Simultaneous Process Design and Control Optimization using Reinforcement Learning

Steven Sachio * **Antonio E. del-Rio Chanona** **
Panagiotis Petsagkourakis ***

* *Department of Chemical Engineering, Imperial College London, SW7 2AZ UK (e-mail: steven.sachio19@imperial.ac.uk).*
** *Department of Chemical Engineering, Imperial College London, SW7 2AZ UK (e-mail: a.del-rio-chanona@imperial.ac.uk).*
*** *Department of Chemical Engineering, University College London, WC1E 7JE UK (e-mail: p.petsagkourakis@ucl.ac.uk).*

Abstract: The performance of a chemical plant is highly affected by its design and control. A design cannot be accurately evaluated without its controls and vice versa. To optimally address design and control simultaneously, one must formulate a bi-level mixed-integer nonlinear program with a dynamic optimization problem as the inner problem; this is intractable. However, by computing an optimal policy using reinforcement learning, a controller with a closed-form expression can be computed and embedded into the mathematical program. In this work, an approach that uses a policy gradient method to compute the optimal policy, which is then embedded into the mathematical program is proposed. The approach is tested in a tank design case study and the performance of the controller is evaluated. It is shown that the proposed approach outperforms current state-of-the-art control strategies. This opens a whole new range of possibilities to address the simultaneous design and control of engineering systems.

Keywords: process design, process control, reinforcement learning, policy gradient, optimization

1. INTRODUCTION

The performance of a chemical plant is substantially affected by its design and its ability to maintain the optimal operating conditions under operational uncertainty (Diangelakis et al., 2017). A design cannot be evaluated without the control and vice versa. Hence, it is essential to simultaneously formulate the design and control of chemical processes to maximize performance.

The Process Systems Engineering (PSE) community has been challenging these problems for decades (Burnak et al., 2019). On one hand, several frameworks have been proposed to avoid simultaneous process design and control problems by avoiding the solution of bi-level mixed-integer dynamic optimization (MIDO) problem by decoupling the approaches, such as in (Flores-Tlacuahuac and Biegler, 2007), model-based flow sheet and process group contribution method (Alvarado-Morales et al., 2010) and control structure selection and design (Skogestad and Morari, 1987). The controller used in previous studies were mainly PI and PID controllers due to their simplicity and robustness. However, this comes with its shortcomings, such as not being able to handle slow disturbances (Sung and Lee, 1996), multiple input multiple output (MIMO) systems optimally, and the fact that they are based on process linearizations.

⋆ This project has received funding from the EPSRC projects (EP/R032807/1) and (EP/P016650/1).

On the other hand, a small proportion of studies have implemented model predictive control (MPC). One of the first significant contributions to implementing the MPC for simultaneous design and control was done by Brengel and Seider (1992). In the formulation, the authors implemented a bi-level optimization problem, where the outer level has the economic objective while the inner level is the MPC formulation as shown in (1).

$$\begin{aligned}
\min_{\boldsymbol{p}} \; & C_p(\boldsymbol{p}) + \kappa C_u(\boldsymbol{x}(\tau), \boldsymbol{y}(\tau), \boldsymbol{u}(\tau), \boldsymbol{p}, \theta^P(\tau)) \\
s.t. \; & f_p(\boldsymbol{p}, \theta^P(\tau)) = 0 \\
& g_p(\boldsymbol{p}, \theta^P(\tau)) \leq 0 \\
& \min_{\pi} \; C_u(\boldsymbol{x}(\tau), \boldsymbol{y}(\tau), \boldsymbol{u}(\tau), \boldsymbol{p}, \theta^P(\tau)) \\
& \quad s.t. \; \dot{\boldsymbol{x}} = f_u(\boldsymbol{x}(\tau), \boldsymbol{y}(\tau), \boldsymbol{u}(\tau), \boldsymbol{p}, \theta^P(\tau)) \\
& \qquad \boldsymbol{u}(\tau) = \pi(\boldsymbol{x}(\tau), \boldsymbol{y}(\tau), \boldsymbol{p}) \\
& \qquad g_u(\boldsymbol{x}(\tau), \boldsymbol{y}(\tau), \boldsymbol{u}(\tau), \boldsymbol{p}, \theta^P(\tau)) = 0 \\
& \qquad h_u(\boldsymbol{x}(\tau), \boldsymbol{y}(\tau), \boldsymbol{u}(\tau), \boldsymbol{p}, \theta^P(\tau)) \leq 0
\end{aligned} \tag{1}$$

where τ is time, $\boldsymbol{x}$ is the vector of states of the system, $\boldsymbol{y}$ is the vector of system outputs, $\boldsymbol{u}$ is the control action described by the policy π, $\boldsymbol{p}$ is the design variable vector including the steady-state manipulated variables, θ^P is the parameter vector of the process and f, g and h are the constraints. C_p and C_u are the economic and controllability cost functions and κ is the design and control integration scaling factor which impacts the trade-

off between controllability and investment cost of the system.

This bi-level optimization problem is extremely expensive to solve online and this is why there was significantly less work done in MPC compared to PI controllers for simultaneous process design and control at the time. In (Chu and You, 2014) the authors have proposed a novel method utilizing a decomposition algorithm to solve bi-level integrated scheduling and optimization for sequential batch processes, while Beykal et al. (2020) have proposed a data-driven framework to tackle bi-level optimization problems. An approach to solve the bi-level problem was presented by Bemporad et al. (2002); Sakizlis et al. (2004) where the authors replaced the inner problem with a multi-parametric MPC (mpMPC) reducing it to a simple look-up table algorithm. In Diangelakis et al. (2017) the authors worked further upon this approach and formulated a "design-dependent offline controller", where it only requires solving the mpMPC problem once offline. The biggest drawback of these mpMPC approach is that they rely in linearizations which do not represent well nonlinear dynamics.

To solve this intractable bi-level optimization problem, we propose to take advantage of the closed form (explicit) nature of reinforcement learning controllers. Reinforcement learning (RL) controllers (Sutton and Barto, 2018) are a natural choice, as they can address disturbances in highly nonlinear and complex processes (Petsagkourakis et al., 2020).

2. METHOD AND INTEGRATION

2.1 Problem Statement

Given a simultaneous process design and control problem, first a bi-level optimization problem is formulated and then split into two, the design problem and the control problem. The design problem is formulated as a mixed integer dynamic optimization (MIDO) problem as shown in (2).

$$\begin{aligned}
\min_{\boldsymbol{Y},\boldsymbol{p}}\ & J_{SDC} = \int_0^{T_F} \mathcal{P}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{u},\boldsymbol{d},\boldsymbol{Y},\boldsymbol{p})d\tau \\
s.t.\ & \frac{dx}{d\tau} = f(\boldsymbol{x},\boldsymbol{y},\boldsymbol{u},\boldsymbol{d},\boldsymbol{Y},\boldsymbol{p}) \\
& \boldsymbol{y}_{min} \leq \boldsymbol{y} = g(\boldsymbol{x},\boldsymbol{u},\boldsymbol{d},\boldsymbol{Y},\boldsymbol{p}) \leq \boldsymbol{y}_{max} \\
& \boldsymbol{u}_t = \pi_\theta(\boldsymbol{x}_t^{obs},\boldsymbol{x}_{t-1}^{obs},\boldsymbol{y}_t,\boldsymbol{y}_{t-1},\boldsymbol{p}^{obs})\ \forall t \\
& \boldsymbol{Y} \in \{0,1\},\ \boldsymbol{x}_t^{obs} \in \boldsymbol{x}_t,\ \boldsymbol{p}^{obs} \in \boldsymbol{p} \\
& [\boldsymbol{x}_{min},\boldsymbol{d}_{min}] \leq [\boldsymbol{x}_\tau,\boldsymbol{d}_\tau] \leq [\boldsymbol{x}_{max},\boldsymbol{d}_{max}]\ \forall \tau \\
& \boldsymbol{p}_{min} \leq \boldsymbol{p} \leq \boldsymbol{p}_{max} \\
& \tau \in [0,T_F],\ t \in \{1,...n_T-1\}
\end{aligned} \tag{2}$$

where τ is the continuous variable time $\tau \in [0, T_F]$, T_F is the time horizon discretized in n_T time steps of size $\Delta\tau = T_F/n_T$. The subscript t is the time step number, it represents the value of a variable at time $\tau = t\Delta\tau$ where $t \in \{0,\ 1, ...,\ n_T-1,\ n_T\}$. While $\boldsymbol{x} \in R^{n_x}$ is the vector of states of the system, $\boldsymbol{y} \in R^{n_y}$ is the vector of system outputs, $\boldsymbol{u} \in R^{n_u}$ is the control action represented discretely as a piece-wise constant variable, $\boldsymbol{d}$ is the disturbance vector, $\boldsymbol{Y}$ is the design binary variable vector, $\boldsymbol{p}$ is the design continuous variable vector including the steady-state manipulated variables, $\mathcal{P}$ is the integral function of the objective function J_{SDC} (usually an economic/efficiency related cost function), f are the differential equations describing the process dynamics and g are the constraints. The vectors with superscript *obs* are the values of the vector which is observable by the controller, given that not the whole state of the process is necessarily observable.

The controller in (2) is represented by a policy π parameterized by a set of parameters θ. The policy could be given as a linear controller, e.g. (Chu and You, 2014; Diangelakis et al., 2017). However, this approach has a limitation; there is no close form expression for nonlinear dynamic systems. As a result an appropriate approximation via linearization is implemented (Diangelakis et al., 2017) and subsequently a multi-parametric programming technique is applied to compute the explicit solution of the optimal control problem. In this paper, we avoid the use of approximations in the dynamic systems and reinforcement learning is utilized to construct closed form expression for the general stochastic closed-loop optimal control problem (see (3) and substitute in (2)). Specifically policy gradient technique is proposed to construct an optimal policy π_θ as an artificial neural network (ANN).

The goal of the design problem in (2) is to find combinations of the design and binary variables (e.g. reactor type, process layout), given an optimized controller to minimize an objective function J_{SDC} over a time horizon T_F discretized in n_T number of time steps of size $\Delta\tau = T_F/n_T$ while satisfying the constraints f and g. To get the optimized controller, an optimal control problem (OCP) is formulated in a way that the controller would also take in the observable design variable as its input. The formulation is shown in (3).

$$\begin{aligned}
\max_{\pi_\theta}\ & \mathbb{E}[J_{OCP}(\boldsymbol{x}_t^{obs},\boldsymbol{y}_t^{obs},\boldsymbol{u}_t,\boldsymbol{p})] \\
s.t.\ & \boldsymbol{x}(0) = \boldsymbol{x}_0 \\
& \boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t,\boldsymbol{u}_t,\boldsymbol{d}_t,\boldsymbol{p})\ \forall t \\
& \boldsymbol{u}_t = \pi(\boldsymbol{x}_t^{obs},\boldsymbol{x}_{t-1}^{obs},\boldsymbol{y}_t^{obs},\boldsymbol{y}_{t-1}^{obs},\boldsymbol{p}^{obs}) \\
& \boldsymbol{x}_t^{obs} \in \boldsymbol{x}_t,\ \boldsymbol{y}_t^{obs} \in \boldsymbol{y}_t,\ \boldsymbol{p}^{obs} \in \boldsymbol{p} \\
& [\boldsymbol{x}_{min},\boldsymbol{u}_{min}] \leq [\boldsymbol{x}_t,\boldsymbol{u}_t] \leq [\boldsymbol{x}_{max},\boldsymbol{u}_{max}]\ \forall t \\
& \boldsymbol{y}_{min} \leq \boldsymbol{y}_t \leq \boldsymbol{y}_{max}\ \forall t \\
& t \in \{1,...,n_T-1\}
\end{aligned} \tag{3}$$

where J_{OCP} is the cost function of the OCP problem e.g. setpoint error (not to be confused with the objective in (2)). The goal of this optimal control problem is to find the optimum parameters of the policy that maximize the expectation of the objective function. Notice that this problem is intractable as a closed-loop policy π needs to be found. To solve this problem the policy is parameterized by an ANN (π_θ) and then a policy gradient method is used (Petsagkourakis et al., 2020; Sutton and Barto, 2018) to solve the optimal control problem.

This is a bi-level optimization problem, with the outer level being a MINLP and the inner level an OCP. By using RL this problem can be addressed. The bi-level optimization problem can be separated and formulated as presented in (3). Notice that the OCP takes into account the design variables to calculate the control output. When this is solved by RL, the final form of the policy is an

explicit function, that takes states and design variables as input, and outputs an optimal control action, this can be embedded into the MIDO problem formulated in (2). Remark: If ReLU activation functions are used, these result in a mixed integer linear function, which can be easily appended to the MINLP problem.

2.2 Artificial Neural Networks (ANNs)

The simplest form of neural networks, feed-forward neural network, is used in this work. An ANN is composed of individual nodes/neurons in which each neuron may have the same or different parameters and activation functions. To make the concept more concrete, let us consider a simple neural network shown in Fig. 1.

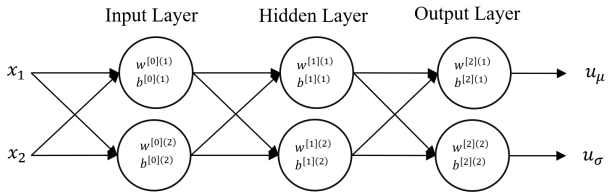

Fig. 1. Simple feed-forward artificial neural network.

Each neuron has its own parameters called weights and biases and they are denoted by $w^{[l](n)}$ and $b^{[l](n)}$ respectively. The superscript $[l]$ corresponds to the layer number while the superscript (n) corresponds to the node number in that layer. In each neuron, the weight is a vector with the same size as the input to that node and the bias is a scalar.

There are two steps of calculations done in a neuron, first is using the weights and bias to calculate a value $z^{[l](n)}$ and then using the activation function of that layer to calculate $a^{[l](n)}$. Activation functions are used to help with the non-linearity, without it the neural network would just be the same as a linear regression. More details and equations on how the calculations are done in the neural network can be found in (Bebis and Georgiopoulos, 1994).

The policy in our formulation is in the form of an ANN just like the one shown in Fig. 1 but different numbers of layers and nodes. The input and all of the hidden layers use the tanh activation function while the output layer nodes use the ReLU6 and leaky ReLU function for the mean and std of the control action, respectively. Details on the ReLU6 and leaky ReLU functions can be found in (Agarap, 2018).

2.3 Reinforcement Learning

The reinforcement learning policy is computed based on the OCP formulated in Problem (3). The method used in this work to optimize the policy is a policy gradient type method and the structure of the controller a feed-forward neural network. To give the controller a head start, a pre-training (supervised learning) scheme is used (for example, see (Torabi et al., 2018)). The main idea is that labelled data (X, Y) are fed into the algorithm and the algorithm will try to find a relationship between them and then predict the output (Y) given the state (X).

In a process control context, the data could be a trajectory of states ($\boldsymbol{x}_{PT}$) and the control action trajectory ($\boldsymbol{u}_{PT}$) taken given the state trajectory. Then, the neural network is trained so that if it sees a similar state $\boldsymbol{x}_i$, it would return a control action which is close to the control $\boldsymbol{u}_i$. The algorithm is shown in Algorithm 1.

Algorithm 1 Pre-training (supervised learning).

Input: Initialize policy with parameters $\theta = \theta_0$, with $\theta_0 \in \Theta_0$, learning rate α, number of iterations n_{iter} and labelled data $\boldsymbol{x}_{PT}$ and $\boldsymbol{u}_{PT}$
Output: Pre-trained policy.

for m = 1,..., n_{iter} **do:**

(1) Generate control actions, $\hat{\boldsymbol{u}}$, using the current policy with parameters θ from the labelled state trajectory $\boldsymbol{x}_{PT}$.
(2) Improve π_θ by AdamOptimizer($\hat{\boldsymbol{u}}, \boldsymbol{u}_{PT}$) (Kingma and Ba, 2017).
(3) $m := m + 1$

After the pre-training, the controller is deployed into the environment to begin the reinforcement learning. The Reinforce (Williams, 1992) (Monte Carlo) algorithm was used with the addition of baseline and decaying learning rate. The algorithm is shown in Algorithm 2.

Algorithm 2 Policy Gradient Algorithm.

Input: Initialize policy parameter $\theta = \theta_0$, with $\theta_0 \in \Theta_0$, learning rate, its update rule α, $m := 0$, the number of episodes K and the number of epochs N.
Output: policy $\pi(\cdot|\cdot, \theta)$ and Θ **for** m = 1,..., N **do:**

(1) Collect $\mathbf{u}_t^k, \mathbf{x}_t^k$ for T time steps for K trajectories along with $J(\mathbf{x}_T^k)$, also for K trajectories.
(2) Update the policy, using a policy gradient estimate $\theta_{m+1} = \theta_m + \alpha_m \frac{1}{K} \sum_{k=1}^{K} \left[(J(\boldsymbol{\tau}^k) - b) \nabla_\theta \sum_{t=0}^{T-1} \log \left(\pi(\mathbf{u}_t^k | \hat{\mathbf{x}}_t^k, \theta) \right) \right]$
(3) $m := m + 1$

The update rule used in the algorithm (step 2 of Algorithm 2) is based on the one used by Petsagkourakis et al. (2020) derived from the policy gradient theorem (Sutton and Barto, 2018). In this update rule, b, is the baseline. Due to the stochasticity of the policy coupled with Monte Carlo sampling, the approximation of the expectation can have high variance, however, a baseline can be used to reduce this variance without a bias (Sutton and Barto, 2018). The baseline used in this work is simply the mean expectation of rewards under the current policy, this is the most commonly used baseline in policy gradient projects and it has been proven to be effective (Petsagkourakis et al., 2020).

While the baseline helps reducing the variance, the decaying learning rate on the other hand helps the training to be faster during the early epochs, and more stable during the later epochs. In this work we use Reinforce for simplicity of presentation, however, other policy gradient-based algorithms (e.g. TRPO (Schulman et al., 2017a), PPO (Schulman et al., 2017b)) can be used.

2.4 Full Approach

First, a bi-level optimization problem is formulated and split into two problems, the design problem and the control

problem. The control problem is solved via reinforcement learning, such that an optimal policy is produced. This optimal policy is a controller in the form of an ANN such that it takes states as inputs and outputs optimal control actions. A range of design variables was defined on which the controller was trained on. Second, the policy is embedded into the design problem and solved using standard MIP methods.

The approach is shown in Algorithm 3. For training the controller, any policy optimization algorithm can be used (e.g. TRPO (Schulman et al., 2017a), PPO (Schulman et al., 2017a), DDPG (Lillicrap et al., 2015)) but for simplicity the Reinforce algorithm with baseline (Sutton and Barto, 2018) is used in this work.

Algorithm 3 Full approach for simultaneous design and control optimization using RL.

Input: Simultaneous process design and control problem.
Output: Optimal design and controller.

(1) **State design-control simultaneous optimization**: formulate the design problem as a mixed-integer dynamic optimization (MIDO) problem as presented in (2).
(2) **Optimal Control problem**: Pose the optimal control problem (OCP) based on the design problem as in (3).
(3) **OCP as RL**: Formulate the OCP to be solved via a RL policy optimization framework.
   (a) **Pre-training**: pre-train from an initial policy either via simulation or plant data, this can be done via supervised or apprenticeship learning, amongst others.
   (b) **Policy learning via policy gradients**: Begin full- training using a policy optimization algorithm.
(4) **Embedded bi-level program**: Embed the final policy obtained from **step 3** into the design (MIDO) problem and solve via a MIP algorithm.

It is important to highlight, that without a policy based method, the OCP is an optimization problem, hence the integrated formulation in (2) is an intractable bi-level optimization problem.

3. RESULTS AND DISCUSSIONS

The tank design case study from (Diangelakis et al., 2017) was used to evaluate the approach. In this case study, it is desired to design a simple tank with continuous inlet and outlet flows. There are no reactions involved, the inlet flow is in the form of a sinusoidal signal and it has a nominal inlet flow of F_{nom} with a maximum deviation of F_{dev}. The outlet flow is the manipulated variable of the PG (policy gradient) controller. A set-point which depends on the values of F_{nom} and F_{dev} is inputted into the controller. Given the coupling of design and control, F_{dev} is a design (upper level) variable, which is passed to the controller (lower level) problem as a parameter to the policy.

The design problem was formulated as a MIDO problem presented in (4). In order to reduce the size of the problem while still maintaining a reasonable sampling time to capture the dynamics of the sinusoidal signal, the final time is set to allow only one period of the sinusoidal signal.

$$
\begin{aligned}
&\max_{V_{tank},F_{dev},F_{nom},V(0)} \; J_{SDC} = \int_0^1 F_{dev} d\tau \\
s.t.\; &\frac{dV(\tau)}{d\tau} = F_{in}(\tau) - F_{out}(\tau) \\
&F_{out}(\tau) = \alpha_t V(\tau) \\
&F_{in}(\tau) = F_{nom} + F_{dev}\sin(\tau/freq) \\
&freq = \frac{1}{2\pi} \\
&V_{SP} = F_{nom} + F_{dev} \leq V_{tank} \\
&err_{\pi_\theta} = \int_0^1 \frac{||V(\tau) - V_{SP}||}{V_{SP}} d\tau \\
&\text{End-point constraints:} \\
&(1-\varepsilon/100)V(0) \leq V(T_F) \leq (1+\varepsilon/100)V(0) \\
&err_{\pi_\theta} \leq \varepsilon/100 \\
&\text{Interior-point constraints:} \\
&a_t = \pi_\theta(F_{in,t}, V_{SP}, V_t, V_{t-1}) \; \forall t \in \{0, ..., n_T - 1\}
\end{aligned}
\tag{4}
$$

where τ is the continuous time variable, $V(t)$ is the volume of liquid in the tank, $F_{in}(t)$ and $F_{out}(t)$ are the inlet and outlet flows in $\mathrm{m^3\,s^{-1}}$, respectively, a is the valve position (1 is open, 0 is closed), and $freq$ is the frequency of the sinusoidal wave disturbance in $\mathrm{s^{-1}}$. V_{SP} is the volume set point, V_{tank} is the volume of the tank, err_{π_θ} is the integral set point error, ε is the maximum allowable error and is equal to 1 %, a_t is the control action output from the policy applied to the system.

Constraints are enforced in the formulation to ensure the initial and final states are the same, within some error threshold, to achieve cyclic operation which allows for extrapolation of the operation to larger time horizons. The same error threshold, is also used for the maximum setpoint deviation.

The goal of the design problem is to determine the maximum deviation for which the controller is able to maintain the desired set-point. While the goal of the OCP is set-point tracking. The OCP is presented in (5).

$$
\begin{aligned}
&\max_{\pi_\theta} J_{OCP} = -10 \sum_{t=0}^{n_T} (V_t - V_{SP})^2 \\
s.t.\; &\frac{dV(\tau)}{d\tau} = F_{in}(\tau) - F_{out}(\tau) \\
&F_{out}(\tau) = \alpha_t V(\tau) \\
&F_{in}(\tau) = F_{nom} + F_{dev}\sin(\tau/freq) \\
&freq = \frac{1}{2\pi} \\
&V_{SP} = F_{nom} + F_{dev} \leq V_{tank} \\
&a_t = \pi_\theta(F_{in,t}, V_{SP}, V_t, V_{t-1}) \; \forall t \in \{0, ..., n_T - 1\}
\end{aligned}
\tag{5}
$$

where J_{OCP} is the objective function, it is the sum of negative squared setpoint errors. Therefore, in theory, the maximum reward that is achievable is 0. This objective function is also used for the training to calculate the total reward earned at the end of each episode.

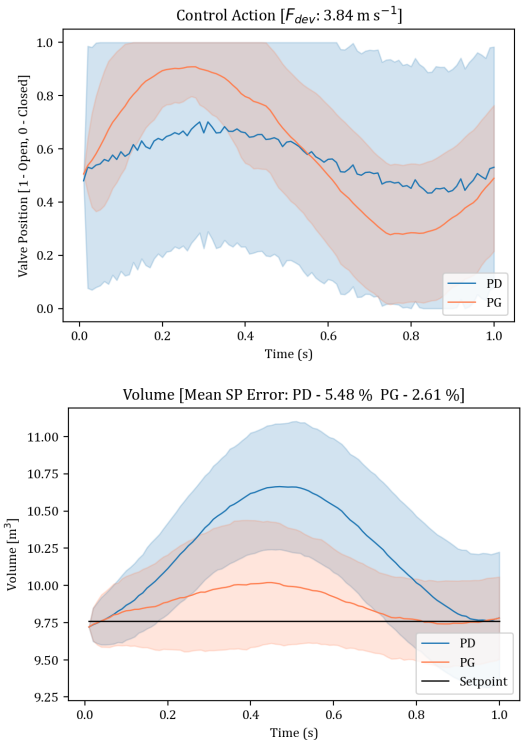

Fig. 2. Tank case study control performances.

In this case study, a proportional derivative (PD) controller is used to compare against the approach proposed in this work. The final results showed that the PG controller was able to handle deviation values of up to 3.84 m^3 s^{-1}. The control performances are plotted in Figure 2.

Figure 2 shows the performance of the controllers in the tank case study at the MIDO solution averaged over 1000 simulations with 2 % measurement noise. It can be observed that the PG controller outperforms the PD controller. The PD control action shows a large variance on its control action indicating very aggressive response. The mean set-point error of the PD controller is two times larger than the PG controller mean error. Furthermore, the PG controller were able to perform better than the mpMPC used by Diangelakis et al. (2017) ($F_{dev} = 2.69ms^{-1}$) as the PG controller managed to handle a higher maximum deviation even with the presence of measurement noise.

4. CONCLUSIONS

In this work we proposed the use of RL to address a long standing challenge for the process design and control community; the simultaneous design and control of sustainable chemical processes. It is shown that using RL it is possible to address the otherwise intractable nonlinear mixed integer dynamic bi-level optimization problem. Through the case study, we show that the PG-based controller was able to perform well in a wide range of design variables enabling the optimization of the process. Policy gradient-based controllers can handle measurement noise and stochastic systems naturally and the final form is an explicit function, which can be directly embedded into optimization problems. For future work, we will add the handling of constraints with high probability such as in (Petsagkourakis et al., 2020), and a further hyper parameter tuning scheme (e.g Snoek et al. (2012) Bayesian optimization or other expensive black box optimization techniques). We aim to address also harder and larger problems in the near future.

REFERENCES

Agarap, A.F. (2018). Deep Learning using Rectified Linear Units (ReLU). *arXiv e-prints*, arXiv:1803.08375.

Alvarado-Morales, M., Hamid, M.K.A., Sin, G., Gernaey, K.V., Woodley, J.M., and Gani, R. (2010). A model-based methodology for simultaneous design and control of a bioethanol production process. *Computers & Chemical Engineering*, 34(12), 2043 – 2061. doi:https://doi.org/10.1016/j.compchemeng.2010.07.003. 10th International Symposium on Process Systems Engineering, Salvador, Bahia, Brasil, 16-20 August 2009.

Bebis, G. and Georgiopoulos, M. (1994). Feed-forward neural networks. *IEEE Potentials*, 13(4), 27–31. doi:10.1109/45.329294.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E.N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1), 3–20. doi:10.1016/S0005-1098(01)00174-1.

Beykal, B., Avraamidou, S., Pistikopoulos, I., Onel, M., and Pistikopoulos, E. (2020). Domino: Data-driven optimization of bi-level mixed-integer nonlinear problems. *Journal of Global Optimization*, 78. doi:10.1007/s10898-020-00890-3.

Brengel, D. and Seider, W. (1992). Coordinated design and control optimization of nonlinear processes. *Computers & Chemical Engineering*, 16(9), 861 – 886. doi:https://doi.org/10.1016/0098-1354(92)80038-B. An International Journal of Computer Applications in Chemical Engineering.

Burnak, B., Diangelakis, N.A., and Pistikopoulos, E.N. (2019). Towards the grand unification of process design, scheduling, and control—utopia or reality? *Processes*, 7(7). doi:10.3390/pr7070461.

Chu, Y. and You, F. (2014). Integrated scheduling and dynamic optimization by stackelberg game: Bilevel model formulation and efficient solution algorithm. *Industrial & Engineering Chemistry Research*, 53(13), 5564–5581. doi:10.1021/ie404272t.

Diangelakis, N.A., Burnak, B., Katz, J., and Pistikopoulos, E.N. (2017). Process design and control optimization: A simultaneous approach by multi-parametric programming. *AIChE Journal*, 63(11), 4827–4846. doi:10.1002/aic.15825.

Flores-Tlacuahuac, A. and Biegler, L.T. (2007). Simultaneous mixed-integer dynamic optimization for integrated design and control. *Computers & Chemical Engineering*, 31(5), 588 – 600. doi:https://doi.org/10.1016/j.compchemeng.2006.08.010. ESCAPE-15.

Kingma, D.P. and Ba, J. (2017). Adam: A method for stochastic optimization. 3rd International Conference for Learning Representations, San Diego, 2015.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv e-prints*, arXiv:1509.02971.

Petsagkourakis, P., Sandoval, I., Bradford, E., Zhang, D., and del Rio-Chanona, E. (2020). Reinforcement learning for batch bioprocess optimization. *Computers*

& Chemical Engineering, 133, 106649. doi:https://doi.org/10.1016/j.compchemeng.2019.106649.

Petsagkourakis, P., Orson Sandoval, I., Bradford, E., Galvanin, F., Zhang, D., and del Rio-Chanona, E.A. (2020). Chance Constrained Policy Optimization for Process Control and Optimization. *arXiv e-prints*, arXiv:2008.00030.

Sakizlis, V., Perkins, J.D., and Pistikopoulos, E.N. (2004). Recent advances in optimization-based simultaneous process and control design. *Computers & Chemical Engineering*, 28(10), 2069 – 2086. doi:https://doi.org/10.1016/j.compchemeng.2004.03.018. Special Issue for Professor Arthur W. Westerberg.

Schulman, J., Levine, S., Moritz, P., Jordan, M.I., and Abbeel, P. (2017a). Trust region policy optimization.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017b). Proximal policy optimization algorithms.

Skogestad, S. and Morari, M. (1987). Control configuration selection for distillation columns. *AIChE Journal*, 33(10), 1620–1635. doi:10.1002/aic.690331006.

Snoek, J., Larochelle, H., and Adams, R.P. (2012). Practical bayesian optimization of machine learning algorithms.

Sung, S.W. and Lee, I.B. (1996). Limitations and countermeasures of pid controllers. *Industrial & Engineering Chemistry Research*, 35(8), 2596–2610. doi:10.1021/ie960090+.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement Learning: An Introduction Second Edition*. MIT Press.

Torabi, F., Warnell, G., and Stone, P. (2018). Behavioral Cloning from Observation.

Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach Learn*, 8, 229–256. doi:10.1007/BF00992696.