Optimizing Operation Recipes with Reinforcement Learning for Safe and Interpretable Control of Chemical Processes

Dean Brandner and Sergio Lucia

TU Dortmund University, Dortmund, Germany
{dean.brandner, sergio.lucia}@tu-dortmund.de

Abstract. Optimal operation of chemical processes is vital for energy, resource, and cost savings in chemical engineering. The problem of optimal operation can be tackled with reinforcement learning, but traditional reinforcement learning methods face challenges due to hard constraints related to quality and safety that must be strictly satisfied, and the large amount of required training data. Chemical processes often cannot provide sufficient experimental data, and while detailed dynamic models can be an alternative, their complexity makes it computationally intractable to generate the needed data. Optimal control methods, such as model predictive control, also struggle with the complexity of the underlying dynamic models. Consequently, many chemical processes rely on manually defined operation recipes combined with simple linear controllers, leading to suboptimal performance and limited flexibility.
In this work, we propose a novel approach that leverages expert knowledge embedded in operation recipes. By using reinforcement learning to optimize the parameters of these recipes and their underlying linear controllers, we achieve an optimized operation recipe. This method requires significantly less data, handles constraints more effectively, and is more interpretable than traditional reinforcement learning methods due to the structured nature of the recipes. We demonstrate the potential of our approach through simulation results of an industrial batch polymerization reactor, showing that it can approach the performance of optimal controllers while addressing the limitations of existing methods.

Keywords: Reinforcement Learning · Interpretable Machine Learning.

1 Introduction

The chemical industry is the largest industrial energy consumer and the third largest industrial emitter of CO2 after the steel and cement industries, making it necessary to achieve high efficiencies together with innovative technologies and recycling to enable achieve net zero emissions [13]. At the same time, chemical processes need to be very carefully operated so that strict quality, safety and regulatory requirements are fulfilled.

The optimal operation of chemical processes can be formulated as an optimal control problem or a Markov decision process (MDP) for which reinforcement learning (RL) has been recently explored [17,22]. However, traditional RL techniques struggle with the consideration of hard constraints and need a large amount of data. Unfortunately, in the field of chemical engineering, many hard constraints related to quality and safety requirements need to be strictly ensured [2] and obtaining a large amount of experimental data for training is typically not possible. The latter challenge can be alleviated by using detailed dynamic models of the chemical processes instead of interacting with the real plant itself but these models are often very complex, making it computationally infeasible to generate large amounts of data. Finally, chemical processes are typically still operated or supervised by humans, for which an interpretable operation strategy is beneficial.

A more established approach to perform advanced operation of chemical processes is the use of optimal control theory methods such as nonlinear model predictive control (NMPC) [19]. In this approach, a dynamic system model is used to obtain predictions and an optimal trajectory of control inputs is calculated by solving an optimization problem every time a new control input needs to be computed. NMPC has been successfully applied in many domains since it can directly deal with nonlinear multivariable systems with hard constraints. However, when the underlying dynamic models of the process are very complex, including for example partial differential equations, multi-phase systems or startup behavior of different unit operations, the resulting optimization problems are often intractable. While some approaches exist to alleviate this problem, such as tailored fast optimization solvers [25], the use of approximate MPC based on neural networks (NN) [6,14] or the combination of RL and NMPC [27,4], it remains challenging to solve the resulting optimization problems in real time.

As a result, even nowadays batch processes are mostly controlled in the following hierarchical fashion. In the upper layer, a reference trajectory of setpoints, called the operation recipe, is provided. These recipes can either be rigorously calculated, or derived by an expert via trial-and-error. The lower layer attempts tracking of the recipe references during execution of a batch run. Usually, simple linear PID controllers are used to track these references. Lots of research was put into optimizing these operation recipes in the past. However, the approaches either focus on bias correction of empirical models once a full batch is completed, such as in run-to-run control [5], or model-based trajectory optimization between or even during runs [15,7]. Although these model-based approaches lead to improvements, they require a detailed control oriented model. In practise, these kinds of models are often not available, inexact, or extremely difficult, if not impossible to use in model-based optimization. Further, model-free optimization approaches such as RL can also be applied to find optimized recipes. Different approaches, ranging from application of standard RL techniques for batch recipe optimization, to newly custom-made RL methods are reviewed in [26]. Still, the authors of [26] identify that data efficiency and constraint handling remain an issue for RL. Due to the practical inapplicability of these approaches, batch

processes are mostly controlled according to manually tuned operation recipes, which are the result of a combination of the experience of experts and heuristics [3,20]. The deployed reference trajectories are often constrained to ramps or constant holding signals, both applied until a certain condition is met. The recipe parameters such as the slope of the ramps or the constant value are usually only tuned by experts and not by rigorous optimization. Further, also the tracking PID controllers must be tuned according to the recipe parameters. All this clearly leads to a significant suboptimal performance of batch processes.

In this work, we propose a new method to incorporate the expert knowledge embedded in operation recipes and combine it with the capabilities of RL when used with detailed dynamic models of complex chemical processes. We use a RL agent to optimally tune the parameters of the operation recipes as well as the parameters of the underlying linear controllers. The goal is to significantly increase the performance of operation recipes, approaching the optimal control solution which typically cannot be computed in real time. Since the amount of deployed actions, which take the form of recipe and PID parameters, to run a full batch is small compared to traditional direct RL techniques, we argue that it is significantly easier to train and also easier to obtain a policy that satisfies hard constrains. In addition, the resulting strategy is easily interpretable, as it retains the structure of operation recipes and linear controllers that is typical in chemical engineering. We showcase the potential of the approach with simulation results of an industrial semi-batch polymerization reactor. This example can serve as a benchmark from chemical engineering for other methodologies, as it is a challenging system with strongly nonlinear dynamics, multiple inputs and several hard constraints for which traditional RL techniques struggle to find a suitable policy.

2 Background

2.1 Reinforcement Learning

RL aims at solving MDPs [24]. An MDP is composed of an agent and an environment. At each time instance, the environment is in state $\boldsymbol{s} \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$ and receives an action $\boldsymbol{a} \in \mathcal{A} \subseteq \mathbb{R}^{n_a}$ that is calculated according to the agent's policy π. The sets $\mathcal{S}$ and $\mathcal{A}$ denote the sets of possible states and actions. The policy can either be stochastic $\boldsymbol{a} \sim \pi(\cdot|\boldsymbol{s})$, so a mapping from a state to a probability distribution over the action space, or deterministic $\boldsymbol{a} = \pi(\boldsymbol{s})$, so a direct mapping from a state to a specific action. For ease of notation, we will focus on deterministic policies for the rest of this work. However, all presented concepts also work with stochastic policies. When action $\boldsymbol{a}$ is applied to the environment, the environment transitions from the current state $\boldsymbol{s}$ to the subsequent state $\boldsymbol{s}^+ \in \mathcal{S}$ according to its underlying transition probability $p(\boldsymbol{s}^+|\boldsymbol{s}, \boldsymbol{a})$, leading to:

$$\boldsymbol{s}^+ \sim p(\cdot|\boldsymbol{s}, \boldsymbol{a}). \tag{1}$$

Often times, the transition from $\langle \boldsymbol{s}, \boldsymbol{a} \rangle$ to $\boldsymbol{s}^+$ can also be expressed as a dynamic system model $f : \mathcal{S} \times \mathcal{A} \times \mathbb{W} \to \mathcal{S}$ in which the uncertainty of (1) is accounted

for via the random variable $\mathbf{w} \sim \mathbb{W}$ leading to:

$$\boldsymbol{s}^+ = f(\boldsymbol{s}, \boldsymbol{a}, \mathbf{w}). \tag{2}$$

In the case of a deterministic environment, the random variable is always zero $\mathbf{w} = 0$ and the transition probability p becomes the Dirac impulse. After the environment transitions one step, the agent receives the next state $\boldsymbol{s}^+$ and a scalar reward $r \in \mathbb{R}$, which measures how good the state-action-pair $\langle \boldsymbol{s}, \boldsymbol{a} \rangle$ is according to a previously designed objective. After that, the cycle of providing an action $\boldsymbol{a}$ given state $\boldsymbol{s}$ and observing the subsequent state $\boldsymbol{s}^+$ and reward r is repeated. The overall goal of RL is to find the optimal policy $\pi^\star$ that maximizes the expected cumulative reward $J(\pi) \in \mathbb{R}$ based on the interaction of the agent with the environment only, by collecting the transition to the subsequent states $\boldsymbol{s}^+$ and rewards $r(\boldsymbol{s}, \boldsymbol{a})$ when being in state $\boldsymbol{s}$ and taking action $\boldsymbol{a}$. Once the optimal policy $\pi^\star$ is found, the MDP is assumed to be solved. To calculate the expected cumulative reward $J(\pi)$, we introduce the state value function $V^\pi : \mathcal{S} \to \mathbb{R}$. The state value function is recursively defined in (3) as the sum of the immediate reward $r(\boldsymbol{s}, \boldsymbol{a})$ when being at state $\boldsymbol{s}$ and taking action $\boldsymbol{a}$ according to policy π, and the expected value of the state value $V^\pi(\boldsymbol{s}^+)$ over all possible subsequent states $\boldsymbol{s}^+$ according to the transition probability p, that is:

$$V^\pi(\boldsymbol{s}) = r(\boldsymbol{s}, \boldsymbol{a})|_{\boldsymbol{a}=\pi(\boldsymbol{s})} + \gamma \mathbb{E}_{\boldsymbol{s}^+}[V^\pi(\boldsymbol{s}^+)]. \tag{3}$$

where $0 < \gamma \leq 1$ is a discount factor. By taking the expected value over the initial states $\boldsymbol{s}_0$, the expected cumulative reward $J(\pi)$ can be derived as

$$J(\pi) = \mathbb{E}_{\boldsymbol{s}_0}[V^\pi(\boldsymbol{s})]. \tag{4}$$

A policy is optimal when it maximizes the expected cumulative reward

$$\pi^\star = \arg\max_\pi J(\pi). \tag{5}$$

However, the maximization problem in (5) requires to solve an infinite dimensional optimization problem which is intractable in general. The most common solution to this problem is to approximate the optimal policy $\pi_{\boldsymbol{\theta}^\star}(\boldsymbol{s}) \approx \pi^\star$ with a function approximator with parameters $\boldsymbol{\theta} \in \mathbb{R}^{n_{\boldsymbol{\theta}}}$. Due to its universal approximation capabilities [12], often NNs are used. The maximization problem boils down to finding the optimal parameters $\boldsymbol{\theta}^\star$

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}). \tag{6}$$

RL gives a rich toolbox of algorithms to solve (6). Among others, common approaches try to find the optimal policy by directly updating the policy parameters $\boldsymbol{\theta}$ with the policy gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \in \mathbb{R}^{n_{\boldsymbol{\theta}}}$ according to the gradient-ascent optimization algorithm

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}). \tag{7}$$

Since the calculation of the policy gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ from the interaction of the agent with the environment is not straightforward, many algorithms exist that address this task. For deterministic policies, state-of-the-art performance can be achieved among others with the twin delayed deep deterministic policy gradient (TD3) algorithm [9] while proximal policy optimization (PPO) [21] and soft-actor-critic (SAC) [11] are state-of-the-art for stochastic polices.

Although serious advances were made in recent years, the sample efficiency of RL algorithms is still poor and scales badly with increasingly complex tasks, as well as with the state and action space dimension. This is especially the case for safe RL. An extensive overview is given in [10]. In the context of process control, the consideration of hard constraints remains a major challenge. Although constrained RL methods exist, they are mostly accounted for via penalty terms in the reward function (that is, as soft constraints) in practice.

2.2 Standard and Advanced Control Approaches

Linear PID Control The majority of chemical processes is operated in steady state or at least a pseudo steady state, which is equivalent to having only slowly changing global process dynamics. Since often the nonlinear system dynamics can be approximated sufficiently with a linear model close around a steady state, and also due to its easy practical implementation, most controllers in chemical industry are proportional-integral-differential (PID) controllers [23].

We want to highlight the differences between the RL state $\boldsymbol{s}$ and RL action $\boldsymbol{a}$, and their physical counterparts. Therefore, we introduce the physical dynamic system state $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{n_{\boldsymbol{x}}}$ and the physical control input $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^{n_{\boldsymbol{u}}}$. The discretized dynamics of the physical system are described by the potentially nonlinear model $f_{\mathrm{p,d}} : \mathcal{X} \times \mathcal{U} \times \mathbb{W} \to \mathcal{X}$ that can also be stochastic, accounted for by the random variable $\mathbf{w}$

$$\boldsymbol{x}_{k+1} = f_{\mathrm{p,d}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \mathbf{w}_k) \tag{8}$$

with k being the current sampling time. We want to emphasize that the RL state and action can in fact be the physical state and control input, so $\boldsymbol{s} = \boldsymbol{x}$ and $\boldsymbol{a} = \boldsymbol{u}$, but the RL state and RL action can also augment further effects such as states from past time steps $\boldsymbol{x}_{k-1}$ or past control actions $\boldsymbol{u}_{\mathrm{prev}}$.

Given a desired steady state $\boldsymbol{x}_{\mathrm{ss}}$ and $\boldsymbol{u}_{\mathrm{ss}}$ of (8), PID controllers try to minimize the error $\boldsymbol{e} \in \mathbb{R}^{n_{\boldsymbol{x}}}$ between the state $\boldsymbol{x}$ and the desired steady state $\boldsymbol{x}_{\mathrm{ss}}$

$$\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_{\mathrm{ss}}. \tag{9}$$

To achieve this, the applied control input $\boldsymbol{u}$ is calculated according to

$$\boldsymbol{u}_k = \boldsymbol{u}_{\mathrm{ss}} + K_{\mathrm{P}}\, \boldsymbol{e}_k + K_{\mathrm{I}} \sum \boldsymbol{e}_k\, \Delta t + K_{\mathrm{D}}\, \dot{\boldsymbol{e}}_k. \tag{10}$$

The control input $\boldsymbol{u}_k$ is therefore determined by adapting the steady state input $\boldsymbol{u}_{\mathrm{ss}}$ with three different correction terms. The displayed correction terms reflect a correction due to the immediate error $\boldsymbol{e}_k$, the integrated error $\sum \boldsymbol{e}_k\, \Delta t$

and the differentiated error $\dot{\boldsymbol{e}}_k$. Each correction term is multiplied by a controller gain K_i, which must be tuned by an expert to achieve the desired performance. Most chemical processes are controlled with PI controllers only, which is done by setting $K_\mathrm{D} = 0$. The full PID parameter vector $\boldsymbol{\Theta}_\mathrm{PID} \in \mathcal{T}_\mathrm{PID}$ includes all parameters that influence the controller performance, such as the setpoints $\boldsymbol{x}_\mathrm{ss}$ and the controller gains K_i.

Cascade control is a strategy that utilizes an inner and an outer control loop to control a system with two subsystems exhibiting differently fast dynamics. The outer, slower control loop determines the setpoint for the inner, faster control loop, which then rapidly adjusts to follow this setpoint set by the outer loop [23]. Cascade control is frequently used in the chemical industry, such as in controlling the reactor temperature in chemical reactors. In this scenario, the outer loop controls the reactor temperature by providing setpoints for the jacket temperature, which is controlled by the inner loop.

Although PID controllers are widely used in chemical industry, they have limitations when applied to highly nonlinear systems without a steady state in the desired operating range. The theory behind PID controllers assumes linear dynamics, making it difficult to handle high nonlinearity. Additionally, process constraints, like maximum or minimum reactor temperatures, can only be addressed indirectly through proper controller tuning. Online consideration of these constraints is not possible. The performance of PID controllers heavily depends on their tuning. Poorly tuned controllers result in unsatisfactory control performance. Finding reasonable controller gains can be a cumbersome task.

Nonlinear Model Predictive Control NMPC is an advanced control approach that is used frequently in chemical engineering, and addresses the shortcomings of linear PID control such as rigorous constraint consideration and optimized performance. To account for both, NMPC solves the following optimal control problem each sampling time [19]:

$$\mathbf{u}^\star = \arg\min_{\mathbf{u}} \quad F_\mathrm{f}(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} \ell(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{11a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = f_\mathrm{p,d}(\boldsymbol{x}_k, \boldsymbol{u}_k), \quad \boldsymbol{x}_0 = \boldsymbol{x}(t_k), \tag{11b}$$

$$g(\boldsymbol{x}_k, \boldsymbol{u}_k) \leq 0. \tag{11c}$$

Within this optimization problem, the system states $\boldsymbol{x}_k$ are internally simulated according to (11b) for a prediction horizon of N steps starting from the most recently measured system state $\boldsymbol{x}(t_k)$. On this considered prediction horizon, process constraints $g : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_g}$ in the form of (11c) are evaluated at each sampling instance. The performance is then optimized by finding the optimal control input sequence $\mathbf{u}^\star = [\boldsymbol{u}_0^\star, \ldots, \boldsymbol{u}_{N-1}^\star]^\top$ according to the objective function (11a), which includes the stage cost $\ell(\boldsymbol{x}_k, \boldsymbol{u}_k)$, and the terminal cost $F_\mathrm{f}(\boldsymbol{x}_N)$, which compensates the truncation errors due to a finite prediction horizon. The first element $\boldsymbol{u}_0^\star$ of the optimal control sequence $\mathbf{u}^\star$ is then applied to the system. Note that in classical control theory, typically minimization problems are solved,

while RL aims at maximizing the reward. Thus, the stage cost in NMPC can be interpreted as the negative reward in RL.

Despite alleviating many problems from linear PID control and performing optimally on constrained systems, the performance of NMPC can significantly deteriorate with model errors. To avoid model errors, very detailed models can be formulated, which in turn can lead to very complex optimization problems due to highly nonlinear dynamics or a large amount of optimization variables preventing its solution in real-time.

2.3 Heuristics and Operation Recipes

In control of chemical systems, the operation of batch processes is a major challenge, despite its omnipresence in the production of pharmaceuticals and special chemicals. Advanced model-based optimal control approaches can often not be applied due to the high model complexity and the potential loss of real-time applicability. Batch operation is often done by a hierarchical approach, which is composed of an upper recipe layer, which provides setpoints for the lower tracking layer with mostly simple linear PID controllers. The reference trajectories of the recipe layer can be calculated by model-based optimization, provided that a good model exists that can be used for optimization. Since this is rarely the case due to high complexity, optimization cannot be performed. Hence, these recipes are designed by experts and are therefore mostly constrained to simple patterns.

In this contribution, we will align the definition of an operation recipe, which is tailored by expert knowledge, to the definition used in [3]. An operation recipe is a sequential procedure that separates the whole batch cycle into smaller batch phases $z = 1, \dots, n_z$. As an illustrative example, one can consider a tank that is filled until a certain level is reached in the first phase $z = 1$, operated at the desired level in the second phase $z = 2$, and emptied in the third phase $z = 3$. These phases themselves are also made up out of smaller sub-steps $c = 1, \dots, n_c$, which are the decisions that are applied to the system. In general, batch phase z can only be completed if the final step c_z of the phase z is reached. The procedure in each sub-step c is determined by a qualitative decision, like setting some value of an actuator or waiting until a certain condition is met. The values that are assigned by these manual adaptations are part of the recipe parameters $\boldsymbol{\Theta}_\mathrm{R} \in \mathcal{T}_\mathrm{R}$. The total set of the parameters for the whole batch cycle are thus the combination of the recipe and PID parameters $\boldsymbol{\Theta}^\top = [\boldsymbol{\Theta}_\mathrm{R}^\top, \boldsymbol{\Theta}_\mathrm{PID}^\top] \in \mathcal{T} = \mathcal{T}_\mathrm{R} \times \mathcal{T}_\mathrm{PID}$. The quantitative value that is set to the qualitative operation in step c can then be read from the c-th element of the full parameter vector $\boldsymbol{\Theta}_c \in \mathcal{T}_c \subseteq \mathbb{R}$. In classic operation recipes, these values are either set once or can be adapted by an expert according to the current plant situation. Table 1 summarizes the concept of operation recipes at the example of the considered tank above.

While this approach has the advantage that the decisions are made by an expert and that the operation recipe is highly interpretable, this approach leads typically to conservative results and is sensitive to PID controller tuning.

Table 1: Example recipe for filling and emptying a tank.

Phase z	Step c	Type	Description
1	1	set	Start the feed pump to a feed rate of $\boldsymbol{\Theta}_1$
1	2	condition	Wait until the level reaches $\boldsymbol{\Theta}_2$
2	3	set	Stop the feed pump and wait for a time of $\boldsymbol{\Theta}_3$
3	4	set	Open the outlet valve to a percentage of $\boldsymbol{\Theta}_4$
3	5	condition	Wait until the level falls below $\boldsymbol{\Theta}_5$

3 Proposed Approach: Recipe-based Reinforcement Learning

In our proposed approach, we train an RL agent with a NN policy that receives the current RL state of the system and computes the next optimal recipe and PID parameters. Contrary to the classical implementation of expert-tuned operation recipes and PID controllers, in which the parameters are typically fixed once and not adapted afterwards, our approach delivers the optimal parameters depending on the current physical system state allowing an optimized control performance. While classical RL aims at finding the optimal control policy directly, which should in theory result in a similar performance as NMPC, it often struggles to find a reasonable policy when considering complex systems with hard constraints. The same problem also occurs when the optimized trajectory is calculated in advance and tracked as in the hierarchical batch operation setting. In addition, the resulting policy is usually not interpretable because the computed control action does not give any information about future decisions, and also process constraints are not considered explicitly as in NMPC. In contrast, our approach adapts the parameters of a fixed operation recipe structure and the parameters of the PID controllers. This results in a highly structured policy so even in the beginning of the RL training process, the resulting policies have an acceptable performance, leading to improved learning behavior. Also, both the operation recipe and the PID controllers were originally designed by experts and should therefore be operationally safe within a certain expert-certified parameter space. Lastly, since the parameters are part of the operation recipe, the derived policy is easy to interpret. Figure 1 summarizes all presented control approaches (left column) and contrasts them with our proposed approach (right column).

In the classical RL setting (see Figure 1a), the RL state $\boldsymbol{s}_{\mathrm{cl}}$ and the RL action $\boldsymbol{a}_{\mathrm{cl}}$ are usually the physical state $\boldsymbol{s}_{\mathrm{cl}} = \boldsymbol{x}$ and the physical control input $\boldsymbol{a}_{\mathrm{cl}} = \boldsymbol{u}$, or at least closely related to it. The agent therefore learns the optimal policy $\boldsymbol{a}_{\mathrm{cl}} = \pi_{\boldsymbol{\theta}}(\boldsymbol{s}_{\mathrm{cl}})$ directly. The underlying environment dynamics are governed by (8). The reward r_{cl} typically depends on a single transition of the physical system only. Based on this information, the optimal policy is to be derived.

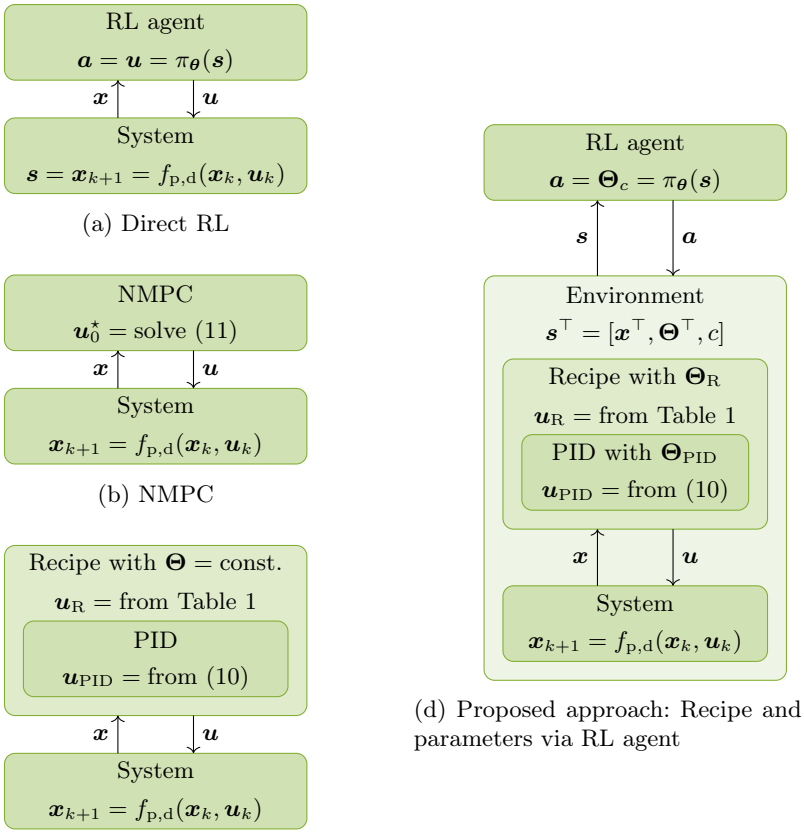
We propose a different design of the environment, specialized for learning recipe and PID parameters. Instead of only the physical system, the environ-

ment consists also of the operation recipe and PID structure. We define the RL state $\boldsymbol{s}_\mathrm{R} \in \mathcal{X} \times \mathcal{T} \times \mathbb{N}$ as the combination of the physical state $\boldsymbol{x}$, the recipe and PID parameters $\boldsymbol{\Theta}$ and the current recipe step c

$$\boldsymbol{s}_\mathrm{R}^\top = [\boldsymbol{x}^\top, \boldsymbol{\Theta}^\top, c]. \tag{12}$$

We will assume that the parameter vector $\boldsymbol{\Theta}$ is ordered, meaning that the c-th element in the parameter vector corresponds to the c-th recipe step. The RL action $\boldsymbol{a}_\mathrm{R}$ reduces from the full physical control input $\boldsymbol{u}$ to the c-th parameter $\boldsymbol{a}_\mathrm{R} = \boldsymbol{\Theta}_c$ that is required in the c-th recipe step. The agent therefore learns to predict the best recipe or PID parameter with its policy $\boldsymbol{a}_\mathrm{R} = \pi_{\boldsymbol{\theta}}(\boldsymbol{s}_\mathrm{R})$. The dynamics of the RL state $\boldsymbol{s}_\mathrm{R}$ now have to be adapted. Since in each interaction of the agent with the environment, the parameter vector $\boldsymbol{\Theta}$ changes due to the applied action $\boldsymbol{a}_\mathrm{R} = \boldsymbol{\Theta}_c$, the dynamics of the parameter vector $\boldsymbol{\Theta}$ are given as

$$\boldsymbol{\Theta}^+ = \boldsymbol{\Theta} + \boldsymbol{i}_c \, \boldsymbol{\Theta}_c \tag{13}$$



(a) Direct RL

(b) NMPC

(c) Recipes and PID ($\boldsymbol{\Theta} =$ const.)

(d) Proposed approach: Recipe and PID parameters via RL agent

Fig. 1: Established control approaches (left) vs. proposed approach (right).

with $\boldsymbol{i}_c$ being the standard unit vector in the c-direction. Further, as the c-th parameter is set, the step counter c must be increased

$$c^+ = c + 1. \tag{14}$$

Since the physical system now either does not transition at all, which is the case when not all parameters of a certain batch phase z are set ($c \neq c_z$), or the physical system transitions through a whole batch phase z, which is the case when all parameters in batch phase z are set ($c = c_z$), the transition dynamics must be adapted accordingly. We introduce the dynamic system $\hat{f}_{\mathrm{p,d}}$, which models the transition through a whole recipe step z. This is equivalent to sequentially evaluating the dynamic system (8) with the control actions $\boldsymbol{u}$ according to recipe phase z until the criterion to switch from phase z to $z+1$ is met. The overall transition dynamics of the physical system can be summarized as

$$\boldsymbol{x}^+ = \begin{cases} \boldsymbol{x} & \text{if } c \neq c_z, \\ \hat{f}_{\mathrm{p,d}}(\boldsymbol{x}, \boldsymbol{u}) & \text{else.} \end{cases} \tag{15}$$

Finally, also the reward r_{R} must be adapted accordingly. If only the parameters are changed according to (13), the physical system does not change and the reward is always zero for those transitions. However, if a full batch phase is carried out ($c = c_z$), the classical reward r_{cl} can be evaluated at each time instance until the end of the batch phase after $n_{\mathrm{end}}(\boldsymbol{s}_{\mathrm{R}})$ transitions. The resulting reward r_{R} is then summed together and weighted with the discretization time step Δt. The calculation can be summarized as

$$r_{\mathrm{R}}(\boldsymbol{s}_{\mathrm{R}}, \boldsymbol{a}_{\mathrm{R}}) = \begin{cases} 0 & \text{if } c \neq c_z, \\ \displaystyle\sum_{i=k}^{n_{\mathrm{end}}(\boldsymbol{s}_{\mathrm{R}})} r_{\mathrm{cl}}(\boldsymbol{x}_i, \boldsymbol{u}_i)\, \Delta t & \text{else.} \end{cases} \tag{16}$$

4 Experiments

We investigate the proposed approach with the example of a semi-batch polymerization reactor [16]. Figure 2 shows a sketch of the reactor. The reactor content consists three components: water, monomer and product. Their masses are given as m_{W}, m_{M} and m_{P} respectively. The reactor content with a temperature T_{R} is in direct contact with the walls of temperature T_{S}. The reactor contains a jacket with a temperature T_{J}, and an external heat exchanger with temperature T_{EHE}. Both can be used for heating and cooling of the reactor content. Further, the temperature of the water on the cooling side of the external heat exchanger is given as $T_{\mathrm{CW,EHE}}$. The reactor can be filled via the feed stream $\dot{m}_{\mathrm{feed}}$ that consists of monomer and water. The jacket temperature T_{J} and the temperature of the external heat exchanger T_{EHE} can be controlled by the inlet water temperature to these devices, which are $T_{\mathrm{J,in}}$ and $T_{\mathrm{CW,EHE,in}}$. Lastly, although being no real physical states, we also model the accumulated feed mass m_{acc} and

the adiabatic temperature T_{ad} as for both process constraints exist. Hence, the resulting states $\boldsymbol{x}$ and control inputs $\boldsymbol{u}$ are

$$\boldsymbol{x} = [m_{\mathrm{W}}, m_{\mathrm{M}}, m_{\mathrm{P}}, T_{\mathrm{R}}, T_{\mathrm{S}}, T_{\mathrm{J}}, T_{\mathrm{EHE}}, T_{\mathrm{CW,EHE}}, m_{\mathrm{acc}}, T_{\mathrm{ad}}]^{\top}, \tag{17}$$

$$\boldsymbol{u} = [\dot{m}_{\mathrm{feed}}, T_{\mathrm{J,in}}, T_{\mathrm{CW,EHE,in}}]^{\top}. \tag{18}$$

The governing equations and parameter values can be found in [16] and online[1].

A classical recipe-based control approach, which is inspired from the closed-loop trajectory of economic NMPC, is shown in Table 2. The whole batch process can be divided into $n_z = 3$ batch phases with 14 parameters $\boldsymbol{\Theta}$ in total. In the first two phases, the feed stream $\dot{m}_{\mathrm{feed}}$ is ramped up until a certain value is reached. After that, the feed stream $\dot{m}_{\mathrm{feed}}$ is kept at a constant rate until the whole batch cycle terminates. During all three batch phases, the reactor temperature T_{R} is controlled via PID controllers in a cascade control structure. The outer slower controller tracks the reference temperature $T_{\mathrm{R,ref}}$ by calculating setpoints for the jacket temperature $T_{\mathrm{J,ref}}$ and the temperature of the external heat exchanger $T_{\mathrm{EHE,set}}$, which are both each controlled by a much faster inner controller. All differential gains K_{D} of the PID controllers are set to zero.

Optimal control of the polymerization reactor usually has the goal to produce most product in the shortest amount of time, while satisfying process constraints. Often, also a smooth control trajectory is preferred to avoid damage to the actuators. Since the considered polymerization reaction always results in full conversion of the monomer to the product, the batch cycle is assumed to be finished when 99 % of all possible reactable monomer is reacted. When setting up an optimization problems for this batch cycle, it turns out that maximizing

[1] `www.do-mpc.com`

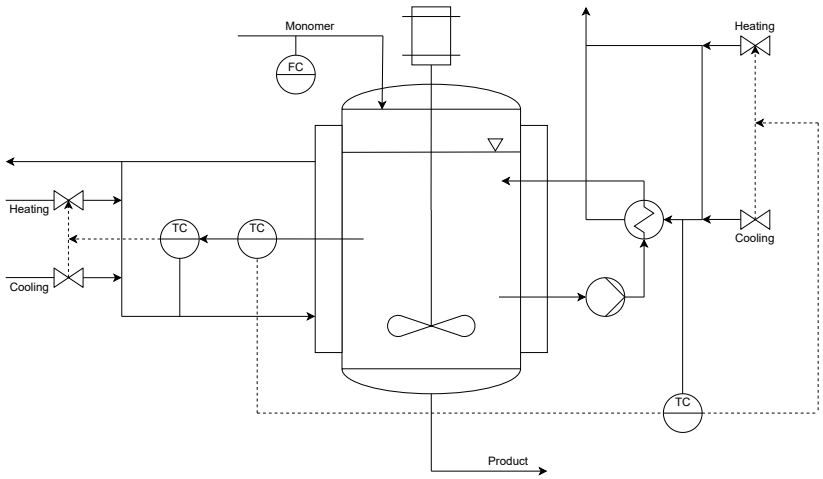


Fig. 2: Sketch of the polymerization reactor.

Table 2: Parameterized operation recipe of the polymerization reactor.

Phase z	Step c	Type	Description
1	1	set	Set slope of feed stream ramp to $\mathbf{\Theta}_1$
1	2	set	Set $T_{\mathrm{R,set}}$ of outer PID controller to $\mathbf{\Theta}_2$
1	3	set	Set K_{P} of outer PID controller to $\mathbf{\Theta}_3$
1	4	set	Set K_{I} of outer PID controller to $\mathbf{\Theta}_4$
1	5	condition	Run phase 1 until total mass reaches $\mathbf{\Theta}_5$
2	6	set	Set slope of feed stream ramp to $\mathbf{\Theta}_6$
2	7	set	Set $T_{\mathrm{R,set}}$ of outer PID controller to $\mathbf{\Theta}_7$
2	8	set	Set K_{P} of outer PID controller to $\mathbf{\Theta}_8$
2	9	set	Set K_{I} of the outer PID controller to $\mathbf{\Theta}_9$
2	10	set	Set maximal allowed value of feed stream to $\mathbf{\Theta}_{10}$
2	11	condition	Run phase 2 until elapsed times is larger than $\mathbf{\Theta}_{11}$ or feed stream becomes larger than $\mathbf{\Theta}_{10}$
3	12	set	Close feed and set $T_{\mathrm{R,set}}$ of outer PID controller to $\mathbf{\Theta}_{12}$
3	13	set	Set K_{P} of outer PID controller to $\mathbf{\Theta}_{13}$
3	14	set	Set K_{I} of outer PID controller to $\mathbf{\Theta}_{14}$

product mass and minimizing the batch time result in the same solution. Solving time-optimal control problems is challenging in general, which is the reason why mostly product maximization is performed in practise. However, RL can theoretically deal with both objectives. We investigate our approach on three different learning scenarios

1. Maximize product mass m_{P},
2. Minimize batch time t_{batch},
3. Minimize batch time t_{batch} and maximize product mass m_{P} (hybrid).

The designed reward therefore encodes the objectives above. In addition to that, all rewards also encode that the optimal control policy shall not violate constraints and should be reasonable smooth. For this, constraint violations are penalized with a high cost, and a reasonably smooth policy is penalized by taking large steps in the physical control input. As it is a special case for operation recipes, in which PID controllers are tuned, all rewards also penalize error between the reactor temperature T_{R} and the setpoint $T_{\mathrm{R,set}}$.

For all three scenarios, a hyperparameter gridsearch with all possible 96 combinations from Table 3 is carried out. The policy and Q-function are approximated with feedforward NNs with ReLU activation functions. All other hyperparameters remain at the default values of the stable-baselines3 [18] implementation. We observe that almost all agents converge to a good or at least reasonable policy. Still, there is room for improvement. Figure 3 show the learning curves for the best agents trained for each scenario. The agents learn rapidly in the beginning and start to converge after $40 \cdot 10^3$ iterations at the latest and only improve marginally afterwards. Fastest convergence is achieved by the hybrid reward scenario, which is expected as it carries most externally provided

Table 3: Considered hyperparameters for gridsearch.

Hyperparameter	Range	Hyperparameter	Range
RL Algorithm	SAC, TD3	Policy architecture	[50, 50], [50, 25, 10]
Batch size	512, 4 096	Learning rate	$3 \cdot 10^{-4}$, 10^{-5}
Exploration noise	0, 0.1	Buffer size	10^6, 10^5, 10^4

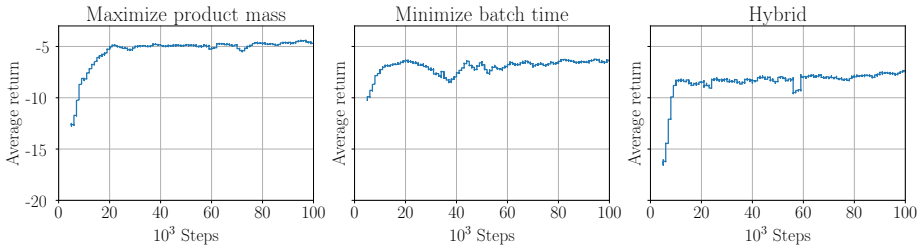


Fig. 3: Learning curves of the RL agents for all three different scenarios.

extra information in the form of two non-conflicting objectives. Note that the absolute value of the return does not provide information on the policy quality as it encodes different information.

All trained agents are evaluated with respect to their common performance metrics, which are the average batch time $\bar{t}_{\text{batch}}$ and the averaged absolute and relative number of constraint violations $\bar{n}_{\text{CV}}$ and $\bar{n}_{\text{CV,rel}}$. To evaluate the average batch time $\bar{t}_{\text{batch}}$, the agents control the system from 50 initial conditions, which were sampled from a different seed than the training was performed on. The metrics for all initial conditions are measured and averaged. We compare the performance of the three agents to NMPC (see Figure 1b), which uses the exact system model and a large prediction horizon of $N = 30$ with a discretization interval of 30 s. This NMPC deals as an estimate of the optimal policy and provides a benchmark performance. Furthermore, we compare the agents to a non-adaptive recipe with reasonable fixed parameters (see Figure 1c). This fixed recipe deals as a baseline. Lastly, we also try to compare our method to direct RL (see Figure 1a) to evaluate our method at a reference. Like for the recipe RL agents, we trained the direct RL agents for all three scenarios and performed a hyperparameter gridsearch, considering all possible combinations from Table 3. From all 288 investigated agents, only 58 terminated all 50 batches, while the remaining agents did not achieve 99 % conversion within five hours and were truncated consequently. From the remaining 58 agents, only two agents violated constraints in less than 5 % of all observed states. The agent with lowest percentage of constraint violations is considered the best and used as a comparison.

The results are illustrated in Table 4. As expected, the NMPC delivers the shortest average batch time and can therefore be referenced as the benchmark. The manually tuned baseline recipe results in an average performance. Since

Table 4: Performance evaluation of the different approaches and scenarios.

Method	$\bar{t}_{\mathrm{batch}}/[\mathrm{h}]$	$\bar{n}_{\mathrm{CV}}/[-]$	$\bar{n}_{\mathrm{CV,rel}}/[\%]$
Baseline (Recipe and PID)	3.29 ± 0.07	0.14 ± 0.49	0.03 ± 0.12
Benchmark (NMPC)	1.37 ± 0.04	0	0
Reference (Direct RL)	3.42 ± 0.04	6.32 ± 1.97	1.54 ± 0.52
Scenario 1 (Maximize m_{P})	2.28 ± 0.10	0	0
Scenario 2 (Minimize t_{batch})	2.25 ± 0.05	0	0
Scenario 3 (Hybrid)	2.21 ±0.06	0	0

all direct RL approaches struggled during training due to stability and overall convergence, even the best obtained agent has a larger average batch time than the baseline recipe. Further, more constraint violations than the baseline can be observed. This illustrates that despite serious tuning effort, tuning of the environment and the RL algorithm can be a cumbersome task. On the other hand, from all three investigated recipe-based training scenarios, scenario 3 (hybrid) shows the best control performance, which is likely as the most expert knowledge is put in the design of the reward. This is congruent with the best learning speed as shown in Figure 3. Still, all scenarios resulted with a final policy that outperforms both, the baseline recipe and direct RL, and appears to be similar in all cases. Also, all investigated scenarios showed a good learning performance and had no stability issues as in direct RL. We argue that this behavior originates from to the structured recipe environment. All resulting operation recipes are in average more than 1 h faster than the manually tuned baseline recipe. Further, all recipe-based RL agents do not violate constraints on the investigated batches, while the direct RL agent violates the constraints in 1.54 % of all states. This also highlights that constraining the RL agent to the structure of operation recipes can improve the overall safety of the RL agent.

The code with all experiments is available online[2]. All RL training runs are performed with the toolbox stable-baselines3 [18] and CasADi [1]. The results for NMPC are obtained with the toolbox do-mpc [8].

5 Conclusion

This paper proposes a novel approach for the optimal operation of chemical processes by integrating reinforcement learning with expert knowledge encapsulated in structured operation recipes. This method addresses key challenges in chemical engineering, such as handling hard constraints related to quality and safety, and the limited availability of experimental data for training. By optimizing the recipe and controller parameters, our approach achieves near-optimal performance with significantly less data and improved constraint handling.

[2] `https://github.com/DeanBrandner/Optimizing_Recipes_with_RL.git`

The simulation results of an industrial batch polymerization reactor illustrate the effectiveness of our method. Compared to traditional reinforcement learning and model predictive control, our approach offers enhanced interpretability, leveraging the structured knowledge of operation recipes.

Future work will focus on extending this methodology to integrate further expert knowledge via reinforcement learning with human feedback. Additionally, real-world implementation and validation of our approach will be performed to confirm its practical viability and benefits in industrial settings. Also, larger case studies will be investigated to assess the scalability of the proposed approach.

Acknowledgments. This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 466380688 – within the Priority Program "SPP 2331: Machine Learning in Chemical Engineering".

References

1. 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 **11**(1), 1–36 (2019)
2. Arendt, J.S., Lorenzo, D.K.: Evaluating Process Safety in the Chemical Industry: A User's Guide to Quantitative Risk Analysis. A CCPS Concept Book, American Chemistry Council ; Center for Chemical Process Safety, Arlington, Va. : New York (2000)
3. Brand Rihm, G., Esche, E., Repke, J.U.: Efficient dynamic sampling of batch processes through operation recipes. Computers & Chemical Engineering **179**, 108433 (2023)
4. Brandner, D., Lucia, S.: Reinforced model predictive control via trust-region quasi-newton policy optimization. In: 2024 European Control Conference (ECC). pp. 2299–2305 (2024)
5. Campbell, W., Firth, S., Toprac, A., Edgar, T.: A comparison of run-to-run control algorithms. In: Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301). pp. 2150–2155 vol.3. IEEE, Anchorage, AK, USA (2002)
6. Chen, S., Saulnier, K., Atanasov, N., Lee, D.D., Kumar, V., Pappas, G.J., Morari, M.: Approximating explicit model predictive control using constrained neural networks. In: 2018 Annual American control conference (ACC). pp. 1520–1527. IEEE (2018)
7. Chen, T., Morris, J., Martin, E.: Particle filters for state and parameter estimation in batch processes. Journal of Process Control **15**(6), 665–673 (Sep 2005)
8. Fiedler, F., Karg, B., Lüken, L., Brandner, D., Heinlein, M., Brabender, F., Lucia, S.: do-mpc: Towards fair nonlinear and robust model predictive control. Control Engineering Practice **140**, 105676 (2023)
9. Fujimoto, S., van Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 1587–1596. PMLR (2018-07-10/2018-07-15)
10. Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., Knoll, A.: A Review of Safe Reinforcement Learning: Methods, Theory and Applications (May 2024)

11. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 1861–1870. PMLR (2018-07-10/2018-07-15)
12. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2**(5), 359–366 (1989)
13. IEA: Tracking Clean Energy Progress 2023 (2023), `https://www.iea.org/reports/tracking-clean-energy-progress-2023`, licence: CC BY 4.0
14. Karg, B., Lucia, S.: Efficient representation and approximation of model predictive control laws via deep learning. IEEE Transactions on Cybernetics **50**(9), 3866–3878 (2020)
15. Kim, B., Huusom, J.K., Lee, J.H.: Robust Batch-to-Batch Optimization with Scenario Adaptation. Industrial & Engineering Chemistry Research **58**(30), 13664–13674 (Jul 2019)
16. Lucia, S., Andersson, J.A., Brandt, H., Diehl, M., Engell, S.: Handling uncertainty in economic nonlinear model predictive control: A comparative case study. Journal of Process Control **24**(8), 1247–1259 (2014)
17. Nian, R., Liu, J., Huang, B.: A review on reinforcement learning: Introduction and applications in industrial process control. Computers & Chemical Engineering **139**, 106886 (2020)
18. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research **22**(268), 1–8 (2021)
19. Rawlings, J.B., Mayne, D.Q., Diehl, M.: Model Predictive Control: Theory, Computation, and Design. Nob Hill Publishing, Santa Barbara, California, 2nd edn. (2020)
20. Reepmeyer, F., Repke, J.U., Wozny, G.: Time optimal start-up strategies for reactive distillation columns. Chemical Engineering Science **59**(20), 4339–4347 (2004)
21. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms (Aug 2017)
22. Shin, J., Badgwell, T.A., Liu, K.H., Lee, J.H.: Reinforcement learning–overview of recent progress and implications for process control. Computers & Chemical Engineering **127**, 282–294 (2019)
23. Skogestad, S., Postlethwaite, I.: Multivariable Feedback Control: Analysis and Design. John Wiley, Hoboken, NJ, 2nd ed edn. (2005)
24. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. Adaptive Computation and Machine Learning Series, The MIT Press, Cambridge, Massachusetts, 2nd edn. (2018)
25. Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., Duijkeren, N.v., Zanelli, A., Novoselnik, B., Albin, T., Quirynen, R., Diehl, M.: acados—a modular open-source framework for fast embedded optimal control. Mathematical Programming Computation **14**(1), 147–183 (2022)
26. Yoo, H., Byun, H.E., Han, D., Lee, J.H.: Reinforcement learning for batch process control: Review and perspectives. Annual Reviews in Control **52**, 108–119 (2021)
27. Zanon, M., Gros, S.: Safe reinforcement learning using robust mpc. IEEE Transactions on Automatic Control **66**(8), 3638–3652 (2020)