Computers and Chemical Engineering 144 (2021) 107133

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Reinforcement learning based optimal control of batch processes using Monte-Carlo deep deterministic policy gradient with phase segmentation

Haeun Yoo[a], Boeun Kim[b], Jong Woo Kim[c], Jay H. Lee[a,*]

[a] Department of Biomolecular and Chemical Engineering, Korea Advanced Institute of Science and Technology, 291 Daehak-ro, Yuseong-gu, Daejeon, 34141, Republic of Korea
[b] Department of Chemical and Biological Engineering, University of Wisconsin-Madison, Madison, WI 53706, USA
[c] School of Chemical and Biological Engineering, Institute of Chemical Processes, Seoul National University, 1, Gwanak-ro, Gwanak-gu, Seoul 08826, Republic of Korea

ARTICLE INFO

Article history:
Received 29 July 2020
Revised 6 October 2020
Accepted 20 October 2020
Available online 30 October 2020

Keywords:
Batch process
Reinforcement learning
Optimal control
Actor-Critic

ABSTRACT

Batch process control represents a challenge given its dynamic operation over a large operating envelope. Nonlinear model predictive control (NMPC) is the current standard for optimal control of batch processes. The performance of conventional NMPC can be unsatisfactory in the presence of uncertainties. Reinforcement learning (RL) which can utilize simulation or real operation data is a viable alternative for such problems. To apply RL to batch process control effectively, however, choices such as the reward function design and value update method must be made carefully. This study proposes a phase segmentation approach for the reward function design and value/policy function representation. In addition, the deep deterministic policy gradient algorithm (DDPG) is modified with Monte-Carlo learning to ensure more stable and efficient learning behavior. A case study of a batch polymerization process producing polyols is used to demonstrate the improvement brought by the proposed approach and to highlight further issues.

1. Introduction

Batch or semi-batch processing is widely used in the process industry, mostly for producing low-volume, high-value added products. Operating condition of a batch process is determined to meet given requirements for end product quality (e.g., composition, particle size and shape) in a manner that assures safety and economic feasibility (e.g., maximization of the productivity or minimization of the cost). However, its inherent features, such as 1) non-stationary operation covering a wide operating envelope 2) consequent exposure of the underlying nonlinear dynamics, and 3) existence of both path and end-point constraints, present a significant challenge to control engineers. These problems are exacerbated by oftentimes significant variabilities in the feedstock quality and condition as well as other process uncertainties (e.g. disturbances, noises, model errors).

For optimal control of batch processes, nonlinear model predictive control (NMPC) has been the most widely studied method (Qin and Badgwell, 2000; Mayne, 2000; Chang et al., 2016). NMPC can be designed in two different forms. First it can be designed to follow a prespecified recipe (e.g., setpoint trajectories), which are determined from an off-line (or run-to-run) optimization. Alternatively, it can be designed to optimize an economic or other performance index directly on-line while respecting process and quality constraints (Rawlings and Amrit, 2009; Ellis et al., 2014). In this setting, as an economic optimization tends to drive the operating recipe towards active constraints, it is important for the controller to assure constraint satisfaction. While NMPC offers the advantage of including constraints directly in the optimal control calculation, its constraint handling capability can degrade when the model used for the prediction has significant uncertainty (Lucia et al., 2014). Conventional NMPC addresses this problem by reperforming the optimization at each sample time after receiving feedback measurements and also by detuning the controller and tightening the constraints to account for the uncertainty ("back-off") but this can lead to significant suboptimality (Paulson and Mesbah, 2018; Santos et al., 2019). Robust NMPC formulations have also appeared which are based on rigorous uncertainty models (e.g., bounded parameter sets, scenario trees) but they come at the cost of

* Corresponding author.
E-mail address: jayhlee@kaist.ac.kr (J.H. Lee).

https://doi.org/10.1016/j.compchemeng.2020.107133
0098-1354/

Nomenclature

$X_n = G_n + D_n,\ n = 0, 1, \ldots, N$	polyol product chains
$Y_n = Q_n + R_n,\ n = 0, 1, ., N$	unsaturated chains
G_n	the growing product chains of length $n(P_nO^-K^+)$
D_n	the dormant product chains of length $n(P_nOH)$
Q_n	the growing unsat chains of length $n(U_nO^-K^+)$
R_n	the dormant unsat chains of length $n(U_nOH)$
$MX_i = \sum_{n=1}^{N} n^i X_n$	moment of X
$MY_i = \sum_{n=1}^{N} n^i Y_n$	moment of Y
$MG_i = \sum_{n=1}^{N} n^i G_n$	moment of G
$MQ_i = \sum_{n=1}^{N} n^i Q_n$	moment of Q
n_{KOH}	the total amount of catalyst
MW_{PO}	molecular weight of polyol
V	liquid volume
W	water
M	monomer
k	reaction rate constant

significantly increased complexity preventing practical use Morari and H. Lee (1999), Lucia et al. (2013), Thangavel et al. (2018).

In this regard, reinforcement learning (RL), which tries to learn an optimal control policy and value function from data may be worthy of consideration. Learning data can be real operation data or simulation data and such data contain (or can be made to contain) the effect of the uncertainty which will be experienced online. Therefore, RL provides a flexible framework wherein a control policy can be learned to address the uncertainty contained in the learning data and its potential has been examined through several set-point tracking control problems (Lee and Lee, 2005; Spielberg et al., 2019; Kim et al., 2020). For such problems, the set point tracking error was used as the (negative) reward in the training of the RL based controller. On the other hand, just a few studies have examined RL in the context of dynamic optimization of batch process operation (Wilson and Martinez, 1997; Martinez, 1998; Petsagkourakis et al., 2020).

In this paper, we examine the use of RL for on-line dynamic optimization and control of a batch process with high dimensional and continuous state and action spaces. We focus on the issues that arise in such applications, e.g., the choice of reward functions, esp. in handling constraints, construction of time-dependent approximators, and choice of learning algorithm. First of all, we suggest three types of reward functions that combine economic performance index and degree of violation of path and end-point constraints. To address the time dependence of batch operation, a phase segmentation approach is proposed to tailor the reward functions and the function approximators to suit distinct characteristics of the different phases of batch run. For the agent training, deep deterministic policy gradient (DDPG) algorithm (Lillicrap et al., 2016) is adopted as it is known to be effective in handling high dimensional continuous state and action spaces. However, the traditional DDPG algorithm, which uses temporal difference learning, is modified with Monte-Carlo (MC) learning so as to ensure stable learning behavior. This is particularly important for batch process control as the reward function is designed such that violation of the end-point constraints affect the overall reward significantly. The NMPC and different RL control strategies will be examined and compared in a case study that involves a semi-batch polymerization reactor producing polyols (Nie et al., 2013).

In summary, the intended contributions of this paper are in 1) suggesting the RL method for optimal control of batch processes with uncertainty, 2) proposing the phase segmentation approach for designing the reward functions and the function approximators and 3) modifying the DDPG algorithm with MC learning to improve the agent learning in order to better reflect the future value, particularly with respect to the end point constraints satisfactions.

In the next section we briefly introduce MPC, RL, MC vs. TD (Temporal-Difference) learning, and DDPG. Section 3 then proposes a RL based optimal control strategy with specific components like the phase segmentation, and MC-DDPG learning. The case study problem is introduced next and the training environment of the RL based optimal controller is described in Section 4. In Section 5, the improvement by using the phase segmentation approach is demonstrated and the learning results of the conventional DDPG and MC-DDPG algorithms as well as NMPC are compared. In addition, sensitivity analysis with respect to the choice of phase segmentation point and to the hyperparameters used in the reward design are investigated. Section 6 summarizes and concludes the paper.

2. Preliminaries

2.1. MPC and RL

The optimal control problem formulation we adopt in this study is given in Eq. (2.1). A general method to solve this problem is by solving the Hamilton-Jacobi-Bellman (HJB) equation, which can be derived by applying Bellman's principle of optimality. On the other hand, the HJB equation is a PDE with boundary conditions and cannot be solved in most cases with just a few well-known exceptions, e.g., the linear quadratic optimal control problem. MPC and RL represent two different approaches to solve the problem indirectly or approximately.

MPC employs a mathematical programming technique to solve an open-loop optimal control problem for a specific state encountered at each given time. Thus, MPC at each time step numerically calculates optimal input trajectory $u(t)$ using appropriate parameterization and discretization of the model (Eq. (2.1c), (2.1e)) and the input trajectory. Typically, the objective function minimized is the set-point tracking error (Eq. (2.1a)), but an economic objective function (Eq. (2.1b)), e.g., the cost or energy consumption, can be used instead and this is often referred to as *Economic MPC (eMPC)* to distinguish it from the conventional MPC. Other inequality constraints such as input and output bounds (Eq. (2.1f), (2.1g)) can be imposed in the optimization providing much needed flexibility in fitting industrial control problems into the standard optimal control problem. When the model is nonlinear, a nonlinear programming (NLP) solver is needed which employs either sequential or simultaneous strategies (Barton et al., 1998; Biegler, 2007a). Thus, the on-line computational requirement can be very high. Note that the optimization is resolved at each sample time as the starting time with the current state updated by measurements as the initial state (Eq. (2.1d)).

$$\min_{u(t)} \int_0^T |y_{sp} - y|_Q^2 + |u_{sp} - u|_R^2 \, dt \tag{2.1a}$$

$$or \max_{u(t)} \int_0^T l_t(x, u) \, dt \tag{2.1b}$$

$$\dot{x} = f(x, u) \tag{2.1c}$$

$$x(0) = x_0 \tag{2.1d}$$

$$y = g(x, u) \tag{2.1e}$$

$$u_{lb} \le u \le u_{ub} \tag{2.1f}$$

$$y_{lb} \le y \le y_{ub} \tag{2.1g}$$

MPC has become *de facto* the advanced control method of the process industry mainly owing to its ability to handle constraints on inputs and states explicitly for multi-variable systems (Eq. (2.1f), (2.1g)). This model-based control method offers an important favorable feature that the objective function and constraints can be changed on the fly, affording much needed flexibilities in fast changing industrial operation environments. Its main weakness is the high on-line computational requirement and the limited ability to incorporate information on uncertainty such as parameter errors and drifts. MPC is based on open-loop trajectory calculation and uncertainties are handled mostly by repeating the optimization after a feedback update, which is reactive in nature. In the context of batch process operation, such reactive mechanism may not be sufficient, thus leading to frequent constraint violations and off-spec products. There are robust MPC formulations such as min-max MPC, tube-based MPC, and multi-stage MPC (Lee and Yu, 1997; Mayne et al., 2011; Lucia et al., 2013), but they tend to be conservative and require significantly increased on-line computation.

RL is a data-based method to learn optimal decision policies in sequential decision making problems. "Data-based" here means that the agent (i.e. the decision maker or the controller) receives data while it is interacting with the environment (i.e. as it applies actions calculated by the policy which is being evolved). The environment can be the real system or a simulated system (Sutton and Barto, 2018). The data received typically consists of next state (s_{t+1}) and reward (r_t) taken by the result of action (a_t) following the policy (π) at current state (s_t). The reward is defined to reflect merits or demerits of each state with respect to achieving the goal. The following are the elements of the tuple that define a Markov decision process (MDP) (Puterman, 2014):

- State space S and state $s \in S$ (discrete or continuous),
- Action space A and action $a \in A$ (discrete or continuous),
- State transition model T, given by transition probability $p(s_{t+1}|s_t, a_t)$,
- Reward function $r(s, a)$ where $r : S \times A \to R$,
- Policy π mapping state s_t to a_t

s here corresponds x, the state variables of the ODE, a the manipulated variables u. The state transition model T is defined by the ODE, which is normally deterministic but can be made stochastic by appropriately adding stochastic uncertainties. For the rest of the paper, we will adopt the notations used in MDP.

Return value is defined as the (discounted) cumulative sum of the received rewards along the travelled trajectory from the current state to the terminal state ($G(s_t) = \sum_{\tau=t}^{T} \gamma^{\tau-t} r(s_\tau)$) with a discount factor $\gamma \in [0, 1]$. Value function ($V(s_t)$) is the expectation of the return value which implies the expected long-term value of the current state, and Q-function ($Q(s_t, a_t)$) is the expected return value of the state and action pair. Based on these values, the agent tries to improve the policy (π) toward the direction of getting higher return or expected return value (policy gradient methods) (Sutton et al., 2000). The design of the reward and its utilization for the agent training have a direct impact on the performance of the RL agent. Thus, in designing an RL controller, it is critical to engineer proper reward functions which reflect the characteristics and intended goal of the batch process operation.

The RL agent can eventually learn an optimal policy for a given long-term goal in an uncertain environment by observing the evaluative responses to the decisions implemented assuming an appropriate reward is used which matches the agent's goal. When the environment can be simulated, the learning can be carried out off-line and the off-line derived optimal policy can be implemented on-line, thus minimizing the potentially negative impact of the learning process on the real system and also drastically reducing the on-line computational load.

2.2. MC and TD learning

RL agent continuously improves its policy interacting with the environment. For the improvements, there are two different approaches: Monte-Carlo (MC) learning and Temporal-Difference (TD) learning. MC learning learns directly from episodes of experience which means the agent updates the value or Q-function and policy using the return values which can be calculated when the episodes terminate. Therefore, this method can only be applied to episodic MDPs (Sutton and Barto, 2018). The REINFORCE algorithm which was used for Alpha-Go training is an MC based policy gradient algorithm for learning stochastic policies (Sutton et al., 2000; Silver et al., 2016).

TD learning updates the value function by using the current estimate of the value function of next state, so this can be executed while an episode proceeds, thus speeding up the agent training process (Sutton and Barto, 2018). This learning method uses *bootstrapping* which means an estimator is updated by using an estimated target value. Although many leading RL algorithms such as Deep Q Network (DQN) and DDPG employ the TD learning method for the efficiency of training, the bootstrapping can cause the value function approximation to get stuck in local minima or even diverge in the presence of a nonlinearly parameterized function approximator or arbitrary sampling distributions (Tsitsiklis and Van Roy, 1997).

2.3. Deep deterministic policy gradient (DDPG) algorithm

DDPG is an actor-critic, model-free algorithm which adopts the strengths of Deep Q Network (DQN) and deterministic policy gradient (DPG) (Lillicrap et al., 2016). The DQN algorithm uses a deep neural network to approximately estimate the Q-function in Q-learning with continuous state space and the agent follows the ϵ-greedy policy in discrete action space (Mnih et al., 2015). The DPG algorithm deterministically maps the state to a specific action by parameterizing the actor function (e.g. by a neural network) and updates the actor parameters following the gradient of the policy's performance which is called *policy gradient*(Sutton et al., 2000; Silver et al., 2014). As shown in Algorithm 1, DDPG employs critic and actor networks. The critic network (Q) is updated toward accurate Q-function approximation and the actor network (μ) is updated toward maximizing the critic network's output using the sampled policy gradient. These updates are executed while an episode proceeds by using the current estimated value of the Q-function (y_i), which is TD learning. DDPG uses a *replay buffer* to eliminate serial correlations of the samples and also separates the target networks (Q' and μ') from the updated networks (Q and μ) to promote stable bootstrapping.

DDPG is known to be effective in problems with continuous state and action spaces like the chemical process control problems. However, there are several obstacles to applying this algorithm to chemical batch process control. First, the actor can be updated toward a wrong direction based on inaccurate critic values which in turn leads to bad samples with low rewards (Tsitsiklis and Roy, 2000; Fujimoto et al., 2018). Actor parameters can even exceed the bounds when the critic provides gradients that encourage the actor network to continue increasing a parameter that already exceeds the bounds (Hausknecht and Stone, 2015). In addition, the agent with a deterministic actor explores spaces by adding just a small noise term to the given action, which makes it even harder

Algorithm 1: DDPG algorithm (Lillicrap et al., 2016).

Initialize the critic network $Q(s, a|\theta^Q)$ and the actor network $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ
Initialize the target networks Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$
(Initialize the replay buffer R)
for *episode = 1, ..., M* **do**
  Initialize a random process ν for action exploration
  Receive initial observation state s_1
  for t = 1, ..., T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \nu_t(noise)$ according to the current policy and random exploration noise
    Receive r_t and s_{t+1} from the environment with a_t
    Store (s_t, a_t, r_t, s_{t+1}) in R
    if *number of samples in* $R \geq N$ **then**
      N random sampling (s_i, a_i, r_i, s_{i+1}) from R
      Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
      Update the critic by minimizing the critic loss: $L_c = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
      Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

      Update the target networks: $\theta^{Q'} \leftarrow \eta\theta^Q + (1-\eta)\theta^{Q'}$, $\theta^{\mu'} \leftarrow \eta\theta^\mu + (1-\eta)\theta^{\mu'}$ with $\eta \ll 1$
    end
  end
end

Table 1
Reward types.

r_{path}	$= \begin{cases} \alpha_{path} \\ -\alpha_{path} \end{cases}$	if satisfies all path constraint otherwise
r_{end}	$= \sum_j r_{end,j}$	
$r_{end,j}$	$= \begin{cases} \alpha_{end,j} \\ -\alpha'_{end,j} - \lvert\triangle c_j\rvert/s_j \end{cases}$	if satisfies end point constraint j otherwise
r_{prod}	$= \nu/s_\nu$	(ν is a measure of process performance)

to get out of the local minima or the divergence problem. Chemical batch processes are particularly vulnerable to the aforementioned drawbacks, because most chemical reactions are irreversible and thus undesired states resulting from wrong decision(s) early on can hardly be steered to desired ones later unlike in robot path planning or game playing. For instance, a faulty action during the early stage of a batch operation can inevitably lead to significant violations of end-point constraints. To address such potential problem intrinsic to chemical batch processes, a modified algorithm of Monte Carlo (MC)-DDPG with phase segmentation is suggested in the next section.

3. RL based batch process control strategy

3.1. Reward design

The goal of batch process control is to produce target products in an economical and safe manner. As in classical optimal control, goal engineering is needed to express the goal as the reward function (i.e. the objective function). In this regard, three types of reward terms (as shown in Table 1) are suggested to allow the agent to be trained effectively for the purposes of achieving high economic performance, and satisfying relevant path and end-point constraints. The first term (r_{path}) denotes reward for satisfying the path constraints. If all the path constraints are satisfied, the agent gets the corresponding reward (α_{path}), otherwise the penalty ($-\alpha_{path}$). The second term (r_{end}) is determined based on whether given end-point constraints are satisfied or not. It is the sum of the reward/penalty assigned to each end-point constraint ($r_{end,j}$). If the constraint j is satisfied, the agent gets the reward ($\alpha_{end,j}$). Otherwise, the agent receives both the default penalty ($\alpha'_{end,j}$) and the additional penalty which is proportional to the degree of constraint violation ($|\triangle c_j|$) scaled by the hyperparameter s_j. This additional penalty term is used to reflect how much the product quality deviates from the spec. The last reward term (r_{prod}) is to be designed to reflect the process's economic performance, so it is problem-specific. For example, in the ensuing case study, the total mass in the reactor ν is used as the reward in order to maximize the process's productivity. α_{path} and α_{end} are hyperparameters which should be tuned along with the discount rate γ. Note that, ν and $|\triangle c_j|$ should be appropriately scaled with s_ν and s_j, respectively, for the effective training.

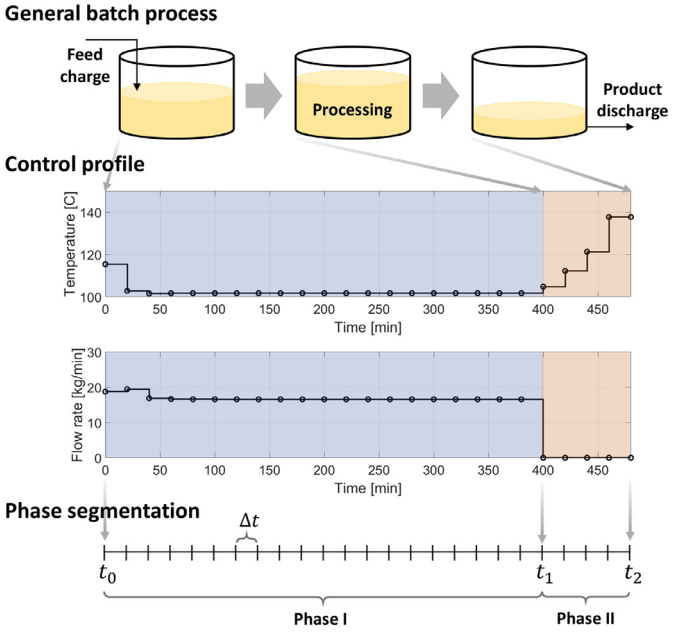

Fig. 1. Example for phase segmentation approach. Based on the physical knowledge, we can divide the control profile into two or more phases.

3.2. Phase segmentation

Most batch processes are operated with recipes encompassing time-varying trajectories, which can include abrupt changes like shutting off of the feed supply (Abel et al., 2000; Bonvin et al., 2001; Peroni et al., 2005). However, in RL, a single pair of actor-critic networks are oftentimes used to represent the value function and the policy. In batch process control, it can be a difficult task to train single networks to cover the entire duration of a batch run since they must represent functions of abrupt (e.g. nondifferentiable) changes in their slopes. Therefore, we propose an approach of phase segmentation to reflect distinct characteristics of different phases of a batch run to reflect their distinct dynamic and reward characteristics. In this approach, separate actor-critic networks are trained for different phases.

A batch operation can oftentimes be divided into phases based on available physical knowledge and reasoning. For example, as shown in Fig. 1, a typical semi-batch polymerization process operation comprises three sequential phases: feed charge, processing,

and discharge. The process should be controlled according to each phase's purpose. Excluding the discharge step, the example semi-batch operation can simply be divided into two phases: Phase I for the feeding and Phase II for carrying out the reaction. One should be able to identify the time sections for each phase and design the reward function for each phase to reflect its purpose to train separate actor-critic networks. If a process is more intricate than this simple case, the process can be segmented into more phases, but this may require significant prior knowledge. One can conceivably automate the segmentation task, e.g., via machine learning, but this is not addressed in this work and is left for future research.

Even though rewards may take on different forms for different phases, the return value (or the value function) which is the cumulative sum of the stage-wise rewards (or its expectation) is calculated along over the entire batch interval. Thus, the critic network can predict the value reflecting the future performance including the final product quality. Assuming the process is divided into P phases (phase p $= [t_{p-1}, t_p)$, $p \in \{1, \ldots, P\}$) and the reward for each phase is represented by r_p, the return value can be calculated as shown in Eq. (3.2).

$$G_\tau = \sum_{t=\tau}^{T} \gamma^{t-\tau} r_t, \;\; r_t = r_p \text{ for } t \in [t_{p-1}, t_p) \tag{3.2}$$

3.3. Monte-Carlo DDPG (MC-DDPG)

Typical environment of a chemical batch process involves continuous state and action variables. In general, state variables include concentrations of the reactants and products, the reactor temperature, etc. Common action variables are the inlet and heating/cooling jacket temperatures, the inlet flow rate, and so on. Therefore, DDPG is a suitable choice for batch process control among the available RL algorithms since it is known to be effective for problems involving continuous state and action spaces as explained in Section 2.3. However, chemical reactions oftentimes involve irreversible transitions and require a good prediction of the terminal reward early on in order to meet the end-product quality. In this regard, we hypothesize that the MC learning method may be a better choice than the TD learning method for batch process control problems.

In this paper, we modified the conventional DDPG algorithm to adopt the MC learning in the place of the TD learning and also to accommodate the phase segmentation described earlier. The resulting MC-DDPG algorithm with phase segmentation is shown in Algorithm 2. First, the critic ($Q_p(s, a|\theta^Q)$) and actor networks ($\mu_p(s|\theta^\mu)$) are initialized for the chosen number of phases, P. To initialize their parameters in a reasonable way, they can be trained by using supervised learning using closed-loop data with a sub-optimal controller like NMPC. This is called "imitation learning." After the initialization, the environment simulator generates one episode data for given batch time, T, with the actions obtained from the initialized actor network. Small noise terms are added to the action variables for exploration. When the episode terminates, the return value G for each time step is calculated and stored with the corresponding state and action in the replay buffer R_p for the time range of each phase p. This procedure is repeated for a given number of episodes L. Then, for a certain number of iterations M,N_p data set is randomly sampled from each replay buffer R_p, and based on this, the critic and actor networks are updated. The critic networks are updated by minimizing the critic's loss which is the mean squared error between the return value and the predicted Q value. The actor networks are also updated based on the same gradient calculation as in DDPG, but target networks and bootstrapping are not used. The learning performance of the DDPG algorithm with MC and TD learning methods will be compared in the ensuing case study.

Algorithm 2: MC-DDPG algorithm with phase segmentation.

Initialize the critic networks $Q_p(s, a|\theta^Q)$ and the actor network $\mu_p(s|\theta^\mu)$ with weights θ^{Q_p} and θ^{μ_p} for $p = 1, \ldots, P$
(Initialize the replay buffer R_p for $p = 1, \ldots, P$)
for *episode = 1, ..., L* **do**
  Initialize a random process ν for action exploration
  Receive an initial observation state s_1
  for *t = 1, ..., T* **do**
    Select action $a_t = \mu_p(s_t|\theta^{\mu_p}) + \nu_t(noise)$, where $t \in [t_{p-1}, t_p)$
    Receive r_t and s_{t+1} from the environment with a_t
  end
  Calculate $G_t(s_t, a_t) = r_t + \gamma r_{t+1} + \ldots \gamma^{T-t} r_T$ for $t = 1, \ldots, T$
  Store (s_t, a_t, G_t) in R_p where $t \in [t_{p-1}, t_p)$
  if *number of samples in* $R_p \geq N_p$ *for* $p = 1, \ldots, P$ **then**
    for *iter = 1, ..., M* **do**
      N_p random sampling (s_i, a_i, G_i) from R_p for $p = 1, \ldots, P$
      Update the parameters of the critic networks by minimizing the critic's loss:
      $L_{c,p} = \frac{1}{N_p} \sum_i (G_i - Q_p(s_i, a_i|\theta^{Q_p}))^2$
      Update the parameters of the actor networks using the policy gradient:

$$\nabla_{\theta^{\mu_p}} J \approx \frac{1}{N_p} \sum_i \nabla_a Q_p(s, a|\theta^{Q_p})|_{s=s_i, a=\mu_p(s_i)} \nabla_{\theta^{\mu_p}} \mu_p(s|\theta^{\mu_p})|_{s_i}$$

    end
  end
end

4. Case study

4.1. Propylene oxide (PO) batch polymerization

To evaluate and illustrate the performance of the proposed MC-DDPG with phase segmentation strategy for batch process control, a polyether polyol process for polypropylene glycol production is chosen as a case study. This process has significant nonlinear dynamics and involves both path and end constraints. The monomer PO first reacts with the alkaline anion and then the oxy-propylene anion undertakes the propagation, followed by the cation-exchange and proton-transfer reactions. A first-principles dynamic model including the population balance equation of polymer chains and monomers and overall mass balance is reformulated with the method-of-moments as shown in Eq. (4.3) for the reactor simulation and NMPC calculation (Nie et al., 2013; Mastan and Zhu, 2015). The mathematical symbols are summarized in the Nomenclature table at the end of this paper.

$$\frac{d(V[MX_0])}{dt} = Vk_i[M][G_0] \tag{4.3a}$$

$$\frac{d(V[MX_1])}{dt} = Vk_i[M][G_0] + Vk_pMG_0 \tag{4.3b}$$

$$\frac{d(V[MX_2])}{dt} = Vk_i[M][G_0] + Vk_p[M](2MG_1 + MG_0) \tag{4.3c}$$

$$\frac{d(V[MY_0])}{dt} = Vk_i[M][Q_0] \tag{4.3d}$$

Table 2
End-point and path constraints in PO polymerization example.

End-point constraints
Final NAMW ($NAMW$) $\geq$3027.74 [g/mol]
Final Unsat. Value (USV) $\leq$0.02 [mmol/g polyol]
Final unreacted PO ($Unrct$) $\leq$2000 [ppm]
Path constraints
Heat removal duty $\leq$Allowed maximum cooling capacity: $r_p(-\Delta H_p) - F(-\Delta H_f) \leq UA(T - T_w)/MW_{PO} = 430$[J/s]
Adiabatic end temperature $\leq$192°C

$$\frac{d(V[MY_1])}{dt} = Vk_i[M][Q_0] + Vk_pMQ_0 \tag{4.3e}$$

$$\frac{d(V[MY_2])}{dt} = Vk_i[M][Q_0] + Vk_p[M](2MQ_1 + MQ_0) \tag{4.3f}$$

$$\frac{d(V[X_0])}{dt} = V(2k_h[W] - k_i[G_0])[M] \tag{4.3g}$$

$$\frac{d(V[Y_0])}{dt} = -Vk_i[Q_0][M] + Vk_tn_c[M] \tag{4.3h}$$

$$\frac{d(V[M])}{dt} = F - V\{k_h[W] + (k_i - k_p)([G_0] + [Q_0]) + (k_p + k_t)n_c\}[M] \tag{4.3i}$$

$$\frac{d(V[W])}{dt} = -Vk_h[W][M] \tag{4.3j}$$

$$\frac{dm}{dt} = F \cdot MW_{PO} \tag{4.3k}$$

This reactor should be operated while satisfying two path constraints, one on the heat removal duty and the other on the adiabatic end temperature. The product should satisfy the quality specification, i.e., the end-point constraints on the final number average molecular weight ($NAMW$), the final unsaturated chains per mass (USV), and the final concentration of unreacted monomer ($Unrct$). The constraints are reported in Table 2 and other details are given in Nie et al. (2013). The manipulated variables are the reactor temperature $T(t)$ and the monomer (PO) feeding rate $F(t)$.

In this case study, the kinetic parameter of the propagation reaction A_p which is one of the most critical parameters affecting the polymerization progression is assumed to vary from batch to batch according to a uniform distribution in the range of $\pm$10% of its nominal value. The key reaction parameters, molecular weights and initial mass are reported in Table 3. Total reaction time and sampling interval are set as 480 min and 20 min, respectively, as in the previous studies (Jung et al., 2015; Jang et al., 2016). At first, we solved this problem with NMPC, which solves the dynamic optimization problem with a shrinking horizon (i.e. a horizon stretching from current time to the end of batch), to gather sample data for the initial training of the actor-critic networks and to get insights for the rewards design. Referring to the simulated NMPC results as well as the results found in the previous studies, we divided this batch operation into two phases, the second phase starting from the 400 min mark. Important features of each phase will be addressed in the next section.

Table 4
Reward for each phase.

Phase I: PO feeding phase	**Phase II: PO digestion phase**
$r = r_{path} + r_{prod}$	if not terminal: $r = r_{path}$ if terminal ($t = 480$): $r = r_{end}$

4.2. RL problem formulation

To solve the problem with RL, it should be formulated as an MDP. With the assumption of perfect measurements, the 11 physical variables in the model (Eq. (4.3)) are used as the state variables and time is also included ($s = [t;\ MX_0;\ MX_1;\ MX_2;\ MY_0;\ MY_1;\ MY_2;\ X_0;\ Y_0;\ M;\ W;\ m]$). Action variables are the aforementioned manipulated variables ($a = [T; F]$).

In this case study, a batch run is divided into two phases, depending on the operating temperature and feeding rate profiles. Phase I covers from the start to the 400 min mark. During this phase, the feed flow should be kept as high as possible in order to achieve high productivity while satisfying the safety constraints (i.e., maintaining the reactor temperature low), so this will be referred to as the *'PO feeding phase'*. Phase II, which ensues, will be referred to as the *'PO digestion phase'*, because the reactor is operated at a high temperature without further feeding in order to achieve the end product quality by quick monomer digestion. Thus, the reward of Phase I is chosen as the summation of the rewards for satisfying the path constraints (r_{path}) and for achieving sufficient amount of products which is related to the productivity (r_{prod}) without regard to the end constraints. v in r_{prod} is defined as the total mass m in the reactor. Phase II must emphasize the satisfaction of the end product quality specs without violating path constraints, so r_{path} and r_{end} are assigned to the non-terminal state and terminal state, respectively as shown in Table 4. Hyperparameters for the reward values of the constraints in Table 5 were tuned

Table 3
Key model parameters (Jang et al., 2016; Jung et al., 2015).

Kinetic parameters ($k_r = A_r \exp -E_r/RT$)				
Reaction (r)	Hydrolysis (h)	Initiation (i)	Propagation (p)	Transfer (t)
A_r [$m^3/mol \cdot s$]	8.64x10^4	3.964x10^5	1.35x10^4	1.509x10^6
E_r [kJ/mol]	82.425	77.822	69.172	105.018
Heat capacity coefficients ($c_{pi} = A_i + B_iT + C_iT^2 + D_iT^3$)				
	A_i	B_i	C_i	D_i
Feed (f)	53.347	0.51543	-1.8029×10^{-3}	2.7795×10^{-6}
Bulk (b)	1.10	2.72×10^{-3}	0	0
Mass parameters				
	H_2O	*PO*	*KOH*	*Alcohol*
MW [g/mol]	18.02	58.08	56.11	92.09
Initial mass [g]	0.6	5822	36	178

Table 5
Hyperparameters for RL agent training.

Hyperparameter	Value
γ	0.98
Total reaction time	480 min
Time step	20 min
Phase change (t_1)	400 min
Buffer size for phase 1 (R_1)	2400
Buffer size for phase 2 (R_2)	600
Minibatch size for phase 1 (N_1)	60
Minibatch size for phase 2 (N_2)	12
OU noise μ	0
OU noise θ	0.15
OU noise σ	0.2
Actor network	(12, 40, 30, 2)
Activation function for the actor	*ReLU*,*Tanh*
Critic network	((12, 30), (2, 30), 60, 30, 1)
Activation function for the critic	*ReLU*
α_{path}	0.5
α_{end}	1
$\alpha_{end}{}'$	0.5
s_{NAMW}	1/1500
s_{USV}	3100
s_{Unrct}	1/3900
s_{ν}	0.715

so that the maximum reward or penalty values have similar orders as other values during the training process.

4.3. Training environment

To initialize the network weight parameters with reasonable values, the closed-loop system with NMPC was simulated for 21 episodes to gather samples from the environment with $-10, -9, \ldots, +9, +10\%$ deviations from the nominal value of A_p. The objective function of NMPC in this case study is given by Eq. (4.4) subject to the constraints in Table 2. The problem was solved by using the simultaneous approach with three collocation points (Biegler, 2007b) and for this, a python package PYOMO with IPOPT solver was employed (Hart et al., 2011; 2017; Wächter and Biegler, 2006). These initial samples could be obtained using other controller types, such as PID controllers or any existing operation data.

$$\min_{T_t, F_t, t\in[0,480]} -m_{(t=480)} + 0.0001\sum \Delta T_t^2 + \sum \Delta F_t^2 \tag{4.4}$$

Using the collected samples, the neural networks were trained for 1000 iterations by supervised learning to initialize the neural network parameters before starting the RL agent training. After the initialization, we trained the resulting networks further using the proposed MC-DDPG algorithm for 2500 additional episodes ($L = 2500$ in Algorithm 2) with the value of A_p randomly sampled from the uniform distribution of $\pm 10\%$ of its nominal value. This uncertain parameter is held constant over each episode but updated after each episode. We set the number of episodes to provide enough episodes for the critic loss value and the total rewards to converge. We used Ornstein-Uhlenbeck (OU) process to sample the noise ν for constructing the exploration part of the policy (Uhlenbeck and Ornstein, 1930; Lillicrap et al., 2016). For the actor networks, *ReLU* and *tanh* activation functions were used, and for the critic networks, *ReLU* was used. To train the neural networks, we employed the deep neural network training package PyTorch (Paszke et al., 2017) in Python. Discount factor γ was set as 0.98 and other hyperparameters used are summarized in Table 5.

5. Results & discussion

To demonstrate the benefit of using the suggested reward design strategy with phase segmentation and the MC-DDPG algorithm, we compare 1) the results of the agent trained with and without the phase segmentation, 2) the learning performance of the agents using the MC-DDPG and the conventional DDPG (TD-DDPG), and 3) the control performances of the agent trained with the MC-DDPG algorithm, the agent initialized by the supervised learning with the NMPC closed-loop data (i.e., imitation learning), and NMPC. Furthermore, we discuss the reason for less-than-perfect performance of the proposed approach, and suggest some ways to further improve its performance. In addition, sensitivity analysis with respect to the phase segmentation point and to the hyperparameters are conducted.

Table 6
Performance results of the agent trained without the phase segmentation for the three cases (*: constraint violation).

ΔA_p	−10%	Nominal	+10%	
NAMW	3247.17	3599.38	3953.41	≥3027.74
USV	0.0297*	0.0279*	0.0264*	≤0.02
Unrct	3	3	2	≤2000
Total mass	797.67	797.64	797.63	

5.1. Phase segmentation

For applying the phase segmentation approach, two pairs of actor-critic networks were trained with the respective rewards as explained in Section 4.2. On the other hand, without the phase segmentation approach, one pair of actor-critic networks was trained with the sum of r_{path} and r_{prod} as the reward of the non-terminal states, and the r_{end} of the terminal state was designed the same as in Section 4.2. Their performances (i.e. end values of $NAMW, USV, Unrct$, and total mass) was evaluated for the cases of −10, 0, and +10% perturbations in A_p.

As shown in Fig. 2a, the action profile of the actor trained without the phase segmentation converged to an unreasonable trajectory which leads to a small amount of the product with unsatisfactory quality for all cases (see Table 6). This actor which failed to learn a good policy suggested a low temperature and zero feeding rate except for the first decision. Note that, in this case study, good performance would be achieved by a high feeding rate with a low temperature during Phase I and a low feeding rate with a high temperature during Phase II. Fig. 2b shows that the agent trained with the phase segmentation approach was able to give good action profiles, fully reflecting the aforementioned phase characteristics.

5.2. TD-DDPG vs MC-DDPG

To compare the learning performances of the TD-DDPG and MC-DDPG algorithms, we trained the agents in the same training environment. For a fair comparison, the phase segmentation approach was applied to the both algorithms with the same reward functions as in Section 4.2, and the same initial data were used for the initialization as in Section 4.3. Fig. 3 shows the total (cumulative) rewards of each episode ($\sum_{t=0}^{T} r_t$)) resulting from the two algorithms as the training proceeds. This value is getting lower with TD-DDPG which means the agent is being trained toward a wrong direction, due to the bootstrapping. As mentioned previously, when the critic network is updated with inaccurate target values through bootstrapping, the actor can also be updated with gradients of wrong directions based on the erroneous critic networks. The resulting actor policy can then lead the training to a wrong space and the actor cannot get the training process out of this situation, especially with a deterministic actor which performs limited exploration with a small noise term. Furthermore, the reason why the bootstrapping can be fatal in the case of batch

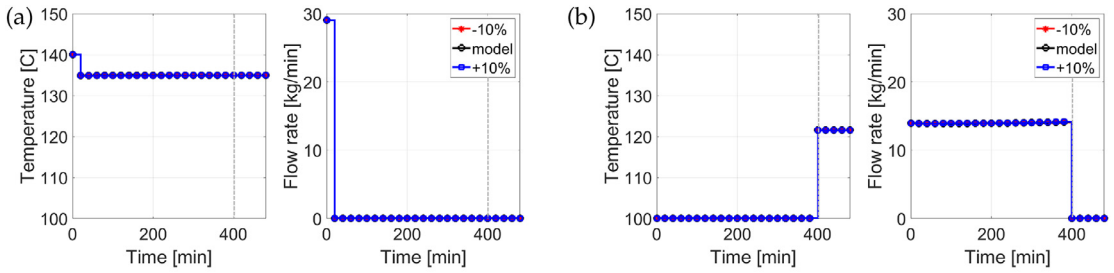

Fig. 2. Action profiles given by the actor trained (a) without the phase segmentation and (b) with phase segmentation. The actor trained without the phase segmentation failed to learn the phase characteristics reflected policy.

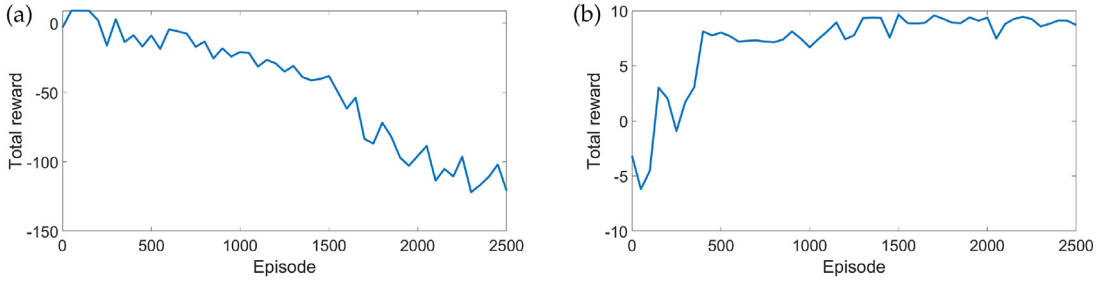

Fig. 3. Learning curves of (a) TD-DDPG and (b) MC-DDPG. The agent trained with TD-DDPG diverged toward a lower reward, but the agent trained with MC-DDPG converged to the maximum possible total reward.

Table 7
Performance results of (a) NMPC, (b) Imitation learning, and (c) MC-DDPG for three cases (*: constraint violation).

(a) eNMPC				
ΔA_p	**−10%**	**Nominal**	**+10%**	
NAMW	3158.95	3866.70	4680.56	≥3027.74
USV	0.0204*	0.0200	0.0198	≤0.02
Unrct	2444.77*	1974.27	1612.73	≤2000
Total mass	5555.16	6955.77	7839.42	
(b) Imitation learning				
ΔA_p	**−10%**	**Nominal**	**+10%**	
NAMW	3634.79	3654.19	3669.74	≥3027.74
USV	0.0228*	0.0205*	0.0187	≤0.02
Unrct	11077*	8675*	7077*	≤2000
Total mass	6941.19	6944.61	6947.64	
(c) Reinforcement learning (MC-DDPG)				
ΔA_p	**−10%**	**Nominal**	**+10%**	
NAMW	3044.69	3059.14	3071.22	≥3027.74
USV	0.0214*	0.0192	0.0174	≤0.02
Unrct	1898	1094	635	≤2000
Total mass	5798.88	5808.77	5817.14	

process control is because the high bias in the TD based learning is inappropriate for the irreversible environment like the polymerization process. Meanwhile, the learning curve of MC-DDPG gradually arrives at a high value indicating that the agent is trained to the direction of maximizing the total reward.

5.3. Control performance comparison

The developed RL based controller with the MC-DDPG algorithm and phase segmentation approach, the NMPC controller, and the controller from the imitation learning were tested on the three parameter perturbation cases as in Section 5.1, and the results are summarized in Fig. 4 and Table 7. As shown in Fig. 4a and b, the NMPC controller gives the profiles of the feeding rate which vary with the perturbation in A_p, and violations of the both path constraints are observed, especially the upper limit on the adiabatic temperature. NMPC with an economic objective typically operates the system very close to boundaries of its constraints. Hence, when the environment has significant uncertainty, constraint violations become inevitable due to the inaccurate predictions used in the NMPC calculations. All the end-point constraints are also violated in the case of the −10% perturbation in A_p as reported in Table 7a.

The performances of the agent trained by using the imitation learning are shown in Fig. 4c, d and Table 7b. The policy gives nearly the same profile for the three cases and this policy does not fully satisfy the path and end-point constraints either. This is because the actor of this agent was updated to closely mimic the sub-optimal controller's behavior without considering the constraint satisfaction.

From this initialized neural networks, the agent was trained by using the MC-DDPG algorithm and its performance results are represented in Fig. 4e, f and Table 7c. The resulting agent provides consistent action profiles despite the perturbations in A_p, and also outperforms the NMPC controller and the controller with imitation learning in terms of satisfying the required constraints. Slight violations occur only in the case of the −10% perturbation in A_p for the path constraint on the adiabatic end temperature and the end-point constraint on the final unsaturated value (*USV*). Of course, constraint satisfactions require a sacrifice on the productivity because an excessively high reactor temperature violating the safety constraints would be unavoidable if we try to produce qualified products from more monomers within given batch time.

To analyze the constraint violations seen with the −10% perturbation in A_p, we conducted the deterministic dynamic optimization for that case. Fig. 5 shows the resulting input profiles: The feeding step is finished one step earlier (380 min) than the predetermined segmentation point in time (400 min) in Section 4.2. In this regard, we can deduce that the predetermined reaction time of Phase II was not sufficient to satisfy the end constraint on *USV* in the case of the −10% perturbation in A_p. Thus, we should

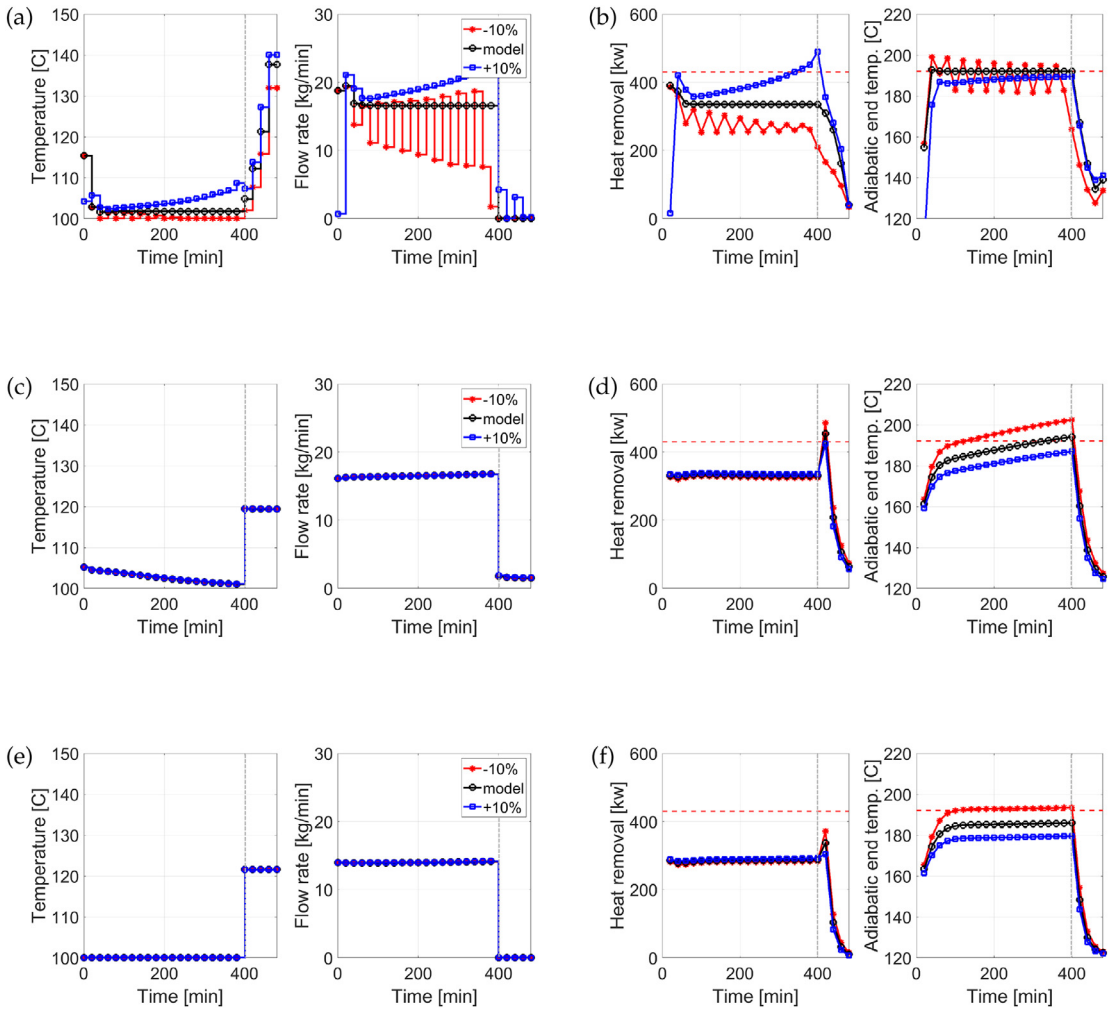

Fig. 4. (a) Action profiles of NMPC. (b) Path constraints in NMPC. (c) Action profiles of imitation learning. (d) Path constraints in imitation learning. (e) Action profiles of MC-DDPG. (f) Path constraints in MC-DDPG.

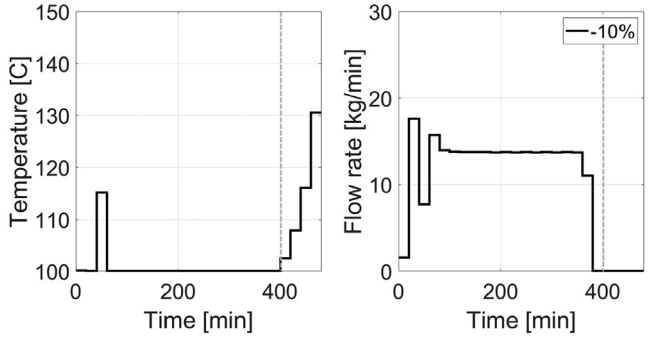

Fig. 5. Dynamic optimization result in the case of −10% perturbation in A_p.

develop a method to employ more flexible phase segmentation and this will be left for future work.

5.4. Sensitivity analysis of phase segmentation

Determining the time duration for each phase requires prior physical knowledge or engineer's insights and it can sometimes be difficult to choose a proper phase segmentation point. To see the impact of choosing an erroneous segmentation point on the optimality of the control policy, we trained the RL agent with phase I lasting longer by 1-time step ($t_1 = 420$ min) than in the previous case ($t_1 = 400$ min). As shown in Fig. 6 and Table 8, the resulting policy violates the path constraint on the adiabatic end temperature in the case of the −10% perturbation in A_p and the end-point constraint on the final unsaturated value (USV) in the cases of the −10% and nominal cases. The control performance is also a little worse than in the previous case, but the phase characteristics are well reflected. If the performance of a converged policy is not sufficient, adjusting the phase segmentation point would be a potential way to improve the policy. However, this approach has limitations in that it still requires prior knowledge. To address this problem, an RL algorithm that can determine the phase segmentation point(s) automatically should be developed in future works.

Table 8
End-point constraint results for the agent trained with a different phase duration (*: constraint violation).

ΔA_p	−10%	Nominal	+10%	
NAMW	3226.67	3234.60	3241.63	≥3027.74
USV	0.0248*	0.0219*	0.0197	≤0.02
Unrct	68	27	11	≤2000
Total mass	6162.93	6162.81	6162.67	

5.5. Sensitivity analysis of reward hyperparameter

Determining the hyperparameter values associated with the reward and penalty is as important as designing appropriate reward

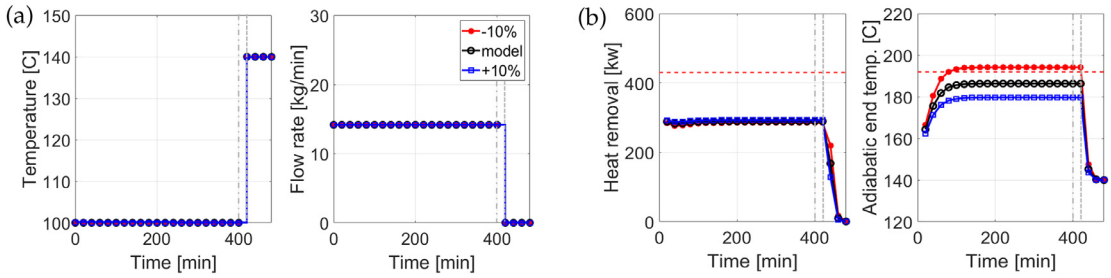

Fig. 6. (a) Action profiles and (b) Path constraints of the agent trained with a different phase duration.

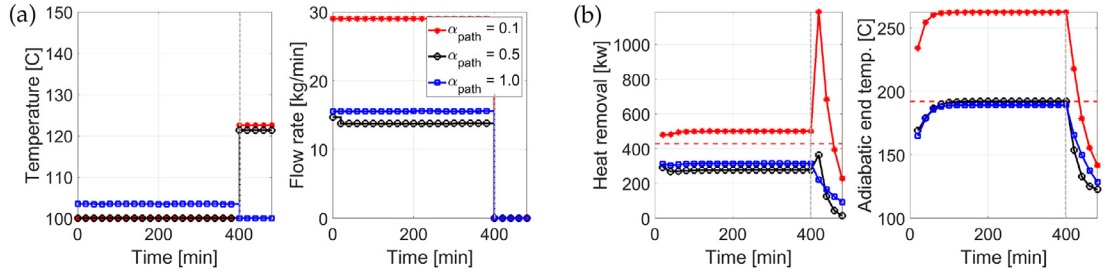

Fig. 7. (a) Action profiles and (b) Path constraints of the agents with three different α_{path} values.

Table 9
End-point constraint results for different values of α_{path} (*: constraint violation).

α_{path}	0.1	0.5	1.0	
NAMW	5736.13	3044.69	3279.10	≥3027.74
USV	0.0233*	0.0214*	0.0203*	≤0.02
Unrct	26699*	1898	38692*	≤2000
Total mass	11830.60	5798.88	6443.26	

functions. The suggested reward functions contain several hyperparameters which should be carefully decided to achieve good training performance. To show its effect on the training, three α_{path} values of 0.1, 0.5 (original), and 1.0 were used for the agents training (under the same training environment as in Section 4.3). The resulting three agents were tested for the case of the −10% perturbation in A_p as shown in Fig. 7 and Table 9. When α_{path} was set to a lower value (i.e. 0.1), the path constraints were violated more because the reward or penalty for satisfying the path constraints had less impact on the total reward. When the α_{path} was set to a higher value (i.e., 1.0), the path constraints were satisfied, but the reactor temperature converged to a lower value which led to a violation of the end constraints. For a highly nonlinear process like this polymerization process, the hyperparameters often do not have linear relationships with the training performance, and the ratios between the hyperparameters may be critical. In addition, the discontinuous reward function can result in a high sensitivity of the controller to the hyperparameter values. For more convenient design and training of the RL controllers, it is important to have a systematic way to address the sensitivity with respect to the choice of these hyperparameter values. For this, adaptation of these hyperparameters during training is a promising route and this will be addressed in our future work.

6. Conclusion

An RL based batch process control strategy was proposed, with particular attention given to the design of the reward to properly reflect the process's economic performance and (path/end-point) constraint satisfaction. In addition, the phase segmentation approach and the Monte-Carlo DDPG algorithm were suggested to handle the non-stationary and irreversible characteristics of most batch processes. The suggested RL strategy was tested on a batch polymerization example and the beneficial effects of the phase segmentation and the MC modification of DDPG on the training and control performance were observed. Comparing with the NMPC controller and the agent trained by imitation learning, the RL-based controller showed enhanced ability to satisfy the path and end-point constraints in the presence of parameter errors. The impact of the choice of phase segmentation point on the optimality was discussed by training the agent with different phase durations. The importance of determining proper hyperparameters was discussed with sensitivity analysis of the reward/penalty value. The proposed approach and algorithm can be applied to other batch or semi-batch process (e.g. bio-reactor) problems, especially those that have significant uncertainties and irreversible and nonlinear dynamics. For future work, the RL based control method will be extended to allow for more flexible phase segmentation and batch time adjustment and to systematically vary the hyperparameters during training for more intuitive and reliable learning.

Declaration of Competing Interest

The authors declare no conflict of interest.

CRediT authorship contribution statement

Haeun Yoo: Conceptualization, Methodology, Software, Writing - original draft. **Boeun Kim:** Validation, Writing - review & editing. **Jong Woo Kim:** Resources, Writing - review & editing. **Jay H. Lee:** Supervision, Writing - review & editing.

Acknowledgement

This research was supported by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (P0008475, The Competency Development Program for Industry Specialist).

References

Abel, O., Helbig, A., Marquardt, W., Zwick, H., Daszkowski, T., 2000. Productivity optimization of an industrial semi-batch polymerization reactor under safety constraints. J. Process Control 10 (4), 351–362.

Barton, P.I., Allgor, R.J., Feehery, W.F., Galán, S., 1998. Dynamic optimization in a discontinuous world. Ind. Eng. Chem. Res. 37 (3), 966–981.

Biegler, L.T., 2007. An overview of simultaneous strategies for dynamic optimization. Chem. Eng. Process. 46 (11), 1043–1053.

Biegler, L.T., 2007. An overview of simultaneous strategies for dynamic optimization. Chem. Eng. Process. 46 (11), 1043–1053.

Bonvin, D., Srinivasan, B., Ruppen, D., 2001. Dynamic Optimization in the Batch Chemical Industry. Technical Report.

Chang, L., Liu, X., Henson, M.A., 2016. Nonlinear model predictive control of fed–batch fermentations using dynamic flux balance models. J. Process Control 42, 137–149.

Ellis, M., Durand, H., Christofides, P.D., 2014. A tutorial review of economic model predictive control methods. J. Process Control 24 (8), 1156–1178.

Fujimoto, S., Van Hoof, H., Meger, D., 2018. Addressing function approximation error in actor-critic methods. In: 35th International Conference on Machine Learning, ICML 2018, vol. 4, pp. 2587–2601.

Hart, W.E., Laird, C.D., Watson, J.-P., Woodruff, D.L., Hackebeil, G.A., Nicholson, B.L., Siirola, J.D., 2017. Pyomo–optimization modeling in Python, vol. 67, second ed. Springer Science & Business Media.

Hart, W.E., Watson, J.-P., Woodruff, D.L., 2011. Pyomo: modeling and solving mathematical programs in Python. Math. Program. Comput. 3 (3), 219–260.

Hausknecht, M., Stone, P., 2015. Deep reinforcement learning in parameterized action space. arXiv preprint arXiv:1511.04143.

Jang, H., Lee, J.H., Biegler, L.T., 2016. A robust NMPC scheme for semi-batch polymerization reactors. IFAC-PapersOnLine 49 (7), 37–42. doi:10.1016/j.ifacol.2016.07.213.

Jung, T.Y., Nie, Y., Lee, J.H., Biegler, L.T., 2015. Model-based on-line optimization framework for semi-batch polymerization reactors. IFAC-PapersOnLine 28 (8), 164–169. doi:10.1016/j.ifacol.2015.08.175.

Kim, J.W., Park, B.J., Yoo, H., Oh, T.H., Lee, J.H., Lee, J.M., 2020. A model-based deep reinforcement learning method applied to finite-horizon optimal control of nonlinear control-affine system. J. Process Control 87, 166–178.

Lee, J.H., Yu, Z., 1997. Worst-case formulations of model predictive control for systems with bounded parameters. Automatica 33 (5), 763–781.

Lee, J.M., Lee, J.H., 2005. Approximate dynamic programming-based approaches for input-output data-driven control of nonlinear processes. Automatica 41 (7), 1281–1288. doi:10.1016/j.automatica.2005.02.006.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2016. Continuous control with deep reinforcement learning. In: Conference Paper at ICLR 2016.

Lucia, S., Andersson, J.A., Brandt, H., Diehl, M., Engell, S., 2014. Handling uncertainty in economic nonlinear model predictive control: a comparative case study. J. Process Control 24 (8), 1247–1259.

Lucia, S., Finkler, T., Engell, S., 2013. Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. J. Process Control 23 (9), 1306–1319.

Martinez, E.C., 1998. Learning to control the performance of batch processes. Chem. Eng. Res. Des. 76 (6 A6), 711–722. doi:10.1205/026387698525414.

Mastan, E., Zhu, S., 2015. Method of moments: a versatile tool for deterministic modeling of polymerization kinetics. Eur. Polym. J. 68, 139–160. doi:10.1016/j.eurpolymj.2015.04.018.

Mayne, D., 2000. Nonlinear model predictive control: challenges and opportunities. In: Nonlinear Model Predictive Control. Springer, pp. 23–44.

Mayne, D.Q., Kerrigan, E.C., Van Wyk, E., Falugi, P., 2011. Tube-based robust nonlinear model predictive control. Int. J. Robust Nonlinear Control 21 (11), 1341–1353.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533. doi:10.1038/nature14236.

Morari, M., H. Lee, J., 1999. Model predictive control: past, present and future. Comput. Chem. Eng. 23 (4–5), 667–682. doi:10.1016/S0098-1354(98)00301-9.

Nie, Y., Biegler, L.T., Villa, C.M., Wassick, J.M., 2013. Reactor modeling and recipe optimization of polyether polyol processes: polypropylene glycol. AIChE J. 59 (7), 2515–2529. doi:10.1002/aic.14144.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A., 2017. Automatic differentiation in PyTorch. NIPS 2017 Workshop Autodiff Submission doi:10.1145/24680.24681.

Paulson, J.A., Mesbah, A., 2018. Nonlinear model predictive control with explicit backoffs for stochastic systems under arbitrary uncertainty. IFAC-PapersOnLine 51 (20), 523–534.

Peroni, C.V., Kaisare, N.S., Lee, J.H., 2005. Optimal control of a fed-batch bioreactor using simulation-based approximate dynamic programming. IEEE Trans. Control Syst. Technol. 13 (5), 786–790.

Petsagkourakis, P., Sandoval, I.O., Bradford, E., Zhang, D., del Rio-Chanona, E.A., 2020. Reinforcement learning for batch bioprocess optimization. Comput. Chem. Eng. 133, 106649.

Puterman, M.L., 2014. Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons.

Qin, S.J., Badgwell, T.A., 2000. An overview of nonlinear model predictive control applications. In: Nonlinear Model Predictive Control. Springer, pp. 369–392.

Rawlings, J.B., Amrit, R., 2009. Optimizing process economic performance using model predictive control. In: Nonlinear Model Predictive Control. Springer, pp. 119–138.

Santos, T.L., Bonzanini, A.D., Heirung, T.A.N., Mesbah, A., 2019. A constraint-tightening approach to nonlinear model predictive control with chance constraints for stochastic systems. In: 2019 American Control Conference (ACC). IEEE, pp. 1641–1647.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al., 2016. Mastering the game of go with deep neural networks and tree search. Nature 529 (7587), 484.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M., 2014. Deterministic policy gradient algorithms.

Spielberg, S., Tulsyan, A., Lawrence, N.P., Loewen, P.D., Bhushan Gopaluni, R., 2019. Toward self-driving processes: a deep reinforcement learning approach to control. AIChE J. 65 (10), e16689.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction. The MIT Press.

Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y., 2000. Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems, pp. 1057–1063.

Thangavel, S., Lucia, S., Paulen, R., Engell, S., 2018. Dual robust nonlinear model predictive control: amulti-stage approach. J. Process Control 72, 39–51.

Tsitsiklis, J.N., Roy, B.V., 2000. Analysis of temporal-difference learning with function approximation. J. Adv. Neural Inf.Process. Syst. (1988).

Tsitsiklis, J.N., Van Roy, B., 1997. Analysis of temporal-difference learning with function approximation. In: Advances in Neural Information Processing Systems, pp. 1075–1081.

Uhlenbeck, G.E., Ornstein, L.S., 1930. On the theory of the brownian motion. Phys. Rev. 36 (5), 823.

Wächter, A., Biegler, L.T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. 106 (1), 25–57.

Wilson, J.A., Martinez, E.C., 1997. Neuro-fuzzy modeling and control of a batch process involving simultaneous reaction and distillation. Comput. Chem. Eng. 21 (SUPPL.1). doi:10.1016/s0098-1354(97)87671-5.