



Universidade Federal  
do Rio de Janeiro  

---

Escola Politécnica

## APLICAÇÃO DE ALGORITMOS DE REINFORCEMENT LEARNING PARA CONTROLE DE NÍVEL DE UM TANQUE

Rodrigo Moysés Lima

Projeto de Graduação apresentado ao Curso de Engenharia de Controle e Automação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores: Bruno Didier Olivier Capron  
Daniel Machado Thomaz

Rio de Janeiro  
Setembro de 2018

APLICAÇÃO DE ALGORITMOS DE REINFORCEMENT LEARNING PARA  
CONTROLE DE NÍVEL DE UM TANQUE

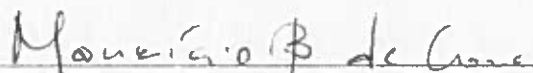
Rodrigo Moysés Lima

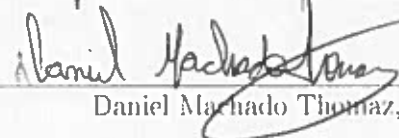
PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO  
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO DA ESCOLA  
POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO  
PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU  
DE ENGENHEIRO DE CONTROLE E AUTOMAÇÃO.

Examinado por:

  
Prof. Bruno Didier Olivier Capron, D.Sc.

  
Prof. Jomar Gozzi, D.Sc.

  
Prof. Mauricio Bezerra de Souza Jr., D.Sc.

  
Daniel Machado Thomaz, M.Sc.

RIO DE JANEIRO, RJ - BRASIL  
SETEMBRO DE 2018

Moysés Lima, Rodrigo

Aplicação de algoritmos de Reinforcement Learning para Controle de Nível de um Tanque/Rodrigo Moysés Lima. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2018.

XIII, 45 p.: il.; 29, 7cm.

Orientadores: Bruno Didier Olivier Capron

Daniel Machado Thomaz

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Controle e Automação, 2018.

Referências Bibliográficas: p. 43 – 44.

1. Reinforcement Learning. 2. Machine Learning. 3. Controle de Processos. 4. Actor-Critic. I. Capron, Bruno Didier Olivier *et al.* II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Controle e Automação. III. Título.

*A nenhum homem arrogante  
jamaiz foi permitido ver a  
Natureza em toda sua beleza  
– John Constable*

# Agradecimentos

Só há uma forma de começar os meus agradecimentos. Tudo o que eu sou e tudo o que conquistei, devo aos meu pais, Beth e Luiz. Muito obrigado por estarem presentes em todos os momentos da minha vida, por todos os sacrifícios e por sempre me oferecerem um amor incondicional. Sem vocês nada disso seria possível.

Tenho muito a agradecer também ao meu irmão Luiz André – o meu maior exemplo. Crescer ao seu lado foi uma experiência incrível que mais ninguém no mundo teve, só eu. Obrigado também à sua esposa Bia, pelo carinho e companheirismo.

Agradeço muito à minha família, sobretudo à minha avó Maria, meu padrinho Emilio, minha tia Márcia, minha prima Duda e minha afilhada Mariana. Temos uma família muito linda e unida e tenho muito orgulho disso. Obrigado também a todos os primos, tios, parentes e agregados que formam essa enorme família que tenho o privilégio de chamar de minha.

Se dentro de casa sempre tive o apoio dos meus familiares, fora de casa estive sempre acompanhado de amigos verdadeiros. Todos os amigos que tive e que ainda tenho são essenciais para mim. Gostaria de agradecê-los em ordem cronológica.

À Velha Guarda, meus amigos de sempre e para sempre.

A todos os meus amigos do São Bento, e em especial à Rapeize. Nossos encontros são sempre repletos de nostalgia e de bons sentimentos. Guardo na minha memória, com muito carinho, todos os momentos que passamos juntos no Colégio.

A todos os meus amigos de faculdade, sobretudo os colegas de Controle e Automação. Gostaria de agradecer primeiramente aos melhores companheiros de grupo que levo para a vida toda, Gabriel Pelielo e Rafael Accácio. Preciso agradecer também aos meus colegas mentores, que foram essenciais para a minha graduação, Isabella Quintanilha e João Monteiro. E a todos os grandes amigos que acompanharam a minha jornada na UFRJ, Lucas Schlee, Hannah, Marcelle, Jean, Gabriel Eiras, Philipe, Felipe, Henrique e demais colegas de turma, calouros e veteranos, todos foram extremamente importantes para a minha formação e para a minha vida.

Aos meus orientadores Bruno e Daniel. Eu jamais encontraria pessoas melhores para me ajudar a concluir a minha graduação na UFRJ. Meu muito obrigado pela dedicação, pela paciência e pela disponibilidade.

Por fim, agradeço a Deus pelas pessoas e oportunidades colocadas na minha vida.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Controle e Automação.

## APLICAÇÃO DE ALGORITMOS DE REINFORCEMENT LEARNING PARA CONTROLE DE NÍVEL DE UM TANQUE

Rodrigo Moysés Lima

Setembro/2018

Orientadores: Bruno Didier Olivier Capron  
Daniel Machado Thomaz

Curso: Engenharia de Controle e Automação

Metodologias baseadas em dados são cada vez mais utilizadas para a resolução das mais diversas tarefas. Nesse trabalho, a metodologia *reinforcement learning* (RL) é aplicada no escopo de controle de processos. Realizou-se o projeto de um controlador de nível de um tanque com a utilização de RL e aplicação da metodologia *actor-critic*. Essa abordagem foi escolhida pois utiliza tanto o espaço de estados quanto o de ações contínuos, que é o mais indicado para uma representação adequada de processos contínuos.

Os resultados mostram que o controle desenvolvido tem desempenho similar a um controlador clássico do tipo PI. Essa validação desse controle baseado em dados de processo para uma planta não-linear simples abre a porta para a sua implementação em processos cada vez mais complexos.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

APPLICATION OF REINFORCEMENT LEARNING ALGORITHMS FOR  
LEVEL CONTROL OF A LIQUID TANK

Rodrigo Moysés Lima

September/2018

Advisors: Bruno Didier Olivier Capron

Daniel Machado Thomaz

Course: Automation and Control Engineering

Data driven methodologies are being increasingly used as a way to perform a wide range of tasks. In this work, the reinforcement learning (RL) methodology is applied in the scope of process control. The project of a level controller of a liquid tank via RL is then carried out. The actor-critic approach was the one applied, since it uses a continuous representation for both the state and action spaces. This makes it well-suited for continuous systems, as are the ones tackled by the process control community.

Results show that the RL-based controller has a similar performance to a classical PI controller. This validation of a controller design using only process data for a simple non-linear plant opens the door for its implementation in increasingly more complex processes.

# Sumário

<b>Lista de Figuras</b>	<b>x</b>
<b>Lista de Símbolos</b>	<b>xi</b>
<b>1 Introdução</b>	<b>1</b>
<b>2 Revisão Bibliográfica</b>	<b>5</b>
<b>3 Fundamentos Teóricos</b>	<b>7</b>
3.1 Interação Agente-Ambiente . . . . .	7
3.2 Recompensa e Retorno . . . . .	8
3.3 Estado e a Propriedade de Markov . . . . .	9
3.4 Processo de Decisão de Markov . . . . .	10
3.5 <i>Value Function</i> . . . . .	11
3.6 Exploração . . . . .	11
3.7 Aproximação da <i>Value Function</i> . . . . .	13
3.8 Metodologia <i>Actor-Critic</i> . . . . .	14
3.9 Avaliação de <i>Policy</i> . . . . .	15
3.10 Melhoria de <i>Policy</i> . . . . .	17
<b>4 Desenvolvimento do Controle de Nível</b>	<b>19</b>
4.1 A Planta . . . . .	19
4.2 Definindo a Função de Recompensa . . . . .	21
4.3 <i>Actor-Critic</i> e VFAs . . . . .	22
4.4 Parâmetros da Exploração . . . . .	23
4.5 Desenvolvimento do Aprendizado via RL . . . . .	24
4.6 Sintonia de um Controlador Clássico . . . . .	25
4.7 Adaptações para Variação no Setpoint . . . . .	28
<b>5 Resultados e Discussões</b>	<b>29</b>
5.1 Parâmetros iniciais de Exploração . . . . .	29
5.2 Parâmetros da Avaliação de <i>Policy</i> e Recompensas . . . . .	30



5.3	Métodos de Melhoria de <i>Policy</i> . . . . .	33
5.4	Ganho do Aprendizado . . . . .	34
5.5	Comparação dos Tipos de Exploração . . . . .	36
5.6	Comparação entre RL e Controlador Clássico . . . . .	37
5.7	Demonstração da característica adaptativa do controlador . . . . .	38
5.8	Rastreamento do Setpoint pelo Estado . . . . .	40
<b>6</b>	<b>Conclusão e Trabalhos Futuros</b>	<b>42</b>
	<b>Referências Bibliográficas</b>	<b>43</b>

# Lista de Figuras

1.1	Diagrama de interações da metodologia <i>reinforcement learning</i> . . . . .	2
3.1	Exemplo ilustrativo da VFA por meio de 3 RBFs. . . . .	14
3.2	Exemplo ilustrativo da variação do valor do traço de elegibilidade. . .	16
4.1	Interações da metodologia RL para controle de nível de um tanque. .	19
4.2	Diagrama ilustrativo da planta a ser controlada. . . . .	20
4.3	Visualização das 10 RBFs definidas para as VFAs do <i>Actor-Critic</i> . . .	23
4.4	Implementação do controlador PID na ferramenta Simulink. . . . .	26
4.5	Resultado de uma variação de +5% em $X_V$ no instante $t = 0.5h$ . . . .	26
5.1	Resultado de simulações com $\mathcal{R}_1$ e diversos valores de $\lambda$ e $\gamma$ . . . . .	31
5.2	Resultado de simulações com $\mathcal{R}_2$ e diversos valores de $\lambda$ e $\gamma$ . . . . .	32
5.3	Resultado de simulações com diferentes métodos de melhoria de <i>policy</i> . .	34
5.4	Resultado de simulações para diversos valores de $\alpha_V$ e $\alpha_\pi$ . . . . .	35
5.5	Resultado de simulações com diferentes métodos de exploração. . . .	37
5.6	Comparação de desempenho entre o controle via RL controlador PI discreto. . . . .	38
5.7	Resultado da simulação do controle com rastreamento de setpoint por meio da exploração. . . . .	39
5.8	Resultado de simulações do controle com rastreamento de setpoint por meio do estado, para diferentes sinais de recompensa. . . . .	41

# Lista de Símbolos

$A$	Seção transversal do tanque [m <sup>2</sup> ], p. 19
$C_V$	Constante da válvula [m <sup>2.5</sup> /h], p. 19
$E_\pi$	Valor esperado em função da <i>policy</i> $\pi$ , p. 11
$F$	Número de <i>features</i> , p. 13
$F_i$	Vazão de alimentação do tanque [m <sup>3</sup> /h], p. 19
$F_o$	Vazão de saída do tanque [m <sup>3</sup> /h], p. 20
$G(s)$	Função de transferência de primeiro grau aproximada do sistema, p. 26
$I$	Ganho integral, p. 27
$K$	Ganho da função de transferência $G(s)$ , p. 26
$N(M, \sigma_t)$	Variável aleatória gaussiana com média $M$ e desvio padrão $\sigma_t$ , p. 12
$N(\phi_i)$	Número de vezes que a região definida pela <i>feature</i> $\phi_i$ foi visitada, p. 24
$N_0$	Parâmetro de sintonia da exploração, p. 23
$P$	Ganho Proporcional, p. 27
$P_r$	Função distribuição de probabilidade, p. 10
$R_t$	Retorno esperado no instante de tempo $t$ , p. 8
$T_r$	Tempo e subida, p. 30
$V$	Volume do tanque [m <sup>3</sup> ], p. 20
$V^\pi$	<i>Value function</i> em função da <i>policy</i> $\pi$ , p. 11

$X_V$	Abertura da válvula [%], p. 19
$\Delta t$	Intervalo de amostragem [h], p. 25
$\alpha_V$	Ganho de aprendizado do <i>actor</i> , p. 17
$\alpha_V$	Ganho de aprendizado do <i>critic</i> , p. 17
$\delta_t$	Erro de diferença temporal, p. 15
$\epsilon_t$	Probabilidade de explorar no instante $t$ , p. 12
$\gamma$	Fator de desconto da recompensa, p. 8
$\hat{V}_\pi(s_t)$	<i>Value function</i> estimada no estado $s_t$ , p. 15
$\hat{f}(s)$	Aproximação de uma dada função $f(s)$ , p. 13
$\lambda$	Fator de decaimento do traço de elegibilidade, p. 16
$\mathcal{A}(s)$	Espaço de ações, p. 7
$\mathcal{A}(s)$	Espaço de ações disponíveis no estado $s$ , p. 7
$\mathcal{P}$	Função de transição de estados, p. 10
$\mathcal{R}$	Função de recompensa, p. 8
$\mathcal{R}_1$	Função de recompensa relacionada à aproximação relativa ao setpoint, p. 21
$\mathcal{R}_2$	Função de recompensa relacionada à região do setpoint, p. 22
$\mathcal{R}_{1t+1}$	Valor da função de recompensa $\mathcal{R}_1$ no instante de tempo $t$ , p. 21
$\mathcal{R}_{2t+1}$	Valor da função de recompensa $\mathcal{R}_2$ no instante de tempo $t$ , p. 22
$\mathcal{S}$	Espaço de estados, p. 7
$\nabla_{\theta_\pi} \pi(s_t)$	Gradiente da <i>policy</i> $\pi(s_t)$ em relação a $\theta_\pi$ , p. 17
$\phi(s)$	Vetor de <i>features</i> , p. 13
$\phi_V(s)$	Vetor de <i>features</i> da <i>value function</i> , p. 17
$\phi_\pi(s)$	Vetor de <i>features</i> da <i>policy</i> , p. 17

$\phi_i(s)$	Função de base radial gaussiana que compõe o vetor $\phi(s)$ , p. 14
$\pi(s)$	<i>Policy</i> que determina a ação $a$ de acordo com o estado $s$ , p. 7
$\pi_i(s)$	<i>Policy</i> para uma dimensão no caso de um espaço de ações multidimensional, p. 13
$\sigma_0$	Parâmetro de sintonia do desvio padrão da exploração, p. 24
$\sigma_\phi$	Desvio padrão das funções gaussianas $\phi_i(s)$ , p. 14
$\tau$	Constante de tempo da função de transferência $G(s)$ , p. 26
$\theta$	Vetor de pesos das <i>features</i> , p. 13
$\theta_{V,t}$	Vetor de pesos da <i>policy</i> no instante de tempo $t$ , p. 17
$\theta_{V,t}$	Vetor de pesos da <i>value function</i> no instante de tempo $t$ , p. 17
$a_t$	Ação no instante de tempo $t$ , p. 7
$c_i$	Centro da função gaussiana $\phi_i(s)$ , p. 14
$e_t(s)$	Traço de elegibilidade do estado discreto $s$ no instante de tempo $t$ , p. 16
$e_{ss}$	Erro de estado estacionário [m], p. 31
$h$	Nível do tanque [m], p. 19
$h_t$	Valor do nível no instante de tempo $t$ [m], p. 21
$h_{sp}$	Setpoint do nível [m], p. 21
$m$	Dimensão do espaço de ações, p. 13
$\max(\phi(s_t))$	<i>Feature</i> $\phi_i$ que apresenta o maior valor dentre todas as <i>features</i> no instante de tempo $t$ , p. 24
$n(s)$	Número de vezes em que o estado $s$ foi visitado, p. 23
$r_t$	Recompensa no instante de tempo $t$ , p. 7
$s_t$	Estado no instante de tempo $t$ , p. 7
$z_t$	Vetor traço de elegibilidade no instante de tempo $t$ , p. 16

# Capítulo 1

## Introdução

Nos últimos anos, o advento da Quarta revolução Industrial desencadeou uma grande expansão nos sistemas de automação e aquisição de dados na indústria de processos [1]. Dessa forma, a obtenção de dados confiáveis e de boa qualidade torna-se cada vez mais barata e acessível, o que faz metodologias baseadas em dados cada vez mais aplicáveis.

Dentre essas metodologias, as áreas de *Data Science* e *Reinforcement Learning* estão em constante desenvolvimento e apresentam ótimos resultados. Porém, apesar de ser um tema explorado nas mais diversas áreas, como na robótica, em que esse tipo de metodologia é aplicada para a locomoção de robôs bípedes [2] e no mercado financeiro, para a otimização na gestão de ativos [3], RL ainda é uma abordagem pouco utilizada pela comunidade de controle de processos. As principais publicações da área serão exploradas na revisão bibliográfica no Capítulo 2.

Outro incentivo para a utilização de metodologias baseadas em dados provém das limitações do controle clássico baseado em modelos. A técnica de controle avançado mais utilizada na indústria de processos é o Controle Preditivo Baseado em Modelo (MPC) [4]. A maioria dos MPCs utilizam modelos lineares ou lineares por partes [5], o que não representa bem a maioria dos processos químicos. Existem também aplicações de MPC não-linear (NMPC) [6], porém, o processo de obtenção de modelos dessa natureza é complexo. Além disso, como o MPC é uma técnica que requer a resolução de um problema de otimização em um intervalo de tempo definido, isso pode ter um alto custo computacional, o que limita as suas aplicações em tempo real.

Em um problema de *reinforcement learning*, um agente aprende a realizar uma determinada tarefa ao interagir diretamente com o ambiente, sem supervisão. O agente é a entidade que aprende e toma decisões. Já o que interage com ele, incluindo tudo que é externo ao agente, é chamado de ambiente. Essas duas entidades interagem continuamente, de tal forma que o agente decide que ação tomar e o ambiente responde a essas ações apresentando novos estados ao agente. O ambiente

também fornece ao agente valores numéricos chamados de recompensas, de acordo com o estado em que se encontra. O objetivo do agente deve ser maximizar a recompensa acumulada ao longo do tempo. A Figura 1.1 ilustra essas interações em um problema genérico de RL.

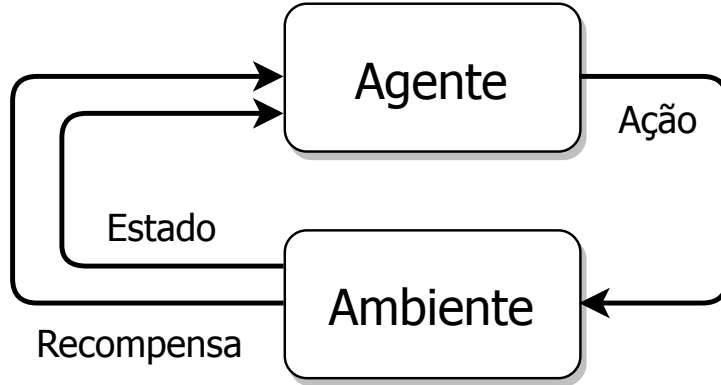


Figura 1.1: Diagrama de interações da metodologia *reinforcement learning*.

Na problemática desenvolvida nesse projeto, a abordagem de *reinforcement learning* será aplicada ao controle de nível de um tanque. Assim, tem-se que o ambiente é um tanque, enchido a uma vazão constante, e o agente é o controlador (no caso, controlador de nível). A ação é o sinal de controle que é dado a uma válvula, que atua de forma a manter o nível do tanque, o estado é o nível do tanque e a recompensa será definida de acordo com o setpoint a ser atingido.

No controle baseado em RL, não há uma lei de controle algébrica. Ao invés disso, o comportamento do controlador é ditado por uma *policy*, um mapeamento dos estados em ações. Dessa forma, dado um determinado estado, a *policy* dita qual deve ser a ação tomada pelo controlador. Essa *policy* vem do aprendizado de uma *value function*, que é a previsão de recompensas futuras esperadas de acordo com o estado. Para cada estado, a *policy* escolhe a ação que maximiza a *value function* relacionada a esse estado [7].

O aprendizado pode se dar através da interação direta com o ambiente e melhorias sucessivas na *policy*, o que é chamado de aprendizado *on policy*. Isso também pode ser feito por meio de uma experiência amostrada por outra *policy*, nesse caso o aprendizado é chamado de *off policy*. Nesse trabalho, consideraremos apenas o aprendizado *on policy*, em que o controlador inicialmente segue uma *policy* gerada aleatoriamente que é melhorada à medida que o sistema é explorado. Essa exploração pode ser feita tomando uma ação aleatória com certa probabilidade que diminui ao longo do tempo, chamada de abordagem  *$\epsilon$ -greedy* [8] ou pela adição de um ruído branco na própria ação, que nesse caso é o sinal de controle [9]. Esses dois tipos de exploração são abordados ao longo desse trabalho.

Outra característica importante e interessante do aprendizado de um controlador

RL é que ele pode continuar a ser treinado enquanto implementado no sistema. Desta forma, distúrbios não medidos e a progressiva degradação do processo podem ser levados em consideração. Essa adição de novos dados confere um comportamento adaptativo ao controlador, sendo essencial para torná-lo robusto, especialmente em relação a distúrbios não medidos que possam levar o sistema a um ponto de operação no qual ele não foi bem treinado.

A implementação mais simples de controle baseado em RL utiliza uma representação de espaços de estados e ações discretos. Dessa forma, achar o maior valor da *value function* se resume em uma busca em um conjunto finito de valores. Contudo, nesse tipo de representação discreta, o volume de dados necessário para o treinamento do controlador aumenta exponencialmente com o número de pares estado-ação, sendo aplicável somente a sistemas de poucas dimensões. Assim, essa abordagem não é adequada para o controle de processos químicos, devido à natureza contínua das variáveis do processo.

De maneira a solucionar o problema da dimensionalidade, uma representação contínua pode ser usada. Para tal, é preciso usar uma aproximação da *value function*, o que permite obter uma função contínua a partir de um conjunto de amostras discreto. Isso pode ser feito com uma combinação linear de funções de base radial dependentes do estado, com redes neurais ou com aproximadores baseados em núcleo [7].

Nesse trabalho, é utilizada a aproximação com uma combinação linear de funções de base radial para obter tanto um espaço contínuo de estados, quanto de ações. A abordagem mais adequada para isso é conhecida como *actor-critic* (AC), pois mantém estruturas separadas para a *value function* relacionada ao estado e para a *policy* que seleciona uma ação. Quem aprende a *value function* é o *critic*, enquanto que o *actor* aprende a *policy*. No método de aprendizado *actor-critic*, a atualização da *policy* é feita a cada iteração conforme um procedimento de dois passos. Primeiro, a *value function* do estado é atualizada pela incorporação de novos dados e em seguida, a *policy* é atualizada a partir dessa nova *value function*. Após o primeiro passo, o *critic* dá um feedback para o *actor*, que geralmente consiste no incremento que foi feito na *value function*. Com isso, o *actor* consegue incrementar também a *policy*. Essa abordagem é descrita detalhadamente no Capítulo 3 (Fundamentos Teóricos).

Os detalhes da aplicação da metodologia AC para o controle de nível de um tanque são apresentados no Capítulo 4 (Desenvolvimento do Controle de Nível). Nessa seção são apresentados os métodos computacionais implementados em MATLAB para o controlador de nível.

Em seguida, no Capítulo 5 (Resultados e Discussões) são apresentados os resultados de diversas simulações feitas para o controle de nível de um tanque. Essas



simulações ilustram os efeitos de diversas variações na arquitetura do algoritmo de aprendizado via RL. Um controlador clássico da literatura do tipo PID é então implementado ao processo, para servir como base de comparação do desempenho do controlador RL.

Por fim, no Capítulo 6 (Conclusão e Trabalhos Futuros) são apresentadas as conclusões sobre o trabalho, além de sugestões para trabalhos futuros.

O objetivo desse projeto é a validação de um controle baseado em dados, no escopo de controle de processos. Para isso, é feita a aplicação de um controle de nível de um tanque via RL, utilizando a metodologia *actor-critic*. Essa abordagem é considerada o estado da arte do RL atualmente, e utiliza uma representação contínua para o espaço de estados e de ações, sendo portanto adequada para sistemas complexos, como é de interesse para a comunidade de controle de processos.

São exploradas diversas variações na sintonia desse controlador. A tarefa de controlar o nível de um tanque é um problema de controle simples, pois apresenta apenas uma variável controlada e uma manipulada. Contudo, essa planta apresenta uma dinâmica não-linear, sendo portanto adequada para uma avaliação inicial desse tipo de metodologia em controle de processos.

Comparando então o controle obtido via RL com um controlador clássico, é possível validar o seu desenvolvimento e aplicação. Com isso, será demonstrada a aplicação de um controlador sintetizado sem qualquer tipo de informação sobre a planta. Esse tipo de abordagem pode ser muito vantajosa se aplicada posteriormente em processos mais complexos, já que pode ser feita sem a necessidade de um levantamento de modelos do sistema, uma tarefa que pode vir a ser muito custosa.

Nesse projeto, é utilizada uma modelagem simples da dinâmica do tanque a ser controlado. Essa modelagem é necessária para que o aprendizado do controlador via RL ocorra por meio de simulações. A ideia do RL é aprender com dados de processo, porém a interação direta do controlador ignorante com o processo não é possível por motivos de segurança e de produtividade.

No entanto, um pré-aprendizado pode ser feito a partir de dados históricos da planta (aprendizado *off policy*). Uma interação direta com o processo pode ser feita por meio de uma limitação das ações de controle, garantindo que o processo não saia de uma determinada zona de operação. Contudo implementações não fazem parte do escopo desse projeto e estão indicadas com sugestões de implementações futuras no Capítulo 6.

# Capítulo 2

## Revisão Bibliográfica

O *Reinforcement Learning* é uma área de *Machine Learning* focada na interação do agente com o ambiente. Dessa forma, o aprendizado, no caso de um controle de processo, é feito sem a utilização de um modelo da planta a ser controlada.

Alguns métodos se desvinculam da necessidade de uma modelagem analítica do processo e utilizam apenas a resposta em frequência da planta como fonte de informação. Isso foi proposto por KEEL e BHATTACHARYYA [10] ao sintonizar controladores do tipo proporcional integral derivativo (PID) e controladores de primeira ordem por meio do diagrama de Bode ou de Nyquist. Dessa forma, não há mais a necessidade de saber a ordem da planta ou o número de polos e/ou zeros nos semiplanos laterais.

Para encontrar os ganhos de controladores PIDs sem o uso de modelos analíticos, KILLINGSWORTH e KRSTIC [11] propõem a utilização do método de busca extremal. Nessa abordagem, os parâmetros do controlador são sintonizados iterativamente de forma a minimizar uma função de custo. A função escolhida foi a integral do erro quadrático, que é avaliada ao final de um experimento de resposta ao degrau.

Essas propostas de controle sem modelo também se estendem a métodos de controle avançado. Para contornar problemas de modelagem não-linear, STENMAN [12] propôs uma alteração no método clássico de MPC ao utilizar a metodologia de modelo em demanda, que é uma identificação local, simples e *online* do sistema.

Nesse trabalho, são consideradas, no contexto de controle de processos, técnicas de RL que foram desenvolvidas pela comunidade de inteligência artificial desde a década de 1980, focando na elaboração de algoritmos de controle puramente baseados em dados, como proposto por BUSONIU *et al.* [7].

No que pode-se considerar a primeira aplicação de RL na engenharia química, HOSKINS e HIMMELBLAU [13] propõem um controle baseado em redes neurais. Dois modelos diferentes de redes neurais foram utilizados para prever tanto a medida de desempenho do controlador quanto suas ações e *policies*. Essa metodologia

foi então aplicada a um reator tanque de agitação contínua (ou CSTR, *Continuous Stirred Tank Reactor*) e os resultados obtidos foram muito similares ao de um controlador PID. Uma observação importante presente nesse trabalho é que mesmo não havendo um modelo das dinâmicas do processo químico, seria possível treinar uma rede neural artificial para poder controlar o processo, e isso ocorreria de maneira similar a como um humano aprende, por tentativa e erro.

A partir desse trabalho pioneiro, o número de aplicações de RL nessa área vem crescendo bastante e com resultados cada vez melhores. ANDERSON *et al.* [14] aplicaram RL combinado com um controle proporcional integral (PI) para controlar uma bobina de aquecimento. Foram utilizados espaços discretos tanto de estados quanto de ações, e concluiu-se que o agente de RL conseguiu melhorar os resultados do controlador proporcional-integral (PI).

Já MARTINEZ [15] usou RL para otimizar processos em batelada, integrando uma estimativa de sucesso probabilística usando um modelo imperfeito com uma abordagem baseada em gradiente. Os resultados dessas simulações comprovaram a robustez do RL em relação a erros de modelagem estruturais e paramétricos.

Há aplicações de técnicas de RL para o controle de fermentação de levedura, sistema em que é difícil se obter um controle robusto por conta das incertezas, das não-linearidades e também das características dinâmicas do processo. SYAFIIE *et al.* [16] adicionaram lógica fuzzy ao RL e obteve ótimos resultados ao compará-lo com um controlador PID avançado: rastreamento mais rápido, *overshoot* menor e sinal de controle mais suave.

SHAH e GOPAL [17] utilizaram RL para o ajuste de parâmetros de um controlador PID em um CSTR não-isotérmico, com espaços contínuos de estados e de ações. Essa abordagem indica uma solução geral para o design de controladores PID em sistemas não-lineares e variantes com o tempo, sem a necessidade de um entendimento detalhado das dinâmicas da planta.

Um problema similar ao desse trabalho é abordado por RAMANATHAN *et al.* [18] ao aplicar RL para controle de nível de um tanque. Porém, o algoritmo apresentado é bastante simplório, com espaço de ações e estados discretos e tanque cônico, de forma a conferir uma característica não-linear ao sistema.

O esquema de controle RL desenvolvido nesse projeto segue a metodologia *actor-critic*. FERNANDEZ-GAUNA *et al.* [19] aplicaram essa metodologia em diversos experimentos computacionais, como o controle de velocidade de um veículo submarino e de uma turbina eólica. Embora a metodologia aplicada aqui seja muito similar àquela proposta por FERNANDEZ-GAUNA *et al.* [19], são explorados diversos cenários de forma a melhorar o processo de aprendizado do controlador RL.

# Capítulo 3

## Fundamentos Teóricos

Quando um bebê está aprendendo a andar, ele ainda não domina a língua dos pais, que não podem explicitamente ensiná-lo a andar. Ele eventualmente aprende por meio da interação direta com o ambiente. A dor causada por uma queda e a felicidade ou orgulho associada com um passo dado com sucesso podem ser considerados sinais de feedback dados ao bebê pelo ambiente. Com isso, ele consegue aprender a andar ao minimizar os sinais negativos enviados pelo ambiente e maximizar os positivos. Essa é a ideia central por trás do *Reinforcement Learning*.

Nesse capítulo serão apresentados os conceitos básicos da metodologia RL, seguindo as definições dos livros de SUTTON e BARTO [8] e de BUSONIU *et al.* [7].

### 3.1 Interação Agente-Ambiente

O aprendizado por RL é fruto apenas da interação direta do agente com o ambiente, segundo os sinais descritos anteriormente na Figura 1.1. Em cada instante de tempo  $t$ , o agente recebe do ambiente uma representação do seu estado  $s_t \in \mathcal{S}$ , em que  $\mathcal{S}$  é o espaço de estados possíveis, e com essa informação ele seleciona uma ação  $a_t \in \mathcal{A}(s_t)$ , sendo  $\mathcal{A}(s_t)$  o espaço de ações disponíveis no estado  $s_t$ . No instante de tempo seguinte, o agente então recebe uma recompensa  $r_{t+1} \in \mathbb{R}$  e um novo estado  $s_{t+1}$ .

De maneira geral, o agente implementa um mapeamento dos estados em probabilidades de realizar as ações possíveis. Essa é a chamada *policy* do agente, denotada  $\pi_t$ , onde  $\pi_t(s, a)$  é a probabilidade de  $a_t = a$ , dado  $s_t = s$ . O processo de aprendizado altera a *policy* de acordo com a experiência do agente.

No contexto desse trabalho, trataremos apenas de *polícies* determinísticas. Sendo assim, a *policy* implementa um mapeamento direto de um estado  $s_t$  para uma ação  $a_t$ , ou seja,  $\pi(s_t) = a_t$ . *Polícies* estocásticas são muito úteis para lidar com processos estocásticos, o que não é o escopo desse projeto.

## 3.2 Recompensa e Retorno

A cada instante de tempo, o agente recebe do ambiente um sinal de recompensa,  $r_t \in \mathbb{R}$ . O objetivo do agente não é maximizar a recompensa imediata, mas sim a recompensa acumulada ao longo do tempo.

Esse sinal de recompensa é uma das principais características determinantes do RL. É uma abordagem bastante flexível e abrangente. Por exemplo, para um robô que aprende a andar, a recompensa pode ser proporcional ao movimento para frente. Já para que aprenda a escapar de um labirinto, a recompensa pode ser zero até o momento em que ele consiga sair, se tornando então  $+1$ . Outra abordagem possível é dar uma recompensa de  $-1$  a cada instante de tempo que o robô passa no labirinto, fazendo com que o agente aprenda a atingir o seu objetivo no menor intervalo de tempo possível. Para um agente aprendendo a jogar xadrez, as recompensas poderiam ser  $+1$  por ganhar o jogo,  $-1$  por perder e  $0$  por empatar e por qualquer movimento feito que não ganha nem perca o jogo.

Dessa forma, o sinal de recompensa deve ser definido de forma que, ao maximizá-lo, o agente irá atingir o objetivo esperado. A recompensa não deve ser usada para indicar ao agente como atingir esse objetivo. No exemplo do jogo de xadrez, a recompensa deve ser dada somente por ganhar o jogo e não por atingir objetivos secundários, como tomar as peças do adversário. Se esse fosse o caso, o agente poderia encontrar uma maneira de receber essa recompensa sem atingir o objetivo real e encontrar um jeito de tomar várias peças do adversário mesmo que isso o faça perder o jogo. O sinal de recompensa é a maneira de comunicar ao agente o que ele deve atingir e não como ele deve fazê-lo.

O valor da recompensa é definido por uma função  $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ , que mapeia um estado  $s_t$ , uma ação  $a_t$  e o estado resultante  $s_{t+1}$  em uma recompensa  $r_{t+1} = \mathcal{R}(s_t, a_t, s_{t+1})$ . Essa recompensa chega para o agente no momento em que ele atinge o estado  $s_{t+1}$ .

O agente procura sempre maximizar a recompensa acumulada ao longo do tempo e isso deve ser definido formalmente. A sequência de recompensas recebidas pode ser definida como:  $r_{t+1} + r_{t+2} + r_{t+3} + \dots$ , lembrando que no instante inicial  $t$  o agente ainda não recebeu nenhuma recompensa  $r_t$ . Assim, o agente deve maximizar o retorno  $R_t$ , que seria uma função desse somatório de recompensas recebidas após cada intervalo de tempo.

Esse somatório pode ser pesado de acordo o tempo, de forma que recompensas imediatas sejam mais importantes do que recompensas futuras. Isso é feito com a inclusão do fator de desconto  $\gamma$ , um parâmetro cujo intervalo de valores é  $0 \leq \gamma \leq 1$ . A expressão do retorno toma a forma da Equação 3.1.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (3.1)$$

Esse fator  $\gamma$  desconta as recompensas quanto mais distantes no futuro elas estiverem. Assim, uma recompensa recebida  $k$  intervalos de tempo no futuro vale apenas  $\gamma^{k-1}$  vezes o que ela valeria se fosse recebida imediatamente. Para  $\gamma < 1$ , o somatório infinito definido em 3.1 tem valor finito, desde que a sequência de recompensas  $r_k$  seja limitada.

Escolhendo  $\gamma = 0$ , o agente irá maximizar apenas a recompensa imediata  $r_{t+1}$ , já que todas as recompensas futuras serão nulas. Contudo, essa abordagem pode limitar o acesso do agente a recompensas futuras e diminuir o retorno. À medida que  $\gamma$  se aproxima de 1, o objetivo do agente passa a considerar mais as recompensas futuras. Sendo assim, é importante encontrar um equilíbrio nesse peso, ao levar em conta as características do processo e o objetivo em questão. Nesse projeto são explorados diversos valores de  $\gamma$  e o seu impacto no aprendizado e desempenho do controlador de nível de um tanque.

### 3.3 Estado e a Propriedade de Markov

A decisão tomada pelo agente, definida pela *policy*, é uma função do sinal de estado. Dessa forma, a entidade estado engloba todo tipo de informação disponível para o agente. O estado não deve informar ao agente tudo sobre o ambiente, mas só o que lhe é útil no aprendizado.

É preciso que o estado resuma a experiência de maneira compacta, mas de forma que toda informação relevante seja mantida. Isso geralmente requer mais do que as condições atuais, mas nunca deve ser mais do que um histórico completo dessas condições. Um sinal de estado que consegue reter todas as informações relevantes atende a propriedade de Markov, ou é simplesmente dito Markov. Por exemplo, a configuração atual de um tabuleiro de xadrez serve como estado Markov pois resume tudo de importante na sequência completa de movimentos dos jogadores que levou àquela configuração. Mesmo que a informação sobre jogadas individuais tenha sido perdida, tudo que realmente importa para o decorrer do jogo está incluso nesse estado.

Para definir formalmente a propriedade de Markov, pode-se assumir que há um número finito de estados e valores de recompensa. Assim é possível defini-la em termos de somatórios e probabilidades ao invés de integrais e densidade de probabilidade, porém, o argumento se estende para incluir estados e recompensas contínuos.

Em um instante de tempo  $t + 1$ , o ambiente gera os sinais de recompensa ( $r_{t+1}$ ) e de estado ( $s_{t+1}$ ) em relação à ação tomada no tempo  $t$ . No caso mais geral, esses

sinais podem depender de tudo o que ocorreu no passado. Assim, essa dinâmica só pode ser definida em termos da distribuição de probabilidade completa, conforme a Equação 3.2.

$$Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0\} \quad (3.2)$$

Por outro lado, se o estado é Markov, então a resposta do ambiente no tempo  $t + 1$  depende apenas do estado e da ação no tempo  $t$  e a Equação 3.2 de dinâmicas do ambiente pode ser reescrita como a Equação 3.3:

$$Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\} \quad (3.3)$$

Sendo assim, o sinal de estado tem a propriedade de Markov se e somente se as Equações 3.3 e 3.2 forem equivalentes para todo  $s', r$  e histórico  $s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0$ . Nesse caso, o ambiente também atende à propriedade de Markov.

Dado que o ambiente é Markov, então a sua dinâmica de um passo (Equação 3.3) é capaz de informar o valor do próximo estado e a próxima recompensa esperada, dados o estado e ação atuais. Sendo assim, estados Markov fornecem a melhor base possível para a escolha de ações. Ou seja, a melhor *policy* para escolha de ações tendo em função um estado Markov é tão boa quanto a melhor *policy* para escolha de ações tendo em função um histórico completo.

## 3.4 Processo de Decisão de Markov

A interação entre agente e ambiente que satisfaz essa propriedade é então chamada de Processo de Decisão de Markov (MDP). Esse MDP é definido pela tupla:  $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ , onde  $\mathcal{S}$  é o espaço de estados,  $\mathcal{A}$  é o espaço de ações,  $\mathcal{P}$  é a função de transição de estados (dinâmica do ambiente) e  $\mathcal{R}$  a função de recompensa.

Tanto  $\mathcal{S}$  quanto  $\mathcal{A}$  podem ser espaços discretos ou contínuos. As aplicações mais simples de RL utilizam espaços discretos, porém nesse trabalho são tratados espaços contínuos, de forma a acompanhar os avanços na área e também pela natureza contínua de processos químicos. Esse tratamento de espaços contínuos é definido na Seção 3.7

$\mathcal{P}$  é a função de transição de estados estocástica  $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ . Essa função indica qual será o estado resultante  $s_{t+1}$  a partir de uma ação  $a_t$  aplicada sobre o estado atual  $s_t$ . Como o processo do trabalho em questão não apresenta nenhum caráter estocástico, essa função é determinística e será definida pela dinâmica do tanque cujo nível será controlado. Essa dinâmica é definida no Capítulo 4.

De maneira análoga,  $\mathcal{R}$  é a função de recompensa  $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ .  $\mathcal{R}$  indica qual será a recompensa resultante  $r_{t+1}$  a partir de uma ação  $a_t$  aplicada sobre o estado atual  $s_t$ , atingindo-se o estado  $s_{t+1}$ . Diferentes funções de recompensa  $\mathcal{R}$  são testadas e comparadas no Capítulo 5, tendo em função o setpoint do controlador de nível.

Definindo-se o MDP, o problema de RL é então formalizado e basta agora definir como ocorre o aprendizado do agente.

### 3.5 Value Function

Conforme apresentado na Seção 3.2, o objetivo do agente é determinar a ação que vai dar o maior retorno esperado no estado atual. Isso é implementado por meio de uma *policy* que mapeia estados em ações, assim, a questão passa ser encontrar a *policy* ótima – aquela que escolhe a ação que vai resultar no melhor retorno futuro possível. A *value function* representa a recompensa acumulada descontada (ou retorno, segundo a Equação 3.1) que o agente espera receber no futuro após atingir um determinado estado. Dessa forma, é atribuído um valor para cada estado.

De fato, as recompensas que o agente espera receber no futuro depende das suas ações futuras. Como essa escolha de ações está definida na *policy*, a *value function* passa a ser uma função do estado e da *policy*. Dessa forma, para MDPs, definimos a *value function*  $V^\pi(s)$  segundo a Equação 3.4.

$$V^\pi(s) = E_\pi\{R_t \mid s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\} \quad (3.4)$$

em que  $E_\pi$  representa o valor esperado de uma expressão, nesse caso do retorno, em função de uma determinada *policy*  $\pi$ .

Desse modo, temos a seguinte dinâmica da interação agente-ambiente: a cada instante de tempo  $t$ , o estado  $s_t$  é observado e o agente então seleciona uma ação  $a_t$  e recebe uma recompensa  $r_{t+1}$  ao chegar no novo estado  $s_{t+1}$ .

Na metodologia desenvolvida nesse projeto, ao final de cada intervalo de tempo, o agente atualiza a sua estimativa de *policy* ótima com a informação fornecida pela última transição ocorrida  $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$ . A *policy* é sempre atualizada de forma a escolher as ações que resultam nos estados de maior valor, ou seja, que maximizam a *value function*.

### 3.6 Exploração

O aprendizado via RL é feito de modo similar a tentativa e erro. Inicialmente, o agente não tem nenhuma informação sobre o ambiente ou sobre a tarefa que ele deve



desempenhar. Ele precisa então descobrir quais estados retornam boas recompensas e melhorar a sua *policy* até maximizar o retorno esperado.

Quando o agente está escolhendo uma ação para tomar, há duas abordagens possíveis: aproveitar ou explorar. Aproveitar significa simplesmente escolher a ação que leva ao estado com a maior *value function* dentre os estados alcançáveis. Enquanto isso, explorar significa escolher outra ação, que terá um retorno esperado menor ou que o retorno esperado ainda é desconhecido. O equilíbrio entre explorar e aproveitar é um dos desafios de RL.

Um exemplo do dia-a-dia bastante elucidativo sobre explorar e aproveitar é proposto por SILVER [20]. Supõe-se que a tarefa em questão seja sair para jantar. Nesse caso, aproveitar seria o agente ir no seu restaurante favorito e explorar seria ir em algum restaurante novo. Claramente, há um risco associado ao desconhecido, pois o novo pode ser melhor ou pior do que o restaurante favorito. Porém, se o agente sempre for ao favorito, ele jamais saberá se existe algum outro lugar melhor do que esse para comer, contentando-se com a solução não-ótima. A mesma lógica se aplica ao aprendizado de qualquer tarefa via RL.

Explorar significa escolher deixar de ganhar uma recompensa esperada com o objetivo de aprender mais sobre o ambiente. Aproveitar significa usar a informação que o agente já tem, de forma a maximizar o retorno.

Nesse trabalho, são exploradas duas técnicas distintas de exploração presentes na literatura. A metodologia utilizada no artigo de FERNANDEZ-GAUNA *et al.* [19], chamada de exploração gaussiana, envolve a adição de uma perturbação ao sinal da *policy*, segundo a Equação 3.5

$$a_t = \pi(s_t) + N(0, \sigma_t^2) \quad (3.5)$$

em que o termo  $N(0, \sigma_t^2)$  produz um número aleatório seguindo uma distribuição gaussiana de média 0 e desvio padrão  $\sigma_t$ . Esse desvio padrão  $\sigma_t$  decresce ao longo do tempo para que o processo de aprendizado convirja. Assim, no início do aprendizado o agente explora bastante, por não conhecer o ambiente ainda. Posteriormente, ele passa a aproveitar mais as informações obtidas e suas ações refletem diretamente a *policy*.

Já a outra abordagem, proposta por SUTTON e BARTO [8], é conhecida como  *$\epsilon$ -greedy*. A cada passo, a ação de explorar é aplicada com probabilidade  $\epsilon$  e a ação de aproveitar com probabilidade  $1 - \epsilon$ , segundo a Equação 3.6.

$$a_t = \begin{cases} \pi(s_t), & \text{com probabilidade } 1 - \epsilon_t; \\ \text{uma ação uniformemente aleatória em } \mathcal{A}(s_t), & \text{com probabilidade } \epsilon_t, \end{cases} \quad (3.6)$$

em que o fator  $\epsilon$  decresce ao longo do tempo, de maneira análoga ao desvio padrão  $\sigma_t$  da Equação 3.5, o que garante um alto índice de exploração no início do aprendizado e diminui conforme o agente acumula mais informações sobre o ambiente.

### 3.7 Aproximação da *Value Function*

Conforme apresentado anteriormente na Seção 3.4, no presente trabalho são tratados espaços de estados e de ações contínuos. Para isso é utilizada a técnica de Aproximação da *Value Function* (VFA). As VFAs utilizam um conjunto de funções de ativação chamadas de *features*  $\phi$ , ilustrado na Equação 3.7.

$$\begin{aligned} \phi : \mathcal{S} &\rightarrow \mathbb{R}^F \\ \phi(s) &= [\phi_1(s), \phi_2(s), \dots, \phi_F(s)]^T \end{aligned} \quad (3.7)$$

onde  $F$  é o número de *features* usadas para representar o espaço de estados.

No presente trabalho são abordadas VFAs lineares, tendo em vista que a sua convergência é garantida, o que não é o caso para VFAs não-lineares. Uma VFA linear é a aproximação de uma dada função  $f(s)$  e é denotada  $\hat{f}(s)$ , segundo a Equação 3.8:

$$\hat{f}(s) = \theta^T \phi(s) \quad (3.8)$$

em que  $\theta$  é o vetor de parâmetros da combinação linear  $\theta = [\theta_1, \theta_2, \dots, \theta_F]^T$ , também chamado de vetor de peso das *features*. Esses parâmetros  $\theta_i$  serão estimados pela interpolação das *features* sobre todo o espaço de estados contínuo, o que é apresentado nas Seções 3.9 e 3.10. A VFA pode ser utilizada para aproximar qualquer função dependente dos estados, como a *value function*  $V^\pi$  e a *policy*  $\pi$ .

Funções multivariáveis podem ser representadas por um conjunto de funções de uma só variável e modeladas por VFAs independentes. Por exemplo, para um espaço de ações multidimensional, a policy precisa escolher uma ação para cada dimensão e assim temos  $\pi(s) = \{\pi_1(s), \pi_2(s), \dots, \pi_m(s)\}$ , sendo  $m$  a dimensão do espaço de ações.

Nesse trabalho foram utilizadas funções de base radial gaussianas (RBF)  $\phi_i(s)$ , cada uma associada a um centro  $c_i$  no espaço de estados  $\mathcal{S}$ , segundo a Equação 3.9.

$$\phi_i(s) = e^{-\frac{\|s - c_i\|^2}{2\sigma_\phi^2}} \quad (3.9)$$

em que  $\sigma_\phi^2$  define a largura da função gaussiana. Os centros das funções foram distribuídos uniformemente ao longo do espaço de estados, e  $\sigma$  foi definido como metade da distância entre os centros de duas RBFs adjacentes, a fim de que as gaussianas não fiquem muito sobrepostas nem muito espaçadas.

Por exemplo, para um espaço de estados unidimensional  $\mathcal{S} \in [0, 1]$ , definindo-se 3 *features*, teremos 3 RBFs, uma com centro em 0, outra em 0.5 e a última em 1. Dessa forma, avaliando o valor das 3 funções para  $s = 0.4$ , tem-se o vetor de *features*  $\phi(0.4) \approx [0.28, 0.92, 0.06]^T$ . Esses 3 valores são os pontos nos quais a reta vertical  $s = 0.4$  destacada em verde intercepta as 3 RBFs conforme ilustra a Figura 3.1.

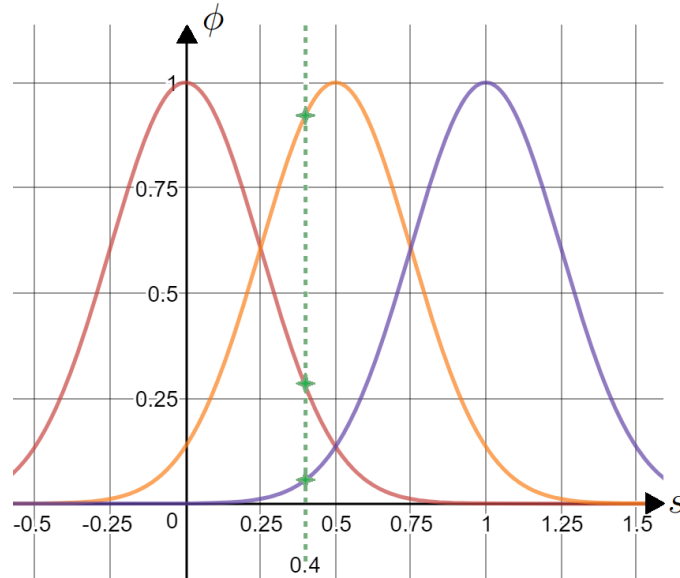


Figura 3.1: Exemplo ilustrativo da VFA por meio de 3 RBFs.

Para calcular então a VFA nesse exemplo, basta fazer uma combinação linear do vetor de *features*  $\phi(0.4)$  com o vetor de pesos  $\theta$ . Supondo que  $\theta = [1, 2, 3]^T$ , então  $\theta^T \phi(0.4) = 1 \times 0.28 + 2 \times 0.92 + 3 \times 0.06 = 0.28 + 1.84 + 0.18 = 2.3$ .

### 3.8 Metodologia *Actor-Critic*

A metodologia de aprendizado aplicada nesse trabalho é chamada de *actor-critic*. O *actor* armazena e atualiza a *policy* enquanto o *critic* faz o mesmo para a *value function* e essas duas entidades interagem continuamente. Nessa abordagem, ao final de cada iteração, com o agente tendo tomado a ação  $a_t$  e atingido o estado  $s_{t+1}$ , é feito um processo de duas etapas. O primeiro, chamado de avaliação da *policy*,

é feito pelo *critic*, que atualiza a sua estimativa da *value function*  $V^\pi$  a partir dos novos dados obtidos com a transição de estados ocorrida. Para que o *actor* possa em seguida fazer a atualização da *policy*, o *critic* fornece a ele um feedback, o erro de diferença temporal  $\delta_t$  que representa o incremento da *value function* no estado  $s_t$ , segundo a Equação 3.10:

$$\delta_t = r_t + \gamma \hat{V}_\pi(s_{t+1}) - \hat{V}_\pi(s_t) \quad (3.10)$$

em que  $\hat{V}_\pi(s_{t+1})$  e  $\hat{V}_\pi(s_t)$  são as *value functions* estimadas nos estados  $s_t$  e  $s_{t+1}$  respectivamente.

Pela natureza contínua dos espaços de estados e ações do processo a ser controlado nesse projeto, tanto a *value function* quanto a *policy* devem ser aproximadas. Essa aproximação é feita conforme descrito na Seção 3.7.

O algoritmo de aprendizado AC implementado aqui para o controle de nível de um tanque é uma adaptação do algoritmo apresentado por FERNANDEZ-GAUNA *et al.* [19]. Segundo a definição dada na Equação 3.8, a *value function* do estado  $s$  é estimada pela Equação 3.11

$$\hat{V}_\pi(s_t) = \theta_V^T \phi(s_t) \quad (3.11)$$

O mesmo vale para a *policy*, segundo a Equação 3.12

$$\pi(s_t) = \theta_\pi^T \phi(s_t) \quad (3.12)$$

Desse modo, o processo de aprendizado consiste em ajustar iterativamente os valores dos pesos  $\theta_V$  e  $\theta_\pi$  e será descrito nas seções seguintes 3.9 e 3.10.

### 3.9 Avaliação de *Policy*

O *critic* utiliza o método conhecido como TD( $\lambda$ ) para a avaliação da *policy* por meio da estimação da *value function* do estado. O seu objetivo é minimizar o erro de diferença temporal definido na Equação 3.10.

A metodologia TD( $\lambda$ ) utiliza o conceito de traço de elegibilidade. Primeiro é apresentado o conceito para o caso mais simples de espaços de estados discretos e em seguida como ele foi aplicado ao espaço de estados contínuo do presente trabalho.

Quando o agente atinge um determinado estado e recebe uma determinada recompensa, não se pode dizer que a última ação que ele tomou foi 100% responsável por providenciar essa recompensa. O traço de elegibilidade é uma memória dos estados recentes e o quanto eles foram visitados ao longo do tempo. Com isso, ele consegue propagar a responsabilidade por adquirir uma certa recompensa ao longo

dos estados visitados, com os mais recentes e mais visitados priorizados.

Dessa forma, em um espaço de estados discreto, cada vez que um estado é visitado, o seu contador é incrementado. Para dar mais valor aos estados recentes, essa contagem de visitas sofre um decaimento ao longo do tempo.

Formalizando, o traço de elegibilidade para um estado  $s$  no instante de tempo  $t$  é denotado  $e_t(s) \in \mathbb{R}^+$ . A cada intervalo de tempo, o traço de elegibilidade de cada estado decai pelo fator  $\gamma\lambda$ , e o traço de elegibilidade do estado atual visitado nesse intervalo de tempo é incrementado por 1, segundo a Equação 3.13

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s), & \text{se } s \neq s_t; \\ \gamma\lambda e_{t-1}(s) + 1, & \text{se } s = s_t, \end{cases} \quad (3.13)$$

para todo  $s \in \mathcal{S}$ , sendo  $\gamma \in [0, 1]$  o fator de desconto previamente apresentado na Seção 3.2 e  $\lambda \in [0, 1]$  um parâmetro de sintonia denominado fator de decaimento do traço. Esse tipo de traço é chamado de traço acumulativo, tendo em vista que o seu valor vai sendo incrementado cada vez que o estado é visitado e decai gradualmente ao longo do tempo.

A Figura 3.2 ilustra a variação do valor do traço de elegibilidade para um determinado estado  $s^*$  em função das suas ocorrências ao longo do tempo. Os traços verticais na parte inferior da figura denotam que o estado atual naquele momento é  $s^*$ , acarretando em um incremento no valor do traço de elegibilidade de  $s^*$ . O traço então decai ao longo do tempo até a próxima visita ao estado  $s^*$ , quando tem seu valor novamente incrementado.

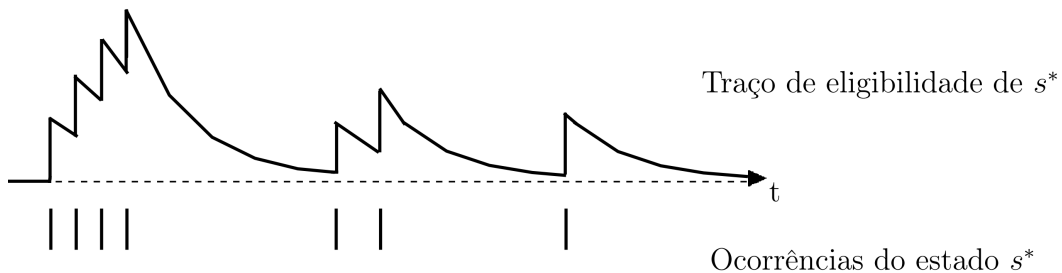


Imagem adaptada de SUTTON e BARTO [8].

Figura 3.2: Exemplo ilustrativo da variação do valor do traço de elegibilidade.

Para um espaço de estados contínuo, não é mais possível obter um traço de elegibilidade individual para cada estado. Então, ele passa a ser um vetor em que cada elemento corresponde a uma *feature*  $\phi_i$ , ou seja, cada função de base radial vai ter o seu valor do traço de elegibilidade. Portanto, o vetor do traço de elegibilidade passa a ser denotado  $z_t \in \mathbb{R}^F$  e é atualizado segundo a Equação 3.14.

$$z_{t+1} = \gamma \lambda z_t + \phi(s_t) \quad (3.14)$$

Assim, o traço de elegibilidade continua decaindo com o fator  $\gamma \lambda$  e o valor de cada *feature* é somado aos seus elementos, representando a visita aos estados. Vale recordar que o valor de cada *feature*  $\phi_i(s_t)$  varia de 0 a 1 de acordo com o valor do estado  $s_t$ , segundo a Equação 3.9.

Para realizar então a avaliação da *policy* e gerar uma nova estimativa da *value function*, é preciso atualizar o valor do vetor de pesos da *value function*  $\theta_V$ . Essa atualização é feita segundo a Equação 3.15:

$$\theta_{V,t+1} = \theta_{V,t} + \alpha_V z_{t+1} \{r_t + \gamma \phi_V(s_{t+1})^T \theta_{V,t} - \phi_V(s_t)^T \theta_{V,t}\} \quad (3.15)$$

em que  $\alpha_V \in [0, 1]$  é um parâmetro de sintonia do aprendizado que ajusta o valor do passo de atualização na direção do parâmetro ótimo. É importante ressaltar também que o termo entre chaves é o mesmo erro de diferença temporal  $\delta_t$  ilustrado na Equação 3.10, e tem aqui a sua representação em termos da VFA do espaço de estados contínuo destacado na Equação 3.16. O *critic* transmitirá esse valor para o *actor* e ele fará parte da atualização do vetor de pesos da *policy*  $\theta_\pi$ .

$$\delta_t = r_t + \gamma \phi_V(s_{t+1})^T \theta_{V,t} - \phi_V(s_t)^T \theta_{V,t} \quad (3.16)$$

## 3.10 Melhoria de *Policy*

Após a avaliação da *policy* feita pelo *critic*, o *actor* então pode atualizar os pesos  $\theta_\pi$ , em direção à *policy* ótima, aquela que maximiza o retorno. Essa atualização é definida na Equação 3.17 [7].

$$\theta_{\pi,t+1} = \theta_{\pi,t} + \alpha_\pi \nabla_{\theta_\pi} \pi(s_t) (a_t - \pi(s_t)) \delta_t \quad (3.17)$$

em que  $\alpha_\pi \in [0, 1]$  é um parâmetro de sintonia do aprendizado análogo a  $\alpha_V$ , que ajusta o valor do passo de atualização na direção do parâmetro ótimo;  $\nabla_{\theta_\pi} \pi(s_t)$  é o gradiente da *policy*  $\pi(s_t)$  em relação a  $\theta_\pi$  e  $a_t$  é a ação realizada no instante de tempo  $t$ . No Capítulo 5 serão apresentadas simulações com diferentes valores de  $\alpha_V$  e  $\alpha_\pi$  para comparar o seu efeito no aprendizado do controlador de nível.

Como ilustra o termo  $\nabla_{\theta_\pi} \pi(s_t)$ , esse é um método de descida de gradiente. Vale recordar que a *policy*  $\pi$  é definida da Equação 3.12 como sendo  $\pi(s_t) = \theta_\pi^T \phi_\pi(s_t)$ . Portanto:

$$\nabla_{\theta_\pi} \pi(s_t) = \nabla_{\theta_\pi} (\theta_\pi^T \phi_\pi(s_t)) = \phi_\pi(s_t) \quad (3.18)$$

Usando a Equação 3.18, a Equação 3.17 pode então ser reescrita como:

$$\theta_{\pi,t+1} = \theta_{\pi,t} + \alpha_{\pi} \phi_{\pi}(s_t) \delta_t (a_t - \pi(s_t)) \quad (3.19)$$

O termo  $(a_t - \pi(s_t))$  da Equação 3.19 reflete a diferença entre a ação que realmente foi tomada  $a_t$  (levando em conta a exploração) e a ação determinada pela *policy* naquele estado  $\pi(s_t)$ . Se essa diferença resultou em uma recompensa mais positiva que o esperado, o erro de diferença temporal  $\delta_t$  também será positivo e a *policy* será atualizada na direção de  $a_t$ . O oposto ocorre caso a recompensa seja mais negativa, com a *policy* se afastando então da direção de  $a_t$ .

Uma outra maneira de se atualizar os pesos  $\theta_{\pi}$  é conhecida como *Continuous Actor Critic Learning Automaton* (CACLA) [21]. Nessa abordagem, o valor só é atualizado caso o erro diferencial temporal comunicado pelo *critic* ao *actor* seja positivo, segundo a Equação 3.20:

$$\theta_{\pi,t+1} = \begin{cases} \theta_{\pi,t+1} = \theta_{\pi,t} + \alpha_{\pi} \phi(s_t) (a_t - \pi(s_t)) & \text{se } \delta_t > 0; \\ \theta_{\pi,t} & \text{se } \delta_t \leq 0, \end{cases} \quad (3.20)$$

Dessa forma,  $\theta_{\pi}$  só é atualizado quando o erro diferencial temporal é positivo, ou seja, uma melhoria na *policy* é esperada. Sendo assim, ao considerar o caso em que o *actor* já aprendeu a melhor *policy* possível mas continua explorando outras alternativas, alterar a *policy* na direção oposta de um erro diferencial temporal negativo não irá melhorar a *policy*. Essa é a justificativa por trás da abordagem CACLA.

Ao longo das simulações implementadas nesse trabalho, tanto a abordagem clássica (Equação 3.19) quanto a CACLA (Equação 3.20) são utilizadas e comparadas para a etapa de melhoria de *policy* durante o aprendizado do controlador de nível.

# Capítulo 4

## Desenvolvimento do Controle de Nível

Nesse capítulo, será apresentada a implementação em MATLAB de um controlador de nível utilizando *Reinforcement Learning*. A Figura 4.1 ilustra as interações de RL no contexto desse projeto.

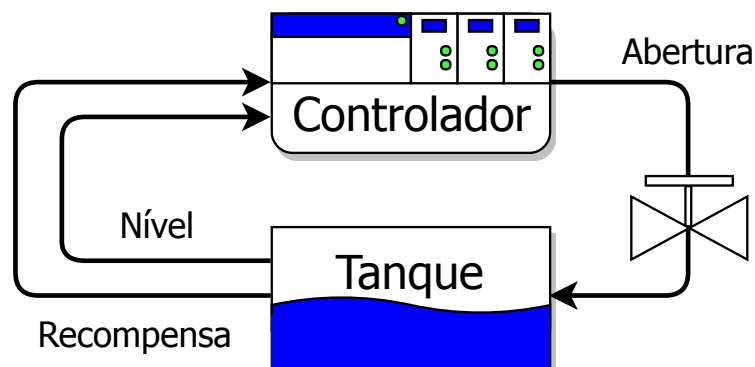


Figura 4.1: Interações da metodologia RL para controle de nível de um tanque.

Nas seções seguintes será definido o MDP, que conforme apresentado na Seção 3.4, é composto pela tupla:  $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ , onde  $\mathcal{S}$  é o espaço de estados,  $\mathcal{A}$  é o espaço de ações,  $\mathcal{P}$  é a função de transição de estados e  $\mathcal{R}$  a função de recompensa.

### 4.1 A Planta

No escopo desse projeto, a planta a ser controlada é um tanque alimentado a uma vazão contínua  $F_i$ . O nível do tanque  $h$  é a variável controlada, e portanto, o estado. A variável manipulada é a abertura da válvula de saída  $X_V$ , considerada a ação. Essa válvula produz uma vazão de saída proporcional à constante  $C_V$  e o tanque tem seção transversal constante  $A$ . A figura 4.2 ilustra a planta com todas



as suas variáveis destacadas.

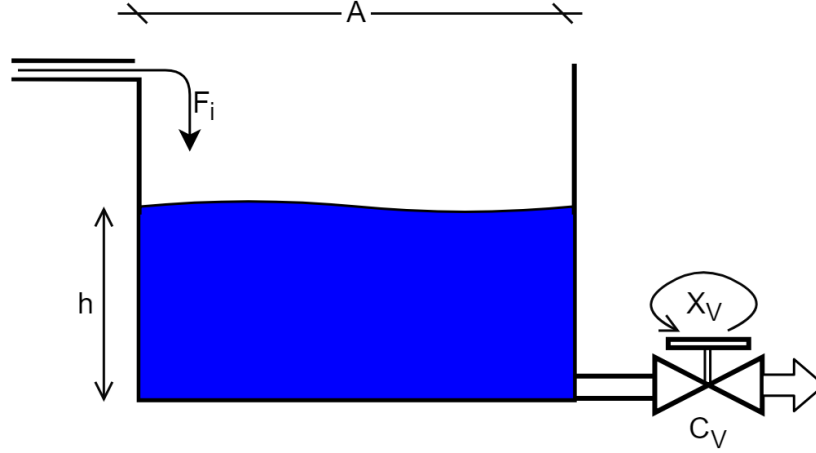


Figura 4.2: Diagrama ilustrativo da planta a ser controlada.

Para o escoamento da válvula de saída, é considerada uma vazão diretamente proporcional à raiz quadrada do nível  $h$ , à abertura  $X_V$  e à constante da válvula  $C_V$ . Essa vazão de saída  $F_o$  é ilustrada na equação 4.1.

$$F_o = C_V \times X_V \times \sqrt{h} \quad (4.1)$$

É possível escrever o balanço de massa do tanque em termos de seu volume, ao se adotar a hipótese de líquido incompressível. Assim, considerando-se uma vazão de entrada  $F_i$ , tem-se a equação dinâmica da planta na Equação 4.2.

$$\frac{dV}{dT} = F_i - F_o = F_i - C_V \times X_V \times \sqrt{h} \quad (4.2)$$

em que  $V$  é o volume do tanque [ $\text{m}^3$ ]. Como o tanque possui seção transversal  $A$  constante e volume  $V = h \times A$ , é possível separar a parcela constante  $A$  da derivada conforme a Equação 4.3.

$$\frac{dV}{dT} = \frac{d(h \times A)}{dt} = \frac{dh}{dt} \times A \quad (4.3)$$

Por fim, ao aplicar a equação 4.3 na equação 4.2, obtém-se a dinâmica do nível da planta segue a Equação 4.4.

$$\frac{dh}{dt} = \frac{F_i - (C_V \times X_V \times \sqrt{h})}{A} \quad (4.4)$$

sendo  $h$  o nível do tanque [m],  $F_i$  a vazão de entrada [ $\text{m}^3/\text{h}$ ],  $C_V$  a constante da válvula [ $\text{m}^{2.5}/\text{h}$ ] e  $X_V$  a abertura da válvula [%].

Considerando um tanque com 1 metro de altura e 3 metros quadrados de seção

transversal, resta definir a constante da válvula  $C_V$ . O ponto de operação considerado para esse dimensionamento é o nível do tanque se manter estável na metade do tanque, ou seja, em 0.5m com uma vazão de entrada de  $100[\text{m}^3/\text{h}]$  e uma abertura da válvula de saída de 50%. Assim, tem-se a constante  $C_V$  segundo a Equação 4.5:

$$C_V = \frac{F}{X_V \times \sqrt{h}} = 282.8427 [\text{m}^{2.5}/\text{h}] \quad (4.5)$$

Dessa forma para uma vazão de entrada  $F_i = 100\text{m}^3/\text{h}$  e o nível pela metade  $h = 0.5\text{m}$ , abrindo a válvula pela metade  $X_V = 50\%$ , tem-se que a vazão de entrada é igual a de saída, mantendo o nível estável. A Tabela 4.1 resume os parâmetros constantes do tanque considerado nesse projeto:

Tabela 4.1: Parâmetros do tanque

Altura [m]	1
Área [ $\text{m}^2$ ]	3
$C_V$ [ $\text{m}^{2.5}/\text{h}$ ]	282.8427

De forma a adicionar uma ação integral no controlador, é possível definir o sinal de ação não como sendo a abertura absoluta da válvula  $X_V$ , mas sim a diferença  $\Delta X_V = X_{V_{t+1}} - X_{V_t}$ . Assim, a ação  $a$  passa a ser o quanto deve variar a abertura da válvula em relação a abertura atual. Dessa forma, quando o sistema atingir o equilíbrio, a ação é zerada,  $a = \Delta X_V = 0$ .

Em suma, tem-se que o estado  $s$  é o nível do tanque  $h$ , a ação  $a$  é a variação na abertura da válvula  $\Delta X_V$  e a função de transição de estados  $\mathcal{P}$  é definida pela Equação 4.4. É preciso então, definir o sinal de recompensa  $\mathcal{R}$ , que será discutido na Seção 4.2.

Para implementar a dinâmica descrita na Equação 4.4, foi utilizada a função `ode45` do MATLAB. Esse é o resolvidor de equações diferenciais ordinárias mais flexível do programa, fornecendo boa precisão com baixo custo computacional. A equação é resolvida a cada iteração, sendo  $F_i$ ,  $C_V$  e  $A$  constantes,  $h = s_t$  e  $X_V$  sendo atualizado com o incremento definido pela ação,  $X_V \leftarrow X_V + a$ . O intervalo de tempo entre cada iteração foi definido como  $\Delta t = 0.1$ , com a unidade de tempo sendo horas ( $0.1 \text{ h} = 6 \text{ min}$ ).

## 4.2 Definindo a Função de Recompensa

Duas abordagens de recompensa são simuladas e comparadas nesse trabalho. A primeira abordagem, que deriva da teoria de controle, pode ser vista na Equação 4.6

e é definida como o quanto o nível se aproximou do setpoint em relação ao estado anterior. Essa função de recompensa será denominada  $\mathcal{R}_1$ .

$$\mathcal{R}_{1_{t+1}} = |h_{sp} - h_t| - |h_{sp} - h_{t+1}| \quad (4.6)$$

sendo  $h_{sp}$  o setpoint do nível,  $h_t$  o estado anterior e  $h_{t+1}$  o estado alcançado.

Dessa forma, ao escolher uma ação que faça o nível se aproximar do setpoint, a recompensa é positiva. Por outro lado, se o nível se afastar do setpoint após a ação determinada pelo agente, então a recompensa é negativa. Assim, no momento de atualização da *policy*, ações com recompensa negativa são desfavorecidas enquanto ações com recompensa positiva são favorecidas.

Por outro lado, a literatura clássica de RL recomenda definir a função de recompensa da maneira mais simples possível [8]. Em tarefas em que se tem um objetivo simples e claro, como o exemplo dado na Seção 3.2 de um jogo de xadrez, basta definir a recompensa como +1 por atingir o objetivo e 0 para o resto do espaço de estados e ações.

No caso do controlador de nível, o objetivo é atingir o setpoint. Porém, não é possível seguir essa abordagem e definir uma recompensa somente para o caso de  $h = h_{sp}$  por questões numéricas. Dessa forma, foi proposta uma região em torno do setpoint em que a recompensa seria diferente de zero. Além disso, para que o controlador não atinja essa região e simplesmente pare de atuar (vale recordar que no equilíbrio  $a = \Delta X_V = 0$ ), é preciso que essa recompensa aumente conforme a distância para o setpoint diminua, sendo máxima quando  $h = h_{sp}$ . Essa função é definida na Equação 4.7 e será denominada  $\mathcal{R}_2$ .

$$\mathcal{R}_{2_{t+1}} = \begin{cases} (0.1 - |h_{sp} - h_{t+1}|) \times 10 & \text{se } |h_{sp} - h_{t+1}| < 0.1; \\ 0 & \text{se } |h_{sp} - h_{t+1}| \geq 0.1, \end{cases} \quad (4.7)$$

Assim a recompensa varia de 0 a 1 quando  $h \in [h_{sp} - 0.1, h_{sp} + 0.1]$ , sendo máxima (= 1) quando  $h = h_{sp}$ .

### 4.3 Actor-Critic e VFAs

O controle de nível do tanque via RL é feito utilizando a metodologia *Actor-Critic*, apresentada na Seção 3.8. Essa abordagem define duas estruturas separadas para a *value function*  $V^\pi$ , aprendida pelo *critic*; e para a *policy*  $\pi$ , aprendida pelo *actor*. Ambos os casos são definidos por uma combinação linear das *features*  $\phi(s)$ , segundo as Equações 3.11 e 3.12.

Em ambos os casos, o vetor de *features*  $\phi(s)$  é o mesmo. Para defini-lo, foram utilizadas 10 RBFs (Equação 3.9), distribuídas ao longo do espaço de estados. A

primeira ( $\phi_1(s)$ ) foi posicionada em  $c_1 = 0$ , e a última ( $\phi_{10}$ ) em  $c_{10} = 1$ . O restantes das RBFs foram distribuídas uniformemente ao longo do espaço de estados ( $[0, 1]$ ), conforme ilustra a Figura 4.3.

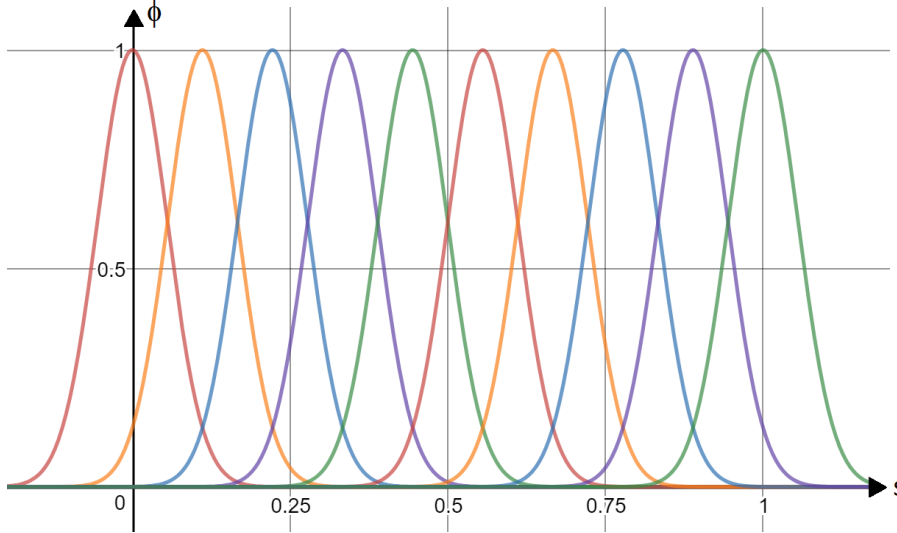


Figura 4.3: Visualização das 10 RBFs definidas para as VFAs do *Actor-Critic*.

## 4.4 Parâmetros da Exploração

Conforme apresentado na Seção 3.6, duas maneiras de exploração são simuladas e comparadas: a exploração gaussiana conforme a Equação 3.5 e a exploração  $\epsilon$ -greedy, segundo a Equação 3.6. É preciso então definir como o parâmetro de exploração  $\epsilon_t$  e o desvio padrão  $\sigma_t$  decaem ao longo do tempo e diminuem o alcance da exploração do agente. Para efeitos de simplicidade, é tratado o decaimento do parâmetro  $\epsilon_t$ , que representa uma probabilidade ( $\epsilon_t \in [0, 1]$ ) e em seguida o seu valor é ajustado por um fator para se obter o valor do desvio padrão  $\sigma_t$ .

O decaimento do parâmetro de exploração pode ser definido com uma função dos estados visitados. Assim, para um estado que foi visitado poucas vezes, assume-se que a informação que o agente tem dele é insuficiente e há grande necessidade de exploração. Por outro lado, para um estado que já foi visitado diversas vezes, é tido que a sua *value function* aproximada deve ser bem próxima da real, e assim o agente não precisaria mais explorar, podendo aproveitar essa informação para decidir realizar a ação que trará o maior retorno esperado.

Em um espaço de estados discretos, essa implementação é trivial. Basta armazenar um contador de número de visitas de cada estado  $N(s)$  e definir o parâmetro  $\epsilon_t$  de forma que ele diminuía conforme o número de visitas  $N(s_t)$  aumente. Dessa

forma, a cada intervalo de tempo o valor de  $\epsilon_t$  é atualizado segundo a Equação 4.8.

$$\epsilon_t = \frac{N_0}{N_0 + n(s_t)} \quad (4.8)$$

em que  $s_t$  é o estado atual,  $N_0$  é um parâmetro de sintonia e  $n(s_t)$  é o número de vezes que o estado atual foi visitado. Assim, para  $n(s_t) = 0$ ,  $\epsilon_t = 1$  e o agente explora e para  $n(s_t) \rightarrow \infty$ ,  $\epsilon_t \rightarrow 0$  e o agente aproveita.

Já em um espaço de estados contínuo, não é mais possível manter um contador de número de visitas de cada estado, pois existem infinitos estados. Sendo assim, o espaço de estados deve ser dividido em regiões e o contador de número de visitas passa a ser feito por região. O número de regiões definido é definido pelo número de RBFs e, portanto, o número de *features*. O número escolhido nesse trabalho foi de 10 RBFs, conforme ilustra a figura 4.3, dividindo o espaço de estados em 10 regiões.

Dado um estado  $s_t$ , para descobrir em qual região ele se encontra, basta descobrir qual é a *feature*  $\phi_i$  com o maior valor para aquele estado. Assim, é possível saber qual é a RBF mais próxima daquele estado e portanto em qual região ele se encontra. Isso é o mesmo que achar o maior valor dentre os elementos do vetor  $\phi(s_t)$  e é denotado  $\max(\phi(s_t))$ .

Dessa forma, sendo  $N(\max(\phi(s_t)))$  o contador que armazena o número de visitas a cada região, o valor de  $\epsilon_t$  passa a seguir a Equação 4.9:

$$\epsilon_t = \frac{N_0}{N_0 + N(\max(\phi(s_t)))} \quad (4.9)$$

A escolha do valor do parâmetro de sintonia  $N_0$  é apresentada no Capítulo 5. Resta então definir  $\sigma_t$  para a exploração gaussiana. O decaimento deve seguir  $\epsilon_t$ , portanto basta ajustá-lo, multiplicando-o por uma constante, segundo a Equação 4.10

$$\sigma_t = \epsilon_t \times \sigma_0 \quad (4.10)$$

em que  $\sigma_0$  é uma constante que escala o valor de  $\epsilon_t$  ( $\epsilon_t \in [0, 1]$ ) para um valor de desvio padrão. Novamente, a escolha do valor de  $\sigma_t$  é apresentada no Capítulo 5.

## 4.5 Desenvolvimento do Aprendizado via RL

Com todas as equações que regem o aprendizado do controle de nível de um tanque via RL desenvolvidas, resta apresentar algumas especificidades das simulações implementadas.

Primeiramente, os vetores de pesos  $\theta_V$  e  $\theta_\pi$  são iniciados com todos os seus elementos iguais a zero. O aprendizado é então feito em forma de episódios. Os episódios começam com  $s_t = h_0$  e  $X_V = X_{V0}$ . A vazão de entrada  $F_i$  e o setpoint

do nível  $h_{sp}$  são mantidos constantes, sendo portanto um problema de controle com caráter regulatório.

Para caracterizar os episódios de aprendizado, eles tem duração de  $5h$ , ou seja, com  $\Delta t = 0.1h$ , os episódios terminam após a iteração de número 50. Se o nível ultrapassar os limites físicos do tanque, esvaziando completamente ( $h \leq 0$ ) ou transbordando ( $h \geq 1$ ), o episódio é terminado. É entendido que a partir desse ponto a simulação passaria a não ser mais representativa do processo, além do fato de atingir esses pontos não ser desejável para o objetivo de controlar o nível do tanque.

Com o fim de um episódio, o nível e a abertura do tanque então voltam aos seus valores iniciais,  $s_t = h_0$ ,  $X_V = X_{V0}$ . Porém os vetores de pesos  $\theta_V$  e  $\theta_\pi$  são mantidos e seus valores constantemente atualizados – esse é o resultado do aprendizado.

Ao final de um número determinado de episódios, o aprendizado é dado como completo. Para avaliar o desempenho do controlador, é rodado um episódio da simulação utilizando apenas o  $\theta_\pi$  calculado, sem que as ações tenham o fator de exploração. Assim, durante essa avaliação, o aprendizado não acontece mais, sendo possível avaliar a atuação da *policy*  $\pi$  sem que ela sofra mais alterações.

Durante essa simulação final para testar o controlador, o valor de  $F_i$  será alterado em um determinado instante de tempo. Dessa forma, será possível testar a robustez do controle em função de perturbações não medidas, com uma variação na vazão de entrada do tanque.

## 4.6 Sintonia de um Controlador Clássico

Para efeitos de comparação, foi sintonizado um controlador clássico do tipo PID para o controle de nível do tanque. Os resultados então são comparados entre esse controlador e o controle via RL. Um controlador PID bem sintonizado pode ser obtido através da ferramenta *PID Tuner* do MATLAB.

Um diagrama do processo foi montado no Simulink, conforme a Figura 4.4. O controlador PID foi então adicionado ao sistema, porém a ferramenta *PID Tuner* não foi capaz de sintonizar um controlador estável devido à presença de não-linearidades no sistema. O controlador PID é do tipo discreto, com o mesmo tempo de amostragem utilizado no RL,  $\Delta t = 0.1h$ .

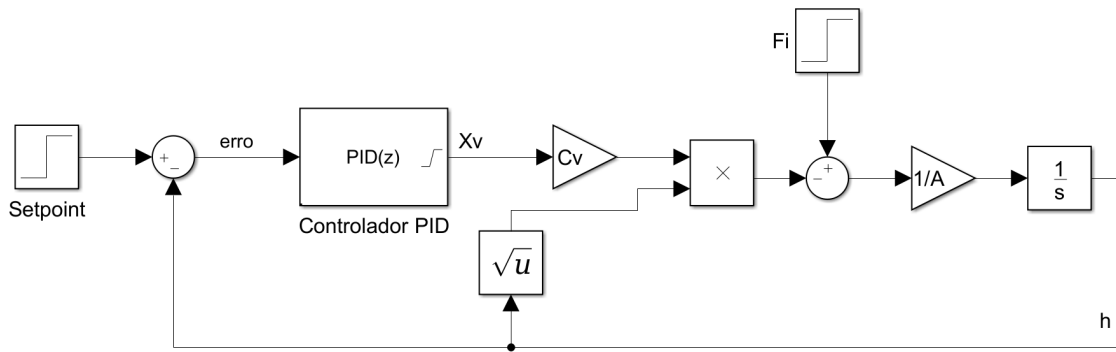


Figura 4.4: Implementação do controlador PID na ferramenta Simulink.

Para sintonizar o controlador, um modelo linear da planta foi obtido linearizando o modelo não-linear em torno do ponto de operação considerado. A modelagem proposta foi uma simples função de transferência de primeiro grau. Essa função de transferência  $G(s)$  é definida segundo a Equação 4.11.

$$G(s) = \frac{K}{\tau s + 1} \quad (4.11)$$

Sendo:

$$K = \frac{\text{Amplitude da variação em } h}{\text{Amplitude da variação em } X_V}$$

$\tau$  = constante de tempo do sistema, o tempo em que a variação da resposta chega a 63,2% do valor final

A linearização foi feita em torno do ponto de operação  $h = 0.5\text{m}$ , com  $X_V = 0.5\%$  e  $F_i = 100\text{m}^3/\text{h}$ . Foi aplicado uma variação de +5% no valor de  $X_V$  no instante  $t = 0.5\text{h}$  e em seguida a mesma variação no sentido oposto (-5%) no instante  $t = 1.0\text{h}$  e o nível resultante pode ser visto na Figura 4.5

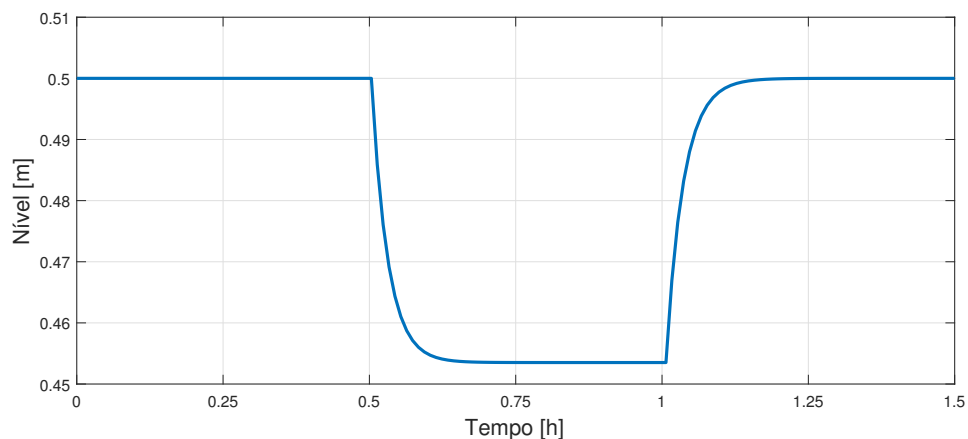


Figura 4.5: Resultado de uma variação de +5% em  $X_V$  no instante  $t = 0.5\text{h}$ .

Foram obtidos os parâmetros  $K = -0.93$  e  $\tau = 0.0287$  e a função de transferência aproximada do sistema é então definida pela Equação 4.12.

$$G(s) = \frac{-0.93}{0.0287s + 1} \quad (4.12)$$

Essa linearização apresentou ótimos resultados, com o sistema de primeira ordem apresentando uma resposta extremamente similar à da planta original ilustrada na Figura 4.5.

A planta então é discretizada com o mesmo tempo de amostragem utilizado para o aprendizado via RL, ou seja,  $\Delta t = 0.1h$ . Para a discretização foi utilizada a aproximação de Tustin, ou método de integração trapezoidal, por garantir uma boa precisão. A aproximação é definida pela Equação 4.13.

$$s = \frac{2}{\Delta t} \frac{z - 1}{z + 1} \quad (4.13)$$

Assim, ao aplicar a Equação 4.13 na Equação 4.12, tem-se:

$$G(z) = \frac{-0.93z - 0.93}{1.287z + 0.713} \quad (4.14)$$

Com o modelo de primeira ordem discretizado da Equação 4.14, foi então possível sintonizar um controlador discreto com a ferramenta *PID Tuner*. Este foi sintonizado com o mesmo tempo de amostragem  $\Delta t = 0.1$  e com o mesmo método de integração trapezoidal. O controlador resultante apresentou ganho derivativo nulo, sendo portanto apenas do tipo PI.

A sintonia gerada automaticamente pelo programa passou então por um ajuste fino de parâmetros manual. A função transferência resultante  $C(z)$  pode ser conferida na Equação 4.15.

$$C(z) = P + I \frac{\Delta t}{2} \frac{z + 1}{z - 1} \quad (4.15)$$

Sendo:

$P = -0.011966$ , o ganho proporcional;

$I = -0.239325$ , o ganho integral e

$\Delta t = 0.1$ , o tempo de amostragem [h].

Os resultados da simulação com esse controlador podem ser conferidos na Seção 5.6.



## 4.7 Adaptações para Variação no Setpoint

Até então, foi tratado o problema de controle de nível de um tanque com setpoint fixo, ou seja, um controle de caráter regulatório. Contudo, como o controlador RL está em constante aprendizado por meio de ações exploratórias, um controlador que continua explorando após sua implementação na planta consegue atualizar a sua *policy* de forma a seguir um novo setpoint.

Também é possível implementar um controle de rastreamento de setpoint de maneira explícita, ao incluir o setpoint no estado. Desse modo, o agente passa então a receber a informação de qual é o setpoint atual e o estado passa a ser bidimensional:  $s_t = [h_t, h_{sp_t}]$ .

Algumas adaptações precisam ser feitas sobre a metodologia apresentada para incluir essa bidimensionalidade, sobretudo nas VFAs. Conforme apresentado na Seção 4.3, são necessárias 10 RBFs para representar o nível  $h \in [0, 1]$ . Como o setpoint  $h_{sp} \in [0, 1]$  representa a mesma grandeza, também pode ser utilizadas 10 RBFs para representá-lo, por consistência.

Para garantir uma simulação mais precisa ao longo do espaço bidimensional de estados, o número de *features* foi aumentado para 20 tanto para o nível quanto para o setpoint. Dessa forma, com 20 RBFs em cada dimensão, é gerada uma rede bidimensional de  $20 \times 20 = 400$  RBFs.

De forma a manter a consistência de todas as equações apresentadas anteriormente, o vetor de *features*  $\phi$  passa então a ser  $400 \times 1$  e o mesmo ocorre para os vetores de pesos das *features*  $\theta_V$  e  $\theta_\pi$ . Todo o resto da arquitetura do controle de nível via RL se mantém.

Como a ação determinada pelo agente  $\Delta X_V$  só consegue agir de forma a mudar o nível  $h$ , apenas a ação exploratória não será o suficiente para garantir que o aprendizado cubra todo o espaço de estados. Dessa forma, durante o aprendizado é preciso variar o setpoint  $h_{sp}$  para fazer com que o agente visite todo o espaço de estados e implemente uma *policy*  $\pi$  abrangente.

Com isso, o controlador deve ser capaz de fazer o nível seguir um determinado setpoint  $h_{sp}$  mesmo com ele variando ao longo da simulação.

# Capítulo 5

## Resultados e Discussões

Nesse capítulo, são apresentados os resultados de diversas simulações feitas para o controle de nível de um tanque via RL. Conforme apresentado nos Capítulos 3 e 4, existem diversos parâmetros a serem sintonizados e diferentes metodologias de aprendizado dentro da abordagem *Actor-Critic*.

As simulações são feitas com os parâmetros definidos no Capítulo 4. Além disso, o setpoint foi definido como  $h_{sp} = 0.7\text{m}$ . Após o término de 300 episódios de aprendizado, é feita uma simulação final para avaliar o desempenho do controlador.

Nessa simulação final, o agente não explora e o valor da vazão de entrada inicial  $F_i$  é alterado de  $F_i = 100\text{m}^3/\text{h}$  para  $F = 90\text{m}^3/\text{h}$  na iteração de número 30, ou seja, instante de tempo  $t = 3\text{h}$ . Assim é testada a robustez do controlador sintetizado em relação a perturbações não medidas.

### 5.1 Parâmetros iniciais de Exploração

Conforme apresentado na Seção 4.4, é preciso definir o parâmetro  $N_0$  para a exploração  $\epsilon$ -greedy. Já para a exploração gaussiana, também é preciso do parâmetro  $\sigma_0$ . Esses dois parâmetros dosam o quanto o agente explora o espaço de estados, ao deixar de seguir a ação definida pela policy  $\pi$  com o objetivo de obter mais informações.

Para a exploração  $\epsilon$ -greedy com um espaço de estados contínuo, é preciso considerá-lo dividido em regiões, cada uma regida por uma RBF. Com 10 RBFs e, portanto, 10 regiões, os contadores de visitas das regiões são incrementados rapidamente e por isso é preciso definir um número alto para  $N_0$ . Após alguns testes, foi verificado que  $N_0 = 50$  define uma exploração satisfatória do espaço de estados. Dessa forma, na visita, por exemplo, de número 50 a uma região, a chance de explorar  $\epsilon_t = 50/(50 + 50) = 0.5$ , ou seja 50% de chance de explorar e 50% de chance de aproveitar. Assim o espaço é bastante explorado até o momento em que as ações que maximizam o retorno esperado passam a ser priorizadas.

É preciso então definir  $\sigma_0$ , que é um fator que multiplica  $\epsilon_t$  para obter-se um desvio padrão segundo a Equação 4.10. No escopo desse projeto, em que a ação é um incremento na abertura da válvula ( $\Delta X_V$ ) em cada instante de tempo, foi definido  $\sigma_0 = 0.05$ , o que garantiu um satisfatório alcance de exploração. Valores maiores de  $\sigma_0$  fazem com que a ação atinja valores extremos, fazendo o tanque esvaziar ( $h \leq 0$ ) ou transbordar ( $h \geq 1$ ) quase que instantaneamente, enquanto que valores menores resultam em pouca exploração e o agente passa a ter pouca informação do ambiente.

## 5.2 Parâmetros da Avaliação de *Policy* e Recompensas

A primeira sintonia apresentada é a dos parâmetros  $\gamma$ , o fator de desconto e  $\lambda$ , o fator de decaimento do traço de elegibilidade. O parâmetro  $\gamma$  é importante em tarefas em que uma ação pode ter pouca influência na recompensa imediata, mas ser determinante para recompensas futuras. Enquanto que  $\gamma$  ajusta diretamente o valor dessas recompensas, o fator  $\lambda$  é importante para diluir a recompensa obtida ao longo dos estados anteriores, pelo traço de elegibilidade. Um valor clássico da literatura para  $\lambda$  em uma tarefa de RL genérica é de  $\lambda = 0.9$  [8].

A escolha desses parâmetros depende então diretamente de como a recompensa é definida. Para a recompensa  $\mathcal{R}_1$ , que é relativa a aproximação absoluta ao setpoint (Equação 4.6), é fácil ver que a recompensa imediata só depende do estado atual e da ação atual. Qualquer aproximação do nível em relação ao setpoint será recompensada. Sendo assim, não há motivo para diluir a recompensa ao longo dos estados anteriores, ou descontar recompensas futuras. Vale ressaltar que para  $\gamma = 0$ , o valor de  $\lambda$  é indiferente, já que ele sempre aparece multiplicado por  $\gamma$  no traço de elegibilidade explicitado na Equação 3.14 e, portanto, quando  $\gamma = 0$ ,  $\lambda$  também é definido como  $\lambda = 0$  por consistência.

Para comparar o desempenho de cada controlador, será considerado o quanto o valor final do nível se aproximou do setpoint e também a velocidade em atingi-lo, medida por meio do tempo de subida  $T_r$ , definido como o tempo que a resposta demora para ir de 10 a 90% do seu valor final. As figuras 5.1a, 5.1b, 5.1c e 5.1d mostram o comportamento do sistema para diferentes configurações de  $\gamma$  e  $\lambda$  com o sinal de recompensa  $\mathcal{R}_1$ , após 300 episódios de aprendizado.

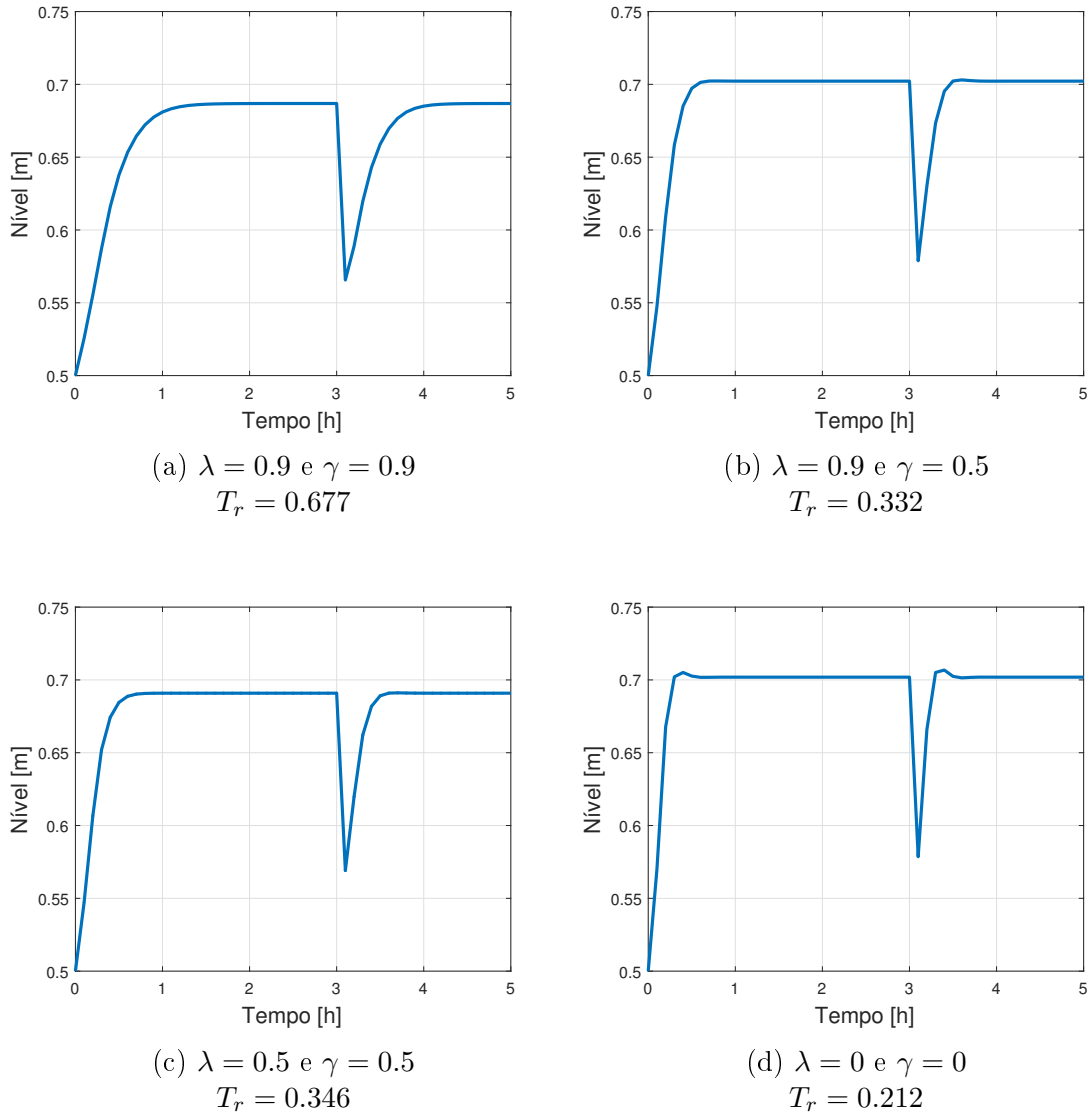


Figura 5.1: Resultado de simulações com  $\mathcal{R}_1$  e diversos valores de  $\lambda$  e  $\gamma$ .

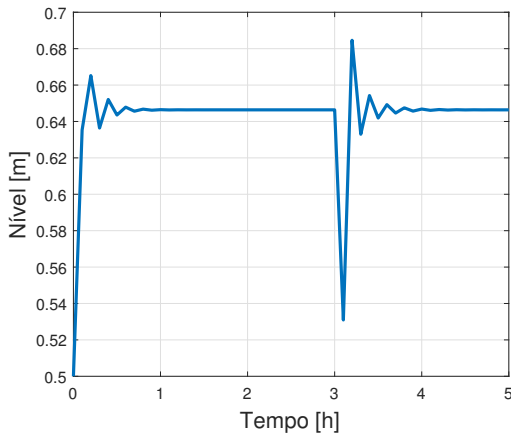
Todas as configurações são capazes de controlar a planta, mesmo após a adição do distúrbio não medido em  $t = 3h$ . Contudo, nas figuras 5.1a e 5.1c, há um offset considerável. O nível não atinge bem o setpoint, estabilizando em  $h = 0.69m$ , além de demorar bastante para atingir essa região, como no caso da Figura 5.1a, que apresentou o maior tempo de subida  $T_r = 0.677h$ .

Os melhores resultados podem ser conferidos nas figuras 5.1b e 5.1d, já que em ambas o nível estabilizou em  $h = 0.702m$ , apresentando um offset mínimo, com erro de estado estacionário  $e_{ss} = |0.7 - 0.702| = 0.002m$ . A única simulação que apresentou overshoot foi aquela com  $\lambda = \gamma = 0$  e mesmo assim o valor desse overshoot foi muito baixo, com o nível atingindo um valor máximo de  $h_{max} = 0.705m$ . Devido a esse controle mais agressivo, essa foi a sintonia que obteve a resposta mais rápida, com  $T_r = 0.212h$ . Dessa forma, a hipótese inicial se confirmou, com o melhor controle sendo obtido zerando-se os parâmetros  $\gamma$  e  $\lambda$ , ilustrado na Figura 5.1d.

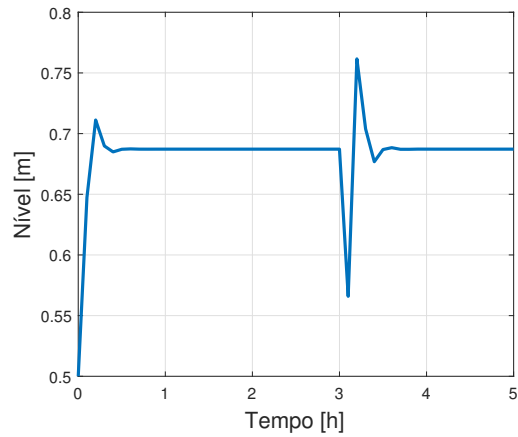
Já para o sinal de recompensa definido na Equação 4.7 (denominado  $\mathcal{R}_2$ ), os parâmetros  $\gamma$  e  $\lambda$  são essenciais. Como a recompensa só passa a ser não-nula dentro de uma faixa de valores próximo ao setpoint, caso essa recompensa não se propague, uma ação que leva o nível na direção do setpoint mas não entra nessa região, nunca vai ser favorecida, por levar a uma recompensa imediata nula.

Porém, com o uso do traço de elegibilidade, que decai conforme  $\gamma\lambda$  (Equação 3.14), quando o agente recebe uma recompensa, por entrar na região próxima do setpoint, por exemplo, essa recompensa é propagada para a *value function* dos estados mais recentes. O parâmetro  $\lambda$  pesa apenas nessa propagação, enquanto o  $\gamma$  influencia também o peso das recompensas futuras no sinal de retorno.

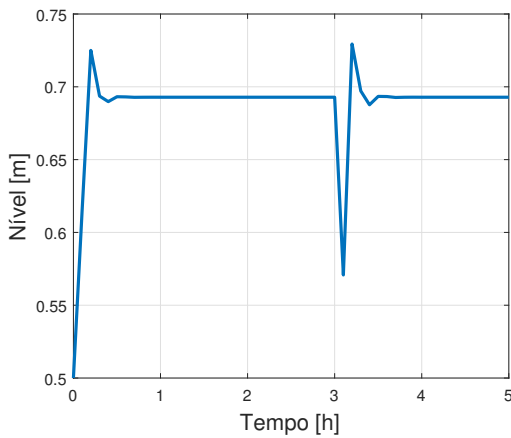
Sendo assim, não deve ser possível sintonizar um controlador de nível utilizando  $\mathcal{R}_2$  com esses parâmetros nulos. O desempenho dos controladores foi novamente comparado para diferentes valores de  $\gamma$  e  $\lambda$ , após um período de aprendizado de 300 episódios. Os resultados podem ser conferidos nas figuras 5.2a, 5.2b, 5.2c e 5.2d.



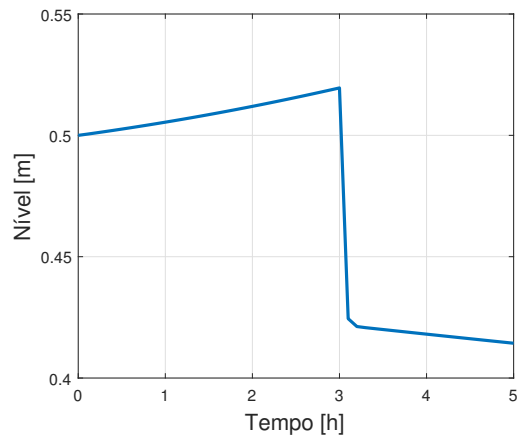
(a)  $\lambda = 0.9$  e  $\gamma = 0.9$   
 $T_r = 0.087$



(b)  $\lambda = 0.9$  e  $\gamma = 0.5$   
 $T_r = 0.121$



(c)  $\lambda = 0.5$  e  $\gamma = 0.5$   
 $T_r = 0.136$



(d)  $\lambda = 0$  e  $\gamma = 0$   
 $T_r$  não definido

Figura 5.2: Resultado de simulações com  $\mathcal{R}_2$  e diversos valores de  $\lambda$  e  $\gamma$ .

Os resultados são bem diferentes daqueles apresentados anteriormente na Figura 5.1, pois os controladores com  $\mathcal{R}_2$  da figura 5.2 são muito mais sensíveis a alterações nos parâmetros  $\lambda$  e  $\gamma$ . Conforme previsto, não foi possível controlar o nível do tanque com os parâmetros nulos, conforme ilustra a Figura 5.2d. No início o nível até parece estar se direcionando muito lentamente para o setpoint, mas após a adição do distúrbio ele passa a assumir a direção oposta do setpoint.

O controle sintonizado apresentou um caráter mais agressivo, evidenciado pelos tempos de subida que foram todos reduzidos e pelos *overshoots* que aumentaram em todas as 3 figuras 5.2a, 5.2b e 5.2c. Para  $\lambda = 0.9$  e  $\gamma = 0.9$  o

Porém um valor alto de  $\lambda$  e  $\gamma$  resultou em um comportamento altamente oscilatório e estabilização longe do setpoint em  $h = 0.65\text{m}$  conforme a figura 5.2a.

Já a sintonização com  $\lambda = 0.9$  e  $\gamma = 0.5$  da figura 5.2b, apresentou um tempo de subida menor do que aquela com  $\lambda = 0.5$  e  $\gamma = 0.5$  da figura 5.2c ( $0.121\text{h} < 0.136\text{h}$ ), porém com um overshoot maior, especialmente após a adição do distúrbio, com o nível atingindo  $h_{max} = 0.761\text{m}$  para  $\lambda = 0.9$  e  $\gamma = 0.5$ , em comparação com um máximo de  $h_{max} = 0.729$  para  $\lambda = 0.5$  e  $\gamma = 0.5$ . Tem-se, então, que o controlador da Figura 5.2c é o mais adequado dentre os demais com a recompensa  $\mathcal{R}_2$ . Este também apresentou o menor erro de estado estacionário estado estacionário  $e_{ss} = |0.7 - 0.693| = 0.007\text{m}$

Comparando-se então os melhores desempenhos de cada recompensa, ilustrados nas figuras 5.1d e 5.2c, tem-se que o melhor controlador é obtido utilizando a recompensa  $\mathcal{R}_1$ . Este apresentou menor overshoot  $0.705 < 0.729$  e um menor erro de estado estacionário  $0.002 < 0.007$ , sendo portanto o controlador de melhor desempenho. Dessa forma, para as simulações seguintes, será considerada então a recompensa  $\mathcal{R}_1$  e os parâmetros  $\lambda = 0$  e  $\gamma = 0$ .

### 5.3 Métodos de Melhoria de *Policy*

Conforme apresentado no Capítulo 3, o aprendizado *actor-critic* ocorre por meio de iterações na *policy*. A cada intervalo de tempo, o *critic* estima o valor da *policy* atual e envia um sinal para o *actor*, que utiliza um método de melhoria de *policy*.

Na Seção 3.10, foram propostas duas metodologias de melhoria de *policy*. A primeira vem da literatura clássica de RL [7], explicitada na Equação 3.19, e é denominado de controlador AC clássico. Já a metodologia CACLA [21], só realiza a atualização da *policy* caso o valor do erro diferencial temporal seja positivo. Essas duas abordagens são comparadas utilizando o controlador com os parâmetros sintonizados segundo os resultados anteriores, expostos na Seção 5.2. Os resultados podem ser conferidos nas figuras 5.3a e 5.3b.

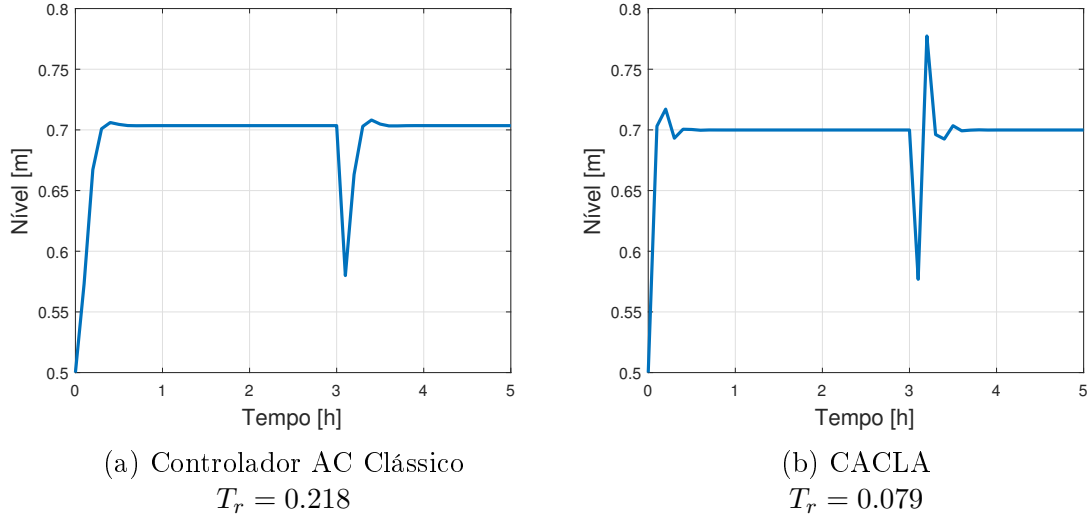


Figura 5.3: Resultado de simulações com diferentes métodos de melhoria de *policy*.

A abordagem clássica da Figura 5.3a reflete o mesmo controlador da Seção 5.2. Alterações mínimas aparecem em relação ao resultado anterior na Figura 5.1d, como o tempo de subida  $T_r = 0.218$ h ao invés de  $T_r = 0.212$ h. Essa pequena diferença se dá porque o aprendizado conta com uma exploração estocástica, o que produz pequenas variações. É importante notar que o eixo do gráfico 5.3a também está diferente, de forma a facilitar a comparação com a Figura 5.3b.

Já o novo resultado com a metodologia CACLA apresentou o menor tempo de subida dentre todos resultados apresentados,  $T_r = 0.079$ h. Além disso, uma característica muito importante desse controlador foi que o erro de estacionário é nulo, com o nível estabilizando em  $h = h_{sp} = 0.7$ m.

Porém, apesar dessas vantagens, após a introdução da perturbação não medida em  $t = 3$ h, a resposta apresentou um overshoot bastante elevado, atingindo  $h_{max} = 0.775$ m. Esse tipo de variação poderia levar o tanque a transbordar, caso o setpoint fosse próximo da altura máxima do tanque.

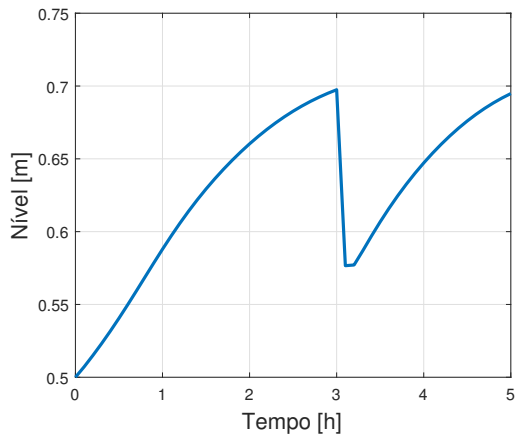
Para um sistema cujo comportamento transitório não seja muito importante, o método CACLA mostra-se mais adequado. Porém, a diferença de erros de estado estacionário entre o CACLA e o controlador AC clássico é tão baixa (0 e 0.002m, respectivamente) que a diferença entre os valores máximos de overshoot (0.775m e 0.708m, respectivamente) passa a ser um fator determinante para avaliar o controlador AC clássico da Figura 5.3a como o controle de melhor performance.

## 5.4 Ganho do Aprendizado

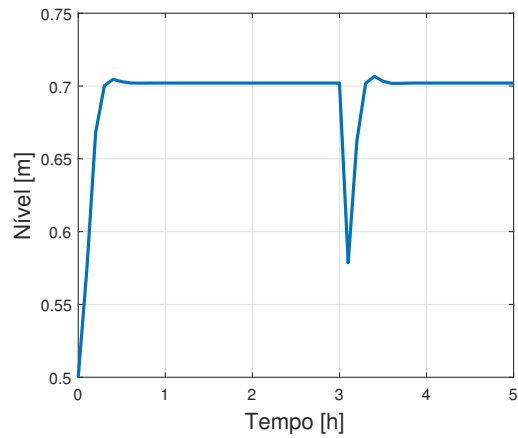
Conforme apresentado nas seções 3.9 e 3.10, a atualização da *value function* e da *policy* são ajustadas pelos parâmetros  $\alpha_V$  e  $\alpha_\pi$ , respectivamente. Eles ditam o

tamanho do passo em direção ao novo valor calculado da *policy* e da *value function*.

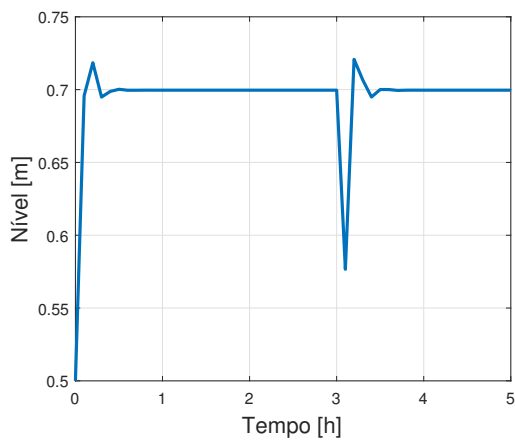
Até então, nas simulações apresentadas nesse capítulo, foi utilizado um valor clássico da literatura, com  $\alpha_V = \alpha_\pi = 0.1$  [8]. Contudo, na Figura 5.4 podem ser vistas variações nos valores de  $\alpha_V$  e de  $\alpha_\pi$  e o resultado na sintonia do controlador.



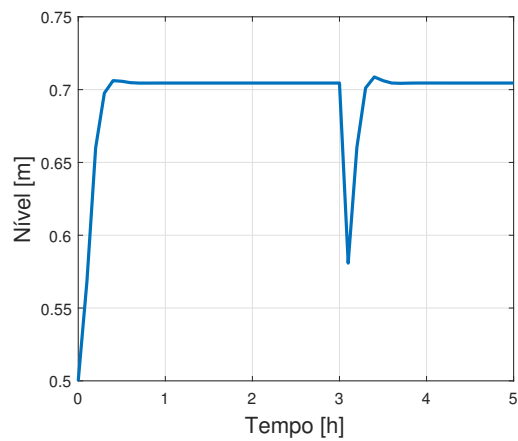
(a)  $\alpha_V = 0.01$  e  $\alpha_\pi = 0.01$



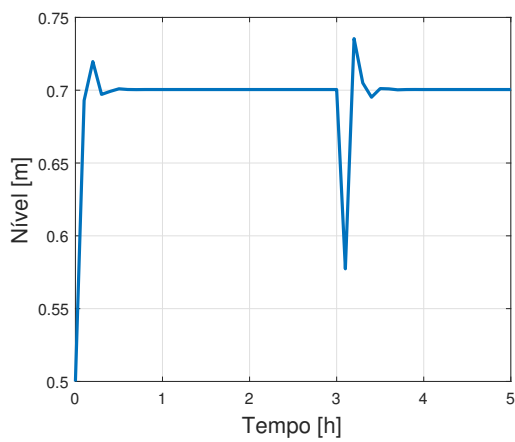
(b)  $\alpha_V = 0.1$  e  $\alpha_\pi = 0.1$



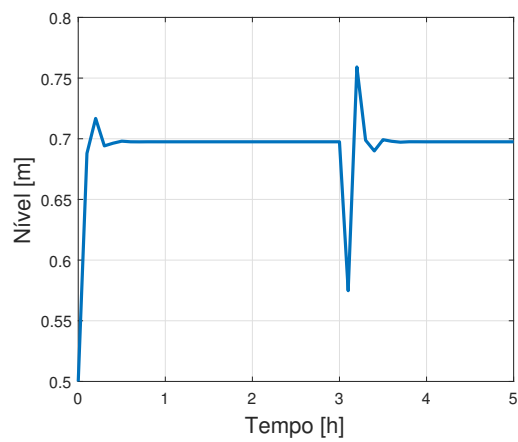
(c)  $\alpha_V = 0.1$  e  $\alpha_\pi = 1$



(d)  $\alpha_V = 1$  e  $\alpha_\pi = 0.1$



(e)  $\alpha_V = 1$  e  $\alpha_\pi = 1$



(f)  $\alpha_V = 0.5$  e  $\alpha_\pi = 0.5$

Figura 5.4: Resultado de simulações para diversos valores de  $\alpha_V$  e  $\alpha_\pi$ .



As figuras 5.4a e 5.4e ilustram bem o efeito da variação de  $\alpha_V$  e  $\alpha_\pi$ . Um valor muito baixo, como 0.01, resulta em um controlador de resposta muito lenta, Já ao aumentar os valores desses parâmetros, tem-se uma resposta mais rápida e agressiva do controlador, o que resulta em um overshoot pronunciado.

Já as figuras 5.4c, 5.4d e 5.4e mostram que o sistema é menos sensível a variações em  $\alpha_V$  do que a variações em  $\alpha_\pi$ . Além disso, a sintonização com os dois parâmetros em 0.5 foi aquela que apresentou o maior overshoot, com  $h_{max} = 0.759m$ .

Os melhores controladores foram aqueles ilustrados nas figuras 5.4b e 5.4d. As duas respostas são muito semelhantes, com uma pequena diferença no erro em estado estacionário. Para  $\alpha_V = \alpha_\pi = 0.1$ , tem-se que  $e_{ss} = 0.02m$ , enquanto que para  $\alpha_V = 1$  e  $\alpha_\pi = 0.1$ , o erro aumenta para  $e_{ss} = 0.05m$ .

Dessa forma, o aprendizado com os valores clássicos da literatura  $\alpha_V = \alpha_\pi = 0.1$  demonstrou o melhor desempenho para o controlador de nível.

## 5.5 Comparação dos Tipos de Exploração

Na Seção 3.6, foram apresentadas duas técnicas de exploração. A exploração gaussiana adiciona uma perturbação na forma de ruído gaussiano à ação determinada pela *policy* e faz com que o agente explore diferentes regiões do espaço de estados ao desviá-lo da *policy*. O desvio padrão desse ruído adicionado diminui ao longo do aprendizado para que no final, após ter explorado o espaço de estados, o agente possa seguir a *policy* calculada.

Já a exploração  $\epsilon$ -greedy envolve substituir a ação determinada pela *policy* por uma ação uniformemente aleatória e com probabilidade  $\epsilon$ . Da mesma forma, essa probabilidade  $\epsilon$  decresce ao longo do aprendizado, pois o agente mais experiente tem mais condições de aproveitar as informações captadas (leia-se, seguir a *policy* calculada) sem a necessidade de continuar explorando tanto o espaço de estados.

Nas simulações apresentadas até então, foi utilizada a exploração gaussiana, usada como base comparativa à exploração  $\epsilon$ -greedy nas figuras 5.5a e 5.5b.

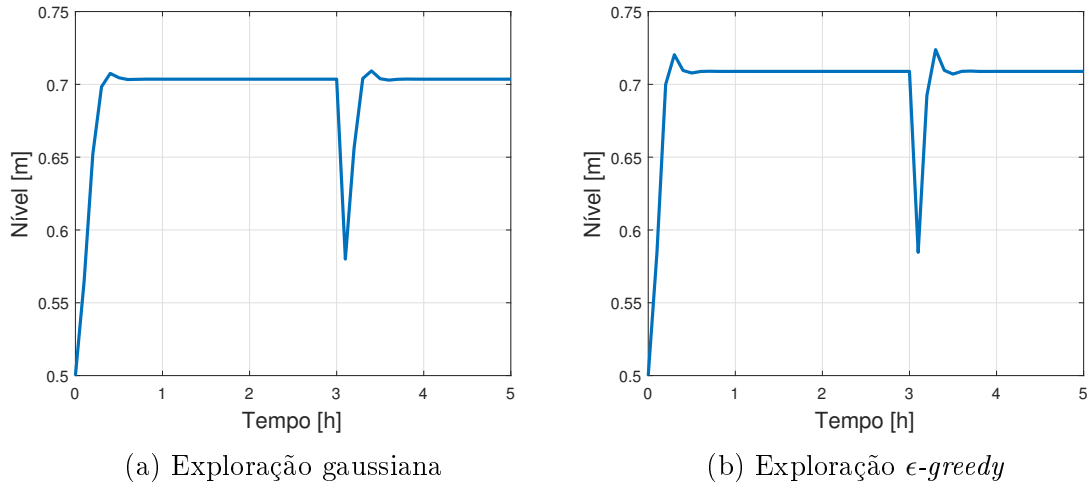


Figura 5.5: Resultado de simulações com diferentes métodos de exploração.

A exploração  $\epsilon$ -greedy da Figura 5.5b apresentou maior overshoot ( $h_{max} = 0.724\text{m}$ ) e erro de estado estacionário ( $e_{ss} = 0.009\text{m}$ ) que a exploração gaussiana da Figura 5.5a ( $h_{max} = 0.709\text{m}$  e  $e_{ss} = 0.004\text{m}$ ).

Sendo assim, a exploração gaussiana provou ser a mais adequada para o aprendizado do controlador de nível via RL.

## 5.6 Comparação entre RL e Controlador Clássico

Com todos os parâmetros devidamente sintonizados, resta agora avaliar o desempenho do controlador via RL com um controlador clássico. A sintonia do controlador baseado na teoria clássica de controle foi descrita na Seção 4.6, e resultou em um PI discreto. Essa comparação tem o objetivo de validar o controle via RL proposto nesse projeto.

A mesma simulação presente nos resultados anteriores foi aplicada ao controlador PI e ao controle via RL, com todos os parâmetros sintonizados segundo as seções anteriores. O resultado pode ser conferido na Figura 5.6, em que a linha contínua representa o controle via RL e a linha tracejada o controlador PI.

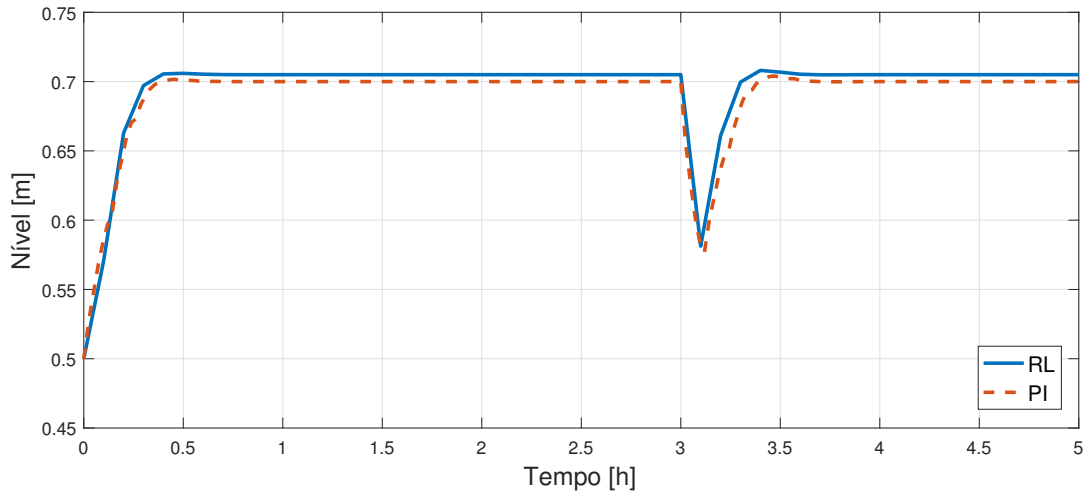


Figura 5.6: Comparação de desempenho entre o controle via RL controlador PI discreto.

Os dois controladores apresentaram uma performance muito similar. O tempo de subida dos dois é próximo, com o controle via RL um pouco mais rápido ( $t_r = 0.235\text{h}$ ) do que o controlador PI ( $t_r = 0.259\text{h}$ ), além de ambos reagiram da mesma forma ao distúrbio introduzido em  $t = 3\text{h}$ . Contudo, o controlador PI apresentou um erro de estado estacionário nulo, enquanto que o RL apresentou  $e_{ss} = 0.04\text{m}$ .

O objetivo aqui não é definir qual é o melhor controlador, mas sim validar o desenvolvimento do controle via RL. Por ser uma planta simples, o controlador PID bem sintonizado apresenta uma performance alta, sendo nesse caso equivalente ao controle via RL apresentado. Porém, para sintonizá-lo foi preciso fazer uma identificação da planta, enquanto que o controle via RL foi sintonizado apenas utilizando dados do processo. Isso pode ser uma grande desvantagem em processos mais complexos, nos quais o levantamento e atualização de modelos sejam tarefas custosas.

Além disso, é importante destacar a relevância do resultado obtido com o controlador RL ao utilizar  $\Delta t = 0.1\text{h}$ . Esse valor é 3.5 vezes maior do que a constante de tempo do sistema ilustrada na Equação 4.12 de primeira ordem ( $0.0287\text{h}$ ). Contudo, o controlador resultante apresentou ótimo desempenho e utilizar uma constante de tempo relativamente grande permitiu realizar o aprendizado com baixo custo computacional.

## 5.7 Demonstração da característica adaptativa do controlador

Outra característica interessante do controlador baseado em RL é que ele pode continuar a aprender uma vez implementado no sistema real, e assim se adaptar a

distúrbios não medidos e/ou mudanças no processos devido ao seu desgaste. Para demonstrar este comportamento adaptativo, foi considerado o caso extremo em que uma há uma mudança de setpoint enquanto o aprendizado só foi realizado para um setpoint. Na prática, e como será feito na próxima seção, o setpoint seria incluído ao estado para que a mudança de setpoint possa ser levada em conta pelo controlador. Mudanças de vazão de alimentação não afetam o controle de forma suficiente para demonstrar a capacidade adaptativa do controlador.

Até então, todos os resultados apresentados foram de simulações feitas sem ações exploratórias. Dessa forma, é possível isolar a *policy* obtida, de forma a fazer comparação e avaliar o desempenho do controlador obtido com diversas variações de aprendizado.

Contudo, a adição da ação exploratória confere ao controlador via RL um caráter adaptativo. Dessa forma, o controlador continua com o seu aprendizado e atualização da sua *policy* conforme ocorrem mudanças no processo. Assim, é possível implementar um controle de rastreamento de setpoint variável ao aplicar o controlador RL sintetizado nas seções anteriores e manter as ações exploratórias.

Foi feita uma simulação de duração estendida, com 200h, em que o setpoint começa em  $h_{sp} = 0.7$ , é alterado primeiro para  $h_{sp} = 0.5$  no instante  $t = 20$ h e em seguida, para  $h_{sp} = 0.8$  no instante  $t = 100$ h. A exploração gaussiana é mantida constante, com  $\epsilon_t = 0.05$ . O resultado pode ser conferido na figura 5.7.

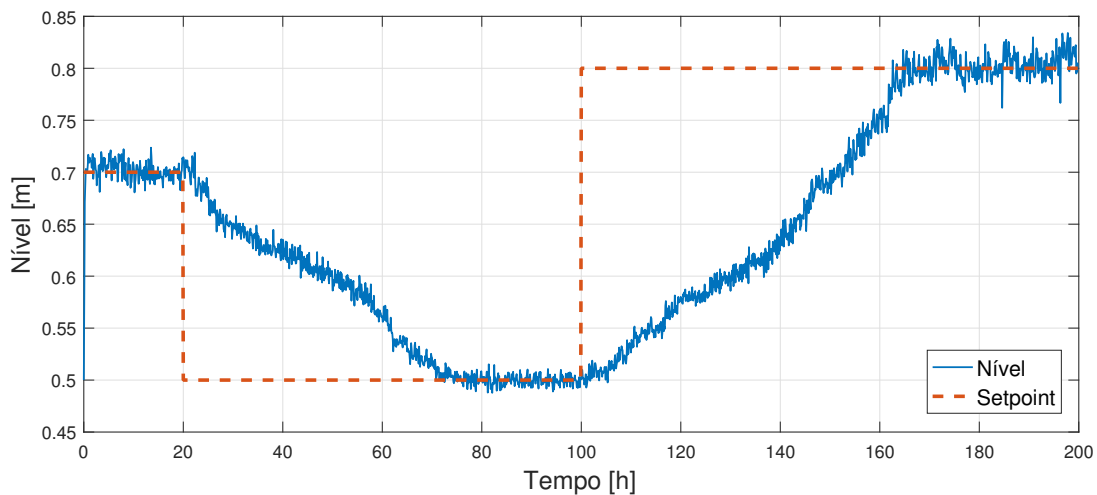


Figura 5.7: Resultado da simulação do controle com rastreamento de setpoint por meio da exploração.

Com a ação exploratória, o nível apresenta pequenas variações durante todo o tempo. Quando o setpoint é alterado em  $t = 20$ h, as ações que diminuem o nível passam a ser mais recompensadas e o controlador vai atualizando a sua *policy* de forma a manter o nível próximo do setpoint  $h_{sp} = 0.5$ m, atingindo-o pela primeira

vez no instante de tempo  $t = 72\text{h}$ . Uma resposta similar ocorre quando o setpoint é alterado para  $h_{sp} = 0.8\text{m}$  no instante de tempo  $t = 100\text{h}$ , com o nível atingindo esse novo setpoint apenas em  $t = 163\text{h}$ .

Sendo assim, pode-se concluir que apesar da resposta à mudança do setpoint ser lenta, o controlador consegue se adaptar a situações diferentes daquelas exploradas durante o seu aprendizado. Além disso, essa abordagem é mais aplicável à variações que ocorrem lentamente, como por exemplo o decaimento de parâmetros do processo ao longo do tempo.

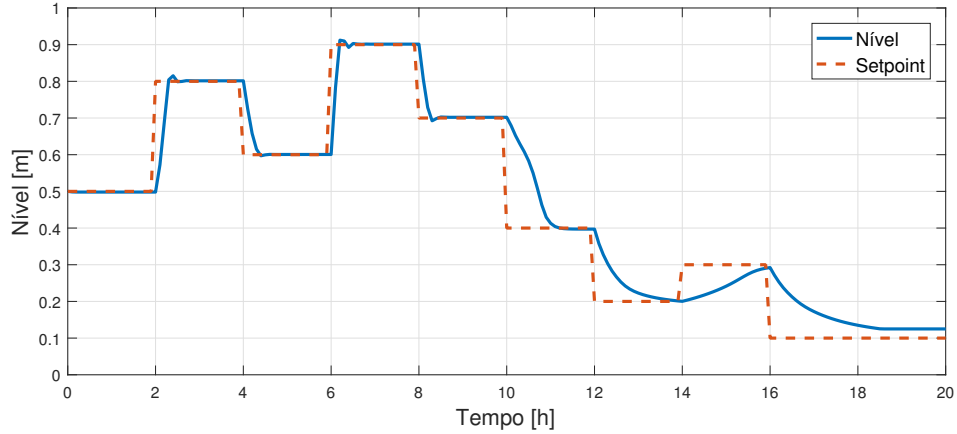
## 5.8 Rastreamento do Setpoint pelo Estado

De forma a acelerar esse rastreamento de setpoint, é possível incluir o setpoint no estado, de forma que a *policy* passe a abranger diferentes situações não só de nível mas também de setpoint. Para isso, é preciso fazer alguns ajustes na modelagem do problema, conforme apresentado na Seção 4.7. Ao incluir o setpoint no espaço de estados, é preciso aumentar a dimensão das *features*  $\phi$  e também dos vetores de pesos das *features*  $\phi_V$  e  $\phi_\pi$ .

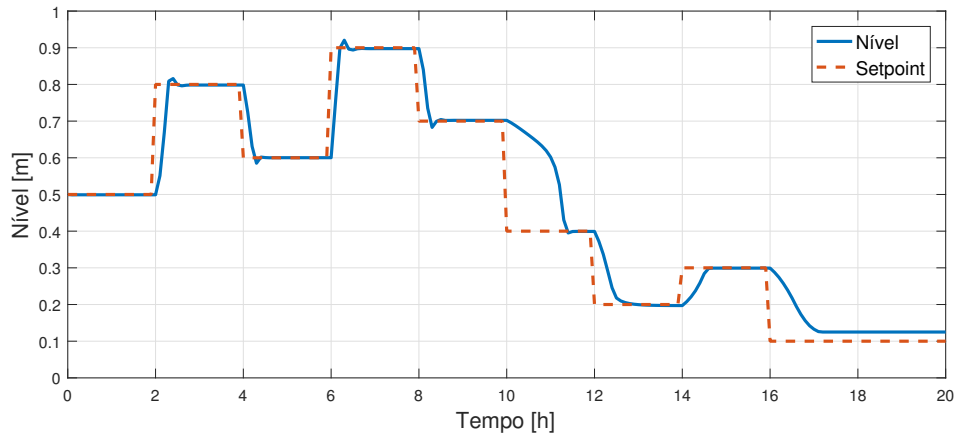
Como o agente só consegue explorar o espaço de espaços na dimensão do nível, já que a sua única ação de abrir e fechar a válvula não altera em nada o setpoint, é preciso forçar um setpoint variável durante o aprendizado para garantir que o agente explore todo o espaço de estados. Dessa forma, o número de episódios de aprendizado foi aumentado para 800. Para garantir que o agente continue explorando o espaço de estados durante o aprendizado, a constante de exploração  $N_0$  foi aumentada para 100.

Para garantir uma simulação mais precisa, o número de *features* foi aumentado para 20, tanto para o nível quanto para o setpoint, o que totaliza  $20 \times 20 = 400$  *features*.

Foram feitas simulações para os dois tipos de recompensas  $\mathcal{R}_1$  e  $\mathcal{R}_2$ , com os melhores valores de  $\gamma$  e  $\lambda$  para cada tipo de recompensa, conforme apresentado na Seção 5.2. Durante a simulação, o setpoint foi alterado para diferentes valores e o resultado pode ser visto nas figuras 5.8b e 5.8a, a seguir. A linha contínua indica o nível enquanto a linha tracejada indica o valor do setpoint do nível.



(a) Recompensa  $\mathcal{R}_1$ ,  $\gamma = 0$  e  $\lambda = 0$



(b) Recompensa  $\mathcal{R}_2$ ,  $\gamma = 0.5$  e  $\lambda = 0.5$

Figura 5.8: Resultado de simulações do controle com rastreamento de setpoint por meio do estado, para diferentes sinais de recompensa.

Os resultados foram similares e satisfatórios, com o nível possibilitado de seguir uma trajetória de referência variável. É importante destacar que o nível não consegue chegar em 0.1m, pois nesse ponto mesmo com a válvula aberta ao máximo ( $X_V = 1$ ), a vazão de entrada  $F_i = 100\text{m}^3/\text{h}$  mantém o nível do tanque em  $h = 0.125\text{m}$ .

O controlador com  $\mathcal{R}_1$  apresentou melhor desempenho na transição de  $h_{sp} = 0.7\text{m}$  para  $h_{sp} = 0.4\text{m}$ . Porém, para os setpoints de baixo valor 0.3m, 0.2m e 0.1m, o controlador com  $\mathcal{R}_2$  teve uma performance significativamente melhor e pode ser considerado o mais adequado dos dois.

# Capítulo 6

## Conclusão e Trabalhos Futuros

Esse trabalho teve como objetivo o desenvolvimento e aplicação de um controlador de nível que utiliza algoritmos de *reinforcement learning*. Esses algoritmos fazem parte da metodologia *actor-critic*, estado da arte do RL atualmente.

Foram exploradas diversas variações durante o aprendizado do controlador, de forma a obter a melhor sintonia do controle. Os resultados apresentados demonstraram um bom funcionamento do controle de nível, com performance similar a um controlador clássico do tipo PI.

A relevância dos resultados desse projeto se deve ao fato do controlador ter sido sintonizado baseado em dados do processo. Uma modelagem da planta foi utilizada somente para realizar as simulações e obter os dados do processo utilizados no aprendizado do controlador.

Dessa forma, essa abordagem pode ser estendida para processos mais complexos. Em muitos casos, o custo de levantamento e atualização de modelos do processo é muito alto, o que torna o controle RL uma alternativa atraente. Além disso, como o controlador continua aprendendo e atualizando a sua *policy*, ele se adapta conforme parâmetros do processo se alteram ao longo do tempo, garantindo robustez.

Para eliminar necessidade de simulações e portanto, de uma modelagem do processo, o controlador deve interagir diretamente com a planta. Isso traz sérias implicações de segurança, já que o controle pode ser instável durante o início do aprendizado. Para contornar esse problema, é possível realizar um pré-aprendizado a partir de dados históricos da planta, o que resultaria em um aprendizado do tipo *off policy*. Outras opções seriam limitar as ações do controlador na planta real, ou basear-se em um modelo simples da planta.

Outra sugestão de trabalho futuro é o uso da metodologia *actor-critic* para processos mais complexos, com múltiplas variáveis manipuladas e controladas, como um reator de Van de Vusse. Além disso, com o aprendizado *off policy*, o controle RL pode partir de um controlador previamente sintonizado, ao invés de iniciar o aprendizado com uma *policy* aleatória por não possuir nenhuma informação de início.

# Referências Bibliográficas

- [1] BRETTEL, M., FRIEDERICHSEN, N., KELLER, M., et al. “How virtualization, decentralization and network building change the manufacturing landscape: An Industry 4.0 Perspective”, *International Journal of Mechanical, Industrial Science and Engineering*, v. 8, n. 1, pp. 37–44, 2014.
- [2] TEDRAKE, R., ZHANG, T. W., SEUNG, H. S. “Stochastic policy gradient reinforcement learning on a simple 3D biped”. In: *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, v. 3, pp. 2849–2854. IEEE, 2004.
- [3] TAN, Z., QUEK, C., CHENG, P. Y. “Stock trading with cycles: A financial application of ANFIS and reinforcement learning”, *Expert Systems with Applications*, v. 38, n. 5, pp. 4741–4755, 2011.
- [4] MORARI, M., LEE, J. H. “Model predictive control: past, present and future”, *Computers & Chemical Engineering*, v. 23, n. 4-5, pp. 667–682, 1999.
- [5] YU-GENG, X., DE-WEI, L., SHU, L. “Model predictive control—status and challenges”, *Acta Automatica Sinica*, v. 39, n. 3, pp. 222–236, 2013.
- [6] QIN, S. J., BADGWELL, T. A. “An overview of nonlinear model predictive control applications”. In: *Nonlinear model predictive control*, Springer, pp. 369–392, 2000.
- [7] BUSONIU, L., BABUSKA, R., DE SCHUTTER, B., et al. *Reinforcement learning and dynamic programming using function approximators*, v. 39. CRC press, 2010.
- [8] SUTTON, R. S., BARTO, A. G. *Reinforcement learning: An introduction*, v. 1. MIT press Cambridge, 1998.
- [9] BRADTKE, S. J., YDSTIE, B. E., BARTO, A. G. “Adaptive linear quadratic control using policy iteration”. In: *American Control Conference, 1994*, v. 3, pp. 3475–3479. IEEE, 1994.



- [10] KEEL, L. H., BHATTACHARYYA, S. P. “Controller synthesis free of analytical models: Three term controllers”, *IEEE Transactions on Automatic Control*, v. 53, n. 6, pp. 1353–1369, 2008.
- [11] KILLINGSWORTH, N. J., KRSTIC, M. “PID tuning using extremum seeking: online, model-free performance optimization”, *IEEE control systems*, v. 26, n. 1, pp. 70–79, 2006.
- [12] STENMAN, A. “Model-free predictive control”. In: *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, v. 4, pp. 3712–3717. IEEE, 1999.
- [13] HOSKINS, J., HIMMELBLAU, D. “Process control via artificial neural networks and reinforcement learning”, *Computers & chemical engineering*, v. 16, n. 4, pp. 241–251, 1992.
- [14] ANDERSON, C. W., HITTLE, D. C., KATZ, A. D., et al. “Synthesis of reinforcement learning, neural networks and PI control applied to a simulated heating coil”, *Artificial Intelligence in Engineering*, v. 11, n. 4, pp. 421–429, 1997.
- [15] MARTINEZ, E. “Batch process modeling for optimization using reinforcement learning”, *Computers & Chemical Engineering*, v. 24, n. 2-7, pp. 1187–1193, 2000.
- [16] SYAFIIE, S., TADEO, F., MARTINEZ, E. “Model-free learning control of chemical processes”. In: *Reinforcement Learning*, InTech, 2008.
- [17] SHAH, H., GOPAL, M. “Model-Free Predictive Control of Nonlinear Processes Based on Reinforcement Learning”, *IFAC-PapersOnLine*, v. 49, n. 1, pp. 89–94, 2016.
- [18] RAMANATHAN, P., MANGLA, K. K., SATPATHY, S. “Smart controller for conical tank system using reinforcement learning algorithm”, *Measurement*, 2017.
- [19] 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.
- [20] SILVER, D. “Lecture 1 - Introduction to Reinforcement Learning”. [http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching\\_files/intro\\_RL.pdf](http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/intro_RL.pdf). Acessado em Julho/2018.

- [21] HASSELT, H. V., WIERING, M. A. “Reinforcement learning in continuous action spaces”, 2007.