Comparison of reinforcement learning techniques for controlling a CSTR process*

Eric Monteiro L Luz[1] and Wouter Caarls[1]

[1*]Department of Electrical Engineering, Pontifical Catholic University of Rio de Janeiro, Rua Marquês de São Vicente, 225, Rio de Janeiro, 22541-041, RJ, Brazil.

Contributing authors: eric.luz@outlook.com; wouter@caarls.org;

Abstract

One of the main promises of Industry 4.0 is the incorporation of computational intelligence techniques in industrial process control. For the chemical industry, the efficiency of the control strategy can reduce the production of effluents and the consumption of raw materials and energy. A possible, although currently underutilized approach is reinforcement learning (RL), which can be used to optimize many sequential decision making processes through training. This work used Van de Vusse kinetics as an evaluation environment for controllers based on reinforcement learning and comparison with conventional solutions like non-linear model predictive control (NMPC). These kinetics contain characteristics that make it difficult for classic controllers such as PID to handle, such as its non-linearity and inversion point. The investigated algorithms showed excellent results for this notable chemical process control benchmark. This study was divided into two experiments: set-point change and operation around the inversion point. The former showed the ability of RL controllers to adjust the controlled variable and simultaneously maximize production. The latter revealed the excellent management capability of the reinforcement learning algorithms and NMPC at the inversion point. In this study, the RL algorithms performed similar to NMPC but with lower computational cost after training.

*This version of the article has been accepted for publication, after peer review (when applicable) but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: http://dx.doi.org/10.1007/s43153-023-00422-y. Use of this Accepted Version is subject to the publisher's Accepted Manuscript terms of use https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms

Keywords: Deep Reinforcement Learning, Chemical Process Control, Van de Vusse's Reactor, Deep Q-Learning, DDPG, SAC, TD3

1 Introduction

In the chemical industry, the process control strategy has a direct impact on process safety, use of inputs and energy, production of effluents, number of process steps, quality and the value of the final product. The most common control strategy used in industry today is the Proportional, Integral and Derivative controller (PID). However, its tuning depends on the degree of linearity of the process. This degree is a function of the thermo-physical and rheological properties. The use of PID controllers may not be satisfactory in non-linear processes.

Another approach used to control complex chemical processes is the model predictive control (MPC). This controller uses a model to predict the future behavior of the system. Based on these predictions and an optimization algorithm, it establishes a set of optimal actions. These controllers naturally handle complex goals, and particularly input, state and time constraints. Depending on the complexity of the model or the prediction horizon, this algorithm may have a high computational cost.

Under the umbrella term of Industry 4.0 [1], various computational intelligence techniques have been proposed as alternatives for processes for simulating high complexity and a considerable degree of non-linearity. In chemical reaction processes, these peculiarities occur due to the non-linearity of chemical kinetics and the possibility of chemical reactions in series and parallel.

One of the most relevant computational intelligence techniques in the control area is Reinforcement Learning [2]. This technique is grounded in the operations research area, and can be used to train controllers that optimize many sequential decision making processes, including chemical plants. In this context, the process control policy is modeled as a mapping of a set of states onto a set of actions or action probabilities. The objective of an RL algorithm is to obtain the policy that maximizes a value function, which is a prediction of the sums of expected future *rewards* from a given state.

One of the main advantages of RL controllers over classical controllers is their adaptability and flexibility. These controllers can operate in complex and nonlinear systems, explore different actions to find the best control strategy and adapt to changes in the environment. Additionally, once trained they are very light in terms of computational load. However, implementing these controllers in process control faces some challenges, such as the relative immaturity of the technology and the lack of stability assessment studies and methods. [3]

One of the challenges in using Reinforcement Learning is establishing the reward function. This function must set a balance between optimizing different process characteristics, such as safety, production and product quality. The reward function also impacts the convergence of the RL algorithm. The degree

of complexity of the function can either facilitate convergence, or make it impossible.

Technological advances in the field of computing allowed the viability and popularization of Deep-Learning techniques [4]. Thus, a new area of research involving deep reinforcement learning algorithms opened up new possibilities for solutions to highly complex and large-dimensional problems [4].

This work aims to train and evaluate reinforcement learning-based controllers for chemical reaction processes. In this evaluation, the performance of different algorithms in the control of the case study, the Van de Vusse reactor, were compared. This comparison involved advanced deep reinforcement learning algorithms like Twin Delayed DDPG and Soft Actor Critic as well as the ILQR NMPC algorithm[5].

2 Related work

The first reports of applications of Reinforcement Learning in chemical processes were neurocontrollers. J.C.Hoskins et al. in 1992, controlled a generic CSTR reactor with Anderson neural controller [6]. Alex et al. in 2001, obtained excellent results in the complete control of the (Tennessee Eastman Process) using the algorithm SANE (Symbiotic Adaptive Neuro-Evolution) [7].

Other works used Approximate Dynamic Programming (ADP) Algorithms to control reactors and other equipment. In these cases, it was necessary to use representations such as K-NN, neural networks, RBF, to approximate the continuous environment. Jong MinLee et al. in 2006, developed controllers for a simplified model of the Van de Vusse reactor and an MMA polymerization reactor [8]. Thidarat Tosukhowong et al. in 2009, implemented an ADP-based algorithm to control a plant consisting of a reactor and a distillation column with recycle. Due to the model's high dimensionality, a selection of variables was necessary [9]. Sudhakar Munusamy et al. and Christian D et al. developed ADP-based controllers for a PFR (Plug Flow Reactor) [10].

In the area of deep reinforcement learning, some works used Deep Q-Learning (DQL, [4]) with some modifications for the control of chemical processes. Zhuang Shao et al. implemented a hybrid DQL with windowing for a wet flue gas desulfurization (WFGD) system [11]. Soonho Hwangbo et al. used an MC-DQL for a downstream separation control system in biopharmaceutical processes. In this DQL change, the temporal difference learning (TD) was changed to learning based on the Monte-Carlo algorithm (MC) [12]. Dong-Hoon Oh et al. used the Advantage Actor-Critic (A2C) algorithm to estimate the ideal operating conditions of the hydrocracking process. HaeunYoo et al. proposes to change the learning by temporal differences (TD) of DDPG to the learning by Monte-Carlo (MC) [13].Yan Ma et al. were successful in controlling a polymerization reaction using DDPG. In this work, several adaptations to transform a chemical process control into an RL problem are addressed [14]. Manee et al. studied the calculation of optimal profiles of a semi-batch crystallizer to control mean size and variance using SAC, TD3, A2C, PPO. In this

work, both off-policy and on-policy algorithms were able to control the mean size and variance of the crystals. However, it has been shown that on-policy algorithms need more training episodes to achieve a near-optimal policy [15]. Alhazmi et al. integrated RL with economical MPC to operate chemical reactors in near-optimal conditions in the presence of incompatibility of plants and models. This structure was studied in ethylene oxidation and demonstrated superior performance and improved yield in the desired product [16].

Most of these studies of reinforcement learning in chemical processes are limited to complex models with reduced dimensionality, as is the case of PFR reactors and polymerization reactions. In the case of high dimensionality, the state and actions of the environment are usually simplified. It is challenging to find studies with both high dimensionality and complexity. Another aspect perceived in these studies is the predominance of the use of off-policy algorithms. These algorithms can find the optimal policy with fewer data and less training.

This work uses deep reinforcement learning algorithms in the Van de Vusse Reactor. Although this model still has a relatively low dimensionality, unlike other works with this reactor, the model was implemented without dimensionality reduction. [8, 17]. The reactor is not isothermal and the variables manipulated were the reactor and thermal jacket feed rates. These changes increase fidelity with real chemical processes and make control more difficult.

3 Background

3.1 Reinforcement Learning

A reinforcement learning agent learns to perform certain actions in an environment, desiring to maximize the sum of future rewards, also called the *return*. It does this by exploiting the knowledge learned through repeated attempts. In a classic reinforcement learning problem, the agent executes the action a_t at time t based on the current state of the environment s_t, and receives feedback from the environment in the form of a reward r_{t+1}, and a new state s_{t+1} (Figure 1) [2].

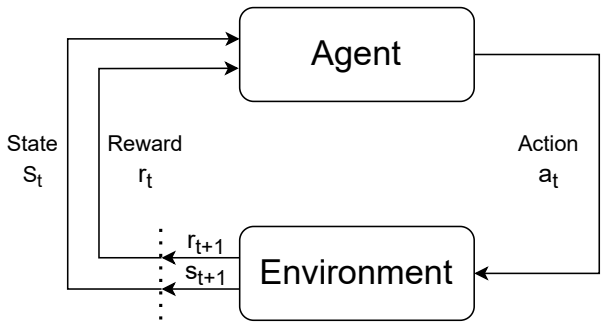

Fig. 1 Workflow of an RL agent

The state s may be beyond immediate perceptions (sensory measurements). Representations of these states can be processed versions of original perceptions, or they may be complex structures built over time from the sequence of perceptions [2]. The state signal should summarize past perceptions in a compact shape, preserving the relevant information. It may require more than immediate perceptions but certainly less than the complete history of all perceptions.

A state signal capable of retaining all relevant information has the Markov property[2]. Due to this property, it is possible to predict the expected next state and next reward, given the current state and action. This feature allows predicting the expected future states and rewards from an iterative process [2].

Most reinforcement learning algorithms estimate a function that stores the expected return of a certain state, called the state value function. The expected return depends on the actions performed by the agent. Consequently, the state value function is related to a specific policy [2].

The policy π is a mapping of each state $s \in S$, and action $a \in A(s)$, to the probability $\pi(a|s)$ of performing an action a when in the state s. The state value function $V^{\pi}(s)$ is the expected return starting from s and following the π policy. It is defined as

$$V^{\pi}(s) = \mathbb{E}(R_t|s_t = s) = \mathbb{E}_{\pi}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s), \tag{1}$$

where $\mathbb{E}_{\pi}(\dots)$ is the expected value given the agent is following the π policy, t is the time, and γ is the discount factor of future rewards to avoid infinite returns in continuous tasks and to set the optimization horizon.

Another way of evaluating the quality of a state and action is through the action-value function. $Q^{\pi}(s, a)$ is the expected return from the state s, performing the action a and following the policy π thereafter. It is defined as

$$Q^{\pi}(s, a) = \mathbb{E}(R_t|s_t = s, a_t = a) = \mathbb{E}_{\pi}(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t = s, a_t = a). \tag{2}$$

The advantage of using an action-value function is that an improved policy can be immediately derived from it:

$$\pi'(s) = \arg\max_{a} Q^{\pi}(s, a). \tag{3}$$

Specific recursive relationships are a fundamental property of the Q^{π} and V^{π} functions used in reinforcement learning. For any policy π and any state s, the following consistency condition holds between the value of s and the value

of its possible successor states

$$Q^{\pi}(s,a) = \sum_{s'} p(s'|s,a)[R(s,a,s') + \gamma V^{\pi}(s')] \tag{4}$$
$$V^{\pi}(s) = \sum_{a} \pi(a|s) Q^{\pi}(s).$$

Where $p(s'|s,a)$ is the probability of the occurrence of the new state s' given action a and state s, and $R(s,a,s')$ is the reward for the transition from s to s' taking action a.

This Bellman equation calculates the average of all possibilities, weighting each one by its probability of occurrence. The value function associated with the initial state s must equal the (discounted) value of the next expected state, plus the expected reward on the transition (Equation 5). Solving an RL problem means finding the policy that provides the greatest long-term returns [2].

3.2 Algorithms

The reinforcement learning algorithms used in this work were Deep Q-Learning (DQL) [4, 18], Deep Deterministic Policy Gradient (DDPG) [19, 20], Twin Delayed DDPG (TD3) [21] and Soft Actor Critic (SAC) [22, 23]. DQL is the foundational algorithm of the field of deep reinformcent learning, and the other three are recognized for presenting excellent results in continuous environments while requiring less samples than competing algorithms.

Deep Q-Learning (DQL) is an off-policy reinforcement learning algorithm that uses a deep neural network to approximate the Q function. Off-policy means it can learn from experience gathered by a different policy than that which is currently being followed by the agent. Specifically, it can learn from experience gathered by previous policies, which is stored in a *replay memory*. DQL creates a neural network $Q(s,a;\ w)$ that approximates the ideal Q^* function value by updating the synaptic weights w using batches of experience $(s_t, a_t, r_{t+1}, s_{t+1})$ drawn from this replay memory, using the Q-learning target

$$Q(s_t, a_t) \leftarrow r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a'). \tag{5}$$

Such networks can automatically extract implicit characteristics of the problem, in addition to being able to work with continuous spaces of high dimensionality. However, it is only applicable to discrete actions [4, 18].

Deep Deterministic Policy Gradient (DDPG) is an off-policy and continuous action reinforcement learning algorithm that combines Deep Q-Learning with an explicit representation of the deterministic policy $\mu(s;\ \theta)$ with synaptic weights θ, called an *actor*. The DDPG consists of an actor and a critic. The actor is the neural network that represents the policy. It receives a certain state and responds with an exact action. The critic is the network that represents the action-value function $Q^*(s,a)$. It takes state and action as input and

returns the expected return. The critic is estimated in the same way as DQL, but the actor is optimized using the *deterministic policy gradient*

$$\nabla_\theta Q(s, \mu(s;\ \theta);\ w) = \nabla_a Q(s, a;\ w)|_{a=\mu(s;\ \theta)} \nabla_\theta \mu(s;\ \theta), \tag{6}$$

which shifts the policy towards higher Q values[19, 20]. Note that, although the optimized policy is deterministic, the *behavior policy* $\pi(a|s;\ \theta) = \mu(s;\ \theta) + \mathcal{N}(0, \sigma)$ is stochastic with exploration noise σ to ensure sufficient exploration during training.

Twin Delayed DDPG (TD3) is an extension of DDPG combining continuous policy gradient, Actor-Critic, and Double Deep Q-Learning methods. While DDPG is capable of achieving great performance, it is often fragile in terms of hyperparameters and other tuning categories. A common fault of DDPG is that the learned function Q starts to drastically overestimate the Q values. This behavior leads to policy breaking because it exploits errors in the Q function. The TD3 adds three improvements to solve the problems reported above. The first improvement is the use of two critic networks. It uses Clipped Double-Q Learning which considers the smallest value of the two critic networks for calculating the update target. The second is to delay updates to the policy. This strategy results in greater stability of the actor network and reduces errors before being used in the Q network. The last is Target Policy Smoothing. This technique adds random noise to the action used to calculate the target, thereby making the estimate more robust.[21].

Soft Actor Critic (SAC) is another actor-critic algorithm, based on the maximum entropy reinforcement learning framework. The actor aims to maximize both reward and entropy. Due to entropy maximization, the algorithm is encouraged to explore the environment more widely, while giving up unpromising paths. It can recognize several near-optimal modes of behavior. As opposed to DDPG and TD3, which use a policy mean $\mu(s;\ \theta)$ with added noise, SAC maintains an explicitly stochastic policy network. In problems where several actions appear equally optimal, the policy will assign an equal mass of probability to these actions. In this way, collapse is avoided due to the repeated selection of a particular action that can exploit some inconsistency of the approximate Q function [22, 23].

ILQR (Iterative Linear Quadratic Regulator, [5]) is an NMPC (Nonlinear Model Predictive Control) algorithm derived from Differential Dynamic Programming[24] that uses a model to predict the future behavior of the system in a finite time window, the horizon. Based on these predictions and the current state of the system, the optimal control inputs about an objective system control function

$$C = \sum_{k=0}^{n-1} \left(\bar{x}_k^T \boldsymbol{Q} \bar{x}_k + \bar{u}_k^T \boldsymbol{R} \bar{u}_k \right) + \bar{x}_n^T \boldsymbol{Q} \bar{x}_n \tag{7}$$

are calculated, and the first control action in the sequence is applied. In Equation 7, n is the length of the horizon, $\bar{x}$ and $\bar{u}$ are the differences between the states and actions and their respective setpoints/desired values, $\boldsymbol{Q}$ is the state cost matrix and $\boldsymbol{R}$ is the action cost matrix. ILQR iteratively solves the dynamic programming problem given by linearizing the system along the current trajectory (forward pass) and maximizing the reward along it using a quadratic model of the value function using second order Taylor expansion (backward pass). For each time step in the predicted sequence, the applied control is modified in the direction of maximizing this quadratic function.

3.3 Case Study: Van de Vusse Reactor

In Van de Vusse kinetics, cyclopentenol(B) is produced from cyclopentadiene, with the formation of cyclopentanediol(C) and dicyclopentadiene(D) by-products, as shown in Figure 2 [25].

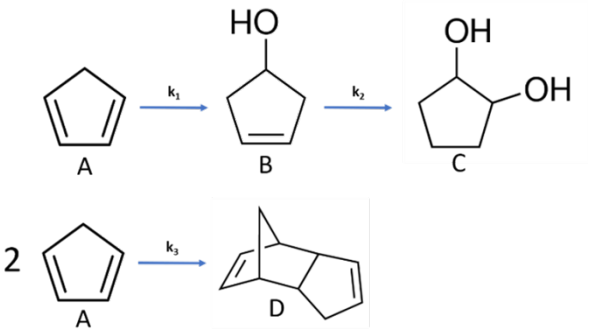

Fig. 2 Van de Vusse chemical reaction

The relationship between the kinetic constants k_1, k_2, k_3 and temperature are regulated by the Arrhenius equation

$$ki(T) = ki_0.\exp\left(\frac{-EA_i}{R.T}\right). \tag{8}$$

Where ki_0 is the kinetic constant at the reference temperature, EA_i is the activation energy, T is the temperature, and R is the universal gas constant. As described in more detail by Engell & Klatt in 1993 [26], the reaction takes place in a jacketed CSTR reactor, due to the exothermic nature of the reaction (Figure 3).

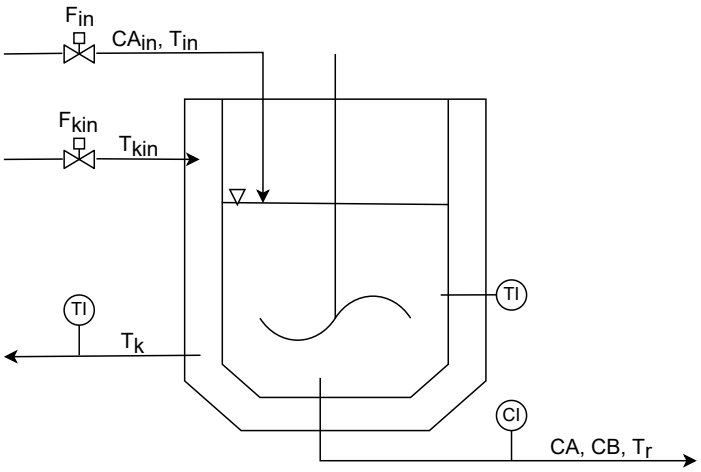

Fig. 3 CSTR reactor with Van de Vusse kinetics

Where C_A is the molar concentration of compound A (cyclopentadiene), C_B is the molar concentration of compound B (cyclopentenol), F_{in} is the reactor feed rate, T_{in} is the temperature of the reactor feed stream, C_{Ain} is the concentration of A in the feed, F_{kin} is the thermal jacket feed flow, T_{kin} is the temperature of the feed stream of the jacket, V_r is the reactor volume, T_r is the reactor temperature, and T_k is the thermal jacket temperature.

The dynamics of the system are described by the differential equations, resulting from the mass and energy balances of the reactor and the cooling jacket (Equation 9). To simulate the system, these equations are numerically integrated.

$$\begin{aligned}
\frac{dC_A}{dt} &= \frac{F_{in}}{V_r}[C_{Ain} - C_A] - k_1(T)C_A - k_3(T)C_A^2 \\
\frac{dC_B}{dt} &= \frac{F_{in}}{V_r}[C_{Bin} - C_B] + k_1(T)C_A - k_2(T)C_B \\
\frac{dT_r}{dt} &= \frac{F_{in}}{V_r}[T_{in} - T_r] + \frac{k_w.A_r}{\rho C_p V_r}[T_k - T_r] + \frac{1}{\rho C_p}[k_1 C_A \Delta H_1 + k_1 C_B \Delta H_2 + k_3 C_A^2 \Delta H_3] \\
\frac{dT_k}{dt} &= \frac{F_{kin}}{V_k}.[T_{kin} - T_k] - \frac{k_w.A_r}{m_k.C_{p_k}}.[T_k - T_r]
\end{aligned} \tag{9}$$

Tables 1 and 2 contain the kinetic parameters of the reaction and properties of the reactor and jacket provided by Engell & Klatt (1993) [26]. These data served as a basis for several other studies on this reactor [27, 28] . Some assumptions underlying the model are constant density, heat capacity, and level along the reactor.

Table 1 Physicochemical properties and dimensions of the reactor.

Parameters	Symbols	Values	Units
Solution density	ρ	0.9342	$kg.L^{-1}$
Thermal capacity	C_p	3.01	$kJ.kg^{-1}.K^{-1}$
Thermal conductivity	K_w	4032	$kJ.m^{-1}.h^{-1}K^{-1}$
Reactor heat exchange area	A_r	0.215	m^2
reactor volume	V_r	10	L
Thermal jacket mass	m_k	5	kg
Thermal capacity of the jacket	C_{p_k}	2.0	$kJ.kg^{-1}.K^{-1}$

Table 2 Reaction kinetic parameters.

Reaction	k_{i0}	EA_i/R	ΔH_{ri}
A $\rightarrow$ B	1.287 x 10^{12} h^{-1}	-9758.3 K	4,2 $kJ.mol^{-1}$
B $\rightarrow$ C	1.287 x 10^{12} h^{-1}	-9758.3 K	-11 $kJ.mol^{-1}$
2A $\rightarrow$ D	9.043 x 10^{12} $L.mol^{-1}h^{-1}$	-8560 K	-41.85 $kJ.mol^{-1}$

The mapping of the C_B at a steady state as a function of feed flow and temperature is shown in Figure 4.a. The longitudinal and transverse profiles for two specific temperatures and flows are found in Figures 4.b and 4.c, respectively. An important feature of this process is gain inversion. This feature is revealed when the same steady-state value for the controlled variable (C_B) is obtained for different values of the manipulated variable (F_{in}). This behavior makes process control even more challenging since disturbances from different sources can cause the process to lose stability.

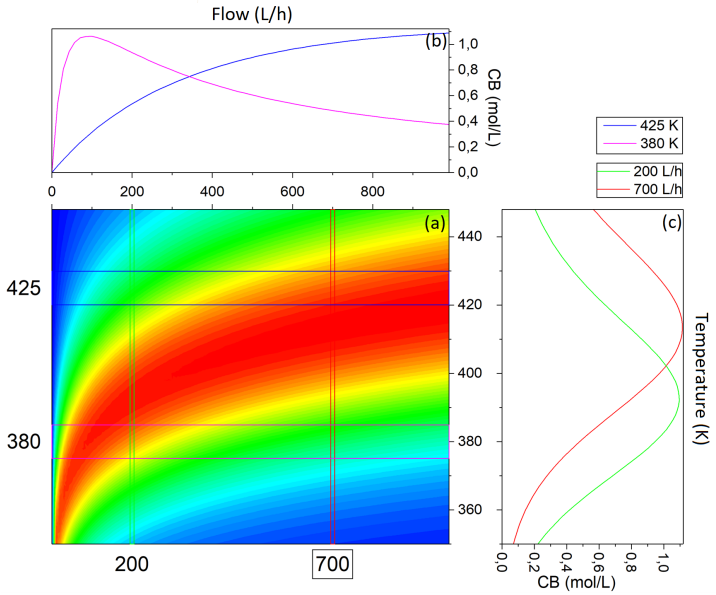

Fig. 4 The steady-states of the Van de Vusse Reactor in terms of C_B, F_{in}, T_r [28]

The non-linear profile is easily evidenced at the 380 K curve in Figure 4.b, where the curve is increasing up to the magnitude of 100 L/h but decreases for higher flows. The point where this behavior change occurs is called the inversion point. This aspect is problematic for proportional controllers because the same action can have opposite results depending on the ambient state. For example, an increase in flow causes an increase in the concentration of B in the region of low flow. However, this same action causes a decrease of C_B in the high flow region.

4 Methodology and Simulations

The Van de Vusse reactor was studied by simulation. It was numerically integrated with the differential equations that describe the process.(Equation 9). This process was implemented in the *Generic Reinforcement Learning Library* [1] (GRL).

The reinforcement learning controller developed for this process is multivariable (5 observed variables and 2 acting variables). The concentrations of compounds A and B, the setpoint of the concentration of B, and the temperature of the reactor and the thermal jacket were considered as observed variables. The manipulated variables were the reactor and thermal jacket feed rates, directly set by the RL actions. The reward function is given as

$$r(t) = W_{F_{in}}.\frac{F_{in}(t)}{700} - (1 - W_{F_{in}}).\mid Sp_{C_B}(t) - C_B(t) \mid, \tag{10}$$

where $r(t)$ is the reward per transition, $Sp_{C_B}(t)$ is the setpoint of the concentration of compound B, $C_B(t)$ is the concentration of B in the output stream, $F_{in}(t)$ is the flow rate at the reactor feed, and $W_{F_{in}}$ is the influence of the flow rate on the reward. The inclusion of $W_{F_{in}}$ allows the agent to balance between maximizing production and product quality. The objective is to obtain the greatest amount of product at the specified concentration of B.

In the NMPC controller, the state variables x were the concentrations of compounds A and B, and the temperature of the reactor and the thermal jacket. The prediction horizon was 50 timesteps, and all states were directly measured. The desired action values were set to 700 L/min and 0 L/min for the reactor and thermal jacket flows. The elements of the objective function of the NMPC are presented in Equation 11. These elements were designed to maintain similarity with the RL reward function, but following the characteristics of a quadratic MPC objective function. Note that the weight for the thermal jacket flow is very low, and only serves to stabilize the algorithm

$$\boldsymbol{Q} = \text{diag}(0, 1 - W_{F_{in}}, 0, 0)$$

[1] https://github.com/wcaarls/grl

Table 3 Operating conditions. F_{in} and F_{ink} are manipulated variables and have no initial condition.

ID	X_0^{tr}	X_0^{te}	Operation Range	unit
C_A	3.3-5.5	5.1	3.3-5.5	mol/L
C_B	0.0-1.0	0.0	3.3-5.5	mol/L
T	285-450	380	285-450	K
T_k	285-450	380	285-450	K
F_{in}	-	-	0.0-700	L/h
F_{ink}	-	-	0.0-400	L/h

$$\boldsymbol{R} = \text{diag}(W_{F_{in}} \cdot \frac{F_{in}(t)}{700}, 10^{-10}) \tag{11}$$

The initial conditions of the reactor (X_0) and the range of each process variable are found in Table 3. The training episodes were randomly initialized respecting the range described in the column X_0^{tr}. During these episodes, the setpoint value and setpoint change time were randomly altered. The test episodes were initialized with the values in the column X_0^{te} and followed predefined setpoint changes.

The simulations for the case study of the CSTR reactor with Van de Vusse kinetics were designed to evaluate the stability and robustness of the reinforcement learning controllers. The first set of simulations consists of the controller's response to the change in the setpoint of compound B (Sp_{C_B}). In the test episodes, the same initial operating conditions were maintained and Sp_{C_B} was changed from 0.9 to 1.1 mol/L at 16 minutes, according to Equation 12.

$$Sp_{C_B}(t) = \begin{cases} 0.9 \text{ if } t \leq 16 \text{ min} \\ 1.1 \text{ if } t > 16 \text{ min} \end{cases} \tag{12}$$

The second set of simulations consists of the gain inversion point test. In this experiment, the setpoint changes to the region around the inversion point. At 12 and a half minutes of simulation, Sp_{C_B} increases from 1.0 to 1.2 mol/L. At 25 minutes of simulation, Sp_{C_B} is reduced to a stable value of 0.9 mol/L (Equation 13).

$$Sp_{C_B}(t) = \begin{cases} 1.0 & \text{if } t < 12.5 \text{ min} \\ 1.2 & \text{if } 12.5 \leq t \leq 25 \text{ min} \\ 0.9 & \text{if } t > 25 \text{ min} \end{cases} \tag{13}$$

The Sp_{C_B}= 1.2 mol/L cannot be reached in a steady-state, as shown in Figure 4. Therefore, the system will alternate between a positive and negative gain during the period between 12.5 and 25 minutes of the simulation. This scenario is challenging for any controller due to these gain changes, which can cause a total lack of control of the process.

Table 4 summarizes the simulations referring to experiments 1 and 2 for the different reinforcement learning algorithms. Each simulation contains 900

Table 4 Experiment 1 and 2 simulations.

ID	Alg.	$W_{F_{in}}$	Learning Rate	Activation
1	DQL	0.1	0.001	relu-relu-linear
2	DDPG	0.1	0.001 e 0.0001	relu-relu-tanh
3	SAC	0.1	0.001 e 0.0001	relu-relu-tanh
4	TD3	0.1	0.001 e 0.0001	relu-relu-tanh

episodes of 35 minutes and was repeated 5 times. The RL training was performed with γ of 0.994, update factor of target network weights (τ) was 0.001, controller time step was 30s and neural networks with two fully connect layers containing 400 and 300 neurons, respectively. In DQL, the algorithm actions were discretized in 11 intervals per manipulated variable, distributed evenly over the entire range of flows.

Although the return is a performance metric, it is not suitable for comparing RL controllers with different environment and reward functions. Apart from the reactor flow, we use the *Integral Absolute Error* (IAE) to compare the performance between different controllers (Equation 14).

$$IAE(t) = \int_0^t \mid sp_{C_B}(t) - C_B(t) \mid dt \tag{14}$$

5 Results and discussions

Experiment 1 consists of the controller's response to the change in the setpoint of compound B (Sp_{C_B}) after 16 minutes of simulation, according to Equation 12. The second experiment of the Van de Vusse Reactor is the gain inversion point test, where the setpoint is changed to be in the region around the inversion point at 12.5 minutes of simulation (Equation 13). The training graphs of the reinforcement learning algorithms are shown in Figure 5.

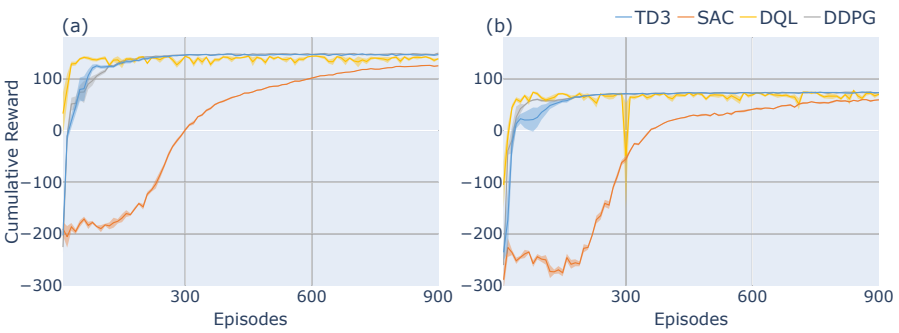

Fig. 5 Learning RL algorithms (a) Experiment 1 (b) Experiment 2;

Analyzing Figure 5, we can see the difference in difficulty between the two experiments. In experiment 1, the RL algorithms obtained faster learning and with less variation, while in experiment 2 they presented a noisier and a little slower learning. For both experiments, the DQL, DDPG, TD3 algorithms required approximately 150 training episodes to converge to a policy with similar returns. SAC presented a slower learning and a lower return for a total of 900 training episodes. Probably, with a greater number of episodes, it would catch up with the other algorithms.

5.1 Experiment 1: Setpoint change

The performance of the algorithms during the test episode for the feed flow weight $W_{FIN} = 0.1$ is shown in Figure 6.

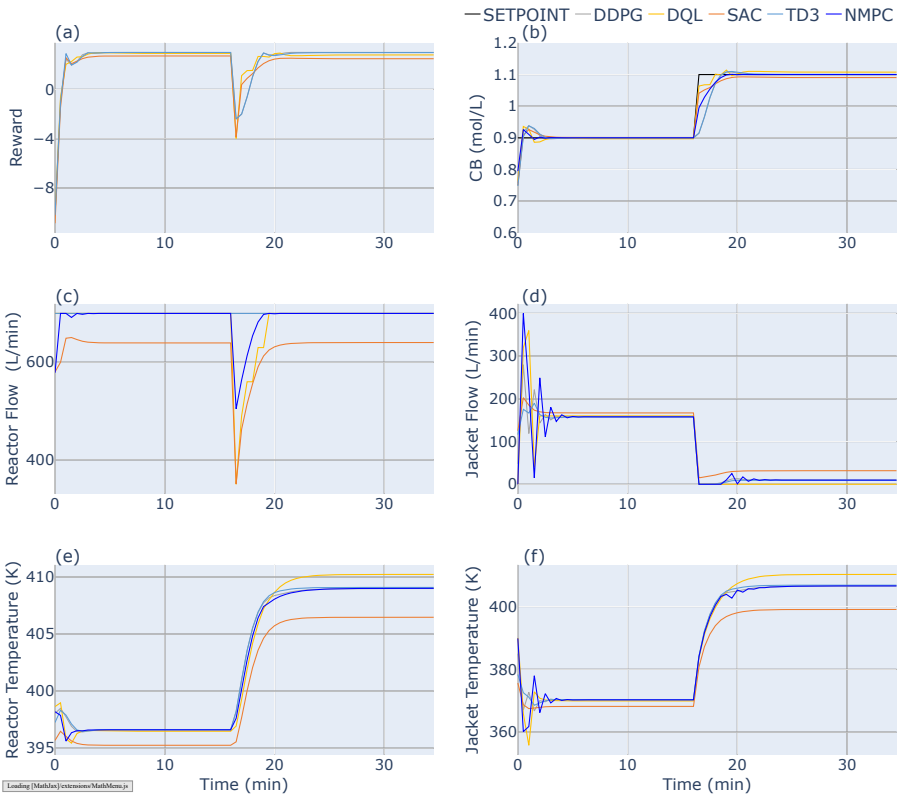

Fig. 6 Simulation results for $W_{FIN} = 0.1$ a) Rewards per step; b) Concentration of B (controlled variable); c) Reactor feed flow (manipulated variable); d) Thermal Jacket supply flow (manipulated variable); e) Reactor Temperature; f) Thermal Jacket Temperature;

For the simulations of $W_{FIN} = 0.1$, the learning algorithms presented similar performances and operating points. The DDPG algorithm presented an

IAE of 41, DQL of 40, SAC of 43, TD3 of 42 and NMPC of 33 (Figure 6.b). The SAC controller showed a small offset of 0.01 to the setpoint of 1.1 mol/L. This offset can probably be eliminated with policy improvement through more training episodes. The SAC had a flow of 647 L/min and a temperature of 406 K. The DDPG, TD3, DQL the temperature was 410 K and a flow of 700 L/min, the maximum stipulated by the model. The NMPC presented an operation similar to RL controllers. It maintained the reactor's feed flow rate at 700 L/min for most of the time, only reducing it during the setpoint change. (Figure 6.c.d).

The high operating point reached by the RL algorithms can be a desirable aspect due to production maximization. When changing the setpoint, the DDPG and TD3 maintained the maximum feed flow of 700 L/min, while the DQL temporarily reduced the flow by 210 L/min and the SAC reduced it by 368 L/min (Figures 6.c.d). Maximizing the feed throughput increased the response time for all tested deep reinforcement learning algorithms (Figure 6.b). In the thermal jacket, all algorithms performed similar actions. The decrease in the flow of the cooling fluid will increase the temperature of the thermal jacket and thus decrease the thermal exchange with the reactor. Consequently, the reactor temperature will increase, due to the Arrhenius equation, there will be an increase in the kinetic rate and the concentration of CB. The DDPG, DQL and NMPC presented a damped oscillatory action in the initial overshoot (Figure 6.d).

One of the advantages of the multivariable controller is the possibility of seeing different ways of controlling a process. This is a consequence of its ability to manipulate several process variables in an integrated way. C_B is modified by both the feed rate (F_{in}) and the temperature (T). By adding the influence of flow on the reward function (W_{FIN}=0.1), the agent's preference for acting on the thermal jacket can be seen. Unexpectedly, this addition made the controller more robust and stable.

5.2 Experiment 2: Inversion Point

The performance of the algorithms during the test episode for the feed flow weight $W_{FIN} = 0.1$ is shown in Figure 7.

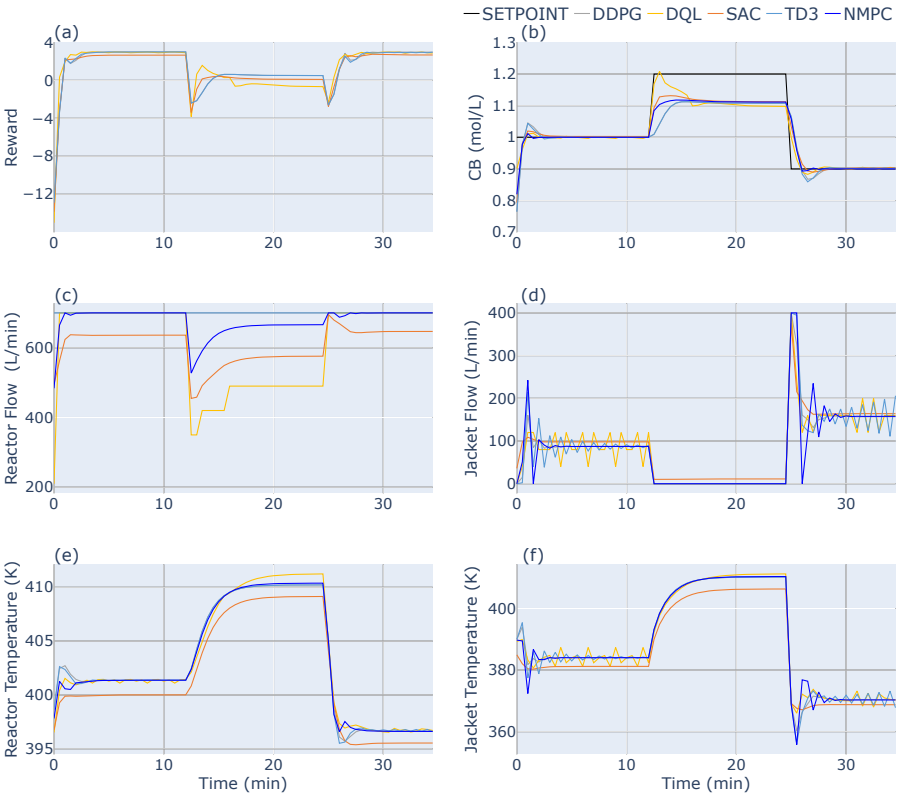

Fig. 7 Simulation results for $W_{FIN} = 0.1$ a) Rewards per step; b) Concentration of B (controlled variable); c) Reactor feed flow (manipulated variable); d) Thermal Jacket supply flow (manipulated variable); e) Reactor Temperature; f) Thermal Jacket Temperature;

The DDPG algorithm presented an IAE of 122, DQL of 100, SAC of 108, TD3 of 123 and NMPC of 107 (Figure 7.b).The RL algorithms and NMPC controller were able to pass the inversion point test with excellence. In this test, the most important thing is the ability of the controller to manage the process after reaching the inversion point. The TD3 and DDPG algorithms kept the reactor feed flow at the maximum value. DQL momentarily reduced the flow to 630 L/min while changing the setpoint. The SAC presented an operating point with a lower throughput than the other algorithms (Figure 7.c). The performance of the NMPC fell between TD3 and SAC. Regarding the concentration of B (CB), the values obtained by the NMPC were consistently lower than those of SAC and higher than those of TD3 during setpoint changes. However, this relationship was reversed in the reactor feed flow rate (FIN). The NMPC exhibited higher flow rates than SAC and lower flow rates than TD3 during setpoint changes.Using this relationship in the context of B production, it is advantageous to maximize the reactor flow rate at 700 L/min. However, this resulted in a slight delay in the response of the controllers. The faster

controllers were those that simultaneously modified the feed flow rates of both the reactor and the jacket.

Performance on the thermal jacket feed flow was similar for all tested RL algorithms. Only DQL and TD3 showed oscillatory actions, mainly at the setpoint of 0.9 mol/L, while DDPG and SAC showed continuous and precise actions (Figure 7.d)

6 Conclusion

In this work, the performance of RL controllers and NMPC for the CSTR reactor with Van de Vusse kinetics was evaluated. This process is a well-known chemical process control benchmark. As described in the 3 chapter, it has characteristics that make its control difficult. Among these characteristics are gain inversion and a multiplicity of inputs.

Most multivariable Reinforcement Learning (RL) controllers demonstrated excellent performance both in setpoint change and operation near the inversion point. In the former, RL algorithms showed similar performances among themselves. However, the Nonlinear Model Predictive Control (NMPC) exhibited a 20% lower Integral Absolute Error (IAE) compared to RL algorithms, at the cost of reduced flow than some RL algorithms. Among the RL algorithms, Soft Actor-Critic (SAC) showed the slowest convergence and achieved the weakest performance, falling 2% below the second-worst performer. In the latter, both RL controllers and NMPC were able to stabilize the process after reaching the inversion point, with no significant differences in IAE values. It is worth noting the main advantage of RL algorithms over NMPCs, which is computational cost. While NMPCs need to perform an optimization process at each iteration to determine the best controller action, RL algorithms have these actions stored in the Q-function or policy, significantly reducing computational cost.

Using a reward function that also values flow rate, the RL controllers completed the tasks with both stability and operational error minimization. A flow rate weight of 10% ($W_{Fin} = 0.1$) was sufficient to guarantee both a small influence on the error and the maximization of the reactor production.

Declarations

Conflict of interest: The authors declare no conflict of interest.

References

[1] Nian, R., Liu, J., Huang, B.: A review on reinforcement learning: Introduction and applications in industrial process control. Computers e Chemical Engineering **139**, 106886 (2020). https://doi.org/10.1016/j.compchemeng.2020.106886

[2] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA (2018)

[3] Shin, J., Badgwell, T.A., Liu, K.-H., Lee, J.H.: Reinforcement learning – overview of recent progress and implications for process control. Computers & Chemical Engineering **127**, 282–294 (2019). https://doi.org/10.1016/j.compchemeng.2019.05.029

[4] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing Atari with Deep Reinforcement Learning

[5] Tassa, Y., Erez, T., Todorov, E.: Synthesis and stabilization of complex behaviors through online trajectory optimization. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4906–4913 (2012). https://doi.org/10.1109/IROS.2012.6386025

[6] Vitthal, R., Rao, C.D.: Process control via artificial neural networks and learning automata, vol. 16, pp. 329–334. IEEE, ??? (1995). https://doi.org/10.1109/IACC.1995.465819. http://ieeexplore.ieee.org/document/465819/

[7] Alex, Alex, Aldrich, C., Aldrich, C.: Plant-wide neurocontrol of the tennessee eastman challenge process using evolutionary reinforcement learning. Proceedings of the Third International Conference on Intelligent Processing and Manufacturing of Materials (2001)

[8] Lee, J.M., Kaisare, N.S., Lee, J.H.: Choice of approximator and design of penalty function for an approximate dynamic programming based control approach. Journal of Process Control **16**, 135–156 (2006). https://doi.org/10.1016/j.jprocont.2005.04.010

[9] Tosukhowong, T., Lee, J.H.: Approximate dpamic programming based optimal control applied to an integrated plant with a reactor and a distillation column with recycle. AIChE Journal **55**, 919–930 (2009). https://doi.org/10.1002/aic.11805

[10] Hubbs, C.D., Li, C., Sahinidis, N.V., Grossmann, I.E., Wassick, J.M.: A deep reinforcement learning approach for chemical production scheduling. Computers e Chemical Engineering **141**, 106982 (2020). https://doi.org/10.1016/j.compchemeng.2020.106982

[11] Shao, Z., Si, F., Kudenko, D., Wang, P., Tong, X.: Predictive scheduling of wet flue gas desulfurization system based on reinforcement learning. Computers and Chemical Engineering **141**, 107000 (2020). https://doi.org/10.1016/j.compchemeng.2020.107000

[12] Hwangbo, S., Sin, G.: Design of control framework based on deep reinforcement learning and monte-carlo sampling in downstream separation. Computers and Chemical Engineering **140**, 106910 (2020). https://doi.

org/10.1016/j.compchemeng.2020.106910

[13] Yoo, H., Kim, B., Kim, J.W., Lee, J.H.: Reinforcement learning based optimal control of batch processes using monte-carlo deep deterministic policy gradient with phase segmentation. Computers and Chemical Engineering **144**, 107133 (2021). https://doi.org/10.1016/j.compchemeng.2020.107133

[14] Ma, Y., Zhu, W., Benton, M.G., Romagnoli, J.: Continuous control of a polymerization system with deep reinforcement learning. Journal of Process Control **75**, 40–47 (2019). https://doi.org/10.1016/j.jprocont.2018.11.004

[15] Manee, V., Baratti, R., Romagnoli, J.A.: Optimal strategies to control particle size and variance in antisolvent crystallization operations using deep rl. Chemical Engineering Transactions **86**, 943–948 (2021). https://doi.org/10.3303/CET2186158

[16] Alhazmi, K., Albalawi, F., Sarathy, S.M.: A reinforcement learning-based economic model predictive control framework for autonomous operation of chemical reactors. Chemical Engineering Journal **428**, 130993 (2022). https://doi.org/10.1016/j.cej.2021.130993

[17] Cassol, G.O., Campos, G.V.K., Thomaz, D.M., Capron, B.D.O., Secchi, A.R.: Reinforcement learning applied to process control: A van der vusse reactor case study. In: Eden, M.R., Ierapetritou, M.G., Towler, G.P. (eds.) 13th International Symposium on Process Systems Engineering (PSE 2018). Computer Aided Chemical Engineering, vol. 44, pp. 553–558. Elsevier, ??? (2018). https://doi.org/10.1016/B978-0-444-64241-7.50087-2. https://www.sciencedirect.com/science/article/pii/B9780444642417500872

[18] 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.: Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015). https://doi.org/10.1038/nature14236

[19] Silver, D., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic Policy Gradient Algorithms

[20] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning (2015)

[21] Fujimoto, S., van Hoof, H., Meger, D.: Addressing function approximation

error in actor-critic methods (2018)

[22] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor (2018)

[23] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., Levine, S.: Soft actor-critic algorithms and applications (2018)

[24] Mayne, D.H., Jacobson, D.Q.: Differential Dynamic Programming. American Elsevier Pub. Co., ??? (1970)

[25] Vusse., J.G.: Plug flow type reactor versus tank reactor **19**, 994–997 (1993)

[26] Engell, S., Klatt, K.-U.: Nonlinear control of a non-minimum-phase cstr, 2941–2945 (1993). https://doi.org/10.23919/ACC.1993.4793439

[27] Montanheiro, C.E.: Estudo de um controlador preditivo não linear multivariável baseado em redes neuronais. Master's thesis, Universidade Federal Do Rio de Janeiro, Rio de Janeiro (2014)

[28] Luz, E.M.L.: Desenvolvimento de controladores inteligenes para reator van de vusse. Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro (2018)