UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Jonny Farias Vicente Ferreira

Vinícius Ribeiro Vieira

Desenvolvimento e aplicação de uma metodologia para a implementação de um controlador baseado em Aprendizado por Reforço do nível de um tanque a partir de simulação

Rio de Janeiro

2022

JONNY FARIAS VICENTE FERREIRA
VINICIUS RIBEIRO VIEIRA

DESENVOLVIMENTO E APLICAÇÃO DE UMA METODOLOGIA PARA A IMPLEMENTAÇÃO DE UM CONTROLADOR BASEADO EM APRENDIZADO POR REFORÇO DO NÍVEL DE UM TANQUE A PARTIR DE SIMULAÇÃO

Trabalho de Conclusão de Curso em Engenharia Química submetida ao Corpo Docente da Escola de Química, como parte dos requisitos necessários à obtenção do grau Engenheiro Químico.

Orientador:

Prof. Bruno Didier Olivier Capron, D.Sc.

Rio de Janeiro

2022

JONNY FARIAS VICENTE FERREIRA
VINICIUS RIBEIRO VIEIRA

DESENVOLVIMENTO E APLICAÇÃO DE UMA METODOLOGIA PARA A IMPLEMENTAÇÃO DE UM CONTROLADOR BASEADO EM APRENDIZADO POR REFORÇO DO NÍVEL DE UM TANQUE A PARTIR DE SIMULAÇÃO

Trabalho de Conclusão de Curso em Engenharia Química submetida ao Corpo Docente da Escola de Química, como parte dos requisitos necessários à obtenção do grau Engenheiro Químico.

Aprovado por:

Kese Pontes Freitas Alberton, D. Sc.

José Rodrigues Torraca Neto, M. Sc.

Marcelo Mendes Viana, D. Sc.

Orientado por:

Bruno Didier Olivier Capron. D. Sc.

Rio de Janeiro

2022

RESUMO

Os avanços computacionais dos últimos 50 anos permitiram a elevação da inteligência artificial a novos patamares. Nesse contexto, surgiu o *reinforcement learning*: uma metodologia que permite aprender a ação a ser tomada e mapear situações de forma a realizar uma tarefa. Nos dias de hoje, essa metodologia pode ser encontrada em diversas áreas, como robótica, finanças, marketing e controle e automação, como ilustra este projeto. Este trabalho utiliza desta técnica para desenvolver uma metodologia em Python com as bibliotecas Numpy, Scipy e Pandas baseada em três passos para implementar um controlador de nível de tanque e aplicá-lo em simulações de sistemas diferentes aos quais fora treinado com o objetivo de avaliar a influência do aprendizado *online* no seu desempenho. O primeiro passo se tratou do treinamento *offline* do controlador. Já o segundo passo, foi realizado o treinamento *online* do controlador no sistema de pior desempenho. Por fim, no terceiro passo foi feito a comparação entre a resposta do controlador referente aos dois treinamentos. Os resultados evidenciam a robustez do controlador, adaptando-se aos diferentes cenários, assim como a melhor resposta que o aprendizado *online* pode proporcionar.

ABSTRACT

Computational advances in the last 50 years have allowed artificial intelligence to reach new standards. In this context, reinforcement learning emerged: a methodology that allows to learn the action to be taken and map situations in order to do a task. Nowadays, this methodology can be found in several areas, such as robotics, finance, marketing, and control and automation, as is shown in this project. This work uses this technique to develop a three step methodology in Python using Numpy, Scipy and Pandas libraries to implement a tank level controller and apply it in simulations of different systems to which it had been trained with the objective of evaluating the influence of online learning on their performance. The first step was offline training of the controller. In the second step, the controller was trained online in the worst performing system. Finally, in the third step, a comparison was made between the controller's response regarding the two trainings. The results show the robustness of the controller, adapting to different scenarios, as well as the improvement in performance that online learning can provide.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A implementação da automação na indústria tem sido cada vez mais recorrente, desde processos simples até o controle total da planta. Essa modernização foi realizada com o avanço da tecnologia, que possibilitou a rápida e eficaz obtenção de dados e o desenvolvimento de I.A., isto é, inteligência artificial. Com isso, foi possível diminuir o erro humano e tornar as operações mais seguras para todos.

Dentro do processo de automação, uma área que vem se destacando é o *Machine Learning* (ML), o aprendizado de máquinas. O objetivo deste ramo consiste em treinar o computador em uma determinada tarefa, onde ele melhorará sua performance através da experiência obtida (BI *et al*., 2019). Com o modelo treinado, pode-se aplicá-lo em diversos segmentos para otimização dos processos.

Este treinamento pode ser realizado de algumas formas que dividem o ML em 3 grandes áreas: supervisionado, não supervisionado e por reforço. De forma resumida, o que define o aprendizado supervisionado é a utilização de dados de treinamento rotulados onde a partir desses dados a inteligência irá classificar os itens em questão e, posteriormente, poderá aplicar essa classificação para itens fora dos conjuntos iniciais. Em contraponto, o aprendizado não supervisionado utiliza dados brutos, não classificados, e cria padrões chamados de *clusters* para a classificação.

Por fim, há o *reinforcement learning* (RL): metodologia à qual a máquina aprende a realizar uma determinada tarefa por tentativa e erro de acordo com sinais numéricos recebidos devido a elas. Esse processo se dá de forma contínua em um *loop* fechado, onde as ações tomadas influenciam nos *inputs* futuros do algoritmo (SUTTON e BARTO, 2014). Entretanto, é inviável a aplicação direta desse método no controle de processos, pois a geração de amostras seria excessivamente cara (NAGABANDI, 2018) e, devido ao controlador ser ignorante no começo do processo, ele pode tomar ações aleatórias potencialmente perigosas, ocasionando problemas de segurança ou que prejudiquem as especificações do produto. Por isso, deve-se realizar um treinamento a partir de simulações do processo real por meio de um modelo com o objetivo de conseguir um controlador suficientemente bom e robusto para uma implementação no processo real. No entanto, o modelo usado na simulação é por natureza incerto, o que pode impactar na performance do controlador no processo real.

Este projeto tem como objetivo propor e testar uma metodologia para o desenvolvimento de um controlador baseado em RL. Uma metodologia em Python em três passos é proposta: Primeiramente, é realizado um treinamento *offline* de controladores a partir de simulações que usam modelos nos quais erros foram acrescentados nos parâmetros considerados para o modelo representando a planta real. Em seguida, é simulada a implementação dos controladores obtidos na primeira etapa no modelo real para testar a robustez do controlador. Nesta etapa, o controlador continua seu aprendizado de forma *online* enquanto é avaliada sua influência no desempenho. O objetivo deste passo consiste em mitigar o efeito das incertezas do modelo e avaliar sua adaptabilidade às condições reais do processo de forma segura. Por fim, a partir do aprendizado adquirido neste último passo, é avaliado novamente o desempenho do controlador em uma nova simulação do processo real.

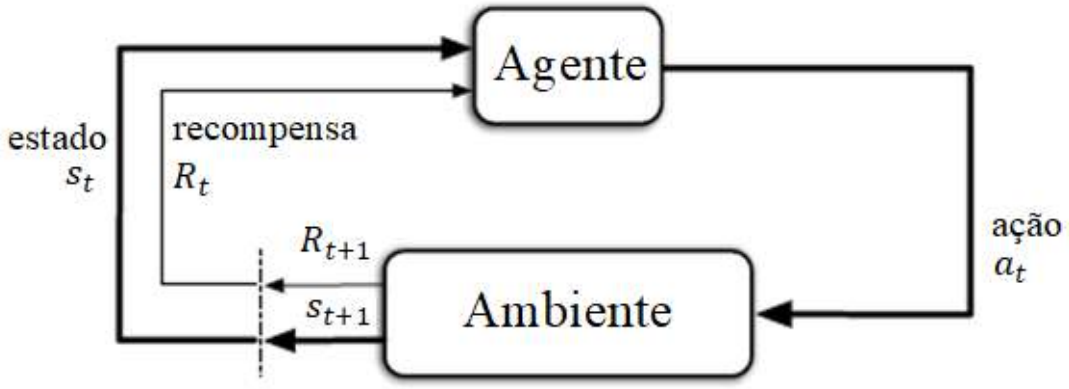
2 FUNDAMENTOS TEÓRICOS DO *REINFORCEMENT LEARNING*

Neste capítulo serão descritos os conceitos básicos do *Reinforcement Learning*, percorrendo os diversos pontos que levam ao seu sucesso como metodologia de *Machine Learning*, desde o conceito das recompensas às especificidades de discretização de um problema contínuo.

2.1 Agente e ambiente

O objetivo do *reinforcement learning* é que o agente aprenda a realizar uma determinada tarefa por meio das interações diretas com o ambiente. Nesse algoritmo, a cada iteração ele recebe uma representação do estado s_t, um conjunto de variáveis que fornece informações sobre o ambiente. Esse estado $s_t \in S$, sendo S é definido como o espaço de todos os estados possíveis. A partir disso, ele irá tomar uma ação a_t a cada intervalo de tempo discreto Δt. A função que define a maneira a qual o agente tomará suas decisões é chamada de *policy* π e se dá por meio da escolha de uma ação $a \in A(s)$, onde $A(s)$ é definido como o espaço de ações para o estado s (SUTTON e BARTO, 2014). Por sua vez, o ambiente sofrerá a ação do agente, evoluirá para um novo estado s_{t+1} e retornará uma recompensa R_{t+1} que deve indicar ao agente o quão boa foi a última ação tomada. A partir da consequência de sua ação que gerou esse novo estado e de sua recompensa, o agente irá adquirir seu aprendizado. O esquema resumido desse processo pode ser visualizado na Figura 1.

Figura 1 - Relação agente - ambiente



Fonte: Adaptado de Sutton e Barto (2014)

O objetivo de um algoritmo de *reinforcement learning* pode ser reformulado como determinar a *policy* que maximize a previsão do acúmulo das recompensas futuras.

2.2 Recompensa, retorno e taxa de desconto

Após tomar uma ação, o agente recebe uma recompensa (*reward*) R, podendo ela ser um valor numérico positivo ou negativo. Sendo assim, o objetivo de um algoritmo de *reinforcement learning* é maximizar o total dos *rewards*, definido como retorno G. O quanto o algoritmo será recompensado ou não é fruto de como será definida a função-recompensa, que pode ser função apenas do novo estado ou de uma comparação entre o estado no instante t e no instante $t + 1$.

Dependendo do sistema, a iteração pode não ter um evento que marque seu fim, como o controle de um processo contínuo de uma planta, por exemplo. Esses processos recebem o nome de tarefas contínuas. Além disso, pelo fato de o modelo do processo ser incerto, as representações futuras dos estados podem propagar essas incertezas conforme as iterações progridem no algoritmo. Dessa maneira, pode-se adotar uma taxa de desconto γ cujo valor seja entre 0 e 1 de forma que as recompensas mais distantes no futuro tenham um valor presente cada vez menor no retorno e a fórmula para o retorno G seja um somatório finito, limitando a função retorno de tarefas contínuas. É possível definir o retorno G_t de um dado episódio da seguinte forma:

$$G_t = R_{t+1} + \gamma \cdot R_{t+2} + \gamma^2 \cdot R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k \cdot R_{t+k+1} \tag{2.1}$$

Onde R_t é a recompensa no instante t. Para valores de γ próximos de zero, o agente irá considerar mais as recompensas imediatas até que para o valor 0 considere apenas a primeira recompensa, recebendo o nome de agente míope. Nesses casos, o objetivo do agente é maximizar apenas os primeiros retornos. Conforme aumenta-se o valor de γ, o agente passa a levar mais em conta as recompensas futuras, até que ele se iguale a 1 e considere igualmente todas as recompensas. Para essas situações, o futuro passa a ter mais peso no retorno final e o agente, portanto, passa a ter uma visão de mais longo prazo.

2.3 Processo de Decisão Markov

No campo dos processos estocásticos, área da probabilidade e estatística que estuda variáveis aleatórias indexadas a uma outra variável, existem os chamados Processos ou Cadeias de Markov. Sistemas com essa propriedade são aqueles que podem ser completamente caracterizados apenas pelo seu estado atual. Isso implica em dizer que o estado contém toda informação relevante e necessária para qualquer predição sobre o futuro, sendo o passado e o processo ao qual o levou ao presente irrelevantes (HULL, 2015). Introduzimos formalmente a formulação de um estado s_t que segue uma Cadeia de Markov por meio da seguinte equação:

$$\mathbb{P}[s_{t+1}|s_t] = \mathbb{P}[s_{t+1} \mid s_1, s_2, \ldots, s_t] \tag{2.2}$$

Onde a probabilidade de se alcançar o estado s_{t+1} dado o estado s_t é a mesma de alcançá-lo dado todos os outros estados anteriores. Assim fica evidente que a transição do estado s_t para o estado s_{t+1} é função apenas do estado s_t.

Como a maioria dos problemas de *reinforcement learning* são modelados como uma Cadeia de Markov, de forma a implementá-lo em um algoritmo de RL foi definido o Processo de Recompensa de Markov (MRP): uma extensão do Processo Markov com a presença de recompensa devido à transição dos estados (SILVER, 2015) e podendo conter ou não uma taxa de desconto γ.

O Processo de Decisão Markov (MDP) vai além do MRP: dessa vez, o processo além das recompensas, também lida com tomadas de decisões em um ambiente no qual todos os estados são Markov (SILVER, 2015). Dessa forma, a transição de estados passa a ser função, além do estado s, da ação a, conforme equação 2.3:

$$\mathcal{P} = \mathbb{P}[S = s_{t+1} \mid S = s_t, A = a_t] \tag{2.3}$$

Onde $\mathcal{P}$ é a probabilidade de transição entre o estado s_t e o estado s_{t+1} dado o estado s e a ação a. É possível afirmar que em um MDP a transição entre esses estados é consequência da decisão tomada, que é escolhida dentro de um espaço de ações $A(s)$ conforme apresentado no tópico 2.1, seguindo uma distribuição de probabilidades.

Para um problema que segue uma cadeia de Markov, o estado atual é a melhor referência possível para a tomada de decisão. Como o estado s_t contém todas as informações relevantes para caracterizar o estado, essa transição de estado será a mesma se for função apenas do estado atual ou de todo o histórico de estados, conforme equação 2.2.

2.4 Função valor

A função valor pode ser entendida como o retorno total esperado por se estar no estado s. Esse retorno esperado é função da *policy* π, pois diferentes maneiras de se tomar a decisão levam a diferentes retornos e uma função valor sempre estará associada a ela. Dessa forma, a função valor pode ser definida formalmente pela seguinte equação:

$$V^{\pi}(s) = E_{\pi}[G_t \mid s] \tag{2.4}$$

A função valor é estimada pela média dos retornos esperados para o estado s pois a recompensa obtida por visitar esse estado não será igual na maioria das vezes, já que se trata de processos estocásticos. Dessa forma, o objetivo de um algoritmo de *reinforcement learning* pode ser formulado como determinar dentro de todas as *policies* a que encontre a função valor ótima $V^{\pi*}(s)$, definida pela equação 2.5:

$$V^{\pi*}(s) = max(V(s)) \tag{2.5}$$

A partir da definição de G_t da equação 2.1, é possível reescrever a função valor da seguinte forma:

$$V^{\pi}(s_t) = E_{\pi}[R_{t+1} + \gamma \cdot V^{\pi} \cdot (s_{t+1}) \mid S = s] \tag{2.6}$$

Essa equação, também conhecida como equação de Bellman, evidencia que a função valor no estado s_t pode ser entendida como o valor esperado da soma entre a primeira recompensa e o valor presente da função valor do estado s_{t+1}.

2.5 Função Valor em espaços contínuos

Em problemas contínuos, como no caso de processos químicos, não é possível seguir exatamente a abordagem para determinação da função valor ótima descrita pela equação 2.5 sem nenhuma discretização dos estados pois teria um número infinito de estados para atribuir uma função valor. Além disso, os dados necessários para o treino do agente aumentam exponencialmente conforme aumenta o número de variáveis de estado e ação, originando a "maldição da dimensionalidade", que impede taxas de discretização altas. Uma forma de contornar esse problema é fazer uso da chamada Aproximação da Função Valor (VFA). A partir de tal técnica, é possível lidar com problemas contínuos ou que tenham um número muito grande de estados discretos (SILVER, 2015).

A VFA utiliza um vetor de características $\Phi(s)$ chamado de *feature*, que pode ser representado da seguinte maneira:

$$\Phi(s) = \begin{bmatrix} \Phi_1(s) \\ \Phi_2(s) \\ . \\ . \\ . \\ \Phi_n(s) \end{bmatrix} \tag{2.7}$$

Neste trabalho, cada coordenada do vetor *feature* foi definida por meio da ordenada de gaussianas que se diferem apenas do seu centro, seguindo a equação 2.8:

$$\Phi_i(s) = e^{\frac{-|s-c_i|^2}{2\cdot\sigma^2}} \tag{2.8}$$

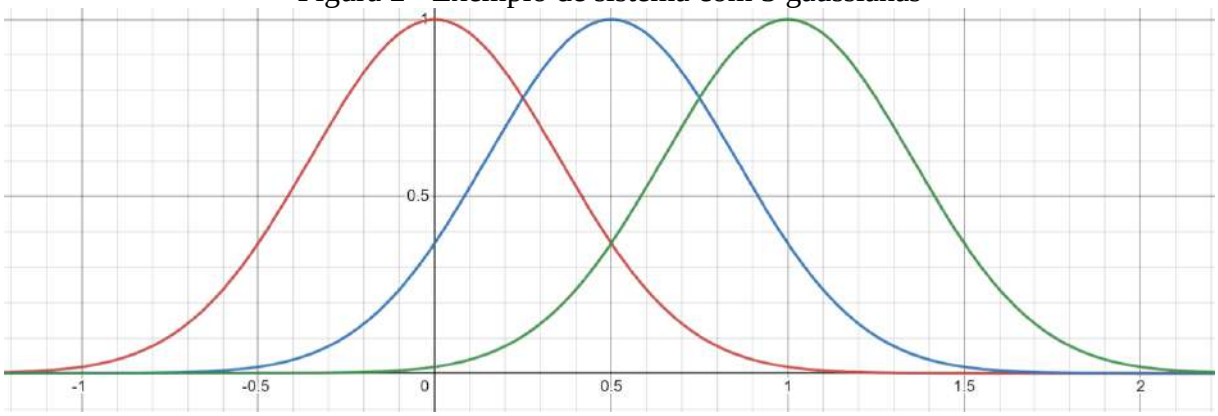
Onde $\Phi_i(s)$ é a coordenada i do vetor de *features*, s é o estado e c_i é o centro da gaussiana i. Também foi definido σ pela equação 2.9:

$$\sigma = \frac{1}{2 \cdot (j-1)} \tag{2.9}$$

Onde j representa o número de coordenadas do vetor $\Phi(s)$, com $j \neq 1$. Essa fórmula define σ como metade da distância entre duas Gaussianas adjacentes dentro do intervalo entre 0 e 1, situação que será apresentada neste trabalho.

Por exemplo, para um sistema que utilize de 3 gaussianas igualmente espaçadas entre 0 e 1 teremos o gráfico da figura 2:

Figura 2 - Exemplo de sistema com 3 gaussianas



O estado s será entendido, então, como o valor de i da maior coordenada do vetor de *features*, que representa o centro da gaussiana mais próximo ao estado. Assim, é possível que, para intervalos pré-definidos de um problema contínuo ou com um número elevado de estados, seja realizada uma aproximação da Função Valor.

A função valor aproximada será dada, portanto, por meio do seguinte produto escalar apresentada na equação 2.10:

$$V^{\pi}(s) = \theta_V^T \cdot \Phi(s) \tag{2.10}$$

Onde θ_V é definido como vetor de peso das *features.*

Também podemos definir uma outra função valor $Q(s, a)$ para o par estado-ação que representa o retorno esperado dado não só o estado s, mas também a ação a. Pode ser representada pela equação 2.11:

$$Q^{\pi}(s, a) = E_{\pi}[G_t \mid S = s, A = a] \tag{2.11}$$

Enquanto a função valor ótima para essa nova definição pode ser representada da seguinte maneira:

$$Q^{\pi*}(s,a) = max(Q^{\pi}(s,a)) \tag{2.12}$$

2.6 *Policy*

A ideia por trás de todo algoritmo de *reinforcement learning* é a sucessão de avaliação e constante melhoria da *policy* π. Ela é uma função a qual representa a ação que será tomada pelo agente em função do estado em que ele se encontra. Pode ser entendida também como a distribuição de probabilidades das ações para um dado estado, conforme ilustra a seguinte equação:

$$\pi(a|s) = \mathbb{P}[A = a \mid S = s] \tag{2.13}$$

Como a partir da ação tomada o sistema evoluirá do estado s_t para o estado s_{t+1}, $\mathbb{P}$ pode ser entendido como uma função de transição de estados. Além disso, é possível notar que ela não é função do tempo, apenas do estado, que em um MDP contém todas as informações que caracterizam a evolução do processo. No caso deste trabalho, a *policy* π é modelada de forma determinística a partir da ação, não sendo uma função de distribuição de probabilidades. Esta modelagem será abordada no capítulo 4.

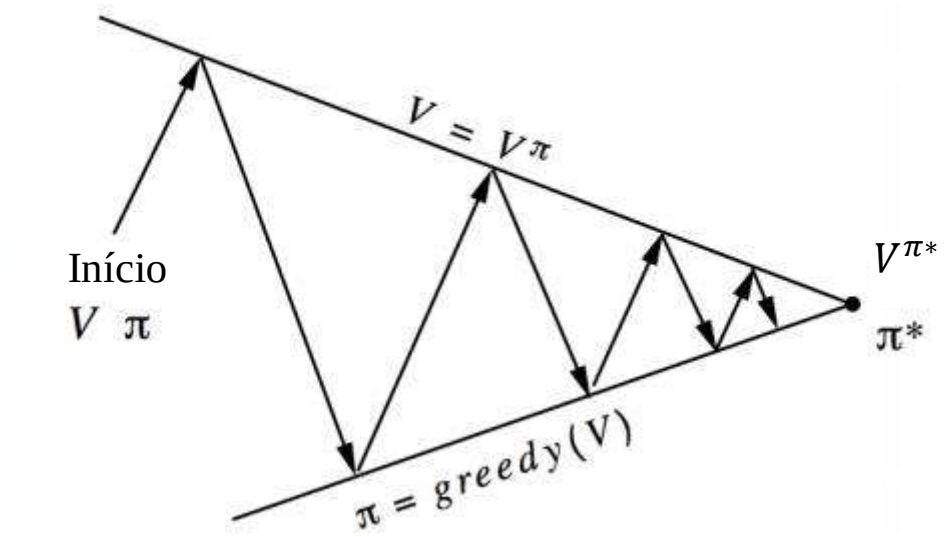
O principal objetivo do algoritmo é justamente encontrar um vetor de parâmetros θ_π ideal por meio de *reinforcement learning* para que o controlador tome as melhores ações possíveis e atinja a função valor ótima definida pela equação 2.5 ou 2.12. As *policies* que permitem encontrar $V^{\pi*}(s)$ ou $Q^{\pi*}(s,a)$ são chamadas de *policies* ótimas, definida como:

$$\pi^*(s) = argmax(V^{\pi*}(s)) \tag{2.14}$$

$$\pi^*(s,a) = argmax(Q^{\pi*}(s,a)) \tag{2.15}$$

A obtenção da *policy* ótima definida nessas equações se dá por meio da resolução iterativa da equação de Bellman. Esse processo se dá em sucessivas avaliações e melhorias de π até que haja sua convergência. A figura 3 ilustra esse processo iterativo:

Figura 3 – Processo iterativo de convergência da *policy*



Fonte: Adaptado de Silver (2015)

Para a avaliação da *policy*, é feito, a partir da combinação das equações 2.1 e 2.4, uma estimativa para $V(s)$. A partir dessa nova estimativa, o agente atuará, inicialmente, com um pouco de exploração para evitar que haja convergência em um mínimo local, porém de forma cada vez mais ambiciosa (*greedy*), para que seja possível atingir π^*.

2.7 Exploração

Para que seja possível evitar soluções subótimas para *policy*, é necessário que o algoritmo percorra suficientemente o espaço estado-ação associado ao sistema. Entretanto, para que o agente aja de forma cada vez mais ambiciosa, essa exploração deve ser feita decaindo com o tempo. Dessa maneira que será possível atingir a *policy* ótima. Para isso, foi adotada a abordagem de exploração gaussiana para visitação dos estados, onde um parâmetro ε decairá conforme os estados são visitados. O valor de ε é definido da seguinte maneira:

$$\varepsilon = \frac{N_0}{N_0 + N(s)} \tag{2.16}$$

Onde N_0 é uma constante definida previamente que define a taxa de diminuição de ε e $N(s)$ é o número de vezes que o centro da gaussiana mais próximo ao estado s foi visitado. Para seu uso no algoritmo, é necessário definir, também, uma constante de exploração c_e. Dessa forma, a exploração será inserida no aprendizado por meio da equação 2.17:

$$a = \pi + \mathcal{N}(0, c_e\varepsilon) \tag{2.17}$$

Onde $\mathcal{N}(0, c_e\varepsilon)$, chamado de ruído de exploração, representa um número aleatório de uma distribuição normal de média 0 e variância $c_e\varepsilon$.

A equação 2.17 torna evidente que a exploração decairá com o tempo pois, conforme os estados vão sendo visitados repetidamente, o valor de ε diminui e, junto dele, a variância da distribuição normal. Com uma menor variância e média 0, o valor de $\mathcal{N}(0, c_e\varepsilon)$ tenderá a 0 em um tempo suficientemente longo.

2.8 *Temporal-Difference Learning*

A ideia por trás de todo algoritmo de *reinforcement learning* é a sucessão de avaliação e constante melhoria da *policy*. O *Temporal-Difference Learning* (*TD Learning*) surge como um método de avaliação da *policy* que permite que o algoritmo atualize a função valor $V(s)$ a cada intervalo de tempo Δt e resolva a equação 2.14. Para problemas de espaços discretos, essa equação é facilmente resolvida pois trata-se de encontrar um valor máximo dentro de um conjunto de valores. Entretanto, para problemas contínuos, surgiria um problema de *Non Linear Programming* (NLP) de difícil resolução.

A metodologia *TD Learning* então apresenta-se como uma solução pois permite que não seja necessário esperar até o final do episódio para que o algoritmo adquira aprendizado, apenas estimativas do futuro, definidas como *bootstrapping* (SILVER, 2015). Isso é ideal ao se tratar com problemas contínuos de *reinforcement learning*, visto que não existe um final. A atualização da função valor será feita a partir da estimação do retorno naquele instante de tempo, chamada de *TD Target*, dado pela equação 2.18:

$$TD\ Target = R_{t+1} + \gamma \cdot V(s_{t+1}) \tag{2.18}$$

Ao comparar a nova estimação da função valor com seu valor anterior, é possível definir o *TD Error* δ_t:

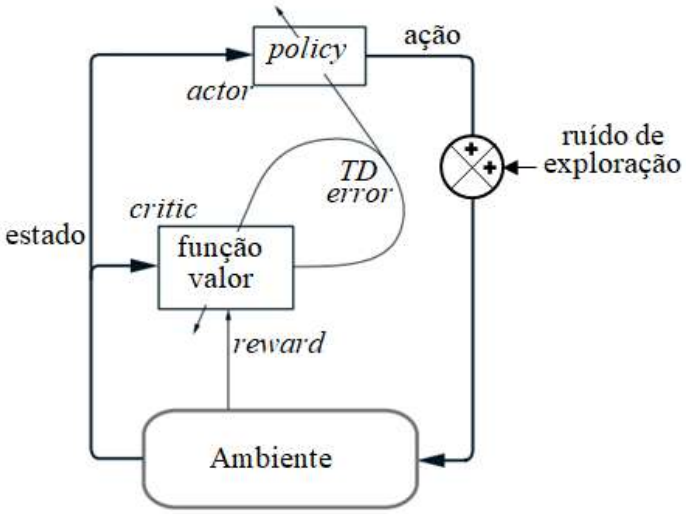
$$\delta_t = TD\ Target - V(s_t) \tag{2.19}$$

O *TD Error*, também conhecido como erro de diferença temporal, representa a diferença entre a estimação mais recente da função valor dada pelo *TD Target* e o seu valor atual.

2.9 Algoritmo *Actor-Critic*

O algoritmo *Actor-Critic* é uma metodologia de Diferença Temporal que tem estruturas separadas na memória explicitamente para a *policy* e a função valor (SUTTON e BARTO, 2014). Essas estruturas são chamadas de *Actor* e de *Critic*. Isso permite que a cada iteração, após iniciá-la no estado s_t, tomar a ação a_t seguindo a *policy* π associando a um ruído de exploração e atingir o estado s_{t+1}, o *Actor* atualize e melhore a *policy* e o *Critic* avalie a *policy* e atualizando a função valor, caracterizando o *TD Learning*. Tal algoritmo pode ser ilustrado pela Figura 4.

Figura 4 - Esquema representativo do algoritmo *Actor-Critic*



Fonte: Adaptado de Surton e Barto (2014)

A utilização da exploração como um ruído no espaço de ações foi a abordagem utilizada nesse trabalho, mas também seria possível aplicá-la sobre os próprios parâmetros da *policy*.

O *Critic* atua por meio da atualização do vetor de parâmetros θ_V de forma a minimizar o *TD Error*. Essa atualização é ilustrada pela equação 2.20:

$$\theta_{V_{t+1}} \leftarrow \theta_{V_t} + \alpha_V \cdot z_{t+1} \cdot \delta_t \tag{2.20}$$

Onde α_V é chamado de taxa de aprendizado e pode ser um valor constante ou dinâmico e z é definido como traço de elegibilidade. No caso deste trabalho, para o aprendizado foi adotado um valor dinâmico de α_V dado pela seguinte equação:

$$\alpha_V = \frac{1}{N(s)} \tag{2.21}$$

O traço de elegibilidade z representa uma forma de atribuir responsabilidade para o valor da última recompensa entre os estados mais recentes e os estados mais visitados. Com o tempo, o valor de z decairá seguindo um fator $\gamma\lambda$, onde λ representa o fator de decaimento do traço de elegibilidade. Visto que o problema deste trabalho é contínuo, o traço de elegibilidade z será um vetor semelhante ao vetor de *features* e pode ser obtido da seguinte maneira:

$$z_{t+1} = \gamma \cdot \lambda \cdot z_t + \Phi(s_t) \tag{2.22}$$

Enquanto isso, o *Actor* irá atuar por meio da atualização do vetor de parâmetros θ_π de forma a maximizar a função valor e, portanto, atingir a *policy* ótima, da seguinte forma:

$$\theta_{\pi_{t+1}} = \theta_{\pi_t} + \alpha_\pi \cdot \delta_t \cdot (a - \theta_{\pi_t}^T \cdot \Phi(s_t)) \cdot \Phi(s_t) \tag{2.23}$$

Onde α_π representa um outro parâmetro de aprendizado que ajusta o passo de atualização na direção do valor ótimo. Seguindo esse método iterativo, a *policy* ótima só poderá ser obtida se os pares estado-ação forem suficientemente explorados.

3 REVISÃO BIBLIOGRÁFICA

De maneira simplificada, pode-se dizer que em um problema de RL há um agente que interage com o ambiente: observa o estado, realiza uma ação e recebe uma recompensa (SHIPMAN; COETZEE, 2019). Essa interação com o ambiente tem o objetivo de maximizar o retorno total das recompensas. A partir dos retornos, o agente obtém o aprendizado. Se comparado aos modelos tradicionais de PID, o RL possui a grande vantagem de permitir a adaptação rápida do controlador às mudanças dos parâmetros do sistema. Além disso, o PID apresenta resultados insatisfatórios quando há tempo morto na planta (WANG; CHENG; SUN, 2007). Esse valor pode chegar a vários minutos em plantas de grande escala, sendo 1 minuto o mais comum (CALLENDER *et al*, 1936).

De forma geral, as aplicações de RL possuem uma abordagem ampla. Sua metodologia foi utilizada desde o controle de sistemas de armazenamento de energia térmica resfriados em edifícios comerciais (HENZE; SCHOENMANN, 2003), até o seu uso rastreamento de trajetória de veículos aéreos não tripulados (CHOI; KIM; JIN KIM, 2017). Em um outro trabalho, Lilicrap et al., 2016, desenvolveram o DDPG (*Deep Deterministic Policy Gradient)*, onde obteve um algoritmo capaz de resolver diversos problemas clássicos, necessitando de menos treinamento se comparado a outros modelos.

Na indústria química, diversos estudos abordaram a otimização de controladores via RL para seus fins. WANG *et al*, 2007, por exemplo, sintonizaram um PID se utilizando de um *Actor-Critic* e demonstraram que esse método supera o mesmo controlador sintonizado por Ziegler-Nichols. Também já foi simulado o uso de *reinforcement learning* em diferentes processos na indústria, como por exemplo foi demonstrado por HOSKINS; HIMMELBLAU, 1992 onde simularam a manutenção de temperatura de um reator CSTR e obtiveram resultados muito próximos ao uso do controlador PID na mesma situação. O aprendizado por reforço nessa área também pode ser ilustrado pelo estudo desenvolvido por ALHAZMI; ALBALAWI; SARATHY, 2022 no qual é integrado o RL ao controle preditivo de modelo econômico, uma metodologia promissora na otimização de processos dinâmicos. Nesse estudo, durante a simulação de oxidação em um CSTR, notou-se que o agente melhorou significativamente o rendimento do processo. Já NIAN, 2020 testou e constatou, em uma escala piloto, a otimização de um sistema de bombas de uma planta gerenciada por RL.

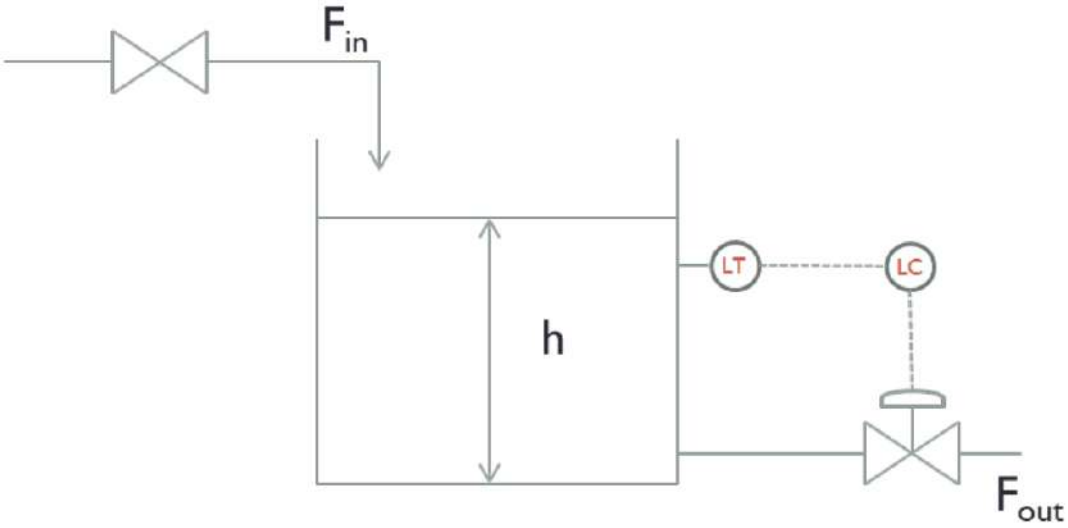
Por fim, pode-se citar o trabalho de LIMA, 2018 que utilizou o aprendizado por reforço com uso de dados simulados do processo e constante atualização de sua *policy* para controlar o nível de um tanque, obtendo o desempenho similar de um controlador PI. O presente trabalho também apresenta um controlador baseado em um algoritmo de RL para o mesmo tanque, porém utilizando para o treinamento e as simulações um menor tempo de amostragem. Isso permite simular o processo de forma mais fidedigna por atualizar o nível do tanque e a abertura da válvula em um intervalo de tempo menor. Além disso, a partir de um treinamento *offline* utilizando um modelo incerto da planta, é avaliado a robustez do controlador. Também foi realizado um teste de aprendizado *online* onde o controlador manteve-se aprendendo durante a simulação de forma a melhorar o seu desempenho. Com esse aprendizado obtido, realizou-se uma simulação final do processo real para avaliar novamente a robustez do controlador.

4 METODOLOGIA

4.1 Apresentação do sistema

O desenvolvimento e as posteriores simulações do controlador terão como base o sistema apresentado na Figura 5:

Figura 5 – Diagrama do sistema



Fonte: CAPRON, 2022

Este sistema possui um tanque alimentado por uma vazão contínua F_{in} que é sujeita a diversos distúrbios e a meta é controlar o nível do tanque h (variável controlada) de forma a mantê-lo no *set-point*. Para isso, o controlador atuará sobre a abertura da válvula de saída X_V (variável manipulada) de forma a ajustar a vazão de saída F_{out} de tal maneira que o sistema se mantenha no *set-point*. A modelagem adotada para a vazão de saída é dada pela seguinte equação:

$$F_{out} = C_V \cdot X_V \cdot \sqrt{h} \tag{4.1}$$

Onde C_V é a constante da válvula com valor de 282,84 $m^{2,5}h^{-1}$ e X_V a abertura da válvula de saída em percentual. Escrevendo o balanço de massa do tanque onde a densidade do fluido ρ é constante temos:

$$\rho \cdot \frac{dV}{dt} = \rho \cdot F_{in} - \rho \cdot F_{out} \tag{4.2}$$

Para esse sistema, adotamos a área transversal do tanque A constante. Dessa forma, ficamos com a seguinte equação após substituir a equação 4.1 na equação 4.2:

$$\frac{dh}{dt} = \frac{F_{in}}{A} - \frac{C_V \cdot X_V \cdot \sqrt{h}}{A} \tag{4.3}$$

4.2 Definições de parâmetros iniciais para aprendizado

Inicialmente, foi necessário definir diversos parâmetros e funções para que o aprendizado fosse possível, como o intervalo de tempo de amostragem Δt, o número de episódios de simulação, o número de iterações por episódio *etc*. Optou-se por utilizar um $\Delta t = 0{,}0001\ h$, ou $0{,}36\ s$, para que fosse possível avaliar o uso do controlador em um caso mais próximo da realidade na indústria, onde seria necessária uma rápida resposta a distúrbios. Desse modo, também foi necessário um elevado número de episódios para aprendizado e de iterações por episódio para que fosse possível armazenar dados suficientes. Foram utilizados os seguintes valores:

Tabela 1 - Distribuição das iterações

Número de episódios	Número de iterações
200	10.000

Totalizando 2.000.000 de pontos no processo de aprendizado.

Além disso, também foram pré-definidos constantes alguns outros parâmetros do algoritmo:

Tabela 2 - Valores de Parâmetros constantes

Parâmetro	Valor
γ	1
λ	0,5
α_π	1

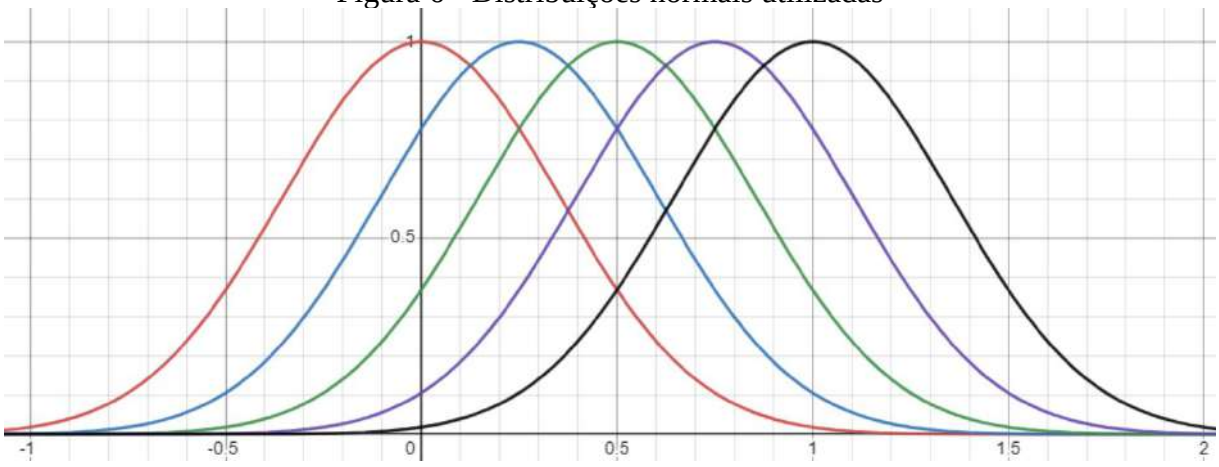
Visto que a simulação é finita devido à quantidade de pontos pré-definidos para aprendizado, foi utilizado para taxa de desconto γ o valor de 1.

4.3 Discretização dos estados

Por se tratar de um problema contínuo onde o estado é o nível do tanque, foi feito uso da técnica de VFA. Para tal, foi feito uso de um vetor de características $\Phi(s)$ de número de coordenadas $j = 5$, cada uma representando uma função normal diferente cuja característica era todas serem igualmente espaçadas dentro do intervalo 0 m e 1 m – os limites superior e inferior do nível do tanque.

Podemos representar em um gráfico as cinco funções normais, conforme ilustra a figura 6.

Figura 6 - Distribuições normais utilizadas



4.4 Definição da ação

Para que a partir da *policy* π obtenhamos a ação, conforme apresentado na equação 2.17, é necessário o vetor θ_π, que é inicializado com todas suas coordenadas zero e atualizado a cada iteração.

O aprendizado por meio da metodologia de *reinforcement learning* utilizado envolveu o uso da exploração gaussiana. Para tal, foi adotado um valor de $N_0 = 1000$, sendo, portanto, ε dado da seguinte forma:

$$\varepsilon = \frac{1000}{1000 + N(s)} \tag{4.4}$$

Além de ε, é necessária uma constante de exploração c_e, definida com o valor de 0,05, de forma que ficasse coerente com a ordem de grandeza dos valores possíveis para X_V. Dessa forma, a ação $a(s)$ foi definida como a variação que se deve fazer na abertura da válvula e é calculada seguindo a equação 4.5:

$$a(s) = \theta_\pi^T \cdot \Phi(s) + \mathcal{N}(0, c_e\varepsilon) \tag{4.5}$$

De posse da ação, a nova abertura de válvula é definida por meio da seguinte equação:

$$X_{V_{t+1}} = X_{V_t} + a \tag{4.6}$$

4.5 Definição da função-recompensa

A função recompensa foi definida de forma que o agente fosse recompensado caso o nível se aproximasse do *set-point* em relação ao estado anterior. A recompensa *R* é representada pela seguinte equação:

$$R = \left|h_{sp} - h_t\right| - |h_{sp} - h_{t+1}| \tag{4.7}$$

Pode-se observar que o valor da recompensa será a diferença entre as distâncias ao set-point no instante t e no instante $t + 1$.

4.6 Algoritmos

Todos os algoritmos que serão apresentados tiveram início do estado estacionário.

4.6.1 Algoritmo de aprendizado

O resultado desejado do algoritmo de aprendizado é a geração de um vetor de parâmetros θ_π e, portanto, de uma *policy* π que seja funcional e possa ser utilizada posteriormente em outras simulações e até mesmo na vida real. Dessa forma, nosso objetivo a cada iteração é atualizá-la até sua convergência após percorrer uma quantidade suficiente de estados. Assim, o algoritmo de aprendizado para cada episódio pode ser resumido da seguinte maneira:

$$\delta \leftarrow R + \gamma \cdot \theta_V^T \cdot \Phi(s_{t+1}) - \theta_V^T \cdot \Phi(s_t) \tag{4.8}$$

$$z \leftarrow \gamma \cdot \lambda \cdot \mathrm{z} + \Phi(s_t) \tag{4.9}$$

$$\theta_V \leftarrow \theta_V + \alpha_V \cdot z \cdot \delta \tag{4.10}$$

$$\theta_\pi \leftarrow \theta_\pi + \alpha_\pi \cdot \delta \cdot (a - \theta_\pi^T \cdot \Phi(s_t)) \cdot \Phi(s_t) \tag{4.11}$$

$$s_t \leftarrow s_{t+1} \tag{4.12}$$

$$a_t \leftarrow a_{t+1} \tag{4.13}$$

$$N(s_t) \leftarrow N(s_t) + 1 \tag{4.14}$$

Tabela 3 – Descrição das equações do algoritmo

Equação	Descrição
4.8	Atualização do TD Target
4.9	Atualização do traço de eligibilidade
4.10	Atualização do vetor de parâmetros da Função Valor
4.11	Atualização do vetor de parâmetros da *Policy*
4.12	Atualização do estado atual
4.13	Atualização da ação mais recente
4.14	Atualização do contador de estados

Ao final de cada episódio, os valores de nível, abertura de válvula e vazão de entrada são reiniciados e ao final de cada iteração o vetor θ_π é atualizado.

Para o treinamento, foi utilizado um tanque com as seguintes dimensões, constante da válvula C_V e constante de exploração c_e:

Tabela 4 - Parâmetros do tanque no aprendizado

Parâmetro (unidade)	Valor
A (m²)	3
h (m)	1
C_V ($m^{2,5}h^{-1}$)	282,84
c_e (adimensional)	0,05
h_0 (m)	0,5
F_{in0} (m^3/h)	100
X_{V0} (%)	50

4.6.2 Algoritmo da simulação da implementação do controlador pré-treinado no processo real (sem aprendizado *online*)

A finalidade da simulação é avaliar a resposta do controlador a um degrau de 50% na vazão de entrada F_{in} a partir do que aprendeu em seus treinamentos. Cada controlador foi

testado através de uma simulação de 8.000 iterações em um episódio único, onde as simulações se iniciaram no estado estacionário e foi aplicado o distúrbio após 25,2 segundos. Dessa forma, o algoritmo das simulações não apresenta aprendizado e não há atualização de valores ao final de cada iteração.

Foram realizadas 5 simulações sem aprendizado *online*, cujos tanques apresentam as seguintes dimensões e constantes da válvula C_V:

Tabela 5 - Parâmetros dos tanques das simulações sem aprendizado *online*

Simulação	A (m^2)	h (m)	C_V ($m^{2,5}h^{-1}$)	h_0 (m)	F_{in0} (m^3/h)	X_{V0} (%)
1	3	1	282,84	0,5	100	50
2	4,5	1	424,26	0,5	200	50
3	1,5	1	424,26	0,5	200	50
4	4,5	1	141,42	0,5	50	50
6	4,5	1	141,42	0,5	50	50

4.6.3 Algoritmo da simulação da implementação do controlador pré-treinado no processo real (com aprendizado *online*)

Fundamentalmente, o algoritmo da simulação com aprendizado *online* se assemelha ao apresentado no algoritmo de treinamento do tópico 4.6.1. As diferenças se dão na aplicação de um degrau de 50% na vazão de entrada F_{in} após 25,2 segundos, na constante c_e com o valor de 0,01 para que ainda haja exploração mas sem saturação da válvula após a aplicação do degrau e na iteração em 2.000.000 de pontos em um episódio único em vez de 200 episódios de 10.000 iterações para melhor avaliação da resposta do controlador ao degrau aplicado.

A simulação com aprendizado *online* - simulação 5 - foi realizada em um tanque de dimensões e constante da válvula C_V iguais ao da simulação 4 e 6.

Tabela 6 - Parâmetros do tanque da simulação 5 com aprendizado *online*

Parâmetro (unidade)	Valor
A (m²)	4,5
h (m)	1
C_V ($m^{2,5}h^{-1}$)	141,42
N_0 (adimensional)	1.000
c_e (adimensional)	0,01
h_0 (m)	0,5
F_{in0} (m^3/h)	50
X_{V0} (%)	50

5 RESULTADOS E DISCUSSÃO

Neste capítulo serão apresentados os resultados do aprendizado e das simulações para as quais o controlador foi utilizado.

5.1 Aprendizado

O aprendizado foi realizado com os seguintes parâmetros:

Tabela 7 - Parâmetros do aprendizado

Parâmetro (unidade)	Valor
Δt (h)	0,0001
N_0	1.000
c_e	0,05
h_{sp} (m)	0,5
γ	1
λ	0,5
α_π	1

Com esses parâmetros foi possível obter o vetor θ_π abaixo, que foi utilizado como o vetor de parâmetros para as posteriores simulações.

$$\theta_\pi = \begin{bmatrix} -0{,}0007185 \\ -0{,}0060959 \\ -0{,}0004883 \\ 0{,}0097080 \\ 0{,}0028049 \end{bmatrix}$$

5.2 Simulação 1

Nesta simulação envolvendo um tanque de 1 m de altura e 3 m^2 de área transversal foi aplicado o degrau de 50% na vazão de entrada F_{in} em um tanque de mesma área transversal A e constante de válvula C_V que o tanque onde o controlador foi treinado.

Foram obtidos os gráficos do nível do tanque (variável controlada) e da abertura da válvula (variável manipulada) respectivamente nas figuras 7 e 8.

Figura 7 - Nível do tanque na simulação 1

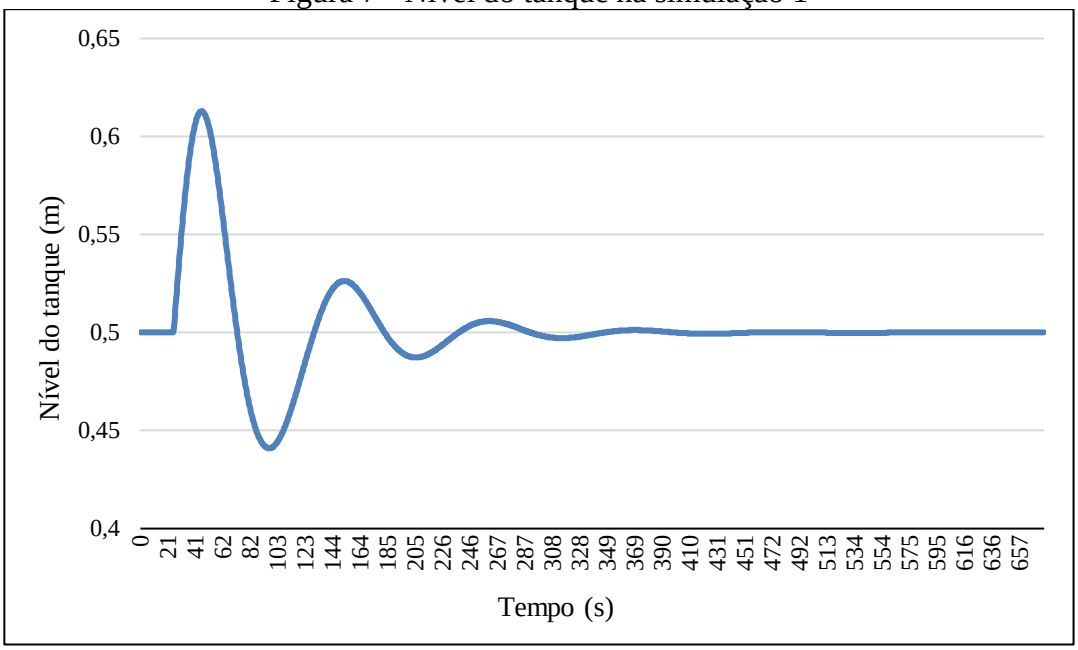
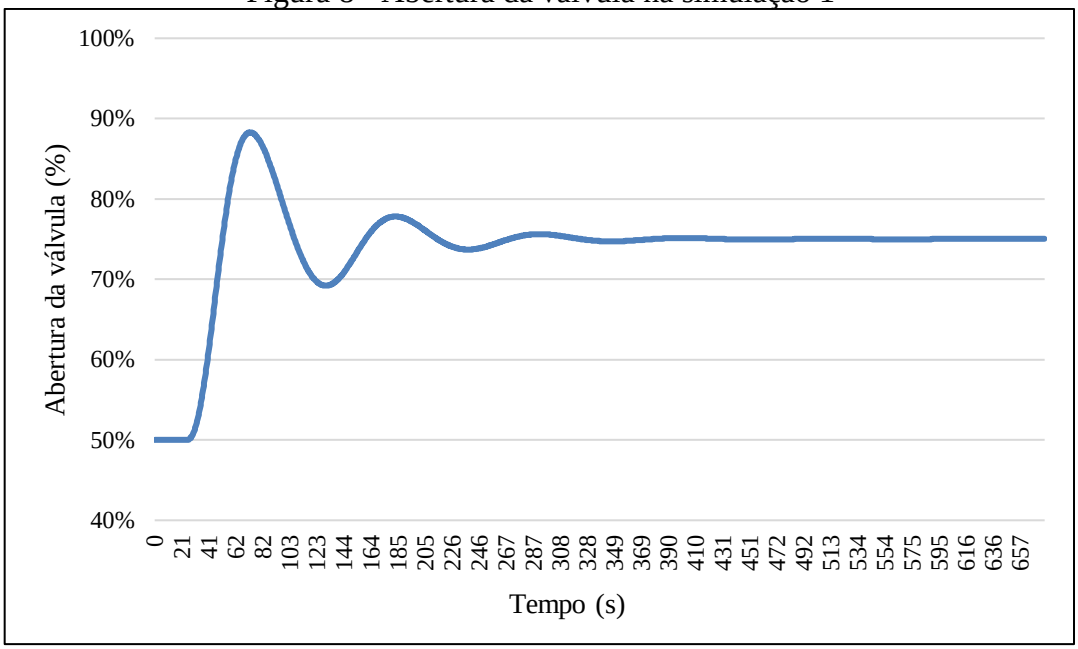


Figura 8 - Abertura da válvula na simulação 1



Observa-se que a *policy* se mostrou eficaz e permitiu que o sistema atingisse novamente o *set-point* em pouco mais de 5 minutos e dentro dos limites físicos de abertura da válvula.

5.3 Simulação 2

A simulação 2 foi realizada em um tanque de área transversal 50% maior, assim como com uma válvula de constante também 50% maior. A partir da simulação foram plotados os seguintes gráficos para observar a resposta do controlador ao degrau nesse sistema diferente ao qual foi treinado.

Figura 9 - Nível do tanque na simulação 2

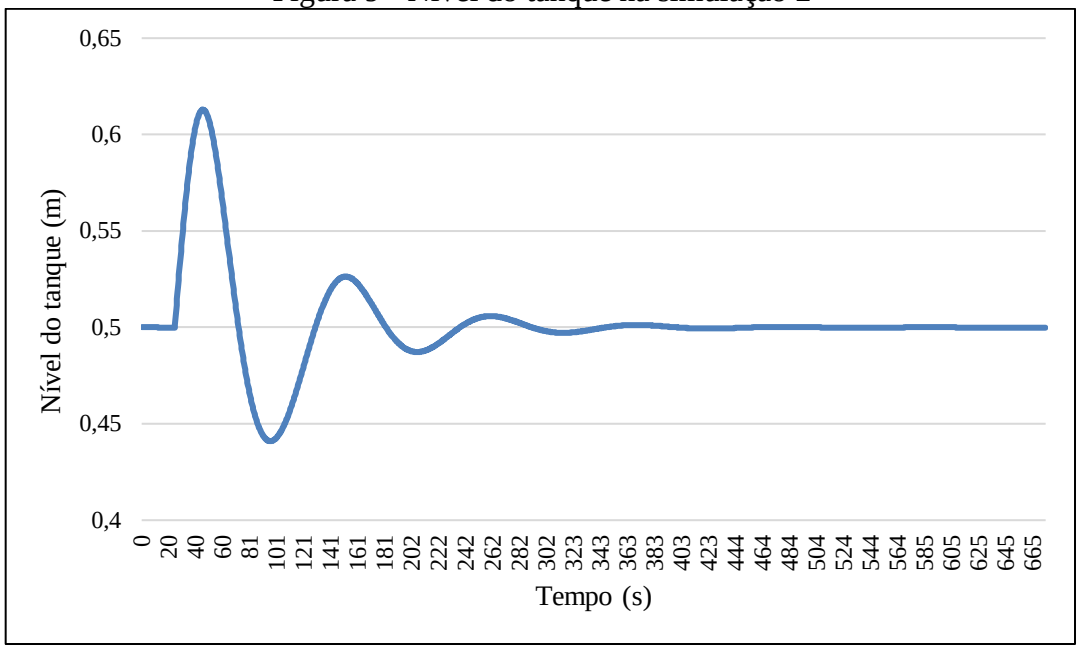
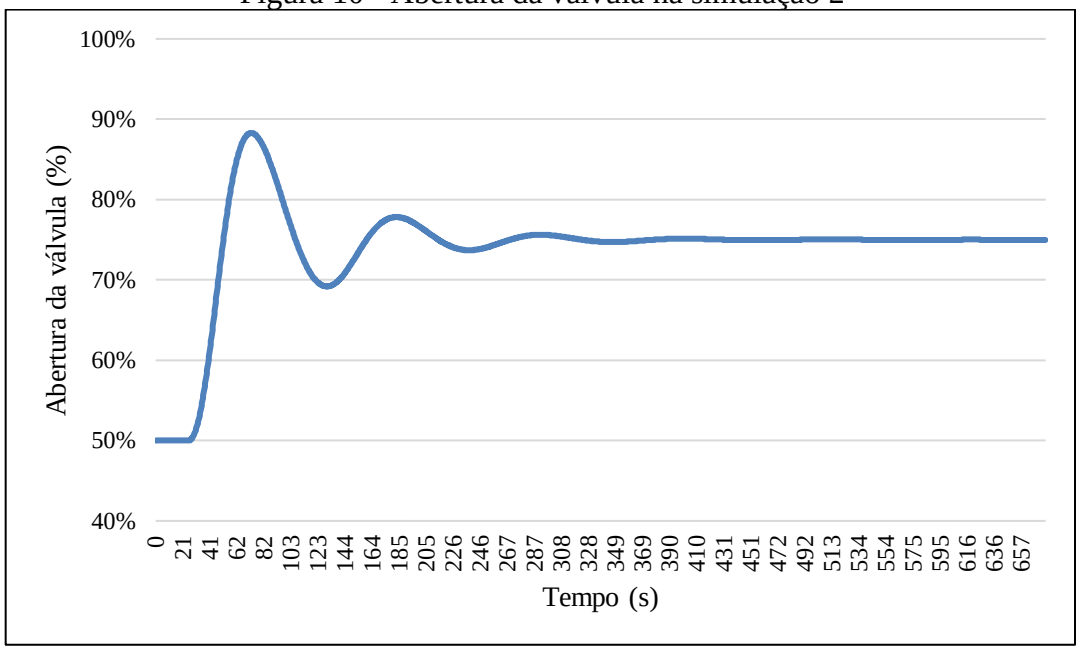


Figura 10 - Abertura da válvula na simulação 2



Pode-se observar que, apesar de sua aplicação em um sistema diferente ao qual foi treinado, o controlador obteve êxito em alcançar o *set-point* de 0,5 m de altura do nível, levando praticamente o mesmo tempo que a simulação anterior para isso.

5.4 Simulação 3

A simulação 3 foi realizada em um tanque de área transversal 50% menor e em uma válvula de constante 50% maior.

Por meio do mesmo vetor de parâmetros θ_π, foram obtidas as seguintes curvas:

Figura 11 - Nível do tanque na simulação 3 versus simulação 1

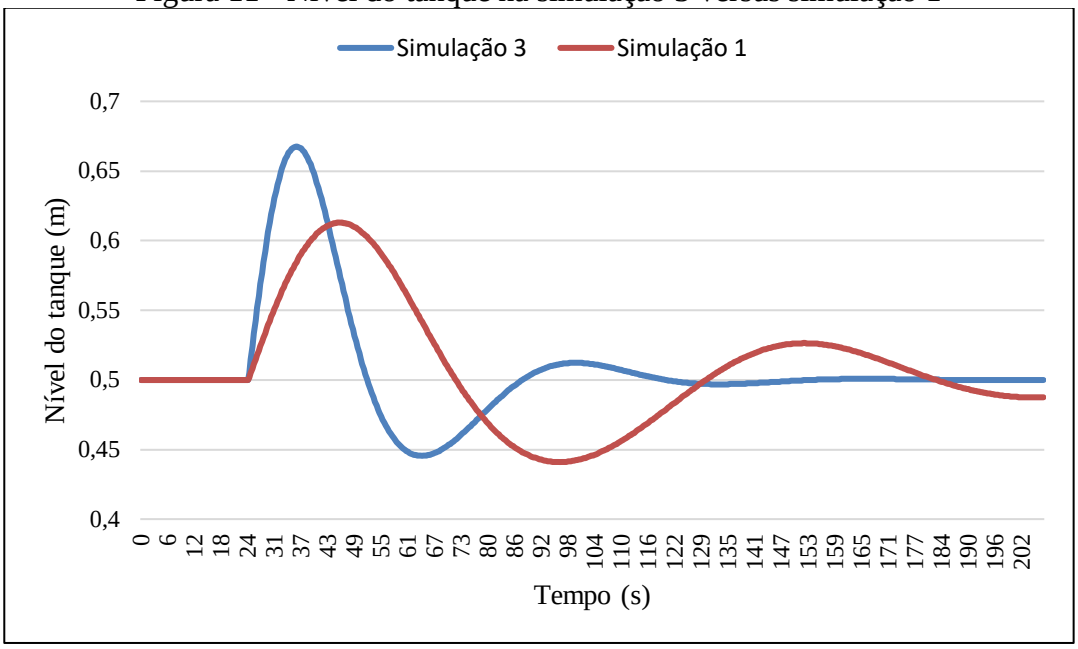
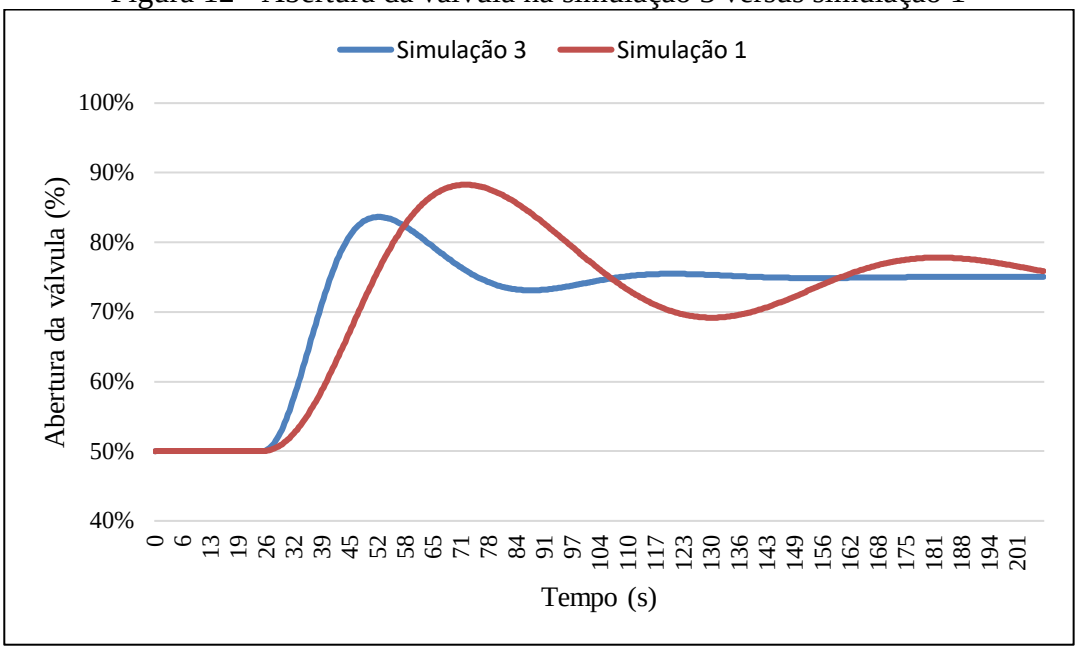


Figura 12 - Abertura da válvula na simulação 3 versus simulação 1



É possível notar que a atuação do controlador nesse sistema diferente ao qual foi treinado obteve um melhor desempenho e uma maior robustez. Isso se deve ao fato de a combinação entre uma redução da área transversal do tanque e um aumento da constante de válvula potencializarem a ação do controlador no segundo termo do balanço de massa, agindo como um aumento no ganho. Isso pode ser demonstrado da seguinte forma:

$$\frac{C_V \cdot 1{,}5 \cdot X_V \cdot \sqrt{h}}{A \cdot 0{,}5} = \frac{3 \cdot C_V \cdot X_V \cdot \sqrt{h}}{A} \tag{5.1}$$

5.5 Simulação 4

Esta simulação foi realizada em condições contrárias à simulação 3: um tanque de área transversal 50% maior com uma válvula de constante 50% menor. Após a aplicação de um degrau também de 50% na vazão de entrada, foram obtidas as seguintes curvas para a altura do nível do tanque e da abertura da válvula contra o tempo:

Figura 13 - Nível do tanque na simulação 4 versus simulação 1

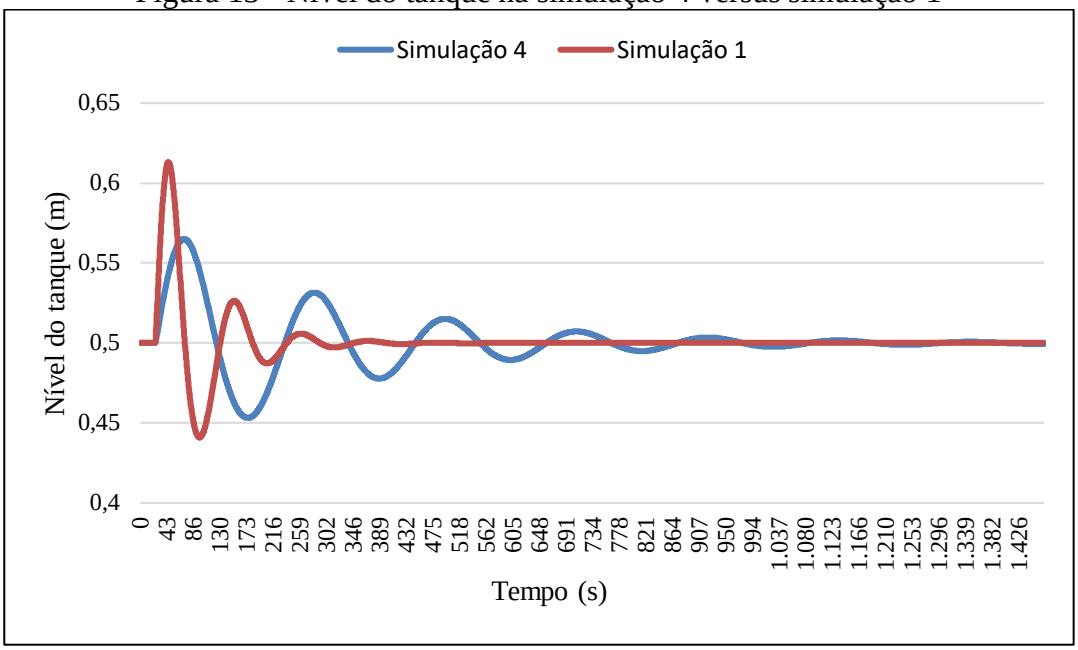
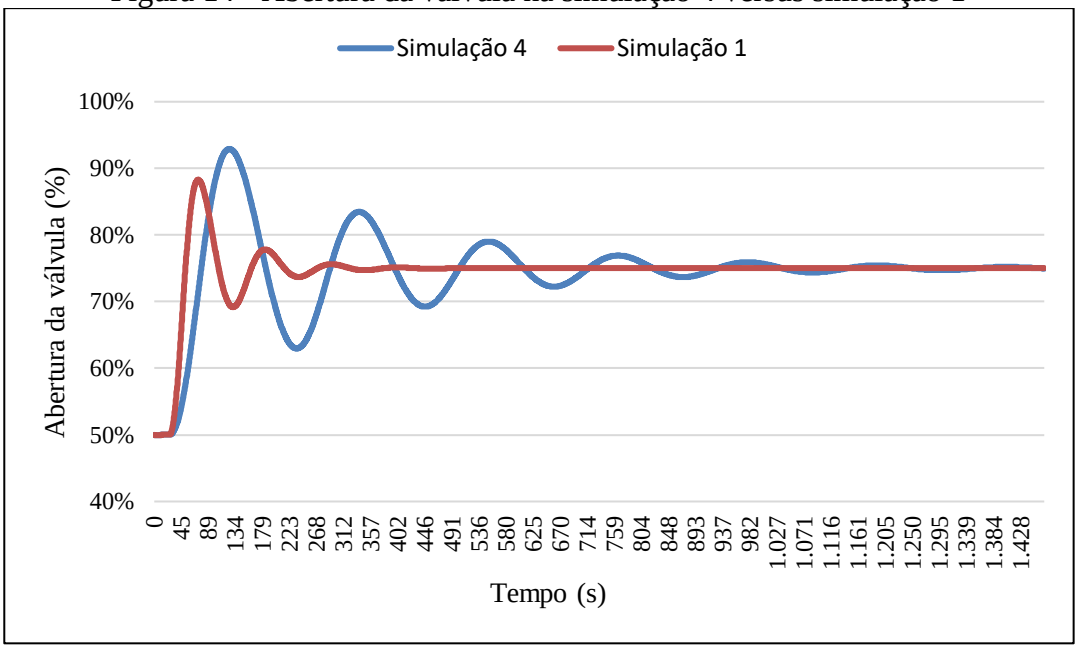


Figura 14 - Abertura da válvula na simulação 4 versus simulação 1



É evidente que para esse caso o controlador demorou mais para atingir o estado estacionário, atuando com um pior desempenho e menos robustez. Vale ressaltar também que o controle exigiu muito mais da válvula, chegando a abri-la quase 95%. Esses fatores podem ser explicados pelo inverso do que aconteceu na simulação anterior: a combinação entre

aumento da área e diminuição da constante de válvula, atuando como uma diminuição do ganho, o que desfavoreceu a atuação do controlador, conforme ilustra a equação 5.2.

$$\frac{C_V \cdot 0{,}5 \cdot X_V \cdot \sqrt{h}}{A \cdot 1{,}5} = \frac{C_V \cdot X_V \cdot \sqrt{h}}{3A} \tag{5.2}$$

5.6 Simulação 5

Com o intuito de mitigar esses efeitos que levaram a um pior desempenho e robustez do controlador na simulação 4, foi simulado para o mesmo sistema a aplicação do mesmo degrau porém com a presença do aprendizado online associado a uma razoável exploração.

A figura 15 ilustra a simulação com aprendizado *online* utilizando o valor de 0,01 para c_e.

Figura 15 – Nível do tanque durante aprendizado *online* com $\boldsymbol{c_e}$ = 0,01

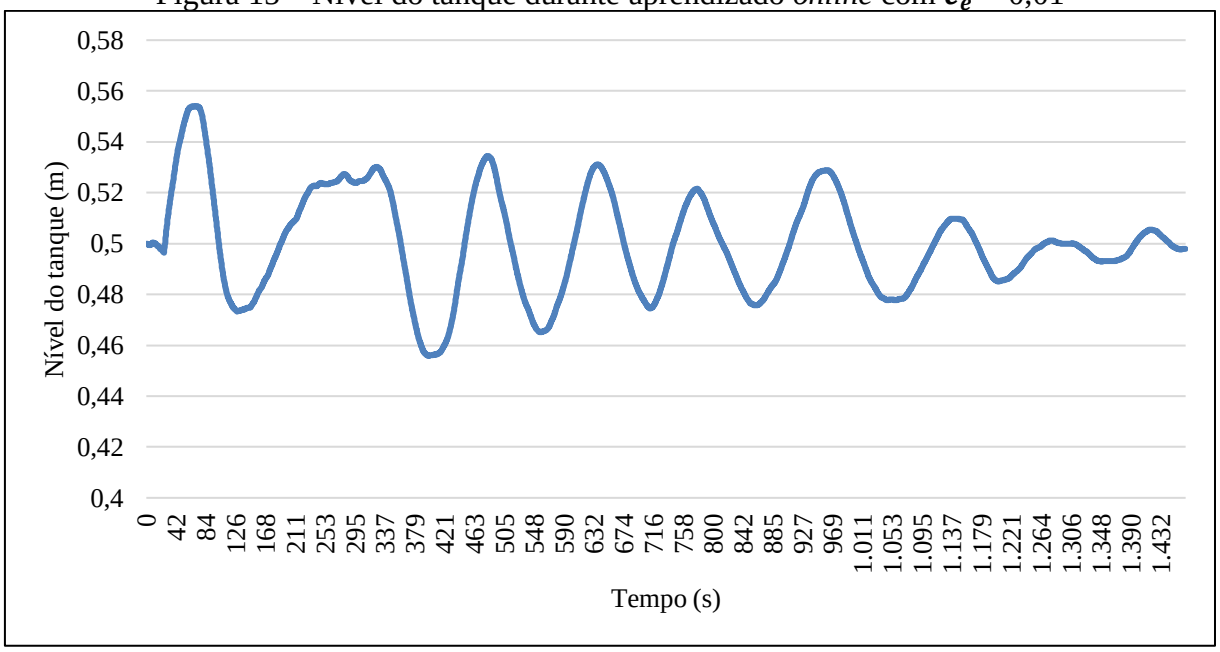
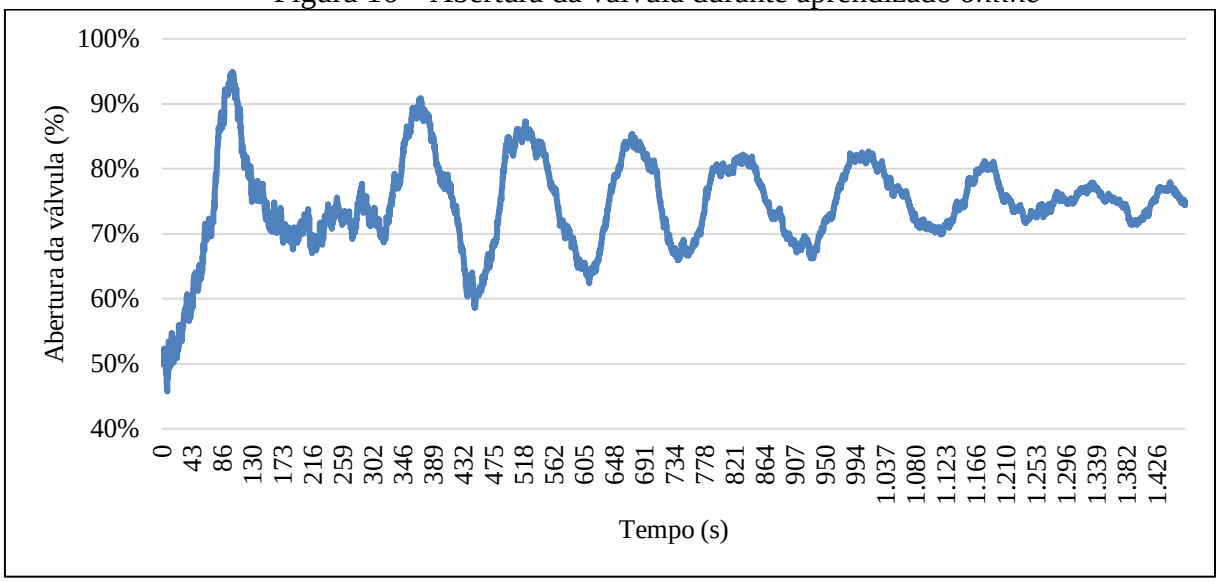
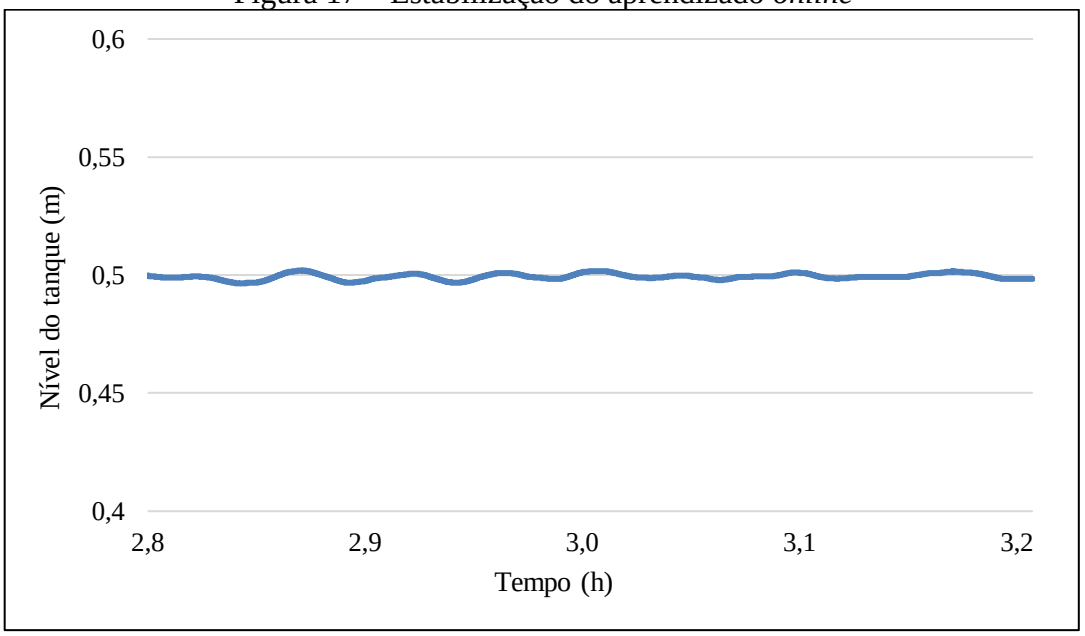


Figura 16 – Abertura da válvula durante aprendizado *online*



É evidente que a exploração provocou ruídos no acréscimo das ações de controle, porém sempre se mantendo dentro dos limites físicos do tanque e da válvula. Inicialmente, essa exploração prejudicou a performance do controlador, porém foi importante para que ocorresse aprimoramento do seu aprendizado. Em um tempo suficientemente longo, o sistema se estabilizou, conforme ilustra figura 17, o que permitiu alcançar um novo agente controlador robusto e seguro.

Figura 17 – Estabilização do aprendizado *online*



$$\theta_\pi = \begin{bmatrix} -0{,}0007185 \\ -0{,}0060998 \\ -0{,}0004623 \\ 0{,}0097216 \\ 0{,}0028050 \end{bmatrix}$$

5.7 Simulação 6

A finalidade da simulação 6 foi também avaliar a resposta do controlador ao degrau de 50% na vazão de entrada no mesmo sistema das simulações 4 e 5, desta vez utilizando a *policy* obtida por meio do aprendizado *online* na simulação 5 e comparar o seu desempenho versus a simulação 4.

Figura 18 - Comparação do nível do tanque nas simulações 4 e 6

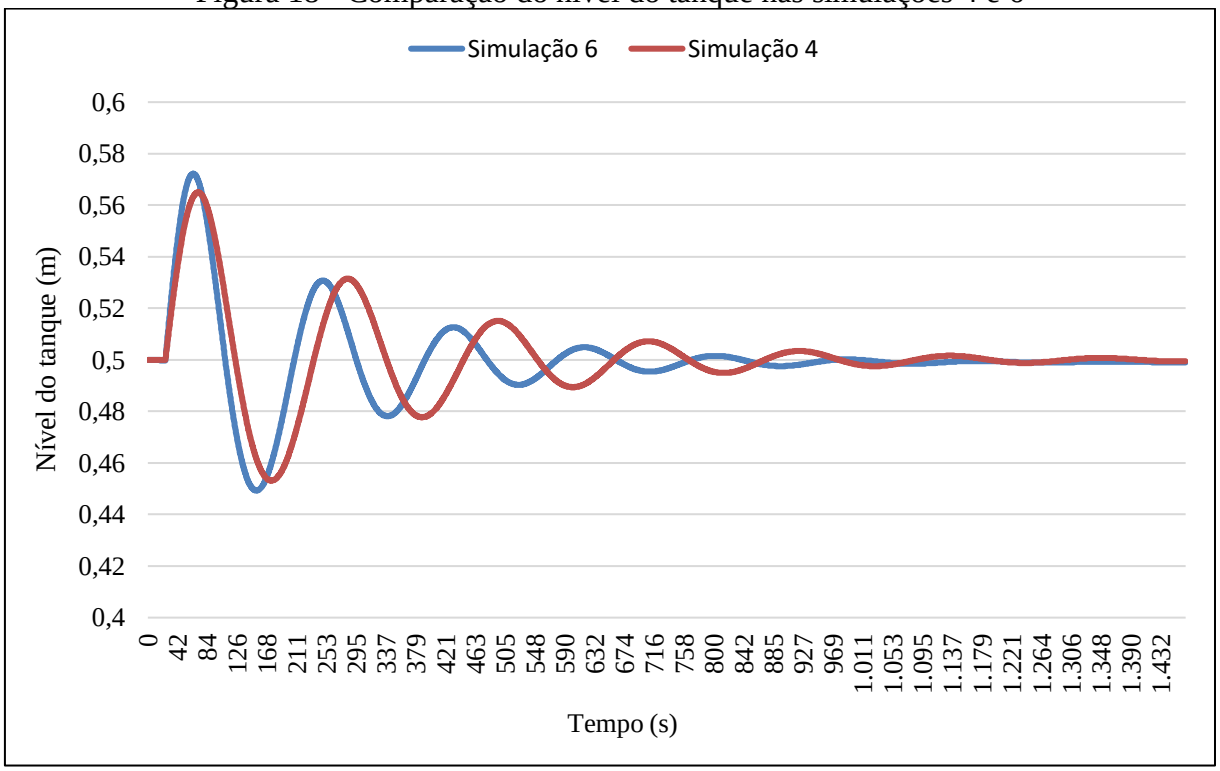
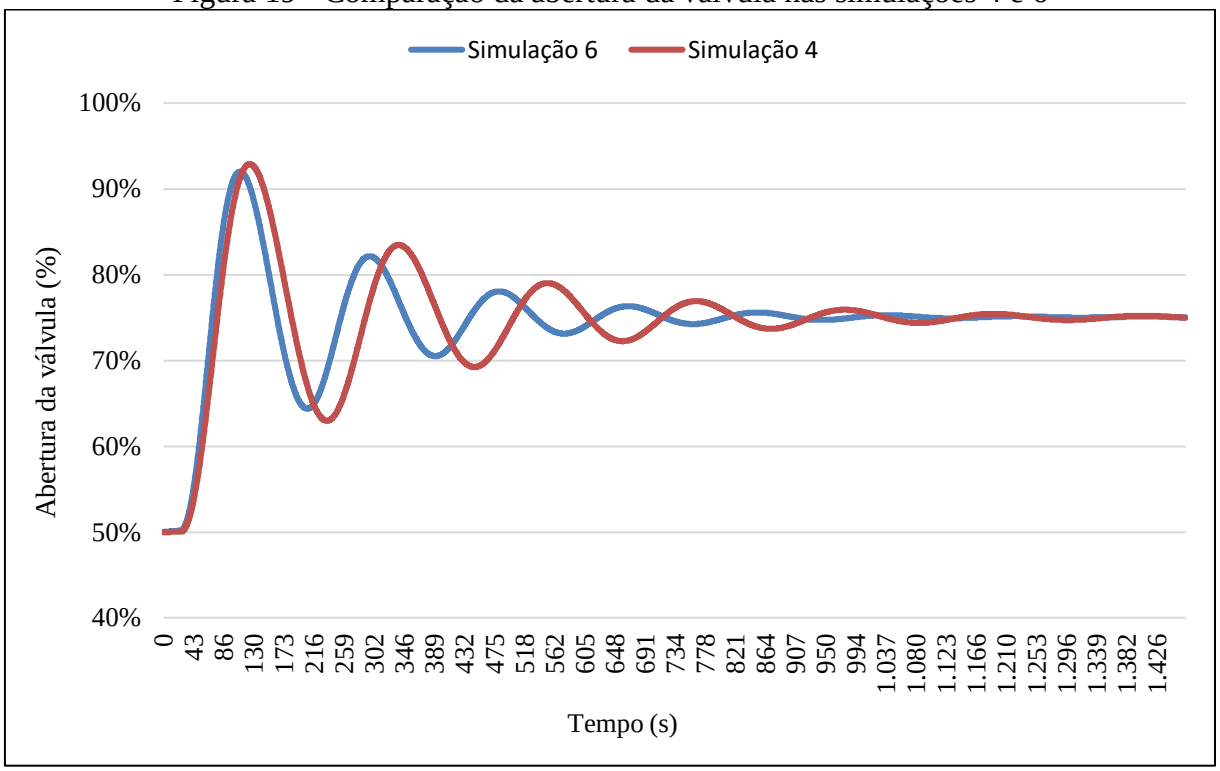


Figura 19 - Comparação da abertura da válvula nas simulações 4 e 6



É evidente que o aprendizado online permitiu aprimorar a performance do controlador, atingindo o estado estacionário de forma ainda mais rápida.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

O presente trabalho buscou desenvolver um controlador utilizando-se de técnicas de aprendizado por reforço, a fim de avaliar se esta metodologia poderia ser empregada na indústria em casos de tanques reais, sejam esses iguais aos usados no treinamento quanto diferentes.

Ao analisar os gráficos da simulação 1, nota-se que o método de treinamento foi eficiente em gerar uma *policy* que consiga responder a perturbações na vazão de entrada do tanque, conseguindo retornar ao estado estacionário em menos de 5 minutos. No entanto, ao variar os parâmetros do tanque e da válvula, ficou claro a influência da modelagem do balanço de massa no controlador, onde na simulação 3 obteve-se resultados melhores de resposta quando comparado à simulação 1 e na simulação 4 ocorreu o contrário, aumentando o tempo para estabilização do sistema.

Dessa forma, a partir do aprendizado *online* adquirido na simulação 5, a simulação 6 conseguiu mitigar esse efeito, mostrando que a implementação do controlador não só é viável em diversos tanques, como também pode ser otimizada por outros métodos, como o aprendizado contínuo durante a operação.

Para possíveis próximos trabalhos, pode-se adicionar ao aprendizado a vazão de entrada como uma outra variável que, junto do nível do tanque, defina o estado do processo. Também pode-se levar em conta o nível do tanque no estado anterior de forma que o agente perceba a tendência da mudança entre os estados e tome uma decisão sobre qual ação tomar. Além disso, a metodologia poderia ser expandida para problemas mais complexos, onde aprendizado e exploração *online* sejam imprescindíveis no controle de sistemas diferentes ao utilizado no aprendizado do agente. Dessa forma, seria possível aumentar o leque de atuação de controladores projetados por meio de algoritmos de *reinforcement learning*.

No que se propôs, este trabalho atendeu ao seu objetivo e demonstrou a aplicabilidade do controlador proposto, adaptando-se no controle de nível em diversas situações.

REFERÊNCIAS

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**, [S.L.], v. 428, p. 130993, jan. 2022. Elsevier BV. http://dx.doi.org/10.1016/j.cej.2021.130993.

BI, Q. et al. What is Machine Learning? A Primer for the Epidemiologist. **American Journal of Epidemiology**, v. 188, n. 12, 21 out. 2019.

CALLENDER, A. et al. Time-lag in a control system. **Philosophical Transactions Of The Royal Society Of London. Series A:** Mathematical, Physical and Engineering Sciences, Londres, v. 235, n. 756, p. 415-444, 21 jul. 1936. The Royal Society.

CAPRON, B. (2021, Fev 15). [Lecture notes on Reinforcement Learning]. Department of Chemical Engineering, UFRJ.

CHOI, S.; KIM, S.; JIN KIM, H. Inverse reinforcement learning control for trajectory tracking of a multirotor UAV. **International Journal of Control, Automation and Systems**, v. 15, n. 4, p. 1826–1834, 10 jul. 2017.

SILVER, D. Introduction to Reinforcement Learning 2015. Londres: Deepmind, 2015. (985 min.), son., color. Disponível em: https://youtube.com/playlist?list=PLqYmG7hTraZDM-OYHWgPebj2MfCFzFObQ. Acesso em: 23 jul. 2022.

DIETTERICH, T. G. **Machine learning**: Annual review of computer science, v. 4, n. 1, Oregon: Annual Reviews Inc.1990. p. 255-306.

FERNANDEZ-GAUNA, B., OSA, J. L., GRAÑA, M. Experiments of conditioned reinforcement learning in continuous space control tasks. **Neurocomputing**, v.271, pp. 38-47, 2018.

GHAHRAMANI, Z. Unsupervised learning. *In:* **Summer school on machine learning.** Berlin: Springer, 2003. p. 72-112.

HENZE, G.; SCHOENMANN, J. Evaluation of Reinforcement Learning Control for Thermal Energy Storage Systems. **HVAC&R Research**, v. 9, n. 3, p. 259–275, 1 jul. 2003.

HOSKINS, J.C.; HIMMELBLAU, D.M.. Process control via artificial neural networks and reinforcement learning. **Computers & Chemical Engineering**, [S.L.], v. 16, n. 4, p. 241-251, abr. 1992. Elsevier BV. http://dx.doi.org/10.1016/0098-1354(92)80045-b.

HULL, J. C. **Options, Futures and Other Derivatives**. 9. ed. Toronto: Pearson, 2015. 892 p.

JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. **Electronic Markets**, Würzburg, n. 31**,** p. 685–695, 2021. https://doi.org/10.1007/s12525-021-00475-2.

LIMA, R. M. **Aplicação de algoritmos de Reinforcement Learning para controle de nível de um tanque**. 2018. 58 f. TCC (Graduação) - Curso de Engenharia de Controle e Automação, Escola Politécnica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2018.

NAGABANDI, A., *et al*. **Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning**. 2018.

NIAN, R.; LIU, J.; HUANG, B. A review On reinforcement learning: Introduction and applications in industrial process control. **Computers & Chemical Engineering**, v. 139, p. 106886, ago. 2020.

NOEL, M. M.; PANDIAN, B. J. Control of a nonlinear liquid level system using a new artificial neural network based reinforcement learning approach. **Applied Soft Computing**, v. 23, p. 444–451, out. 2014.

SHIPMAN, W. J.; COETZEE, L. C. Reinforcement Learning and Deep Neural Networks for PI Controller Tuning. **IFAC-PapersOnLine**, v. 52, n. 14, p. 111–116, 2019.

SUTTON, R.S.; BARTO, A.G. **Reinforcement Learning: An Introduction.** 2. ed. Cambridge: The MIT Press, 2014

WANG, X.; CHENG, Y.; SUN, W. **A Proposal of Adaptive PID Controller Based on Reinforcement Learning**. Journal of China University of Mining and Technology, v. 17, n. 1, p. 40–44, mar. 2007.