the decision space[14–16] and directly parameterizing a policy. There are a number of approaches to policy optimization as underpinned by evolutionary strategies, finite difference and policy gradient methods.[17,18] Policy optimization methods have been deployed for tasks including dynamic optimization of nonlinear stochastic processes[19] and tracking problems[6]. For further review of RL methods and their application within the process industries, we direct the reader to the following works.[7,20]

The learning process encapsulated by RL demands both time and technical investment in policy training. This is highlighted further given that RL-based controllers are currently unable to generalize well across control tasks, for example, different changes of set point, meaning policy training is typically undertaken for each task.[21] As a result, implementation of RL control policies is computation and expertise expensive. To solve this problem, this work proposes a method to reduce the time and resource investment demanded by RL, through leverage of process data to learn from demonstration provided by an existing (but unknown) control policy. Then, the initialized RL is improved by learning from the real process over a short time period, thus outperforming the existing control policy. This two-step strategy has been recently deployed in domains including autonomous helicopter flight[22] and self-driving cars.[23,24] To demonstrate this approach, Section 2 will introduce the preliminaries and motivation, Section 3 will outline the methodology, with Section 4 exhibiting different case studies.

2 | PRELIMINARIES

2.1 | Policy gradients and reinforce

Policy gradient methods directly learn a policy. Through the use of artificial neural networks (ANNs) as parameterization, the policy may be deployed naturally in either discrete or continuous action spaces through appropriate network construction.[25] Policy gradient methods do not explicitly learn the value of the policy. Instead, under the policy gradient theorem, acting with respect to the policy and gaining experience of the process dynamics provides approximation of the direction in which value increases fastest in parameter space. Hence, learning proceeds through gradient ascent to update the parameters of the policy to ensure control policies of high value (or low cost) are more probable.[18]

One policy gradient algorithm, reinforce with baseline, approximates the direction in which the policy observes increased performance through Monte Carlo realizations of the process dynamics under the current policy parameterization. This algorithm has several advantages such as convergence to locally optimal solutions in policy space[26] and efficient exploration of the decision space without requirement for a bandit strategy or further optimization routine for action selection—as is the case in many pure action-value methods.[27] Demonstration of the method is also available.[19] Therefore, it is used in this work to learn an RL parameterization of an existing control policy from process data. Despite favor of the reinforce with baseline algorithm, other RL methods capable of operating in continuous action and state spaces could be implemented, such as entropy regularized policy optimization methods,[16] trust region policy optimization,[14] and proximal policy optimization (PPO) methods.[15]

2.2 | Learning from demonstrations via apprenticeship

Learning from demonstrations encompasses an increasingly prevalent and established group of methods, which leverage data generated from an existing but unknown control policy to aid learning-based control systems. This concept is generally termed as apprenticeship learning (AL). AL has been adopted in a number of complex control domains,[22,24] but to our knowledge, this work is the first to propose use of the method to leverage plant data directly, and this is one of the primary contributions of this work. The concepts of AL are expressed in three main subfields including behavioral cloning (i.e., supervised learning), inverse optimal control, and inverse reinforcement learning (IRL).

This study exploited IRL built upon the framework provided by MDPs.[28] MDPs express process objectives mathematically as a reward function. The reward function provides a scalar feedback signal indicative of the optimality of process evolution. IRL is concerned with the task of mathematically abstracting the reward function given process knowledge and demonstrations from an existing control policy. The IRL problem is formalized as: *given* observations of an existing policy over time, sensory inputs available for determination of the originally demonstrated control law and a model of the process; *determine* the reward function that can mostly justify the demonstrated behavior.[24,29,30] IRL proceeds on the assumption that demonstrated control action is noisily optimal under the reward function derived.[30,31] However, it should be noted that this does not necessarily imply that the policy is optimal in view of the true objectives for process control and optimization.

As such, IRL leverages process data to learn a reward function that encodes the control objectives of an existing scheme into a feedback signal. A control policy that maximizes the utility of this reward function within the MDP framework provides a parameterization of the existing control scheme. Hence the pairing of IRL with RL as an MDP solver, allows for synchronously learning the parameterization of an existing but unknown control policy as described in process data. The generated reward function can be used to compare against the process objective (if known) and suggest if the extracted control policy is suitable for online learning. Moreover, manual modifications are always implemented during process control even if the process objective is known. These manual modifications cannot be quantified by human operators, but can be retrieved from historical data by IRL. Therefore, using IRL to generate a reward function is advantageous for parameterization of the optimal control policy.

2.3 | Motivation

In the following work, we demonstrate a framework for learning and optimization of chemical processes. The framework consists of two steps: offline learning, and online learning and improvement. Here, the use of terminology is converse to that common in the ML community. In this work, offline learning indicates a process of AL (via IRL) to infer control objectives from process data and the learning of a corresponding parameterization of the control policy described by data; online improvement then indicates the transfer of the learned parameterization to the real system for the purpose of further policy improvement under the true process objective. The framework enables the learning of an RL-based control policy, by leveraging process data from existing control schemes (offline) and subsequently improves the learned policy parameterization via further RL (online). The automation of offline learning and the policy tuning process that is associated, provides a significant contribution given the technical, computational and data demands of RL-based policy learning.

Offline learning produces a parameterization of the existing control policy, which could be deployed directly for control. The parameterization will achieve similar performance to that expressed by the original control scheme. If necessary, the parameterization may then be transferred to the second stage of online learning for further policy improvement. It should be emphasized that the leveraging of process data is significant given the practical difficulties in learning an RL-based policy “from scratch”.[19,32] The framework also lends itself to the improvement and recalibration of the control scheme temporally. Figure 2 provides further description of the framework proposed.

3 | METHODOLOGY

3.1 | Problem statement

The following work proceeds on the formulation of the underlying problem of process control as an MDP. The true dynamics of an MDP are described as follows:

$$\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \tag{3.1.1}$$

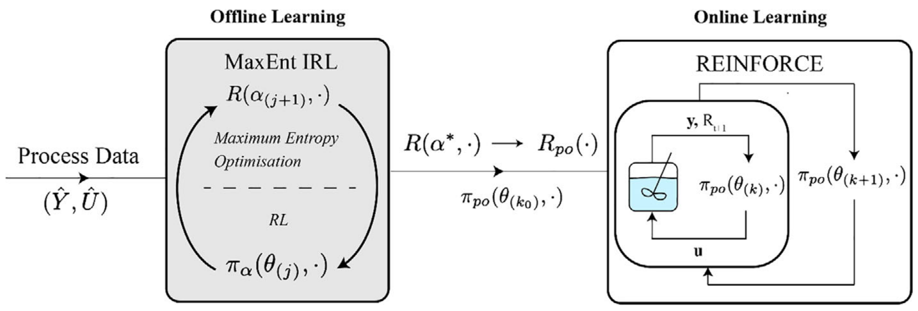
Offline Learning
MaxEnt IRL
$R(\alpha_{(j+1)},\cdot)$
Maximum Entropy Optimisation
RL
$\pi_\alpha(\theta_{(j)},\cdot)$
Process Data
$(\hat{Y},\hat{U})$
$R(\alpha^*,\cdot) \longrightarrow R_{po}(\cdot)$
$\pi_{po}(\theta_{(k_0)},\cdot)$
Online Learning
REINFORCE
y, R_{t+1}
$\pi_{po}(\theta_{(k)},\cdot)$
$\pi_{po}(\theta_{(k+1)},\cdot)$
u

FIGURE 2 The offline–online framework proposed for the learning and optimization of processes. Offline learning utilizes process data to learn a reward function $R(\boldsymbol{\alpha}^*)$ and a parameterization of the demonstrated policy $\pi_{po}\left(\boldsymbol{\theta}_{(k_0)}\right)$. Online learning utilizes the learned parameterization as initialization for further policy optimization under a reward function $R_{po}(\cdot)$ descriptive of the true process objective [Color figure can be viewed at wileyonlinelibrary.com]

$$\mathbf{y}_{t+1} \sim p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}) \tag{3.1.2}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ is a vector of continuous variables representative of the true system state, $\mathbf{u} \in \mathbb{R}^{n_u}$ the manipulated variables (MVs), $\mathbf{y} \in \mathbb{R}^{n_y}$ the observed control variables and t is indicative of the discrete time index.[33] The process evolution between discrete time indices t and $t+1$ is governed by the conditional density function $p(\mathbf{x}_{t+1}|\,\mathbf{x}_t,\ \mathbf{u}_t)$. Similarly, the observation $\mathbf{y}_t$ of the true state of the system $\mathbf{x}_t$ is governed by the conditional density $p(\mathbf{y}_t|\mathbf{x}_t)$. To facilitate learning of a policy prior to transfer to the real system, approximation of the true dynamics proceeds based on state-space models and assumptions regarding process stochasticity, hence:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{d}_t) \tag{3.1.3}$$

$$\mathbf{y}_{t+1} = g(\mathbf{x}_{t+1}) \tag{3.1.4}$$

where $f(\cdot): \mathbb{R}^{n_x \times n_u \times n_d} \rightarrow \mathbb{R}^{n_x}$ is representative of the process dynamics and $\mathbf{d}_t \in \mathbb{R}^{n_d}$ is representative of the process disturbance. The mapping $g(\cdot): \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y}$ is the state observation associated with measurement noise[2].

The following work deploys RL to learn a control policy from process data. The objective of RL is to minimize the expected cost of a dynamic process (or equivalently to maximize its value). In the following, a process trajectory, $\boldsymbol{\tau} = (\mathbf{x}_0, \mathbf{y}_0, \mathbf{u}_0, \dots \mathbf{u}_{T-1}, \mathbf{x}_T, \mathbf{y}_T)$, describes the manner in which a process evolves over a given discrete time horizon of length T. The cost or value $G(\boldsymbol{\tau})$ of the process trajectory over a finite horizon is denoted:

$$G(\boldsymbol{\tau}) = \sum_{t=1}^{T} \gamma^{t-1} R_t \tag{3.1.5}$$

where $\gamma \in (0,\ 1]$ is a discount factor, which provides a net present value interpretation of future value; and R_t is the credit (reward) assigned to the process' evolution between time indices $t-1$ and t. However, in view of process stochasticity, the probability of observing $\boldsymbol{\tau}$ adheres to a conditional density $p(\boldsymbol{\tau}|\,\theta)$ based on the control policy and process dynamics:

$$p\left(\boldsymbol{\tau}|\boldsymbol{\theta}\right) = \bar{\rho}(\mathbf{x}_0) p(\mathbf{y}_0|\mathbf{x}_0) \prod_{t=0}^{T-1} \pi(\mathbf{u}_t|\mathbf{y}_t, \boldsymbol{\theta}) p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) p(\mathbf{y}_{t+1}|\mathbf{x}_{t+1}) \tag{3.1.6}$$

where $\bar{\rho}(\mathbf{x}_0)$ is the probability density of the initial system state; $\pi(\mathbf{u}_t|\,\mathbf{y}_t, \cdot)$ is the conditional density function descriptive of the learned policy, which is parameterized by $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$; and $p(\mathbf{x}_{t+1}|\,\mathbf{x}_t,\ \mathbf{u}_t)$ is the conditional density function representative of the process dynamics.

Note that the definition of a policy as a conditional density function implies it is stochastic. This is important in the scope of the learning process associated with RL but does not necessarily assert the use of a stochastic policy upon deployment for control of the real system (only the mode might be used in practice). The objective of the RL problem and learning process is to find a policy $\pi(\cdot, \boldsymbol{\theta}^*)$ that maximizes the objective $J(\boldsymbol{\tau})$, such that

$$\pi(\cdot, \boldsymbol{\theta}^*) = argmin_{\pi(\cdot, \boldsymbol{\theta})} - J(\boldsymbol{\tau}) \tag{3.1.7}$$

$$J(\boldsymbol{\tau}) = \int p(\boldsymbol{\tau}|\boldsymbol{\theta})\, G(\boldsymbol{\tau})\, d\boldsymbol{\tau} \tag{3.1.8}$$

Equation (3.1.8) describes the probability-weighted average of trajectory value and hence reformulation may utilize equivalence of $J(\boldsymbol{\tau})$ as the expectation of trajectory value under the policy parameters $\boldsymbol{\theta}$, such that

$$J(\boldsymbol{\tau}) = \mathbb{E}_{\tau \sim p(\boldsymbol{\tau}|\boldsymbol{\theta})} [G(\boldsymbol{\tau})] \tag{3.1.9}$$

The description provided in this section formalizes the problem of optimal control under the framework provided by MDPs. One approach to finding approximate solution to the problem described by Equations (3.1.7)–(3.1.9) is encompassed by policy optimization RL methods.

3.2 | Policy gradient and reinforce

Policy gradient methods are a subset of policy optimization methods, which estimate the gradient of the objective detailed by Equation (3.1.8) with respect to the parameters of the current policy. Mathematically, this is described by the policy gradient theorem.[18] The Supporting Information (SI) provides full derivation and explanation of the policy gradient theorem. Given an estimate of the true policy gradient, gradient ascent methods facilitate policy improvement to make trajectories of higher reward more probable. In this manner, the policy parameterization is updated (via Equation (3.2.2)) in the direction provided by the policy gradient (Equation (3.2.1)):

$$\nabla_{\boldsymbol{\theta}_{(j)}} J(\boldsymbol{\tau}) = \nabla_{\boldsymbol{\theta}} \int p(\boldsymbol{\tau}|\boldsymbol{\theta})\, G(\boldsymbol{\tau})\, d\boldsymbol{\tau}$$

$$= \mathbb{E}_{\tau \sim p(\boldsymbol{\tau}|\boldsymbol{\theta})} [G(\boldsymbol{\tau}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}|\boldsymbol{\theta})] \tag{3.2.1}$$

$$\boldsymbol{\theta}_{(j+1)} = \boldsymbol{\theta}_{(j)} + \omega \nabla_{\boldsymbol{\theta}_{(j)}} J(\boldsymbol{\tau}) \tag{3.2.2}$$

where j is the iteration of policy optimization, and ω is the step size in the direction of the policy gradient, $\nabla_{\boldsymbol{\theta}_{(j)}} J(\boldsymbol{\tau})$. The derivation of Equation (3.2.1) leverages the use of a logarithmic identity (see SI). This enables mathematical separation of the conditional probability functions descriptive of the process dynamics and policy (see Equation (3.1.6)). Given the process dynamics are independent of the parameterization, $\boldsymbol{\theta}$, of the policy, $\pi(\boldsymbol{\theta}, \cdot)$, examination of Equation (3.1.6) provides:

$$\nabla_{\boldsymbol{\theta}(j)} \log p(\boldsymbol{\tau}|\boldsymbol{\theta}) = \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}(j)} \log \pi\left(\mathbf{u}_t|\mathbf{y}_t, \boldsymbol{\theta}_{(j)}\right) \tag{3.2.3}$$

Consequently, the policy gradient described by Equation (3.2.1) is reformulated as:

$$\nabla_{\boldsymbol{\theta}(j)} J(\boldsymbol{\tau}) = \mathbb{E}_{\tau}\left[G(\boldsymbol{\tau})\sum_{t=0}^{T-1}\nabla_{\boldsymbol{\theta}(j)}\log\pi\left(\boldsymbol{u}_t|\boldsymbol{y}_t,\boldsymbol{\theta}_{(j)}\right)\right] \tag{3.2.4}$$

Exact computation of the true policy gradient requires full knowledge of the conditional density functions descriptive of process dynamics. Given such knowledge of the process dynamics are unavailable, the policy gradient is approximated by directly sampling the process under the current policy parameterization over a given time horizon via a Monte Carlo method.[5] This is encapsulated by the reinforce with baseline algorithm, which is detailed by Algorithm 1.

```
Algorithm 1 Reinforce with baseline
Input: Initialize: a policy $\pi$ with initial parameters $\boldsymbol{\theta}_0$; learning rate $\omega$; episode length $T$; K episodes for Monte Carlo rollouts of the policy; and, $N$ training epochs. Early stopping conditions may also be implemented.
Output: A policy $\pi(\boldsymbol{u}|\,\boldsymbol{y}, \theta)$
for j = 1, ..., N do
    1. Perform Monte Carlo realizations of the policy for $T$ timesteps and $K$ trajectories. Store all state action pairs observed $\left(\boldsymbol{u}_t^k, \boldsymbol{y}_t^k\right)$, as well as the total return from the episode $G_t^k$ (see Equation (3.1.5))
    2. Estimate the policy gradient and update the parameters of the policy such that $\boldsymbol{\theta}_{(j+1)} = \boldsymbol{\theta}_{(j)} + \omega_{(j)}\frac{1}{K}\sum_{k=1}^{K}\left[\left(G^k - b\right)\nabla_{\boldsymbol{\theta}}\sum_{t=0}^{T-1}\ln\,\pi\left(\boldsymbol{u}_t^k|\boldsymbol{y}_t^k,\boldsymbol{\theta}_{(j)}\right)\right]$, where $b = \frac{1}{K}\sum_{k=1}^{K}G^k$
```

Through utilization of the Monte Carlo method, an unbiased approximation of the true policy gradient is obtained. However, due to the stochastic nature of both the policy and process dynamics, the gradient may observe high variance. In order to reduce the variance of approximation, a baseline b is introduced.[5] This baseline is formulated directly as the expectation of cost associated with the realizations of the policy. In this manner, the update balances the cost of an action against the expected cost from the current policy.

It is of important note that the parameterization of the policy must be continuously differentiable as prescribed by the policy gradient theorem. Naturally, this lends to application of ANNs for function approximation in this work. Specifically, a recurrent long short-term memory (LSTM) neural network was used for parameterization of the control policy. Recurrent LSTM neural networks have demonstrated utility in dynamic stochastic control problems with extension to systems characterized by partial observability.[2] General detail of the mathematical operations specific to LSTMs can be found in the following works,[34,35] with figurative description of the network used in this application provided by Section SI.2 of the SI. The investigation utilized the Pytorch 1.3.1 framework and first-order gradient ascent method Adam to train the LSTM network proposed. The network structure was composed of two hidden layers, each with 20 LSTM cells. A leaky rectified linear unit (ReLU) activation function was applied across both hidden layers and a ReLU6 activation function was applied across the output layer, naturally bounding the output prediction. For a random variable z, the ReLU6 transformation is described as:

$$\mathrm{ReLU6}(z) = \min(\max(0,z),6) \tag{3.2.5}$$

The network designed in the context of this work, predicts the mean ($\boldsymbol{\mu}_t$) and standard deviation ($\boldsymbol{\sigma}_t$) of a unimodal multivariate normal distribution. This distribution describes the conditional density function representative of the control policy, such that: $\boldsymbol{u}_t \sim \pi\left(\boldsymbol{u}_t|\boldsymbol{y}_t,\boldsymbol{H}_t,\boldsymbol{\theta}\right) = \mathcal{N}\left(\boldsymbol{\mu}_t,\boldsymbol{\sigma}_t^2\right)$, where $\boldsymbol{H}_t$ is a learned parameterization of the history of process states provided by the LSTM cells, and $\boldsymbol{\sigma}_t^2$ is the variance. Here, we formally construct the control policy as stochastic. However, upon deployment of the policy to the real system, the policy may be assumed deterministic through selection of the actions corresponding to the mode (equivalently, the mean) of the multivariate normal distribution, such that $\boldsymbol{u}_t = \boldsymbol{\mu}_t$.

In this section, we have presented an approach to solving the MDP characteristic of a control problem through use of the policy gradient method, reinforce with baseline, in combination with an LSTM network for parameterization of the learned policy. In the following, we introduce an approach to policy learning, namely maximum entropy IRL (MaxEnt IRL), which utilizes existing process data to learn from demonstration. Conceptually, this approach is commonly known as AL.

3.3 | AL via IRL

AL via IRL is a general approach to policy learning from demonstration (i.e., process data). The benefits to such an approach are twofold. First, AL via IRL provides a parameterization of the existing control policy expressed in the process data. Second, it facilitates RL-based policy learning under the "real" process objective as it provides an initial policy to hot-start the RL procedure. Otherwise, initially, the agent (or controller) will explore the control action space randomly, which results in a data hungry and time-consuming approach. These benefits are exploited by the framework proposed in Section 2.3 as detailed by Figure 2.

The foundational IRL algorithms construct the reward function $R: \boldsymbol{Y} \to \mathbb{R}$ as a linear combination of state features representative of the system state, $\boldsymbol{\varphi} \in \mathbb{R}^{d \times 1}$, such that:

$$R = \alpha_1\varphi_1 + \alpha_2\varphi_2 + \ldots + \alpha_d\varphi_d \tag{3.3.1}$$

where α_i are feature weightings and $\varphi_i: \boldsymbol{Y} \to \mathbb{R}$ explicitly represent the system state ($\boldsymbol{y}$), but also implicitly encode control objectives.

Typically, $\boldsymbol{\varphi}$ are hand designed based on process and control task knowledge.[29] Knowledge of process objectives can also be applied to place bounds on the weights $\boldsymbol{\alpha}$ in the reward function; however, this may not always be desired as one could assert technical bias on the problem and reduce the feasible region. From this definition of the reward function $R(\boldsymbol{\alpha},\mathbf{y})$, consequent reformulation of the policy optimization objective $J(\boldsymbol{\tau})$ in Equation (3.1.9) yields

$$J(\boldsymbol{\tau})=\mathbb{E}_{\tau\sim p(\tau|\theta)}\left[\sum_{t=1}^{T}\gamma^{t-1}R(\boldsymbol{\alpha},\mathbf{y}_t)\right] \tag{3.3.2}$$

$$J(\boldsymbol{\tau})=\sum_{i=1}^{d}\alpha_i\mathbb{E}_{\tau\sim p(\tau|\theta)}\left[\sum_{t=1}^{T}\gamma^{t-1}\varphi_i(\mathbf{y}_t)\right] \tag{3.3.3}$$

This may be further decomposed through definition of trajectory features, υ_i, such that for the discounted case:

$$\upsilon_i^{\gamma}=\sum\nolimits_{t=1}^{T}\gamma^{t-1}\varphi_i(\mathbf{y}_t) \tag{3.3.4}$$

$$J(\boldsymbol{\tau})=\sum_{i=1}^{d}\alpha_i\mathbb{E}_{\tau\sim p(\tau|\theta)\left[\upsilon_i^{\gamma}\right]} \tag{3.3.5}$$

$$J(\boldsymbol{\tau})=\boldsymbol{\alpha}^T\mathbb{E}_{\tau\sim p(\tau|\theta)[\boldsymbol{\upsilon}^{\gamma}]} \tag{3.3.6}$$

where $\boldsymbol{\alpha}\in\mathbb{R}^{d\times 1}$ and $\boldsymbol{\upsilon}^{\gamma}\in\mathbb{R}^{d\times 1}$. Equivalently, undiscounted trajectory features $\boldsymbol{\upsilon}$ may be recovered by setting $\gamma = 1$. The characterization of a policy and process trajectory in terms of $\boldsymbol{\upsilon}$ enables RL to learn from multiple, distributed trajectories and reduces the problem to learning feature weights $\boldsymbol{\alpha}^*$.[29,30] Conceivably, a number of different reward functions exist that recover the desired behavior. The current study uses the MaxEnt IRL framework proposed by Ziebart et al.,[30,36] which proceeds in identification of $\boldsymbol{\alpha}$ via a probabilistic approach as underpinned by the principle of maximum entropy.

3.4 | Maximum entropy IRL

In AL, we are interested in learning a policy as described by a conditional probability density function $\pi(\mathbf{u}_t|\,\mathbf{y}_t,\cdot)$, such that upon deployment of the policy to the real system, the process observes the same evolution as that described by process data (see Equation (3.1.6)). Explicitly, the investigation learns the expert's policy expressed by process trajectories $\mathbf{T}=\left[\boldsymbol{\tau}_1^E,\ldots,\boldsymbol{\tau}_K^E\right]$ as characterized by trajectory features, $\left\{\boldsymbol{\upsilon}_k^E\right\}$, where $k = 1,\ldots, K$. MaxEnt IRL[30] is an established method and poses solution to the problem of learning such an approximate policy. It learns a reward function that maximizes the likelihood of observing the demonstrated trajectories $\mathbf{T}$ given an accurate model of the process dynamics. Further discussion is provided in SI.3. It follows that the log-probability of observing a given trajectory $\boldsymbol{\tau}$ is proportional to the cumulative undiscounted reward observed between a start and terminal state,[36] such that:

$$p(\boldsymbol{\tau}\,|\boldsymbol{\alpha})=\frac{\exp\{\boldsymbol{\alpha}^T\boldsymbol{\upsilon}(\boldsymbol{\tau})\}}{Z(\boldsymbol{\alpha},\cdot)} \tag{3.4.1}$$

where $\boldsymbol{\upsilon}=[\upsilon_1,\upsilon_2,\ldots,\upsilon_d]$, and $Z(\boldsymbol{\alpha},\cdot) = \sum_{\boldsymbol{\tau}\in\mathbf{T}}\exp\{\boldsymbol{\alpha}^T\boldsymbol{\upsilon}(\boldsymbol{\tau})\}$ is the partition function, which enforces normalization of the distribution. Formally, the approach prescribes that each of the demonstrations, $\boldsymbol{\tau}^E\in\mathbf{T}$, are independently and identically distributed such that the likelihood of observing the set of trajectories, $\mathbf{T}$, expressed in process data is:

$$p(\mathbf{T}|\boldsymbol{\alpha})=\prod\nolimits_{k=1}^{K}p\left(\boldsymbol{\tau}_k^E|\boldsymbol{\alpha}\right)=\prod_{k=1}^{K}\frac{1}{Z(\boldsymbol{\alpha},\cdot)}\exp\left\{\boldsymbol{\alpha}^T\boldsymbol{\upsilon}_k^E\right\} \tag{3.4.2}$$

where $Z(\boldsymbol{\alpha},\cdot)$ is assumed constant for all $\boldsymbol{\tau}^E\in\mathbf{T}$;[30] and $p(\mathbf{T}|\,\boldsymbol{\alpha})$ is the likelihood of observing the set of demonstrations. Under the maximum entropy formulation,[30,31,36,39] optimal solution of the feature weights, $\boldsymbol{\alpha}^*$ is:

$$\boldsymbol{\alpha}^*=\text{argmax}_{\boldsymbol{\alpha}}\,p\left(\mathbf{T}|\boldsymbol{\alpha}\right)=\text{argmax}_{\boldsymbol{\alpha}}\prod\nolimits_{k=1}^{K}p\left(\boldsymbol{\tau}_k^E|\boldsymbol{\alpha}\right) \tag{3.4.3}$$

The gradient of the log-likelihood objective (Equation (3.4.3)) with respect to feature weights, $\boldsymbol{\alpha}$, is formulated as:

$$\nabla_{\boldsymbol{\alpha}_{(i)}}\sum_{k=1}^{K}\log p\left(\boldsymbol{\tau}_k^E|\boldsymbol{\alpha}_{(i)}\right)=\frac{1}{K}\sum_{k=1}^{K}\boldsymbol{\upsilon}_k^E-\nabla_{\boldsymbol{\alpha}_{(i)}}\log Z\left(\boldsymbol{\alpha}_{(i)},\cdot\right) \tag{3.4.4}$$

$$\nabla_{\boldsymbol{\alpha}_{(i)}}\log Z\left(\boldsymbol{\alpha}_{(i)},\cdot\right)=\mathbb{E}_{\boldsymbol{\tau}^{\pi}\sim p\left(\boldsymbol{\tau}^{\pi}|\boldsymbol{\alpha}_{(i)},\boldsymbol{\theta}^*\right)}\left[\boldsymbol{\upsilon}^{\pi}\right] \tag{3.4.5}$$

where $\nabla_{\boldsymbol{\alpha}_{(i)}}\log Z(\boldsymbol{\alpha}_{(i)}\,,\cdot)$ is estimated via policy optimization in the underlying MDP to find a policy, $\pi(\cdot,\boldsymbol{\theta}^*)$, that maximizes the following modified objective, and then subsequently performing Monte Carlo realizations of the solution policy under the process dynamics to provide sample trajectories, $\boldsymbol{\xi}=\left[\boldsymbol{\tau}_1^{\pi},..,\boldsymbol{\tau}_N^{\pi}\right]$ characterized by $\left\{\boldsymbol{\upsilon}_n^{\pi}\right\}$, where $n = 1, \ldots, N$. This is also discussed further in Section SI.3. Equations (3.4.4) and (3.4.5) suggest that the MaxEnt IRL problem finds a weight vector, $\boldsymbol{\alpha}^*$, which minimizes the differences between the expected trajectory features of the learned policy and that which is demonstrated. Gradient-based optimization methods may be deployed to find solution, $\boldsymbol{\alpha}^*$, by stepping parameter values, $\boldsymbol{\alpha}$, in the direction of the gradient.[30,36] This work utilizes the first-order gradient ascent method (Equation (3.4.6)).

$$\boldsymbol{\alpha}_{(i+1)}=\boldsymbol{\alpha}_{(i)}+\kappa\nabla_{\boldsymbol{\alpha}_{(i)}}\log p\left(\mathbf{T}|\boldsymbol{\alpha}_{(i)}\right) \tag{3.4.6}$$

where κ is a learning rate. The problem formulated here constitutes a bi-level optimization, with the upper level task approached by MaxEnt IRL and the lower level task handled by the policy gradient method reinforce. In each iteration i of the upper MaxEnt IRL problem, a new reward function, $R(\boldsymbol{\alpha}_{(i)},\cdot)$, is abstracted. The underlying MDP is subsequently solved by policy optimization and estimation of the partition function and $\mathbb{E}[\boldsymbol{\upsilon}^{\pi}]$ provided. The reinforce method and the approach to solving the lower level optimization task is detailed by Algorithm 1.

It should be noted that the approaches to policy optimization provided by PPO and entropy regularization could provide further stability in learning and accuracy in estimation of the partition function, respectively. In view of the length of the horizon specific to many control tasks, discounted trajectory features $\boldsymbol{\upsilon}^{\gamma}$, as described by Equation (3.3.4), should be used rather than the undiscounted features. This establishes the upper MaxEnt IRL task as a nonconvex optimization[37] but provides performance improvements in the lower level policy optimization task. Algorithm 2 details the MaxEnt IRL algorithm further.

Algorithm 2 MaxEnt inverse reinforcement learning

```
Input: Initialize: a policy $\pi^A_{(0)}$ with initial parameters $\theta_{(0)}$; a weight vector $\boldsymbol{\alpha}$; state feature functions $\boldsymbol{\varphi}(\mathbf{x})$; trajectory features representative of the demonstrated trajectories $\boldsymbol{\upsilon}^E$; maximum iterations $N_{max}$; learning rate $\kappa$;
Output: optimal weights $\boldsymbol{\alpha}^*$ and agent parameterization of the demonstrated policy $\pi_{\text{po}}(\boldsymbol{\theta}_{(k_0)}), \cdot$ for further policy improvement in online learning.
for $n = 1, \ldots, N_{max}$ do
    1. Perform policy optimization of $\pi^A_{(n-1)}$ under the current reward function $R(\boldsymbol{\alpha}_{(n)})$ via Algorithm 1. Return $\pi^A_{(n)}$ as solution to the MDP defined.
    2. Perform Monte Carlo realization of $\pi^A_{(n)}$ (via Algorithm S1) to evaluate the policy. Return the trajectory features characteristic of the expected process evolution under the policy $\mathbb{E}[\boldsymbol{\upsilon}^{\pi_{(n)}}]$.
    3. Approximate the gradient of the likelihood of observing the demonstrated trajectories with respect to the weights $\nabla_\alpha \log p(\mathbf{T}|\boldsymbol{\alpha}) = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{\upsilon}^E_k - \mathbb{E}[\boldsymbol{\upsilon}^{\pi_{(n)}}]$.
    4. Perform gradient ascent such that $\boldsymbol{\alpha}_{(n+1)} = \boldsymbol{\alpha}_{(n)} + \kappa\nabla_\alpha \log p\,(\mathbf{T}|\boldsymbol{\alpha})$
end
```

3.5 | Overview of the proposed methodology

The methodology proposed leverages the large amount of process control data available to industry to learn an RL-based parameterization of a previously implemented control scheme through AL via IRL. This parameterization should express the existing control law as well as the process knowledge of operators provided the available data is sufficiently rich. Once a parameterization is constructed offline, it is deployed as initialization for further RL-based policy improvement (online). This online learning proceeds under a reward function descriptive of the real process objectives. Through this approach, we significantly reduce the computational and technical investment associated with training an RL-based control policy. Specifically, the improvements noted are drawn from the offline section of the framework. Here, we combine simulation with the use of IRL to automate analysis of historical process data. This enables us to directly abstract a reward function, which provides clear preference (discrimination) over controls from: (i) knowledge of the process control task we are concerned with (represented by the basis features, $\boldsymbol{\varphi}$, in the reward function); and (ii) empirical observations of the system and its behavior in response to controls (by optimizing the feature weight $\boldsymbol{\alpha}$). Learning under this reward function provides a parameterization of the existing control scheme expressed in process data. Section 4 presents a number of computational case studies for empirical demonstration of the framework described.

4 | COMPUTATIONAL CASE STUDIES

4.1 | Introduction to the case studies

The optimization objective of the following studies is set point tracking in a multiple-input, multiple-output (MIMO) control scheme. Specifically, the process is a nonisothermal continuous stirred tank reactor under operation of an endothermic isomerism reaction of the form: $A \rightarrow B$. The reaction rate temperature dependence is described by the Arrhenius kinetics. Demonstration is provided in the form of process data generated by the action of a PID control scheme, produced via a discrete time Python 3.7.3 implementation. The controlled variables ($\mathbf{y}$) are concentration of reagent, C_A^{obs} and temperature of the reactor, T^{obs}. The MVs ($\mathbf{u}$) are the temperature of a heating jacket, T_E and concentration of the reagent in the input stream, C_{A0}. Bounds are placed upon the absolute values of the control space. Definition of process variable follows:

$$\mathbf{y} = \left[C_A^{obs}, T^{obs}\right]^T \tag{4.1.1a}$$

$$\mathbf{x} = [C_A, T]^T \tag{4.1.1b}$$

$$\mathbf{u} = [C_{A0}, T_E]^T \tag{4.1.1c}$$

In the case studies presented, the process model is of deviation variable form and was derived from first principles. The deviation variable, z^* of random variable, z is expressed as:

$$z^* = z - z_{ss} \tag{4.1.2}$$

where z_{ss} is the previous steady-state value of z. Process stochasticity (disturbance) is assumed zero mean Gaussian, as is the nature of system observation. Therefore, approximation of the true underlying process dynamics takes the form of a system of stochastic differential equations, such that

$$\mathbf{x}^*_{t+1} = \mathbf{x}^*_t + h\left(\mathbf{x}^*_t, \mathbf{u}^*_t\right)dt + \delta\left(\mathbf{x}^*_t\right)dW_t \tag{4.1.3a}$$

$$\mathbf{y}^*_{t+1} = g\left(\mathbf{x}^*_{t+1}\right) \tag{4.1.4a}$$

where function $h(\cdot)$ is descriptive of the underlying process dynamics; $\delta(\cdot)$ the magnitude of disturbance, as described by the Wiener process, W_t[38]; and, $g(\cdot)$ describes the nature of system observation. In the following studies,

$$h\left(\mathbf{x}^*_t, \mathbf{u}^*_t\right) = \begin{bmatrix} -3.997 & -0.446 \\ -6.092 & -1.581 \end{bmatrix} \mathbf{x}^*_t + \begin{bmatrix} 0.500 & 0 \\ 0 & 0.305 \end{bmatrix} \mathbf{u}^*_t \tag{4.1.3b}$$

$$\delta\left(\mathbf{x}^*_t\right) = \begin{bmatrix} 0.500 & 0 \\ 0 & 0.300 \end{bmatrix} \mathbf{x}^*_t \tag{4.1.3c}$$

$$g\left(\mathbf{x}^*_{t+1}\right) = \begin{bmatrix} 1+\mathcal{N}(0,0.025) & 0 \\ 0 & 1+\mathcal{N}(0,0.025) \end{bmatrix} \mathbf{x}^*_{t+1} \tag{4.1.4b}$$

and the Euler Maryuama method was utilized for system integration.[38] The SI provides formal derivation and parameter values. Given the formulation of the MIMO problem, the investigation is concerned with controlling the evolution of error, ε within both the temperature, T^{obs} and reagent concentration, C_A^{ob} control loops.

4.2 | Design of state features for AL

The introduction provided in Section 3.4 outlines a framework for learning the weight vector $\boldsymbol{\alpha}^*$, which provides a linear mapping from state representations, $\boldsymbol{\varphi}$, to scalar cost. Further, for a given representation, a set of possible process trajectories exist, which match the counts of state features (trajectory features) of the existing policy. Therefore, design of $\boldsymbol{\varphi}$ should consider both the process, optimization objectives and restriction of the possible set of trajectories. As a result, this work proposes the use of three types of state features, all of which provide consistent control objectives temporally and utilize knowledge of the underlying process control task.

4.2.1 | Type I

The first state feature proposed is encapsulated by the radial basis function (RBF). The RBF provides a similarity measure and allocates exponentially lower cost or greater value for those control policies which achieve set point tracking. The feature is formulated as:

$$\hat{\varepsilon} = \frac{y_{sp} - y}{y_{sp} - y_{ss}} \tag{4.2.1}$$

$$\varphi_I(\hat{\varepsilon}) = e^{-(\beta\hat{\varepsilon})^2} \tag{4.2.2}$$

where y_{ss} is the previous observed steady state of the system, y_{sp} is the desired set point, β is the shape parameter and $\varphi_I(\hat{\varepsilon}) = [0,1]$. The closer the value of β to zero, the greater the offset tolerated and the denser the reward landscape. Conversely, higher values of β provide exponentially greater rewards for trajectories closer to the set point, but a sparser reward landscape. In the following case studies, the investigation utilized $\beta = 10$.

4.2.2 | Type II

Although the Type I feature is an absolute measure of control performance, alone it does not fully characterize the evolution of system response. Furthermore, the set of possible process trajectories, which could match the representation of the demonstrated policy $\mathbf{v}^E$ is large. To restrict the possible set, Type II and III features take inspiration from the PID control law, which at a given time is a linear combination of the error, $\varepsilon = y_{sp} - y$, in the control loop at the current time point (proportional), the manner in which the error has evolved over time (integral) and the projected evolution of error in the future (derivative). Hence, the Type II state feature proposed intends to quantify how the absolute error in a control loop evolves temporally. As such, Type II state features are described as:

$$\varphi_{II}(\hat{\varepsilon}) = \int_0^t |\hat{\varepsilon}|\, dt \approx \sum_{j=1}^{tc} |\hat{\varepsilon}|\, \Delta t \tag{4.2.3}$$

where Δt is equivalent to the sampling time or times at which control is provided (in this work, the two are synonymous), $|\cdot|$ refers to the absolute value; j the discrete time index and tc the current time point. The absolute magnitude of the error provides clear control objective regardless of whether the error $\hat{\varepsilon}$ is positive or negative in value. If this was not taken, actions that decrease error in the control loop may be penalized or rewarded in an RL setting depending upon whether the integral of the error becomes positive or negative as a result.

4.2.3 | Type III

The design of Type III state features aims to quantify how the error in the control loop may evolve into the future. As a result, the feature approximates the derivative of the error in the control loop at the sampled time:

$$\varphi_{III}(\hat{\varepsilon}) = \frac{d|\hat{\varepsilon}|}{dt} \approx \frac{|\hat{\varepsilon}_{tc}| - |\hat{\varepsilon}_{tc-1}|}{\Delta t} \tag{4.2.4}$$

where $tc - 1$ is the previous discrete time index. In view of the proposed state features, the investigation is able to characterize control trajectories and provide direct and consistent control objective. As a result, the reward function R of the MDP described is specified as

$$R = \alpha_1 \varphi_I\left(\hat{\varepsilon}_{C_A^*}\right) + \alpha_2 \varphi_I(\hat{\varepsilon}_{T^*}) + \alpha_3 \varphi_{II}\left(\hat{\varepsilon}_{C_A^*}\right) + \alpha_4 \varphi_{II}(\hat{\varepsilon}_{T^*}) + \alpha_5 \varphi_{III}\left(\hat{\varepsilon}_{C_A^*}\right) + \alpha_6 \varphi_{III}(\hat{\varepsilon}_{T^*}) \tag{4.2.5}$$

4.3 | Case study definitions

Three case studies demonstrate the use of the framework in different contexts and control tasks. Table 1 details the specific experimental setup. Case Study I demonstrates the framework proposed for deployment when subjectively near optimal control is provided by an existing control scheme. Case Study II demonstrates the framework is still effective when the control demonstrated by an existing scheme is subjectively suboptimal. Case Study III explores the potential to transfer knowledge within the framework in order to aid efficiency in learning on different control tasks.

5 | RESULTS AND DISCUSSION

5.1 | Case Study I—Learning from near optimal demonstrations

The purpose of this case study is to construct an RL controller which learns from demonstration provided by a near optimal control policy and then to improve it further. As such, we demonstrate the full utility of the offline-online framework proposed. First, offline learning under MaxEnt IRL is deployed to find a linear combination $\boldsymbol{\alpha}^*$ of state features, which infers and encodes control objectives into a feedback signal or reward function. Under this reward function, a parameterization of the control policy expressed in process data is learned in order to match the demonstrated process behavior as characterized through expected trajectory features. The learned parameterization is then improved under the real process objective, which in this case is pure tracking. Here the demonstrated control policy is that of a well-tuned PID controller (PID1 as detailed by the SI).

5.1.1 | Results of AL via MaxEnt IRL

Utilizing 500 Monte Carlo realizations of the PID1 policy, the methodology was able to generate an informative dataset and subsequently characterize the policy using the six basis features presented in Equation (4.2.5), with $\gamma = 0.99$ and $T = 50$ indicates the length of the discrete-time finite horizon. The trajectory feature expectations of PID1 are outlined in Table 2.

From Table 2, it is concluded that under the characterization of the PID1 policy $\boldsymbol{\upsilon}^{\gamma,E}$, Algorithm 2 was able to learn an agent parameterization of the demonstrated policy (i.e., PID controller). This was achieved after just four iterations of the algorithm. Each iteration is composed of solving an MDP via RL (detailed by Algorithm 1) and then updating the weight vector $\boldsymbol{\alpha}$ via Equation (3.4.4). The hyperparameters for Algorithm 2 and each iteration are detailed by the SI. It is worth reiterating that there is a set of possible policies, which observe the same expected trajectory feature counts $\mathbb{E}\left[\boldsymbol{\upsilon}^{\gamma,E}\right]$ as that of the demonstrated policy. In the context of this work, further restricting the possible set is not necessary; however, introduction of further state features $\boldsymbol{\varphi}$ would facilitate such. Given that $\boldsymbol{\varphi}$ compose the reward function and all express inherent set point tracking objectives, intuitively, any of the policies from the possible set, which match the trajectory features of the demonstrated policy should provide good initialization for further policy improvement. The learned weight vector $\boldsymbol{\alpha}^*$ may also be interpreted and provide insight into the dynamics of the respective control loops.

TABLE 1 Conditions of design for the case studies detailed. The real initial state of the controlled variables $\mathbf{x_0}$ is drawn from the respective distributions. The set point y^*_{sp} details the new setpoint of the respective control variables as set at $t = 0$

Case study	System parameter	Concentration (C^*_A) control loop	Temperature (T^*) control loop
I	Initial state distribution $\bar{p}(\mathbf{x}_0)$	$\mathcal{N}$ (0, 0.25)	$\mathcal{N}$ (0, 0.75)
	Set point Y^*_{sp}	−1	4
II	Initial state distribution $\bar{p}(\mathbf{x}_0)$	$\mathcal{N}$ (0, 0.25)	$\mathcal{N}$ (0, 0.75)
	Set point Y^*_{sp}	1	4
III	Initial state distribution $\bar{p}(\mathbf{x}_0)$	$\mathcal{N}$ (0, 0.25)	$\mathcal{N}$ (0, 0.75)
	Set point Y^*_{sp}	−2.5	3

TABLE 2 The expected discounted trajectory features of PID1 ($\boldsymbol{\upsilon}^{\gamma,E}$) and the policy learned through AL ($\boldsymbol{\upsilon}^{\gamma,\boldsymbol{\pi}}$), and IRL's feature weight ($\boldsymbol{\alpha}^*$) generated in CS I. $Y^* - Type$ indicates the type of trajectory feature and the respective control loop error

	Trajectory features					
	$C^*_A - I$	$T^* - I$	$C^*_A - II$	$T^* - II$	$C^*_A - III$	$T^* - III$
$\mathbb{E}\left[\boldsymbol{\upsilon}^{\gamma,E}\right]$	21.63	20.68	4.08	7.93	−22.87	−22.43
$\mathbb{E}[\boldsymbol{\upsilon}^{\gamma,\boldsymbol{\pi}}]$	21.41	20.76	4.31	7.03	−22.28	−22.71
$\boldsymbol{\alpha}^*$	0.137	0.652	−0.067	−0.630	−0.194	−0.343

Abbreviations: AL, apprenticeship learning; IRL, inverse reinforcement learning.

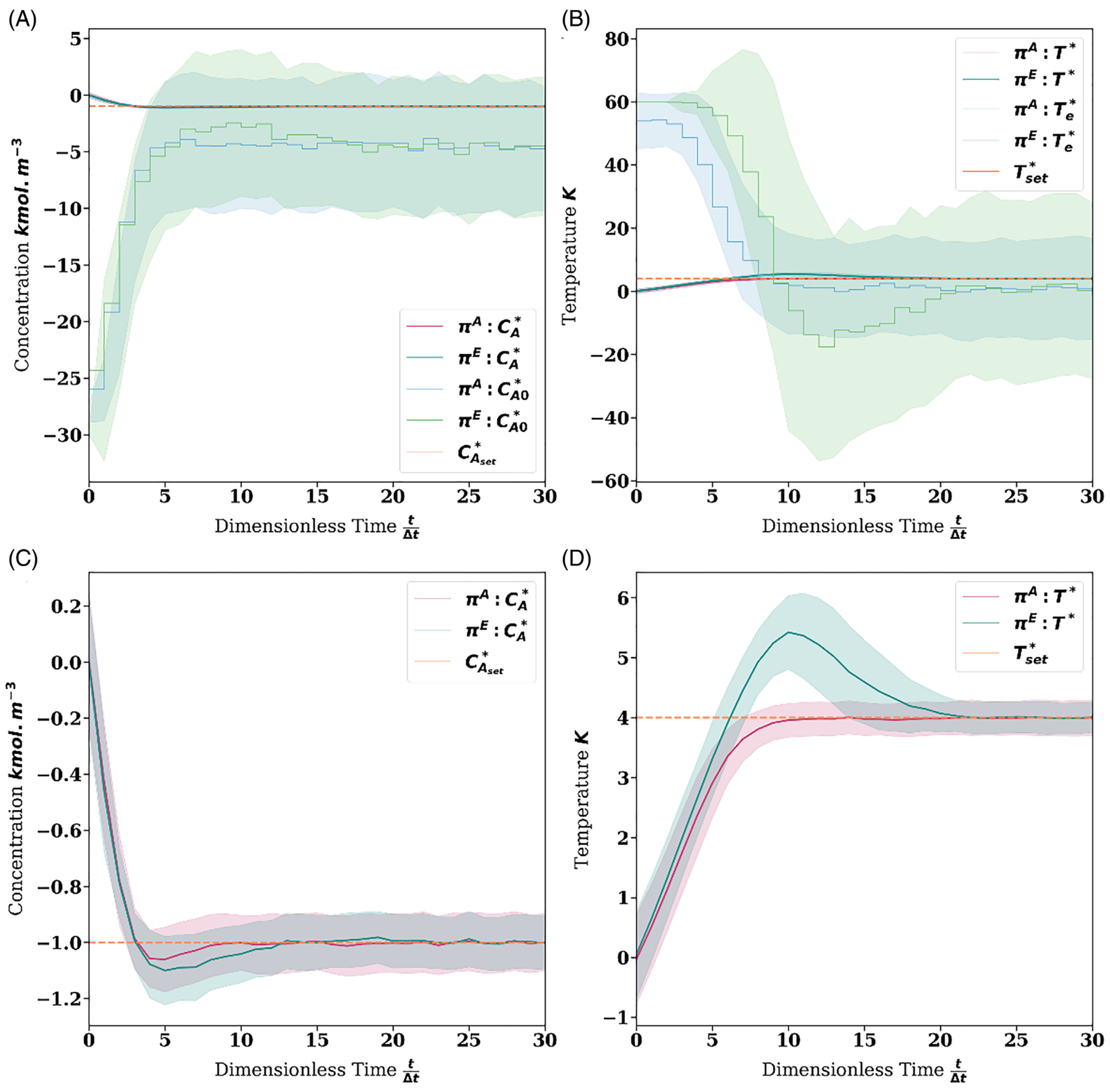

FIGURE 3 Optimal policy of the agent in Case Study I. (A,B) Control and system response of the concentration control loop and of the temperature control loop, respectively. (C,D) Zoomed system response in the concentration control loop and in the temperature control loop, respectively. π^A and π^E indicate the policy of the agent (after online learning) and the PID, respectively. Solid line represents the mean control response and the shaded regions indicate the standard deviation. Line colors of manipulated variables: blue—π^A; light green—π^E. Line colors of control variables: red—π^A; dark green—π^E. Line color of set points: orange [Color figure can be viewed at wileyonlinelibrary.com]

The state features that are specific to the temperature control loop receive a greater weight than the concentration control loop. This is likely reflective of the endothermic nature of reaction and the relative changes of set point in the temperature loop and concentration loop. Compared to changing reactant concentration, an increase in reactor temperature T will likely shift reaction equilibrium more significantly in a manner to increase consumption of reagent. As a result, the system dynamics act in a way to aid the set point change in the concentration control loop. Hence, greater weighting is allocated to control of the temperature control loop.

In this section, we show the utility of the offline learning method proposed in the context of learning by demonstration (or AL). Subsequently, we demonstrate how online learning may be deployed for further policy improvement.

5.1.2 | Online learning and optimal control

Further improvement of the initial policy (Section 5.1.1) utilizes Algorithm 1 and a real process reward function shown as Equation (5.1.2.1), which expresses pure set point tracking objective

$$R = \varphi_l\left(\hat{\varepsilon}_{C_A^*}\right) + \varphi_l\left(\hat{\varepsilon}_{T^*}\right) \tag{5.1.2.1}$$

Here, the parameter β in φ_l (Equation (4.2.2)) is retuned to ensure that high performance set-point tracking is achieved ($\beta = 30$). The final result of the policy obtained is displayed in Figure 3.

Examination of Figure 3(A) describes the control policies of the agent and PID1 within the concentration control loop. Given the initialization provided by IRL, further online RL-based policy improvement learns a control observably similar but relatively smoother, to that demonstrated by the PID controller. Explicitly, the policy improvement was provided by two rounds of online learning, with 10 training iterations (epochs) per round. As a result, the agent is able to facilitate a system response, which meets set point faster with less overshoot observed than using the PID controller (shown in Figure 3 (C)). Similar observations are made in analysis of Figure 3(B,D), which demonstrate the response of the temperature control loop. In this case, the online updated RL yields a better temperature response characterized by a fast rise time with no observable overshoot.

5.2 | Case Study II—Learning from suboptimal demonstrations

In Case Study II, the demonstrations (process data) are derived from a second PID controller (PID2 detailed by the SI). Compared to Case Study I, the demonstrations provided by the PID controller here are of an overdamped control response, which subjectively appears suboptimal.

TABLE 3 The expected discounted trajectory features of the PID2 ($\boldsymbol{\upsilon}^{\gamma,E}$) and the policy learned through AL ($\boldsymbol{\upsilon}^{\gamma,\pi}$), and IRL's feature weight ($\boldsymbol{\alpha}^*$) generated in CS I. $Y^* - Type$ indicates the type of trajectory feature and the respective control loop error

	Trajectory features υ					
	$C_A^* - I$	$T^* - I$	$C_A^* - II$	$T^* - II$	$C_A^* - III$	$T^* - III$
$\mathbb{E}[\boldsymbol{\upsilon}^{\gamma,E}]$	13.76	8.52	8.02	15.53	−22.49	−20.71
$\mathbb{E}[\boldsymbol{\upsilon}^{\gamma,\pi}]$	16.41	7.10	6.46	13.29	−21.82	−18.79
$\boldsymbol{\alpha}^*$	−0.259	−0.182	−0.545	−0.093	−0.545	−0.545

Abbreviations: AL, apprenticeship learning; IRL, inverse reinforcement learning.

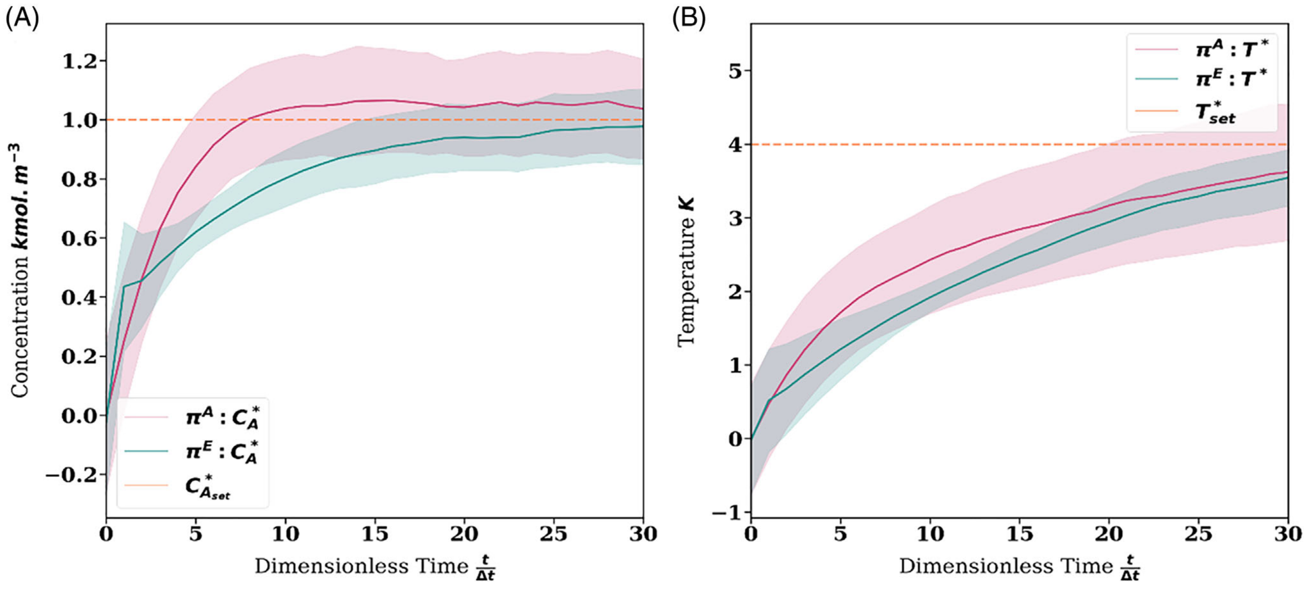

FIGURE 4 System response over the first 30 control interactions from the policy learned from demonstration during apprenticeship learning (AL) in Case Study II. (A,B) System response in the concentration control loop and the temperature control loop, respectively. π^A and π^E indicate the response associated with the policy of the agent (after offline learning) and that demonstrated, respectively. Solid line represents the mean control response and the shaded regions indicate the standard deviation. Line colors of control variables: red—π^A; dark green—π^E. Line color of set points: orange [Color figure can be viewed at wileyonlinelibrary.com]

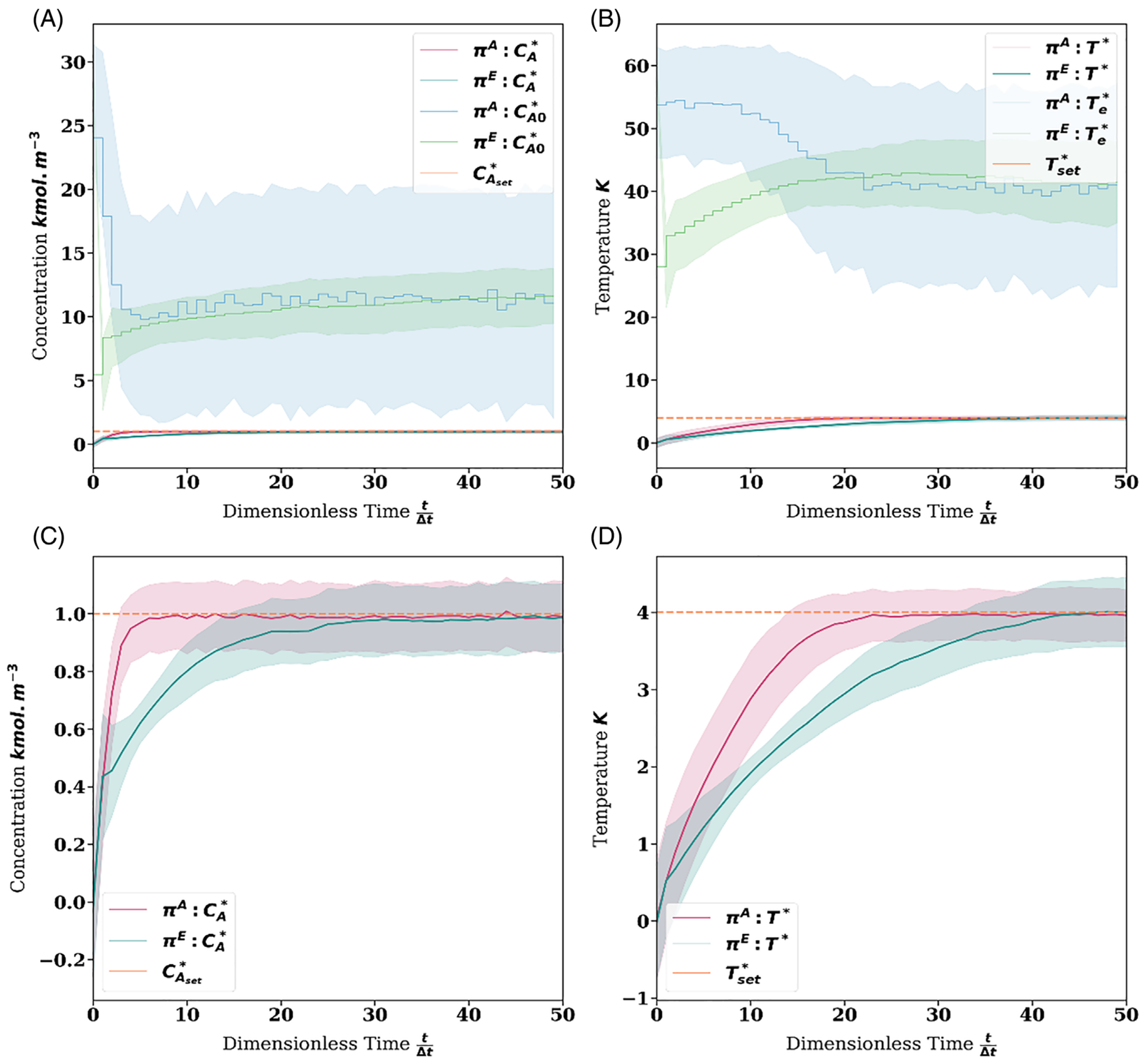

FIGURE 5 Optimal policy of the agent in CS II over the full simulated horizon. (A,B) Control and system response of the concentration control loop and the temperature control loop, respectively. (C,D) Zoom of the system response in the concentration control loop and in the temperature control loop, respectively. π^A and π^E indicate the policy of the agent (after online learning) and the PID, respectively. Solid line represents the mean control response and the shaded regions indicate the standard deviation. Line colors of manipulated variables: blue—π^A; light green—π^E. Line colors of control variables: red—π^A; dark green—π^E. Line color of set points: orange [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 4 The expected discounted trajectory features of the PID1 generated in CS III. Y^*-Type indicates the type of trajectory feature and the respective control loop error

	Trajectory features υ					
	$C_A^* - I$	$T^* - I$	$C_A^* - II$	$T^* - II$	$C_A^* - III$	$T^* - III$
$\mathbb{E}[\upsilon^{\gamma,E}]$	16.07	18.36	8.08	8.35	−21.83	−22.78
$\mathbb{E}[\upsilon^{\gamma,\pi}]$	14.00	18.04	9.37	6.50	−19.94	−21.06
α	0.664	0.052	−0.223	−0.226	−0.403	−0.541

5.2.1 | Results of AL via MaxEnt IRL

In similar fashion to Section 5.1.1, Algorithm S1 was used to characterize the demonstrations from PID2. Table 3 details the resultant trajectory feature expectations $\mathbb{E}[v^{\gamma,E}]$.

Once again, Algorithm 2 facilitates the learning of an agent parameterization of the demonstrated policy in three iterations. It is of note, however, that the methodology was unable to match the trajectory features exactly. Instead, a good approximation of the demonstrated policy was produced. There are two points of discussion here. First, it is likely that the reward function itself is underspecified and further state features, φ, should be proposed. Second, it is possible that the objectives of the demonstrated control policy cannot be described purely as a linear combination of the state features[31]—although the linear approximation in this case is reasonable, given the similarity of the trajectory features.

In this case study, state features relevant to the concentration control loop are allocated the greatest weighting. This is because the set points are changed in the same direction (as detailed by Table 1). Naturally, a rise in reagent concentration will cause a decrease in temperature (endothermic reaction), whilst a rise in temperature will facilitate the conversion of reagent concentration. As the reaction equilibrium is more sensitive to the temperature change, greater weightings must be added to the concentration control loop to reach the new set point.

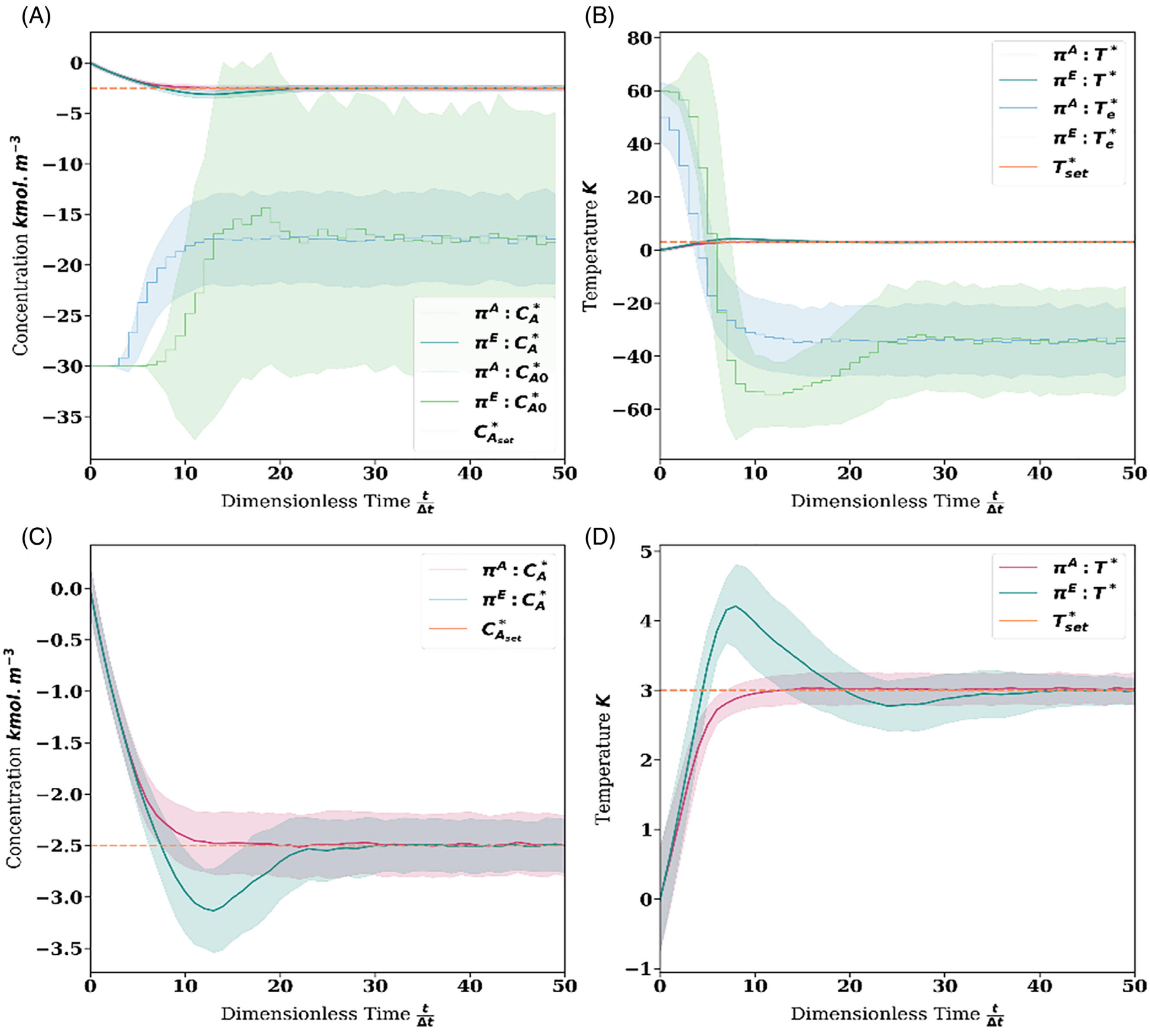

FIGURE 6 Policy π^A generated as a result of knowledge transfer through apprenticeship learning (AL) and online policy optimization. (A,B) Control and system response of the concentration control loop and the temperature control loop, respectively. (C,D) Zoom of the system response in the concentration control loop and the temperature control loop, respectively. π^A and π^E indicate the policy of the agent (after online learning) and the PID, respectively. Solid line represents the mean control response and the shaded regions indicate the standard deviation [Color figure can be viewed at wileyonlinelibrary.com]

Furthermore, Type I state features are allocated negative weights, which is unusual. Intuitively, Type I features represent a similarity measure between the current state of the system and the desired set point. Given that the feature value is non-negative ($\varphi_I = [0, 1]$), a negative reward weighting means that the IRL learnt objective function will prevent the process from reaching the new set point. This is the primarily attributed to the fact that a large proportion of the demonstrations never reached the new set point (Figures 4 and 5) due to the overdamped control response. As AL considers the expert's (i.e., PID controller) actions as a noisily optimal control policy, it will find the optimal solution of weight vector, $\boldsymbol{\alpha}^*$, to reproduce this overdamped control response. Therefore, the current result indicates that if the demonstration data does not contain a good control policy, it is essential to further improve the AL generated policy through online learning.

5.2.2 | Online learning and optimal control

As in Section 5.1.2, online learning is performed to improve the AL policy (initialized for RL). Given that a degree of offset was present in both control loops as detailed by Figure 4, two short rounds of RL policy improvement, again consisting of 10 training epochs, proceeded with hand tuning of the parameter β in each round. Figure 5 details the final results of the update RL model. From Figure 5, it is found that the improved policy of the agent π^A, observes a faster rise time, no overshoot and subjectively better set point tracking than the demonstrated policy (PID). In this way, the methodology shows ability to learn from suboptimal demonstrations and then efficiently improve the learned parameterization of the demonstrated policy through online learning (in this work, 24 min spent online to update the RL).

5.3 | Case Study III—Knowledge transfer in learning from demonstration

Finally, Case Study III demonstrates how knowledge transfer from one task improves the efficiency of offline AL for further set points. Here, we again assume the availability of existing demonstrations as described by process data. The control task (set point change) in this study is described by Table 1 and is different to both tasks examined in Case Studies I and II. Again, we would like to learn a parameterization of the control policy (offline) expressed in the process data and then improve it further (online), but we wish to reduce the computational budget associated with offline AL. Thus, we propose to transfer knowledge from a previous study to improve computational and learning efficiency.

Knowledge transfer is in the form of the offline learned policy parameterization, $\pi_{\text{po}}\left(\boldsymbol{\theta}_{(\text{k}_0)}\right)$ and weight vector, $\boldsymbol{\alpha}^*$, from a previous task. Here, knowledge is transferred from Case Study I, given its better PID performance than Case Study II. Both $\boldsymbol{\alpha}^*$ and $\pi_{\text{po}}\left(\boldsymbol{\theta}_{(\text{k}_0)}\right)$ from Case Study I are provided as initialization for AL of the new task in Case Study III. Update of this initialization only takes 80 epochs. Previously, the two studies recovered demonstrated behavior within a total of 300 and 250 epochs of policy optimization, respectively. This reduction in the computational intensity of policy learning demonstrates that the computational burden of AL via IRL—under the current methodology—may be significantly reduced through knowledge transfer. In this study, process data were generated using PID1. Table 4 details the corresponding trajectory feature expectations, $\boldsymbol{\upsilon}^{\gamma,E}$.

Given the parameterization as learned via IRL, a further two rounds of 10 epochs of RL enabled further policy improvement online. The results are presented in Figure 6. Figure 6(A,B) highlights how the policy learned under knowledge transfer achieves pure set point tracking with a smoother control policy than that demonstrated by PID1. Once again, Figure 6(C,D) shows that this control policy successfully facilitates a system response with fast rise time, but no overshoot or oscillatory behavior around the set point, as is present in the demonstrations.

6 | CONCLUSIONS

In this article, we propose a framework based on AL to learn a control law based on process data, this approach allows us to synthesize a neural network control policy from a previous controller (e.g., PID, MPC, or human controllers) more robustly than with supervised learning. Having learned a parameterization of the control law, subsequent deployment of RL enables further policy improvement by directly interacting with the real process, thus outperforming the existing control law. Here, AL is implemented through IRL. Given the data-driven nature of IRL, the RL-based policy parameterization promises to express the action of the control scheme and process knowledge of the operators. RL is constructed using a policy optimization algorithm, although other methods could be also applied in the future. Based on the case studies, it is concluded that the proposed framework can effectively extract control information from available process data, transfer knowledge between different cases, and can result in a better optimal control policy efficiently. It should be noted that we assume the availability of rich informative datasets. If the data is not informative, the framework is unlikely to be effective. Future work will explore implementation of various data augmentation strategies, based on physical knowledge or statistical analyses, to artificially synthesize informative datasets.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no experimental datasets were generated or analysed during the current study.

ORCID

Max Mowbray https://orcid.org/0000-0003-1398-0469
Dongda Zhang https://orcid.org/0000-0001-5956-4618

REFERENCES

1. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature*. 2015;518(7540):529-52933.

2. Heess N, Hunt JJ, Lillicrap TP, Silver D. Memory-based control with recurrent neural networks
3. Kirk DE. *Optimal Control Theory: An Introduction*. New York: Dover Publications; 1998.
4. Liu D, Wei Q, Wang D, Yang X, Li H. Overview of adaptive dynamic programming. *Adaptive Dynamic Programming with Applications in Optimal Control*. Basel: Springer International Publishing; 2017:1-33.
5. Petsagkourakis P, Sandoval IO, Bradford E, Zhang D, del Rio-Chanona EA. Reinforcement learning for batch bioprocess optimization. *Comput Chem Eng*. 2020;133:106649.
6. Lawrence NP, Stewart GE, Loewen PD, Forbes MG, Backstrom JU, Gopaluni RB. Optimal PID and antiwindup control design as a reinforcement learning problem. *arXiv:200504539 [cs, eess, math]*.
7. Spielberg S, Tulsyan A, Lawrence NP, Loewen PD, Bhushan GR. Toward self-driving processes: a deep reinforcement learning approach to control. *AIChE J*. 2019;65(10):e16689. https://doi.org/10.1002/aic.16689.
8. Kim JW, Park BJ, Yoo H, Oh TH, Lee JH, Lee JM. A model-based deep reinforcement learning method applied to finite-horizon optimal control of nonlinear control-affine system. *J Process Control*. 2020;87:166-178.
9. Kim Y, Lee JM. Model-based reinforcement learning for nonlinear optimal control with practical asymptotic stability guarantees. *AIChE J*. 2020;n/a(n/a):e16544. https://doi.org/10.1002/aic.16544.
10. Gottipati SK, Sattarov B, Niu S, et al. Learning to navigate the synthetically accessible chemical space using reinforcement learning. *arXiv:200412485 [cs]*.
11. Angermueller C, Dohan D, Belanger D, Deshpande R, Murphy K, Colwell L. Model-based reinforcement learning for biological sequence design. International Conference on Learning Representations.
12. Gros S, Zanon M. Data-driven economic NMPC using reinforcement learning. *arXiv:190404152 [cs]*.
13. Zanon M, Kungurtsev V, Gros S. Reinforcement learning based on real-time iteration NMPC. *arXiv:200505225 [cs, eess]*.
14. Schulman J, Levine S, Moritz P, Jordan MI, Abbeel P. Trust region policy optimization. *arXiv:150205477 [cs]*.
15. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. *arXiv:170706347 [cs]*.
16. Schulman J, Chen X, Abbeel P. Equivalence between policy gradients and soft Q-learning. *arXiv:170406440 [cs]*.
17. Lehman J, Chen J, Clune J, Stanley KO. ES is more than just a traditional finite-difference approximator
18. Sutton RS, McAllester D, Singh S, Mansour Y. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*. Neural information processing systems foundation; 2000:1057-1063.
19. Petsagkourakis P, Sandoval IO, Bradford E, Galvanin F, Zhang D, del Rio-Chanona EA. Chance constrained policy optimization for process control and optimization. *arXiv:200800030 [cs, eess]*.
20. Shin J, Badgwell TA, Liu K-H, Lee JH. Reinforcement learning—overview of recent progress and implications for process control. *Comput Chem Eng*. 2019;127:282-294. https://doi.org/10.1016/j.compchemeng.2019.05.029.
21. Beaulieu S, Frati L, Miconi T, et al. Learning to continually learn. *arXiv:200209571 [cs, stat]*.
22. Coates A, Abbeel P, Ng A. Apprenticeship learning for helicopter control. *Commun ACM*. 2009;52(7):97-105.
23. Wu Z, Sun L, Zhan W, Yang C, Tomizuka M. Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving. *IEEE Robot Autom Lett*. 2020;5(4):5355-5362.
24. Silver D, Bagnell JA, Stentz A. Learning from demonstration for autonomous navigation in complex unstructured terrain. *Int J Robot Res*. 2010;29(12):1565-1592.
25. Sutton RS. *Reinforcement Learning: An Introduction*. 2nd ed: The MIT Press; 2018.
26. Zhang K, Koppel A, Zhu H, Başar T. Global convergence of policy gradient methods to (almost) locally optimal policies. *arXiv:190608383 [cs, eess, math, stat]*.
27. Simmons-Edler R, Eisner B, Mitchell E, Seung S, Lee D. Q-learning for continuous actions with cross-entropy guided policies. *arXiv:190310605 [cs]*.
28. Azar NA, Shahmansoorian A, Davoudi M. From inverse optimal control to inverse reinforcement learning: a historical review. *Annu Rev Control*. 2020;50:119-138.
29. Abbeel P, Ng AY. Apprenticeship learning via inverse reinforcement learning. *Twenty-First International Conference on Machine Learning - ICML '04*. ACM Press; 2004:1. https://doi.org/10.1145/1015330.1015430.
30. Ziebart B, Maas A, Bagnell JA, Dey AK. Maximum entropy inverse reinforcement learning. In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*. AAAI'08. AAAI Press; 2008.
31. Wulfmeier M, Ondruska P, Posner I. Maximum entropy deep inverse reinforcement learning. *arXiv:150704888 [cs]*.
32. Karg B, Alamo T, Lucia S. Probabilistic performance validation of deep learning-based robust NMPC controllers. *arXiv:191013906 [cs, eess, math]*.
33. Rohani S. *Coulson and Richardson's Chemical Engineering. Volume 3B, Process Control*. 4th ed. Oxford: Butterworth-Heinemann; 2017.
34. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput*. 1997;9(8):1735-17380.
35. Colah C. Understanding LSTM Networks; 2015. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
36. Ziebart B. Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy.
37. Zhou Z, Bloem M, Bambos N. Infinite time horizon maximum causal entropy inverse reinforcement learning. *IEEE Trans Automat Contr*. 2018;63(9):2787-2802.
38. Mao X. The truncated Euler–Maruyama method for stochastic differential equations. *J Comput Appl Math*. 2015;290(C):370-384.
39. Jaynes ET. Information theory and statistical mechanics. *Phys Rev*. 1957;106(4):620-630. https://doi.org/10.1103/PhysRev.106.620.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of this article.

How to cite this article: Mowbray M, Smith R, Del Rio-Chanona EA, Zhang D. Using process data to generate an optimal control policy via apprenticeship and reinforcement learning. *AIChE J*. 2021;e17306. https://doi.org/10.1002/aic.17306