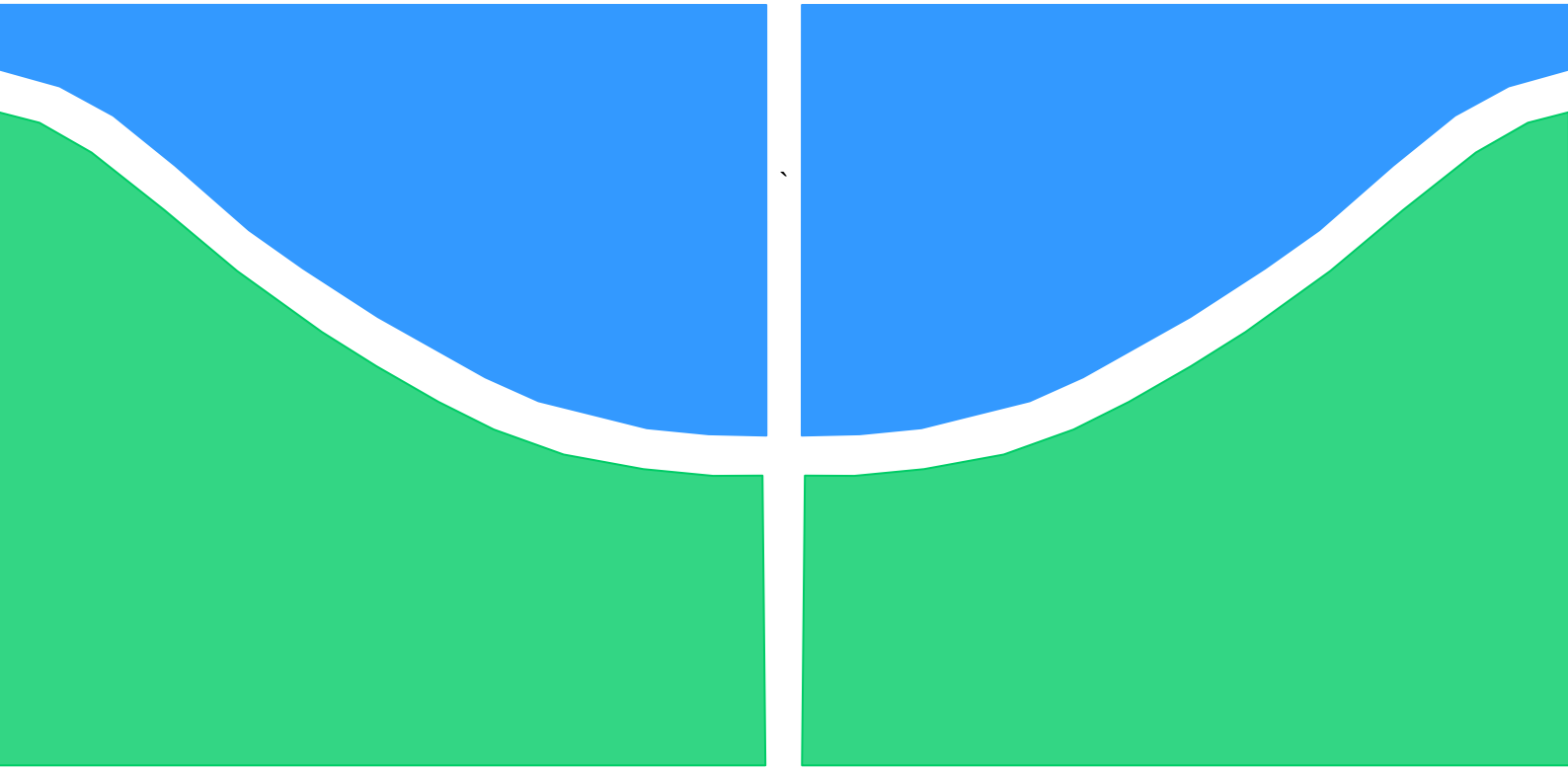
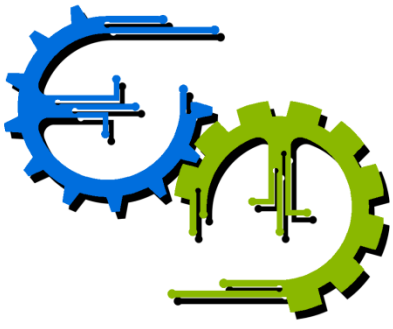


TRABALHO DE GRADUAÇÃO

CONTROLE DE UM SISTEMA VARIANTE NO TEMPO USANDO REDES NEURAIS ARTIFICIAIS E APRENDIZADO POR REFORÇO

Álvaro Queiroz dos Reis Silva

Brasília, dezembro de 2018



ENGENHARIA MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

CONTROLE DE UM SISTEMA VARIANTE NO TEMPO USANDO REDES NEURAIS ARTIFICIAIS E APRENDIZADO POR REFORÇO

Álvaro Queiroz dos Reis Silva

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Controle e Automação.

Banca Examinadora

Prof. Adolfo Bauchspiess, UnB/ ENE
(Orientador) _____

Prof. Alexandre Ricardo Romariz, UnB/ ENE _____

Prof. Lélio Ribeiro Soares Júnior, UnB/ ENE _____

Brasília, dezembro de 2018

FICHA CATALOGRÁFICA

SILVA, ÁLVARO QUEIROZ DOS REIS

Controle de um sistema variante no tempo usando redes neurais artificiais e aprendizado por reforço

Distrito Federal, 2018

xi, 41p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2018). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1.Aprendizado por reforço 2.Controle adaptativo
3.Controle de nível de liquido 4.Sistemas variantes no tempo

I. Mecatrônica/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

QUEIROZ, A. Q. R. S., (2018). CONTROLE DE UM SISTEMA VARIANTE NO TEMPO USANDO REDES NEURAIS ARTIFICIAIS E APRENDIZADO POR REFORÇO. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº , Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 41p.

CESSÃO DE DIREITOS

AUTOR: Álvaro Queiroz dos Reis Silva

CONTROLE DE UM SISTEMA VARIANTE NO TEMPO USANDO REDES NEURAIS ARTIFICIAIS E APRENDIZADO POR REFORÇO.

GRAU: Engenheiro ANO: 2018

Álvaro Queiroz dos Reis Silva
Setor Leste, Q44 L29-A – Gama.
72440-440 Brasília – DF – Brasil.

RESUMO

É proposta deste trabalho avaliar 2 algoritmos de aprendizado por reforço para o controle de sistema não linear de nível de líquidos de 4 tanques interligados. O primeiro algoritmo avaliado será uma implementação do Q-Learning usando tabelas. O segundo será o algoritmo de aprendizado por reforço ator-crítico usando redes de base radial no ator e no crítico. A variação no tempo será introduzida pela variação manual das válvulas em tempos pré-estabelecidos.

No primeiro caso, pretende-se avaliar o desempenho do algoritmo, que é relativamente simples, no controle da planta.

No segundo caso, deseja-se avaliar o desempenho do algoritmo, que foi desenvolvido por Matos [10] e propor possíveis melhorias.

A planta de nível será utilizada em conjunto com simulações (Simulink) para elucidar alguns pontos de sintonia dos algoritmos e, o processo de implementação é descrito, resultados são apresentados e discutidos.

Com o algoritmo Q-Learning, foi possível realizar o controle da planta de nível de líquidos com um erro em estado estacionário de 6.3%, erro devido a discretizações de ações e estados. Com o algoritmo de aprendizado por reforço ator-crítico, foi possível obter ganho de performance do controlador alimentando as redes neurais artificiais do ator e do crítico com os níveis dos 4 tanques. O algoritmo é capaz de se adaptar às mudanças de posições nas válvulas e seguir a referência. O tempo de assentamento foi de 40 segundos e o sobrepasso muito próximo a zero. O melhor controlador, dentre os avaliados neste trabalho, foi de aprendizado por reforço, ator-crítico com uma taxa de amostragem de 0.5 segundos e taxa de aprendizado de 0,1.

Palavras Chave: Aprendizado por reforço, redes neurais, controle de sistemas não lineares, sistemas variantes no tempo, controle adaptativo, Q-Learning

ABSTRACT

It is proposed to evaluate 2 reinforcement learning algorithms for non-linear fluid level control of 4 interconnected tanks. The first algorithm evaluated will be an implementation of Q-Learning using tables. The second will be the actor-critical reinforcement learning algorithm using for both the actor-critical the radial-based networks. The variation in time will be introduced by manually varying the valves at pre-set times.

In the first case, it is intended to evaluate the performance of the algorithm, which is relatively simple, in the control of the plant.

In the second case, we want to evaluate the performance of the algorithm, which was developed by Matos, [10] and propose possible improvements.

The level plant will be used in conjunction with simulations (Simulink) to elucidate some tuning points of the algorithms and, the implementation process is described, results are presented and discussed.

With the Q-Learning algorithm, it was possible to control the liquid level plant with a steady state error of 6.3%, error due to discretization's of actions and states. With the actor-critical reinforcement learning algorithm, it was possible to obtain controller performance gain by feeding the artificial neural networks of the actor and the critic with the levels of the 4 tanks. The algorithm is able to adapt to changes in valve positions and follow the reference. The settling time was 40 seconds and the overshoot close to zero. The best controller, among those evaluated in this study, was reinforcement learning actor-critical with a sampling rate of 0.5 seconds and a learning rate of 0.1.

Keywords: Reinforcement learning, neural networks, non-linear systems control, varying time systems, adaptive control, Q-Learning

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

w	Peso sináptico
S	Par estado-ação
R	Recompensa
G	Retorno esperado
a	ação
A	Área de secção no tanque
V	Valor da função de estado
Q	Função de valor para o par estado ação
H	Altura do fluido
k	Parâmetros das válvulas entre os tanques

Símbolos Gregos

α	Taxa de aprendizado
ξ	Função de custo
φ	Função de base radial
μ	Centro de função de base radial
σ	Dispersão de função de base radial
γ	Fator de desconto
δtd	Erro diferença temporal

Siglas

RBF	Radial Basis Function
RNA	Rede neural artificial
ANN	Artificial Neural Network
RL	Reinforced learning
TD	Diferença-temporal
PI	Controlador proporcional Integral
NARMA	Nonlinear Autoregressive Moving Average
RMSE	Root-mean-square deviation
FMDP	Finite Markov decision process

1. INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO

O controle de nível de líquidos com interação entre múltiplos tanques é comum em processos industriais. Academicamente, o problema é muito interessante por apresentar características não lineares relativamente simples e com constantes de tempo grandes, o que simplifica a construção de plantas educacionais.

Para usar técnicas de controle clássicas, é preciso utilizar o modelo linearizado da planta para um certo ponto de operação, entretanto, distanciando-se deste ponto, o erro tende a crescer, podendo até a chegar a um ponto onde o sistema fica instável em malha fechada. Em malha aberta o processo é sempre estável e está sujeito a saturação dos níveis ou do atuador.

Aprendizado de máquina é um subcampo da ciência da computação que apresenta ferramentas para resolver problemas complexos como é o controle de nível de líquidos citado. Mitchel [17] forneceu uma definição mais formal amplamente citada: "Diz-se que um programa de computador aprende pela experiência E, com respeito a algum tipo de tarefa T e performance P, se sua performance P nas tarefas em T, na forma medida por P, melhoram com a experiência E".

Alguns acontecimentos históricos importantes para o campo de aprendizado de máquina são os seguintes: Em 1958, Rosenblatt [12] apresenta o Perceptron, unidade básica das redes neurais modernas, em 1973, Dreyfus [3] usou um novo algoritmo chamado de *backpropagation* para adaptar parâmetros de controladores em proporção aos gradientes de erro, em 1974 Werbos [16] menciona a possibilidade de usar esse mesmo princípio em redes neurais artificiais, em 1986, Rumelhart, Hinton e Williams mostram independentemente, experimentalmente, que esse método pode gerar representações internas de dados vindos das camadas internas de redes neurais [13].

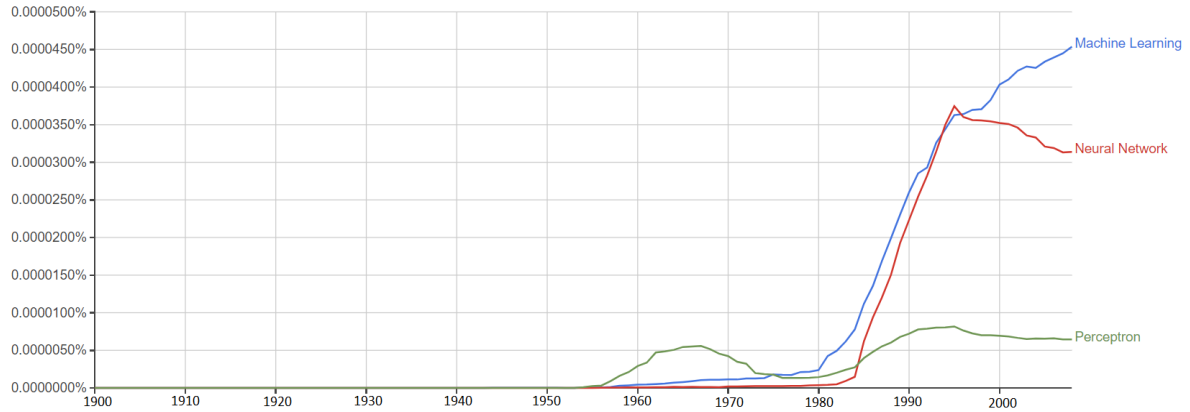


Figura 1.1 Google Ngrams mostra a proporção de cada expressão citada no corpo de livros de seu catálogo por ano.

Redes neurais artificiais são importantes instrumentos para a aplicação na área de controle de sistemas não-lineares, dadas as suas características de aproximadores universais e a capacidade de aprender com treinamento. Muitos sistemas são intrinsecamente não lineares, como a planta de nível de líquidos estudada neste trabalho.

Vários estudos já foram feitos no uso de redes neurais para controle de sistemas de nível de líquidos e na última década os avanços em hardware, teoria e software tornaram possível a implementação dessas técnicas com maior sucesso.

Oliveira [7] apresentou a implementação de um perceptron multicamadas para o controle do mesmo sistema deste trabalho, ao final de seu trabalho o autor escreve “Observou-se que, para o processo de quatro tanques acoplados, o controle neural[perceptron

multicamadas] utilizado apresentou desempenho pouco satisfatório, fato que pode ser atribuído à complexidade, oscilações do tipo stick-slip da planta", entretanto, alguns anos depois, Matos [10] trabalhando na mesma planta usando um algoritmo de aprendizado por reforço ator-crítico com redes neurais de base radial, obteve resultados animadores, o autor escreve "*When compared to a PI the designed controller showed worse results when responding to the first reference values, but over time, it became more and more efficient and evened the PI performance* surpassing *it eventually and being able to steady the reference value faster and with lower overshoot*". Portanto técnicas de controle utilizando aprendizado por reforço e redes neurais se mostram promissoras, algumas vezes, como o último autor escreve, pode-se obter resultados melhores do que os obtidos com técnicas de controle clássicas.

Existem 3 categorias de aprendizado de máquinas, aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço, os dois primeiros tipos serão explorados brevemente e aprendizado por reforço será explorado de forma suficiente para a construção do entendimento do algoritmo usado para o controle de nível.

1.2. OBJETIVOS DO TRABALHO

Este trabalho tem como objetivo geral, explorar técnicas de aprendizado por reforço e o uso de redes neurais artificiais para o controle de um sistema não linear de nível de líquidos de 4 tanques interligados e estudar como um dos algoritmos se comporta com variações dos parâmetros do sistema em tempo de operação.

O primeiro algoritmo a ser apresentado será uma implementação do Q-Learning usando tabela, o segundo será o algoritmo desenvolvido por Matos [10] em sua dissertação de mestrado sob a mesma orientação deste trabalho, o algoritmo é uma implementação de aprendizado por reforço Ator-Crítico, o algoritmo será avaliado em diversos aspectos para que se possa propor eventuais melhorias.

Para comparação, serão usados controladores NARMA-L2 e PI.

1.3. RESULTADOS OBTIDOS

Dos resultados apresentados neste trabalho, é possível concluir que o Q-Learning é um algoritmo inapropriado para controle de sistemas onde existe necessidade de precisão, para a referência de 10 cm e taxa de aprendizado 0,05, o erro obtido após 11 minutos de treinamento foi de 0,6 cm. Essa limitação é devido às discretizações de estados e de ações, entretanto, o algoritmo obtém bons resultados sem qualquer conhecimento do ambiente(da planta) e pode ser usado para o controle de sistemas com requisitos mais permissivos em relação a precisão.

Também foi possível elucidar vários aspectos importantes do algoritmo de aprendizado por reforço ator-crítico, em especial, foi possível obter melhora no desempenho do algoritmo observando os 4 estados(níveis dos tanques) ao invés de apenas um, e o algoritmo se torna adequado para o controle de sistemas variantes no tempo com uma taxa de decaimento da taxa de aprendizado igual a 1.

2. INTRODUÇÃO TEÓRICA

2.1. APRENDIZADO POR REFORÇO

Essa técnica é diferente de aprendizado supervisionado, onde o aprendizado é feito através de treinamentos com situações marcadas, com rótulos, a ação correta a ser tomada é ditada por conhecimento de um agente externo, o supervisor. Cada exemplo é a descrição de uma situação e a qual categoria ela pertence e qual ação deve ser tomada. O objetivo desse tipo de aprendizado é extrapolação ou generalizar sua resposta para situações não presentes nos dados de treinamento, mas em alguns problemas é difícil ou pouco prático conseguir dados para fazer o treinamento.

A técnica também não é considerada como um tipo de aprendizado não supervisionado, que é basicamente um conjunto de técnicas que tenta encontrar estruturas escondidas em um conjunto de dados de treinamento sem rótulos. Talvez seja tentador pensar em aprendizado por reforço como uma forma de aprendizado não supervisionado porque a técnica não precisa de exemplos de comportamentos desejados, mas o objetivo de aprendizado por reforço é maximizar uma função de recompensa ao invés de encontrar padrões escondidos nos dados apresentados, por isso, aprendizado por reforço é considerado uma terceira técnica de aprendizado de máquinas ao lado de aprendizado supervisionado e não supervisionado.

Suponha um agente em um ambiente, o agente é capaz de agir sobre o ambiente de forma a causar uma mudança nesse ambiente, ou seja, o agente é capaz de causar uma mudança de estado no ambiente. Este agente possui metas e suas ações são orientadas por recompensas ou punições de forma a cumprir esse objetivos.

Aprendizado por reforço é aprender o que fazer, como mapear situações para ações de forma a maximizar numericamente o sinal de recompensa. O agente não recebe ordens do que fazer, o que ele deve fazer é descobrir quais ações tem maior compensa. Essas duas características, tentativa e erro e a busca por maximizar uma função recompensa são as duas ideias principais que distinguem o aprendizado por reforço.

Aprendizado por reforço é formalizado usando ideias de teoria de sistemas dinâmicos. Simplificadamente, um agente deve ter a capacidade de medir o ambiente e deve ser capaz de agir sobre esse ambiente de forma a afetar o seu estado. O agente também deve ter metas relacionadas ao estado de ação e meios para chegar a este estado, portanto o processo de decisão deve considerar a medida, a ação e o objetivo.

Uma das dificuldades que surge em aprendizado por reforço e não em outros tipos de aprendizado de máquina, é o balanço entre exploração de novas ações e o uso de ações já conhecidas. Para obter uma grande recompensa o agente deve preferir ações que ele já usou no passado e sabe que são efetivas em gerar recompensa, mas para descobrir novas ações, ele deve tentar coisas novas. O agente deve explorar ações que já foram testadas e explorar novas ações para descobrir ações melhores para selecionar em estados futuros. O dilema é que o agente não pode exclusivamente explorar novas ações, ou explorar ações apenas conhecidas, sem falhar a tarefa dada, o agente deve tentar uma variedade de ações e gradativamente favorecer essas que parecem ser as melhores. Esse dilema será explorado na sessão “O problema do bandido multi-armado”.

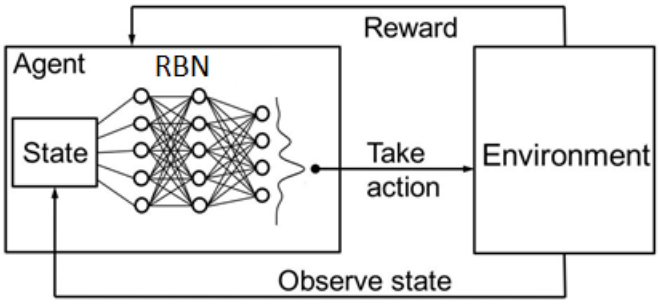


Figura 2.1 Representação algoritmo de aprendizado por reforço utilizando RBN para aproximação da política a = f(S), extraído de [14].

Outra característica chave de aprendizado por reforço, é que o problema do agente com objetivo-recompensa interagindo com um ambiente não conhecido é considerado como um todo, em contraste com muitas técnicas onde um problema é dividido em subproblemas menores e mais simples, aprendizado por reforço vai no sentido contrário, começando com um agente completo, interativo e que busca o objetivo dado. Todos os agentes têm objetivos explícitos, podem medir aspectos do ambiente onde atuam, e podem tomar ações que influenciam esses ambientes.

2.2. APRENDIZADO DIFERENÇA-TEMPORAL

Métodos de Aprendizado diferença-temporal, ou métodos TD, podem aprender diretamente do ambiente sem o modelo de sua dinâmica, podem atualizar suas estimativas baseando-se em parte em outras estimativas aprendidas, sem esperar o fim do processo.

Suponha um rato em um labirinto, sua posição no labirinto será o seu estado, a escolha de caminho será a política e par estado ação(ou política) define a função valor V. O método usa experiência para resolver problemas de predição, dada uma experiência ganha usando uma política π, a estimativa dos valores V e v_π são atualizados para o estado S_t ocorrido nessa experiencia, portanto, métodos TD esperam apenas até a próxima iteração para atualizar o valor da estimativa

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \quad (2.1)$$

No tempo *t+1* A recompensa observada R_{t+1} e a estimativa $V(S_{t+1})$ são usados para atualizar o valor $V(S_t)$, esse método é chamado TD(0), ou TD de passo único.

De forma procedural, o algoritmo é como se segue:

Algoritmo 1 – TD(0)

$Input: Política\ \pi\ a\ ser\ avaliada$
$Inicializar\ V(s)\ arbitrariamente$
$Repetir\ (para\ cada\ episódio)$
    $Inicializar\ S$
    $Repetir\ (Para\ cada\ passo\ do\ episódio)$
    $A \leftarrow ação\ dada\ por\ \pi\ para\ S$
    $Tomar\ ação\ A, observar\ R\ e\ S'$
    $V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$
    $S \leftarrow S'$
$Até\ que\ S\ acabe$

É possível notar que a quantidade obtida é uma espécie de erro, medindo a diferença entre a estimativa do valor S_t e a melhor estimativa $R_{t+1}\ e\ \gamma V(S_{t+1})$. Essa quantidade é chamada de erro TD, e surge em formas variadas de aprendizado por reforço.

Uma das vantagens do método é que ele não precisa de um modelo do ambiente, outra grande vantagem é que o método é implementado *online*, de forma incremental. Essa pode ser uma característica crítica para sistemas dinâmicos que estão em constante mudança.

2.3. O ALGORITMO SARSA

No aprendizado diferença-temporal, apenas as transições de estado para estado e seus valores são considerados. No algoritmo SARSA, são consideradas as transições do par estado-ação para estado-ação e os valores do novo estado.

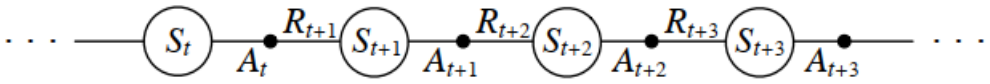


Figura 2.2 Representação gráfica de transições do par estado-ação.

Formalmente, o algoritmo é definido analogamente ao aprendizado diferença temporal

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (2.2)$$

A atualização é feita em toda transição de um estado não terminal S_t. Se S_{t+1} for terminal, então $Q(S_{t+1}, A_{t+1})$ é definido como zero, essa regra usa todos os elementos do quíntuplo de eventos, $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$, que compõe a transição de um par estado-ação para o próximo, sendo esse quíntuplo de elementos os responsáveis pelo nome SARSA para o algoritmo.

2.4. Q-LEARNING

Q-learning é uma técnica de aprendizado de reforço usada em aprendizado de máquina. O objetivo do Q-Learning é aprender uma política, que diz a um agente qual ação tomar sob dadas circunstâncias. Não requer um modelo do ambiente e pode lidar com problemas com transições e recompensas estocásticas, sem exigir adaptações.

Para qualquer processo finito de decisão de Markov (FMDP), o Q-learning encontra uma política que é ótima no sentido de que maximiza o valor esperado da recompensa total sobre todos os passos sucessivos, a partir do estado atual. O Q-learning pode identificar uma política ótima de seleção de ação para qualquer FMDP, dado o tempo infinito de exploração e uma política parcialmente aleatória. "Q" nomeia a função que retorna a recompensa usada para fornecer o reforço e pode ser considerada a "qualidade" de uma ação tomada em um determinado estado.

Variáveis importantes

- **Exploração vs Aproveitamento:** A taxa de aprendizado ou o tamanho da etapa determina até que ponto as informações recém-adquiridas substituem as informações antigas. Um fator de 0 faz com que o agente não aprenda nada (explorando exclusivamente o conhecimento prévio), enquanto um fator de 1 faz com que o agente considere apenas as informações mais recentes (ignorando o conhecimento anterior para explorar as possibilidades). Em ambientes totalmente determinísticos, uma taxa de aprendizado 1 é ideal. Quando o problema é estocástico, o algoritmo converge sob algumas condições técnicas na taxa de aprendizado que exige que ele diminua para zero. Na prática, muitas vezes é usada uma taxa de aprendizado constante.
- **Fator de Desconto:** O fator de desconto determina a importância de recompensas futuras. Um fator de 0 tornará o agente "míope" (ou míope) considerando apenas as recompensas atuais, enquanto um fator que se aproxima de 1 fará com que se esforce

por uma recompensa alta de longo prazo. Se o fator de desconto atender ou exceder 1, os valores da ação podem divergir, sem um estado terminal, ou se o agente nunca atingir um, todos os históricos de ambiente tornam-se infinitamente longos e utilitários com recompensas aditivas não-recontadas geralmente se tornam infinitos. Mesmo com um fator de desconto apenas um pouco menor que 1, o aprendizado da função Q leva à propagação de erros e instabilidades quando a função valor é aproximada com uma rede neural artificial. Nesse caso, começar com um fator de desconto menor e aumentá-lo em direção ao seu valor final acelera o aprendizado.

- **Condições iniciais:** Como o Q-learning é um algoritmo iterativo, ele assume implicitamente uma condição inicial antes que a primeira atualização ocorra. Altos valores iniciais, também conhecidos como "condições iniciais otimistas", podem estimular a exploração: não importa qual ação seja selecionada, a regra de atualização fará com que ela tenha valores menores que a outra alternativa, aumentando assim sua probabilidade de escolha. A primeira recompensa r pode ser usada para redefinir as condições iniciais. De acordo com essa ideia, a primeira vez que uma ação é tomada, a recompensa é usada para definir o valor de Q. Isso permite o aprendizado imediato em caso de recompensas determinísticas fixas. Espera-se que um modelo que incorpore o reset das condições iniciais (RIC) preveja melhor o comportamento dos participantes do que um modelo que assume qualquer condição inicial arbitrária (AIC).
- **Quantização:** Considere o exemplo de aprender a equilibrar um bastão em um dedo. Descrever um estado em um determinado momento envolve a posição do dedo no espaço, sua velocidade, o ângulo do bastão e a velocidade angular do bastão. Isso produz um vetor de quatro elementos que descreve um estado, ou seja, um instantâneo de um estado codificado em quatro valores. O problema é que infinitamente muitos estados possíveis estão presentes. Para reduzir o espaço possível de ações válidas, vários valores podem ser atribuídos a um intervalo.

Usada para a atualização da tabela Q, o princípio de optimalidade de Bellman fornece uma definição recursiva para a função Q ótima [14]. O $Q(S_t, A_t)$ é igual ao somatório da recompensa imediata depois de realizar a ação durante algum tempo no estado e a recompensa futura esperada descontada após a transição para um próximo estado.

A equação aplicada no algoritmo Q-Learning desenvolvido durante este trabalho apresenta uma variável Bônus, que como o nome sugere, será um reforço positivo dado ao ator caso sua ação leve a um estado desejável, mais detalhes serão apresentados na sessão Implementação.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) + B\hat{o}nus] \quad (2.3)$$

Para ilustrar o algoritmo apresentado anteriormente, considere o seguinte jogo: o ator deverá percorrer um tabuleiro. Para vencer, ele deverá chegar à posição de cor verde, com isso ele recebe um bônus de 1 ponto, caso ele chegue a posição de cor vermelha, ele receberá uma punição de -1 ponto e o jogo também acaba. Para esse jogo, cada estado do sistema é sua posição x,y. As ações possíveis são, mover-se para o norte, sul, leste ou oeste.

A tabela Q é inicializada com zeros, abaixo é possível ver a evolução dos valores máximos de Q para cada estado na tabela Q.

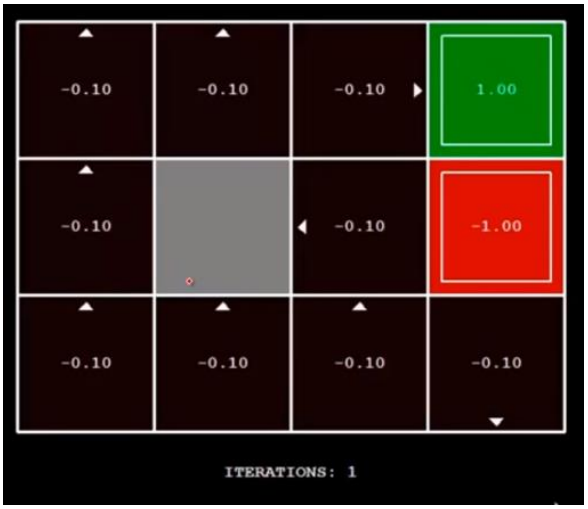


Figura 2.3 Tabela de ações na primeira iteração.

Nas primeiras iterações, o ator irá percorrer o tabuleiro escolhendo direções de forma aleatória principalmente, isso porque ele ainda não tem experiencias passadas como referência, para dizer qual a melhor ação em cada estado, entretanto, toda vez que uma ação é tomada, ela é avaliada, e o par estado ação é atualizado de acordo com o efeito que a ação teve. Se o ator estiver no estado (3,1) e tomar a direção leste, ele será punido com -1 ponto e o jogo acaba, mas da próxima vez que o ator passar por este estado, ele irá saber que tomar a direção leste é uma opção ruim, devido ao valor associado ao par estado-ação, e tomará outra direção.

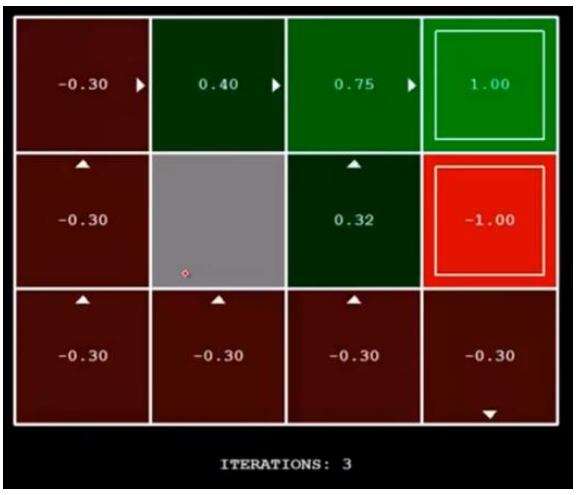


Figura 2.4 A tabela indica qual a melhor ação para ser tomada a cada estado após 3 iterações.

Após 3 jogos, o ator começa a ter uma tabela Q com informações relevantes, por exemplo, se ele estiver no estado (3,3) ele deve tomar a direção leste, assim o jogo acaba e

ele recebe um bônus de *+1*, ele também “descobriu” que se ele estiver no estado (3,1), o melhor que ele tem a fazer é ir para o norte.

Como pode ser observado figura 2.5, em poucas iterações o caminho ótimo é encontrado para esse jogo simples.

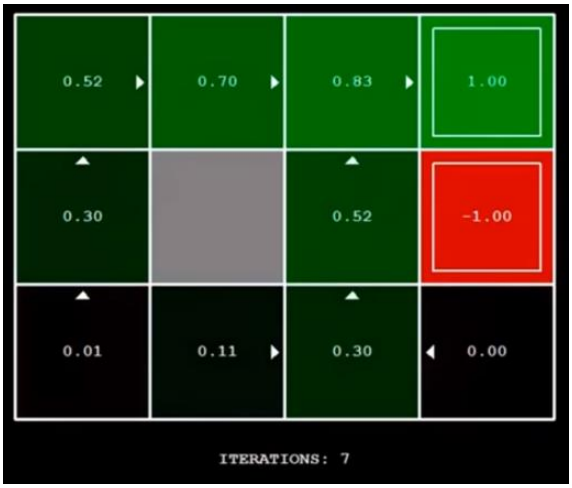


Figura 2.5 A tabela indica qual a melhor ação para ser tomada a cada estado após 7 iterações.

Na tabela Q completa da figura 2.6, é possível ver os valores Q para cada ação correspondente a posição do valor, note, sul, leste, oeste. Com esta tabela, o agente escolherá sempre a ação associada ao maior valor Q, na posição (0,0), por exemplo, ele irá para o norte porque esssa ação tem o maior valor, 0,49.

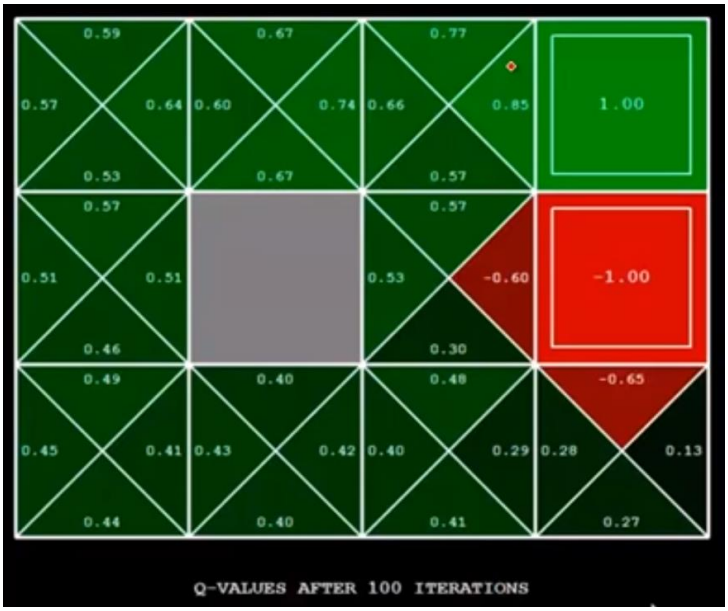


Figura 2.6 Tabela Q completa após 100 iterações.

2.5. O PROBLEMA DO BANDIDO MULTI-ARMADO

Na teoria da probabilidade, o problema do bandido armado (às vezes chamado de problema do bandido K ou N-armado) é um problema no qual um conjunto limitado de recursos deve ser alocado entre escolhas concorrentes (alternativas). Uma maneira que maximiza seu ganho esperado, quando as propriedades de cada escolha são apenas parcialmente conhecidas no momento da alocação, e podem ser melhor entendidas à medida que o tempo passa ou pela alocação de recursos para a escolha. O nome vem de imaginar um apostador em uma fila de caça-níqueis (às vezes conhecidos como "bandidos de um braço"), que tem que decidir quais máquinas jogar, quantas vezes jogar cada máquina e em que ordem jogá-las, e se deve continuar com a máquina atual ou tentar uma máquina diferente.

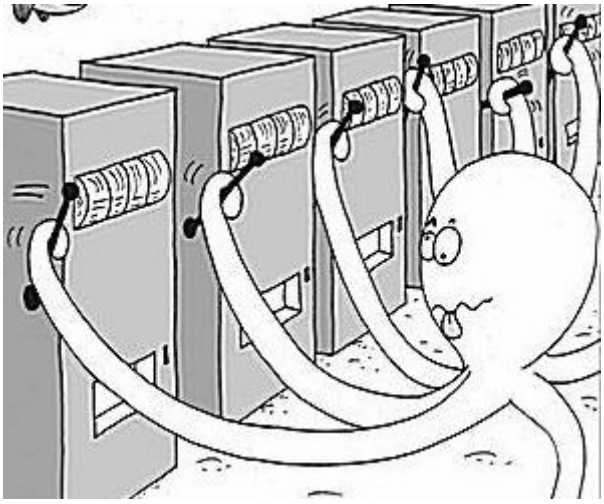


Figura 2.7 O problema de escolher onde alocar recursos limitados em um cassino

Algumas estratégias podem ser empregadas para resolver o problema, dentre elas:

- **Estratégia gananciosa**: a melhor alavanca conhecida é selecionada sempre.

- **Estratégia épsilon-ganancioso**: a melhor alavanca é selecionada para uma proporção *1 – épsilon* dos testes, e uma alavanca é selecionada aleatoriamente (com probabilidade uniforme) para uma proporção épsilon. Um valor do parâmetro épsilon típico é *0,1.* Mas isso pode variar dependendo do problema.

- **Estratégia do épsilon-decrescente:** Similar à estratégia épsilon-ganancioso, exceto que o valor do épsilon diminui conforme o experimento avança, resultando em comportamento altamente exploratório (exploração em largura) no início e comportamento altamente explorador (exploração em profundidade) no terminar.

Outras estratégias podem ser usadas, um outro exemplo seria o de ter períodos completamente exploratórios (testar várias máquinas) e logo após um de aproveitamento (jogar apenas nas máquinas onde mais se obteve lucro).

2.6. APRENDIZADO POR REFORÇO ATOR-CRÍTICO

Algoritmos ator-crítico aprendem tanto políticas quanto valor de funções. O ator é o componente que aprende políticas, e o crítico é o componente que aprende sobre como está se comportando a política utilizada pelo ator para criticar suas escolhas. O crítico usa algoritmos de diferença-temporal para aprender o estado-valor da função para a atual política do ator, o valor da função permite que o crítico critique as escolhas de ações do ator através do envio de erros TD para o ator. Um erro TD positivo significaria que a ação foi boa, porque levou o ambiente a um estado melhor que o esperado, enquanto que o erro TD negativo indica que a ação foi ruim, porque levou a um estado pior que o esperado, então, baseado nessas críticas o ator pode continuamente atualizar suas políticas de ação.

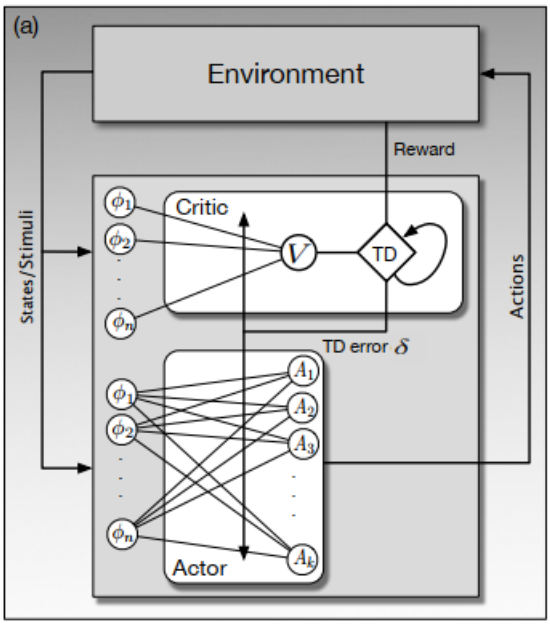


Figura 2.8 Esquema gráfico para aprendizado por reforço ator-crítico, extraído de [14].

Ambos, ator e crítico recebem entradas que consistem de características múltiplas, representando o ambiente do agente. Os pesos no ator parametrizam o valor da função, e os pesos na rede do ator parametrizam a ação, ou política, as redes aprendem conforme os pesos nessas redes mudam de acordo com as regras de aprendizado.

O erro TD produzido pelo crítico é o sinal de reforço para a mudança nos pesos tanto do ator quanto do crítico, esse aspecto da implementação da rede, juntamente com a predição de recompensa e hipótese de erro pode ser a forma como o cérebro humano aprende, segundo Sutton e Barto [14].

2.7. REDES NEURAIS COM FUNÇÕES DE BASE RADIAL

As redes neurais com funções de base radial possuem estrutura de camadas assim como a usada no método perceptron multicamadas, a estrutura da rede é dividida em 3 componentes principais, entradas, camadas ocultas e camada de saída.

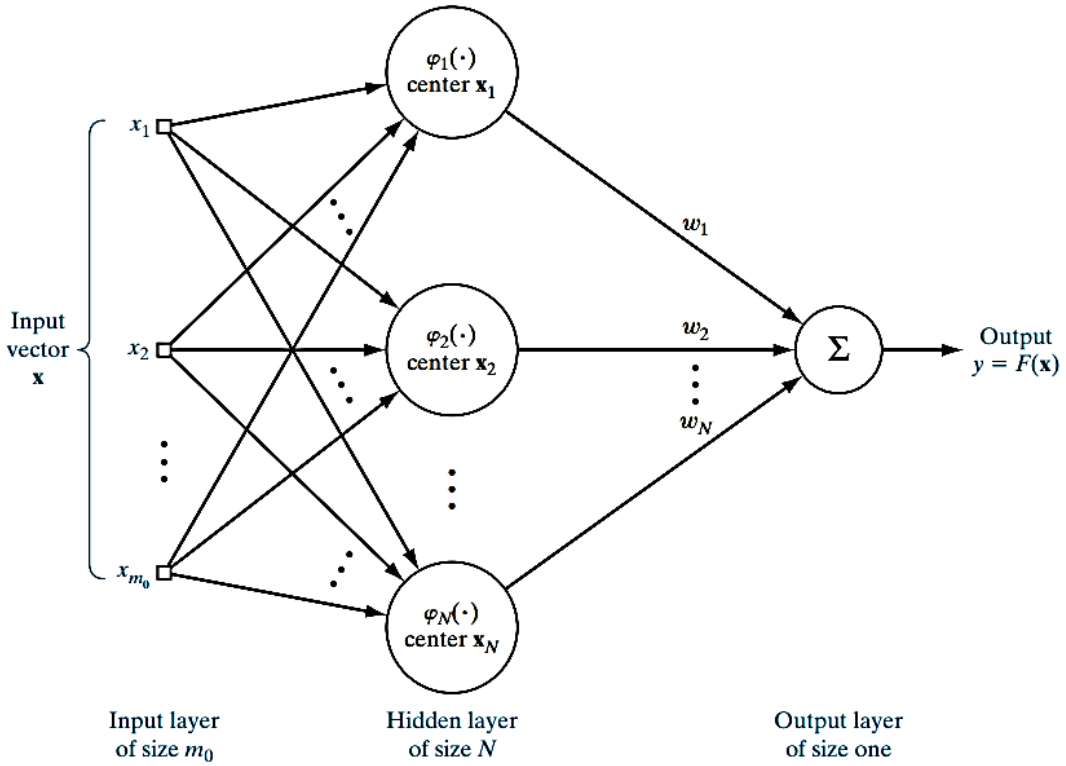


Figura 2.9 Representação de uma rede neural com função de base radial..

A camada de entrada consiste de m_0 nós fonte, onde m_0 é a dimensão do vetor de entradas x.

$$\varphi_j(x) = \varphi(\|x - x_j\|), \quad j = 1, 2, 3, \ldots N \tag{2.4}$$

Onde o ponto x_j define o centro da função de base radial, e o vetor x é o sinal aplicado na camada de entrada, portanto, diferentemente do perceptron multicamadas, a conexão entre a camada de entrada e as camadas ocultas são diretas sem qualquer peso.

A camada de saída consiste tipicamente de uma única unidade computacional, como nesse trabalho a Gaussiana será usada como função de base radial, cada unidade computacional nas camadas ocultas corresponde a

$$\varphi_j(x) = \varphi(\|x - x_j\|) = exp\left(-\frac{1}{2\sigma_j^2}\|x - x_j\|^2\right), j = 1, 2, 3, \ldots N \tag{2.5}$$

onde σ_j é uma medida da largura da Gaussiana na posição j com centro x_j, tipicamente, mas nem sempre, todas as Gaussianas têm a mesma medida de largura σ_j. Esse é o caso deste trabalho, nesses casos, cada camada oculta se difere uma das outras através das diferentes posições de seus centros x_j.

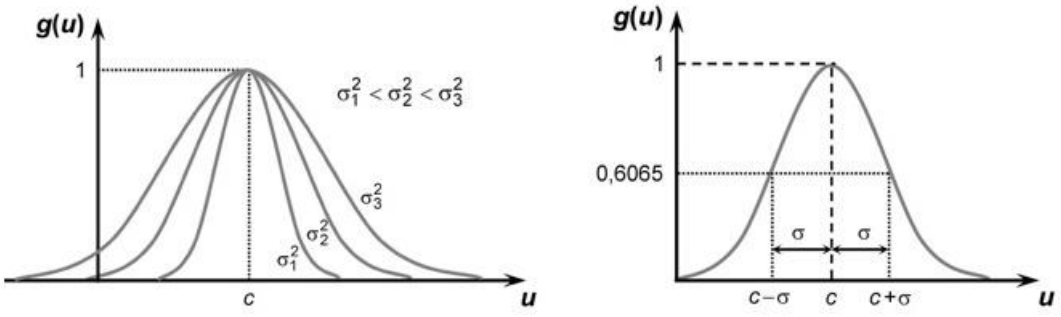


Figura 2.10 Variáveis importantes de uma Gaussiana, RBF usada durante este trabalho.

Portanto, a saída na camada de saída é um único número real definido por:

$$y = F(x) = \sum_{j=1}^{K} \omega_j \varphi_j(x, x_j) \tag{2.6}$$

Ou ainda

$$y = \sum_{j=1}^{K} \omega_j \exp\left(-\frac{1}{2\sigma_j^2}\|x - x_j\|^2\right) \tag{2.7}$$

2.8. NARMA-L2

NARMA, Nonlinear Autoregressive Moving Average, do ingles, é uma representação do sistema não linear nas proximidades de uma regiao de equilibrio.
O modelo aproximador NARMA-L2 foi proposto por Narendra e Mukhopadhayay em 1997. O modelo NARMA-L2 pode ser usado com o modelo das plantas não-lineares. Tipicamente, o número de atrasos desta rede neural incrementa com a ordem da planta. O controlador NARMA-L2 procura transformar o sistema não linear em um sistema linear através do cancelamento das não linearidades.

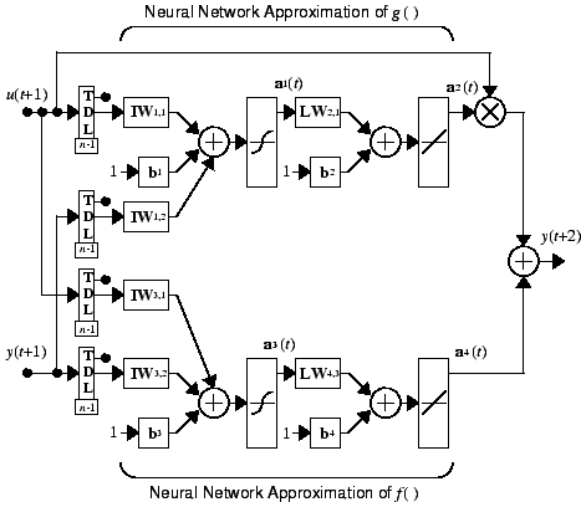


Figura 2.11 Estrutura NARMA-L2.

O controlador NARMA-L2 é um rearranjo da rede neural que modela o processo que é treinado offline. Para isso é preciso treinar uma rede neural para representar a dinâmica do sistema. O modelo NARMA é representado pelo seguinte modelo:

$$y(k+d) = N[y(k), y(k-1), \dots, y(k-n+1)], u(k), u(k-1), \dots, u(k-n+1) \qquad (2.8)$$

onde u(k) é a entrada e y(k) é a saída do sistema.
O objetivo da fase de treinamento é descobrir a função N que aproxima as não linearidades do sistema, que aqui vamos chamar de F e G:

$$\hat{y}(k+d) = f[y(k), y(k-1), \dots, y(k-m+1)] + g\big[[y(k), y(k-1), \dots, y(k-n+1)], u(k-1), \dots, u(k-n+1)\big]u(k) \qquad (2.9)$$

Onde d é o atraso, n é o atraso da saída e m é o atraso da entrada.
O controlador NARMA-L2 é dado por:

$$u(k) = \frac{y_r(k) - f[Y, U]}{g[Y, U]} \qquad (2.10)$$

O desempenho do controlador NARMA-L2 depende essencialmente da identificação do sistema com uma rede neural. Um bom modelo de uma rede neural para o sistema fornece bons resultados nos pontos de operação desejados [7].

3. DESCRIÇÃO DA PLANTA

3.1. A PLANTA DE NÍVEL DE LÍQUIDOS

O código de controle será avaliado na planta didática de quatro tanques que se encontra no Laboratório de Automação e Robótica, LARA. O sistema recebe água de um reservatório que é bombeada por uma bomba para o primeiro tanque. No segundo e no quarto tanque existem saídas para a água retornar ao reservatório.

Três desses tanques possuem dimensões de 49,5x10x6 centímetros e o quarto tanque tem secção variável. As secções transversais dos tanques 1, 2 e 3 são de 60 cm². A largura do tanque 4 a partir de 14,7 cm de altura, e que representa 29% do nível de líquido, passa a aumentar linearmente até 20,2 cm de largura (31,65°) com área da secção transversal passando de 60 cm² para 121,2 cm².

O tanque 1 possui entrada de água, que é bombeada do reservatório para este tanque por meio de uma motor-bomba. O tanque 2 possui uma saída de água, qo2, através de um furo de 4 mm de diâmetro, situado no fundo desse tanque para o reservatório. O tanque 4 possui uma saída de água, qo4, através de um furo de 4 mm, situado no fundo desse tanque para o reservatório.

Entre os compartimentos existe uma ranhura de 2 mm de largura com altura ajustável, que determina o parâmetro k de interconexão entre os tanques. Com isso, é possível variar o fluxo de água entre os tanques, ou seja, será possível variar os parâmetros do sistema durante os experimentos feitos. Essa será uma das formas de avaliar os algoritmos durante o trabalho.

A motor-bomba é acionada por meio de um sistema de potência comandado a partir de uma tensão entre 0 e 10 Volts. O último tanque é dotado de um sensor de nível por pressão. A dinâmica do processo é nitidamente não-linear, uma vez que as vazões nos tanques dependem da raiz quadrada das alturas dos níveis de água para fluxo turbulento [7].

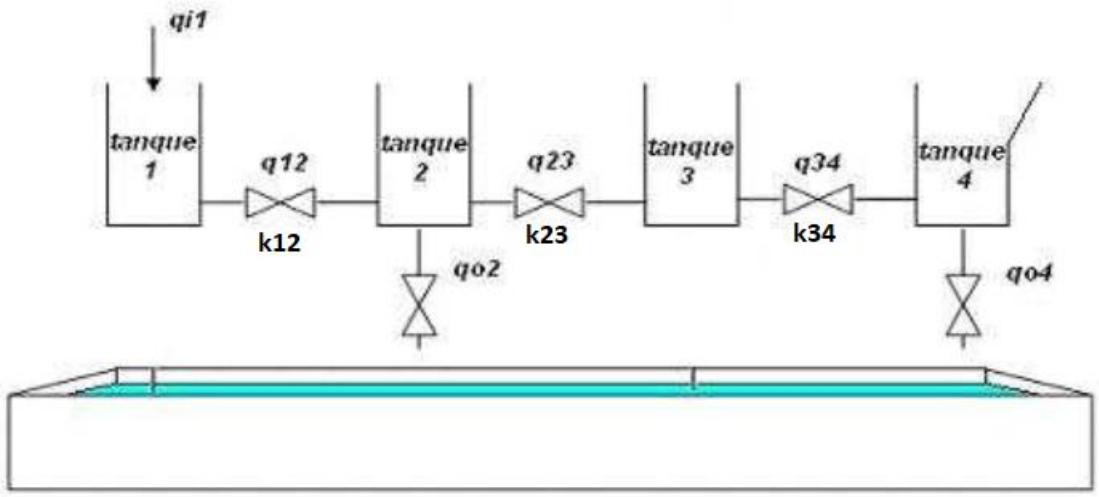


Figura 3.1 Representação esquemática da planta de nível de líquidos, extraído de [7].

A variável a ser controlada, será o nível do quarto tanque, H4, onde está o sensor de nível, o atuador será a bomba que se encontra no reservatório, abaixo dos quatro tanques.

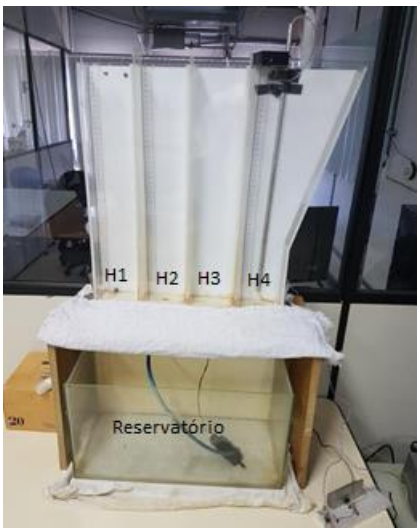


Figura 3.2 Planta de nível de líquidos.

Com base na descrição anterior, é possível escrever as equações do sistema, para respeitar a conservação de massas (equação de continuidade), temos que:

$$\frac{dV}{dt} = A\frac{dh}{dt} = \sum q_{in} - \sum q_{out} \tag{3.1}$$

Agora podemos escrever uma equação para cada tanque:

$$A_1 \frac{dH1}{dt} = q_{i1} - q_{12} \tag{3.2}$$

$$A_2 \frac{dH2}{dt} = q_{12} - q_{02} - q_{23} \tag{3.3}$$

$$A_3 \frac{dH3}{dt} = q_{23} - q_{34} \tag{3.4}$$

$$A_4(h) \frac{dH4}{dt} = q_{34} - q_{04} \tag{3.5}$$

Para o sistema descrito neste trabalho, as equações que determinam as vazões para o reservatório são as seguintes [7]:

$$q_{02}(t) = \ k_{02}\sqrt{h_2(t)} \tag{3.6}$$

$$q_{04}(t) = \ k_{02}\sqrt{h_4(t)} \tag{3.7}$$

As equações que determinam as vazões entre os tanques são:

$$q_{12}(t) = \ k_{12}\sqrt{h_1(t) - \ h_2(t)} \tag{3.8}$$

$$q_{23}(t) = \ k_{23}\sqrt{h_2(t) - \ h_3(t)} \tag{3.9}$$

$$q_{34}(t) = \ k_{34}\sqrt{h_3(t) - \ h_4(t)} \tag{3.10}$$

Como pode ser visto pelas equações, o sistema é de quarta ordem, o parâmetro A_4 varia com a altura e ainda temos saturação e zona morta no atuador e além disso, a variação de nível nos tanques depende da raiz quadrada do nível do tanque a cada momento.

Para as simulações, que serão muito importantes durante os próximos capítulos, será necessário conhecer as constantes de resistência a passagem de fluído nas ranhuras do tanque, que podem ser vistas como válvulas, na terceira ranhura, como exemplo, temos:

$$k_{34} = \frac{q_{34}(t)}{\sqrt{h_3(t) - \ h_4(t)}} \tag{3.11}$$

A figura 3.5 mostra como a posição da gaveta na planta de nível de líquidos causa uma variação de fluxo de água entre os tanques.

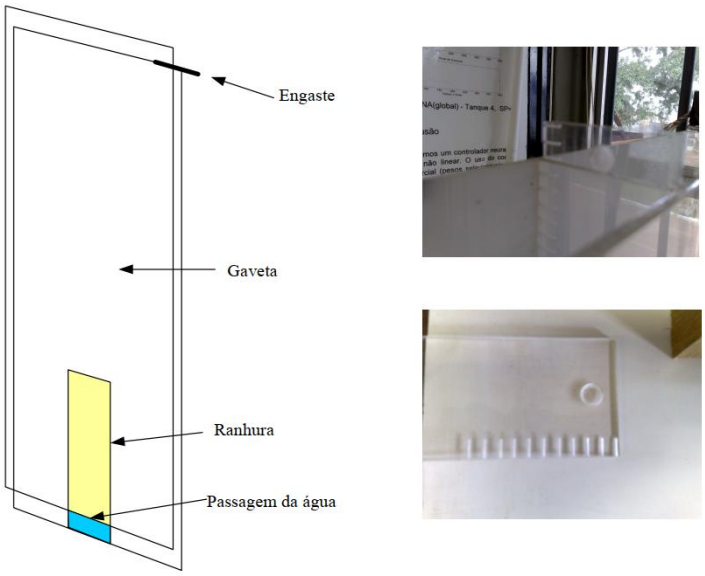


Figura 3.3 Representação de válvulas na planta, extraído de Oliveira [7].

Em experimentos com o fim de identificação desses parâmetros, Oliveira [7] obteve *k2 = 4,313 e k4 = 4,9667* e os valores contidos na tabela 3.1.

Tabela 3.1 Valores de parâmetros da planta, extraído de [7].

Posição da válvula	$k_{12}\left[cm^{2,5}/s\right]$	$k_{23}\left[cm^{2,5}/s\right]$	$k_{34}\left[cm^{2,5}/s\right]$
1ª engaste da régua	2,88	4,64	5,26
2ª engaste da régua	9,06	11,89	10,09
3ª engaste da régua	12,18	20,62	16,65

Como pode ser observado, para diferentes alturas no engaste (posição da válvula), teremos diferentes valores para cada um dos parâmetros na tabela. Essa característica será explorada em simulações para validar a capacidade do controlador de lidar com variações nos parâmetros da planta em tempo de execução.

3.2. ELETRÔNICA DE ACIONAMENTO E AQUISIÇÃO DE DADOS

A saída do ator para cada um dos 2 algoritmos implementados será um sinal enviado ao atuador na foma de um sinal de pulso modulado, ou PWM. Esse sinal digital tem amplitude que varia de zero a 5 volts para o microcontrolador Arduino UNO, que será o usado no controle do processo como interface entre o computador e a planta de nível de líquidos.

A execução do algoritmo ocorrerá em um camputador utilizando MATLAB, o sinal de controle será enviado por USB para o Arduino, este enviará o sinal de controle para o atuador. Analogamente, a medida realizada pelo sensor, que será descrito em breve, será um sinal analógico, a leitura desse sinal também será feita pelo Arduino e então, esse sinal será enviado para o computador, também via USB, para ser processado pelo algoritmo.

Figura 3.4 Arduino UNO usado para interface entre processo e computador.

Os sensor de nível instalado no último tanque infere a altura do nível através da pressão no fundo do reservatório. Para a medição de nível de líquido através da pressão, utiliza-se o princípio de que a pressão só depende da densidade do material e da altura do líquido acima do ponto de referência para um líquido uniforme. O sensor utilizado foi o do fabricante Freescale (modelo MPXM2010GS) que pode operar entre temperaturas de -40 °C até 120°C, e pode medir uma pressão máxima de 75 kPa. O sinal analógico obtido na medida precisa ser multiplicado por 50 para se obter a altura do nível de líquidos em centímetros.

Figura 3.5 Sensor de pressão Freescale modelo MPXM2010GS usado para inferir nível de líquidos.

A bomba usada para levar a água até os tanques opera em uma tensão de 0 a 12 volts e pode operar em tensões de até 16 volts, portanto é necessário um circuito de acionamento que irá elevar a tensão desse sinal de controle e fornecer corrente suficiente para a bomba de água, isso porque o máxima corrente que as portas digitais do Arduino podem oferecer, de forma segura, é 40 mA e a bomba irá precisar de uma corrente maior para operar.

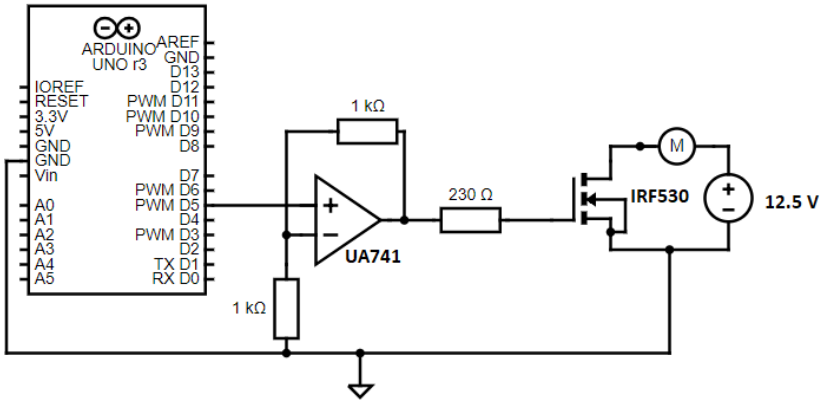


Figura 3.6 Esquema elétrico de circuito de controle PWM.

O circuito usa um amplificador operacional UA741 para elevar a tensão do sinal PWM até aproximadamente 10 Volts, essa sinal estará conectado à porta do transistor MOSFET IRF530N.

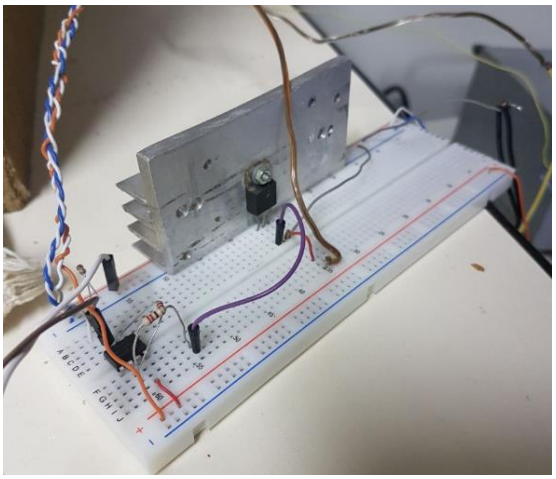


Figura 3.7 Circuito de acionamento PWM, extraído de [10].

A bomba possui zona morta até 2 Volts e saturação a partir de 11 volts. O sinal de controle será um ciclo de trabalho PWM e o transistor funcionará como chave, como um motor elétrico é um filtro passa baixas, a tensão aplicada ao motor será a tensão media durante todo o período do pulso, portanto, com meio ciclo de trabalho a tensão sob o motor será a metade da tensão da fonte, menos a tensão nos terminais do IRF530N (2 volts), ou seja aproximadamente 5,25 volts, e com ciclo de trabalho unitário a tensão deve chegar a aproximadamente 10,5 volts.

4. IMPLEMENTAÇÃO

4.1. APRENDIZADO POR REFORÇO Q-LEARNING

Na implementação do Q-Learning para controle da planta, os estados serão os níveis do tanque 3 e do tanque 4. As ações serão diferentes níveis de ciclos de trabalho do sinal PWM.

Para a construção da tabela Q e implementação do algoritmo, será necessário discretizar as ações e os estados. Cada tanque terá seu nível discretizado em 100 estados e as ações serão divididas em 20 níveis diferentes de ciclos de trabalho PWM. Ou seja, cada estado será determinado pelos níveis dos tanques 3 e 4 e 20 diferentes níveis de ciclo de trabalho PWM serão possíveis. Neste trabalho o sinal PWM poderá variar entre 0,2 e 1, entretanto essa restrição de valores não é necessária para o funcionamento do algoritmo.

A função de recompensa será o valor do módulo do erro entre o nível do tanque 4 e a referência, negativo.

Será usado o algoritmo épisilon-descrescente para que o algoritmo tenha perfil explorador no começo de sua execução e perfil cada vez mais aproveitador conforme o agente conhece as ações com maior retorno, a taxa *épisilon* usada em todos os experimentos será 0,1.

A taxa de amostragem foi determinada de forma empírica, os melhores resultados foram obtidos com um tempo de amostragem de 0,5 segundos.

Além da função de recompensa, um bônus será dado conforme o nível se aproxima da referência, o objetivo desse bônus é dar reforço positivo a ação, já que função recompensa usada é na verdade uma punição – quanto mais distante do objetivo, maior a punição. Diferentes níveis de bônus serão dados quanto maior for a proximidade do nível do quarto tanque ao nível referência.

O algoritmo de controle desenvolvido se encontra no apêndice 2 e foi implementado apenas em simulação.

4.2. O ALGORITMO Q-LEARNING

O Algoritmo Q-Learning pode ser dividido em 5 etapas principais:

Algoritmo 2 – Q-Learning

1. Inicializar tabela Q: Construir uma tabela Q. Existem n colunas, onde n indica o número de ações. Existem m linhas, onde m = número de estados. Inicializar os valores em 0.
2. Escolher ação a ser tomada: A ação é escolhida conforme tabela Q montada
3. Tomar ação escolhida
4. Observar ambiente e medir recompensa
5. Atualizar tabela Q

Os passos 2 a 5 devem ser repetidos até o final do processo.

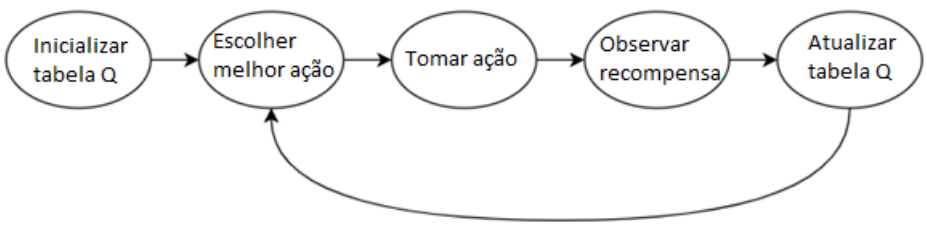


Figura 4.1 Fluxo do algoritmo.

O jogo exemplo apresentado no capítulo 2 explica como a tabela Q evolui com o tempo, uma visualização do algoritmo desenvolvido durante este trabalho seria pouco prática, uma vez que, como veremos, existirão 10000 estados diferentes, e em cada um desses estados, existirão 20 ações diferentes, mas o princípio permanecerá o mesmo.

4.3. APRENDIZADO POR REFORÇO ATOR-CRITICO

O agente é dividido em dois elementos diferentes, o ator e crítico, o agente é o elemento de aprendizado que usará sua experiência para escolher qual ação tomar. O ambiente é o sistema de quatro tanques.

O estado do sistema que compõe a função de valor, como visto pelo algoritmo de aprendizado por reforço, dependerá da referência naquele momento e do nível do quarto taque. O crítico aproxima o par estado e ação, portanto as entradas do crítico serão o estado do sistema e a ação de controle.

Tanto a rede do ator como a rede do crítico serão feitas com uma função de base radial, a gaussiana. A saída da rede do ator será a ação de controle e a saída da rede do crítico será o par estado-ação, também chamado de função de valor.

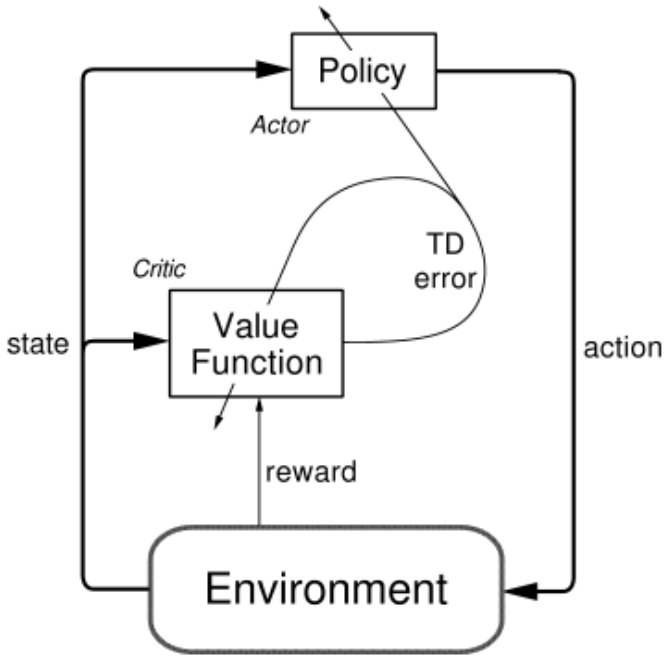


Figura 4.2 Esquema de controlador Ator-Crítico, extraído de [14].

Como o controlador usa o método SARSA, a saída da RBN usada para a função de valor depende do par de ação de estado e é usado pelo crítico para calcular o erro diferença temporal δ_{td}, que será usado para treinar ambas as redes. Usar o δ_{td} como o erro alvo para a atualização significa que δ_{td} é a função de custo a ser minimizada e a função de recompensa será diferença entre o estado desejado e o estado atual.

O controle de nível da planta de quatro tanques foi implementado com o algoritmo de aprendizado por reforço, ator e crítico, usado a seguinte metodologia Matos [10]:

Algoritmo 3 – Aprendizado por reforço ator-crítico

1. Inicializar o estado s_k e todos os parâmetros constantes a serem usados pelo controlador.

2. Inicializar a ação a_k usando a RBN definida para o ator.

3. Inicializar Q_k usando a RBN definida para a função de valor.

4. Tomar ação a_k no modelo para predizer o próximo estado s_{k+1}.
5. Observar a recompensa r_{k+1} (proporcional ao erro).
6. Calcular a ação a_{k+1} usando a RBN definida para o ator.
7. Calcular Q_{k+1} usando a RBN definida para a função de valor.
8. Calcular a diferença temporal δ_{td}.
9. Treinar as redes do ator e do crítico.
10. Realizar nova medida da saída do sistema, calcular nova ação a_k a ser tomada.
11. Repetir passos 4 até o 10.

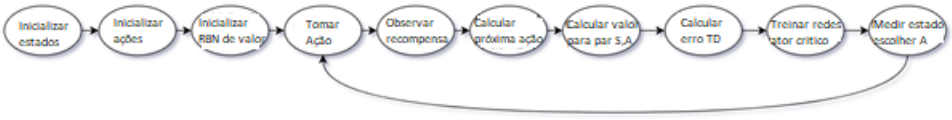


Figura 4.3 Fluxo do programa Aprendizado por reforço Ator-Crítico.

Alguns parâmetros da RBF têm grande peso no desempenho do *Reinforcment Learning,* RL. A taxa de aprendizado, que pode ser melhor analisada em conjunto com a taxa de amostragem, pode ser entendida como o ganho de cada passo do controlador em direção a referência, ou ainda, a largura de cada passo, e a taxa de amostragem determina quantos passos serão dados em um dado intervalo de tempo [10]. Outra observação importante, que é pertinente para sistemas de controle digitais, é que quanto menor for a taxa de amostragem, menor a velocidade de convergência do controlador, e ainda, uma taxa de aprendizado muito pequena fará com que o controlador não responda com a rapidez necessária as oscilações da resposta, e uma taxa de aprendizado muito alta poderá aumentar muito as oscilações do sistema e até mesmo fazer com que ele se torne instável.

O algoritmo implementado pode ser encontrado no apêndice 3.

4.4. IMPLEMENTAÇÃO NARMA-L2

No sistema de nível líquidos estudado neste trabalho, 4 tanques interligados, a saída do último tanque é proporcional a raíz quadrada da diferença entre os níveis do terceiro e do quarto tanque (equações 3.5 e 3.10) e essa é apenas uma das várias não linearidades presentes no sistema.

Para o projeto do controlador NARMA-L2, foi escolhido uma quantidade de 4 atrasos na saída, 2 na entrada, 10 neurônios e tempo de amostragem de 10 segundos.

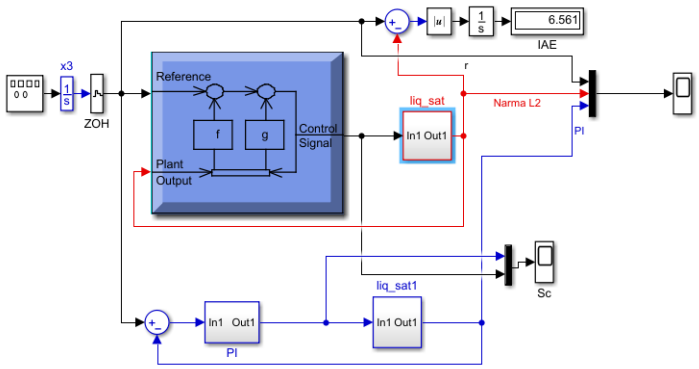


Figura 4.4 Diagrama de blocos Simulink NARMA-L2.

Quatro atrasos se tornam apropriados devido ao sistema ser de quarta ordem, vários testes foram feitos e essa confuguração foi a que demonstrou menor erro quadrático médio.

5. RESULTADOS E DISCUSSÃO

5.1. APRENDIZADO POR REFORÇO Q-LEARNING

A seguir, resultados obtidos em simulações feitas com modelo da planta em diagrama de blocos no software Simulink utilizado em trabalhos anteriores, [7] e [10], que aproxima bem o comportamento do sistema real e que pode ser encontrado no apêndice 1.

Quatro simulações variando a taxa de aprendizado, o bônus e função de recompensa serão apresentados a seguir, outros parâmetros como a taxa de amostragem também possuem grande impacto no desempenho do algoritmo.

Na figura 5.1 é possível ver o comportamento do sistema em resposta ao sinal de controle U escolhido pelo algoritmo a cada instante do tempo de amostragem, o nível y do quarto tanque e a referência ref.

A medida do nível no gráfico está com um ganho 0.02, logo o nível real da referência no gráfico seria 10 centímetros, essa forma de exibição foi escolhida por exibir o sinal de controle e nível de forma conveniente para visualização da evolução das ações do ator com o nível.

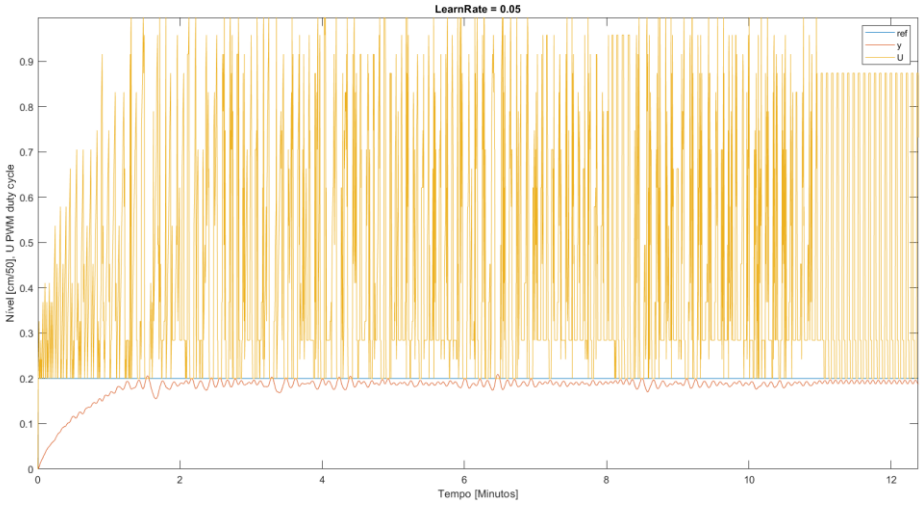


Figura 5.1 Resultado LearnRate = 0,05.

O sistema tem comportamento altamente exploratório nos primeiros minutos, como pode ser observado pelo sinal de controle, entretanto, após 11 minutos de treinamento, o algoritmo passa a usar apenas poucas ações que o levam a recompensa.

O maior erro em relação a referência é 6%.

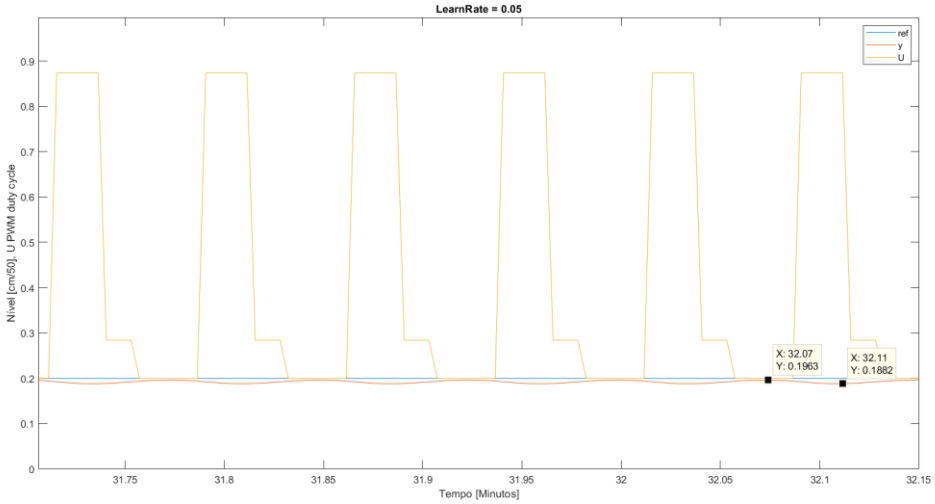


Figura 5.2 Resultado LearnRate = 0,05 aproximado(zoom).

No segundo teste, foi usado taxa de aprendizado de 0,99. O treinamento chegou a um conjunto de ações fixos em menos tempo que no teste anterior, aproximadamente 4,7 minutos. O maior erro após o treinamento foi de 7,3%.

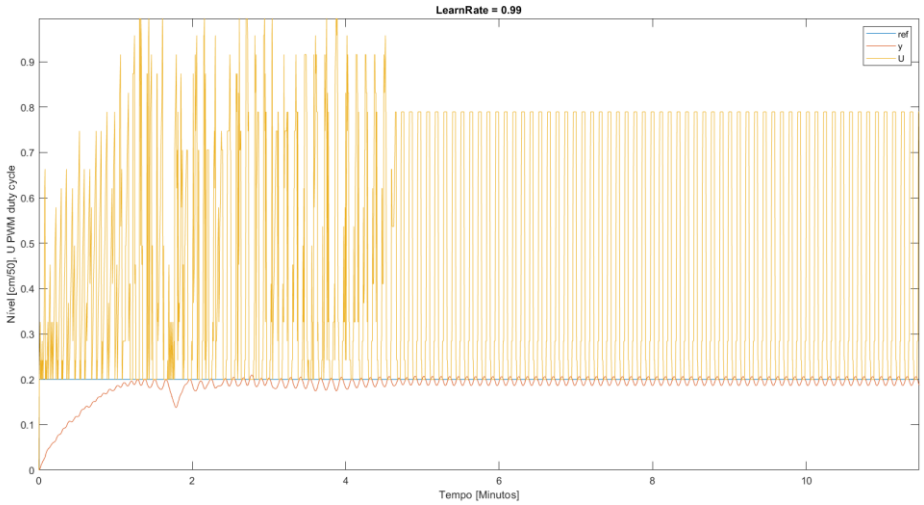


Figura 5.3 Resultado LearnRate = 0,99.

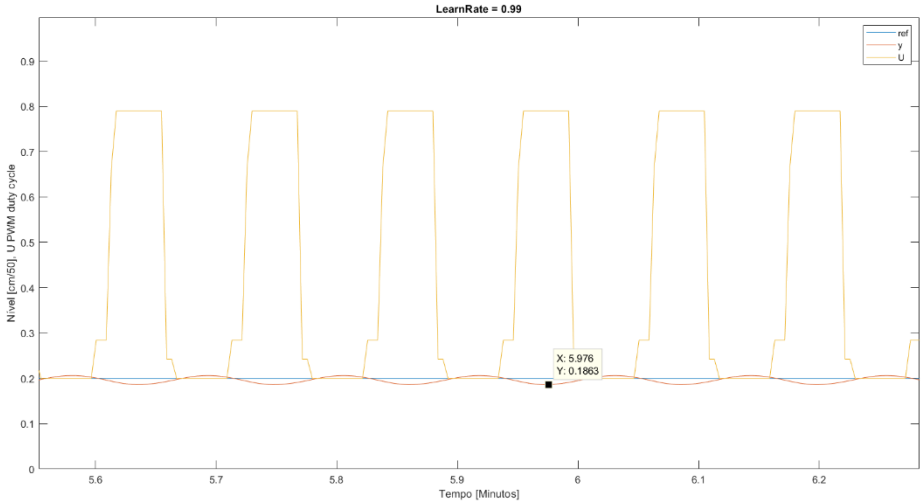


Figura 5.4 Resultado LearnRate = 0,99 aproximado.

No teste a seguir, foi mantida a taxa de aprendizado, entretanto a função de recompensa foi trocada, agora a recompensa é o inverso do erro quadrático entre o nível do quarto tanque e a referência.

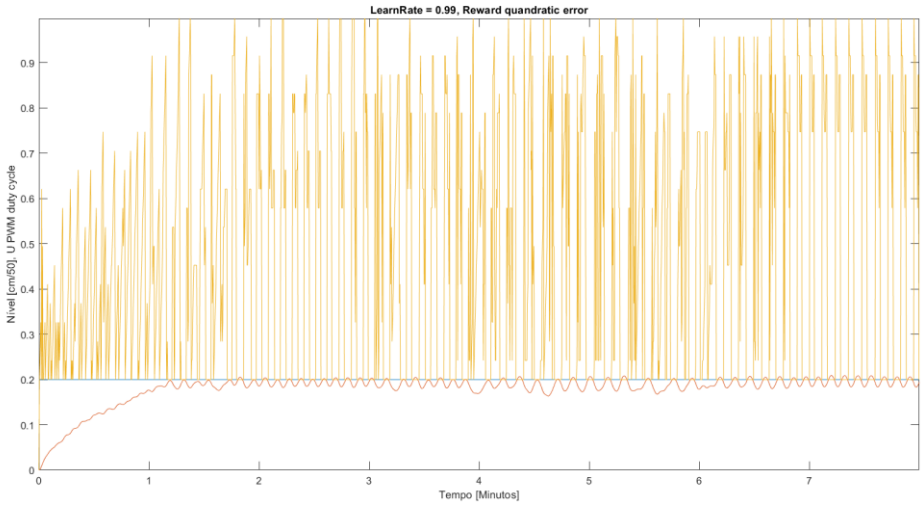


Figura 5.5 Resultado LearnRate = 0,99 e RF quadrática.

É possível notar uma queda no desempenho do algoritmo, tanto o erro máximo quanto o tempo para se parar de explorar ações novas aumentaram, o erro passou a ser 9,17% e a exploração de ações termina em aproximadamente 7 minutos.

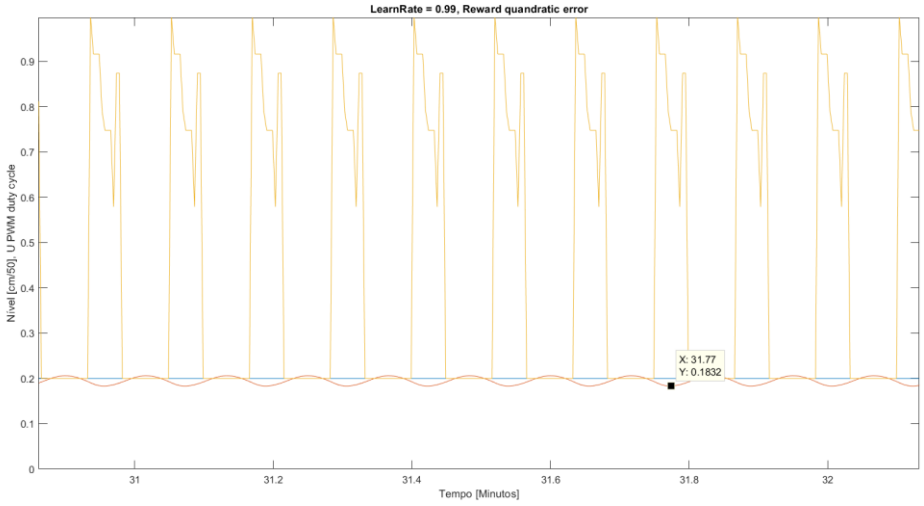


Figura 5.6 Resultado LearnRate = 0,99 e RF quadrática aproximado.

O próximo teste foi feito com taxa de aprendizado de 10, o que é fora da faixa 0-1 normalmente apresentada na literatura.

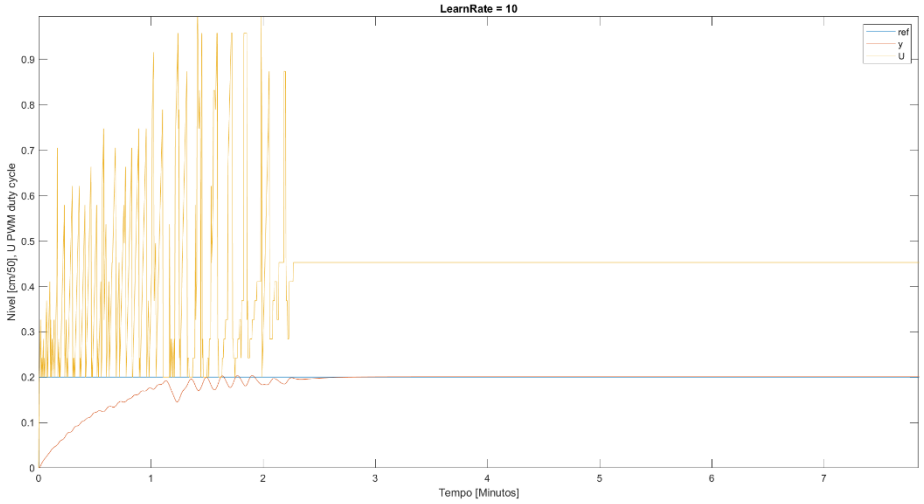


Figura 5.7 Resultado LearnRate = 10.

Como pode-se perceber, após 2 minutos a ação ótima já foi encontrada, e além disso, o algoritmo encontrou uma única ação que segue a referência com um erro mínimo, entretanto, essa alta taxa de aprendizado funciona apenas para uma faixa muito reduzida de valores.

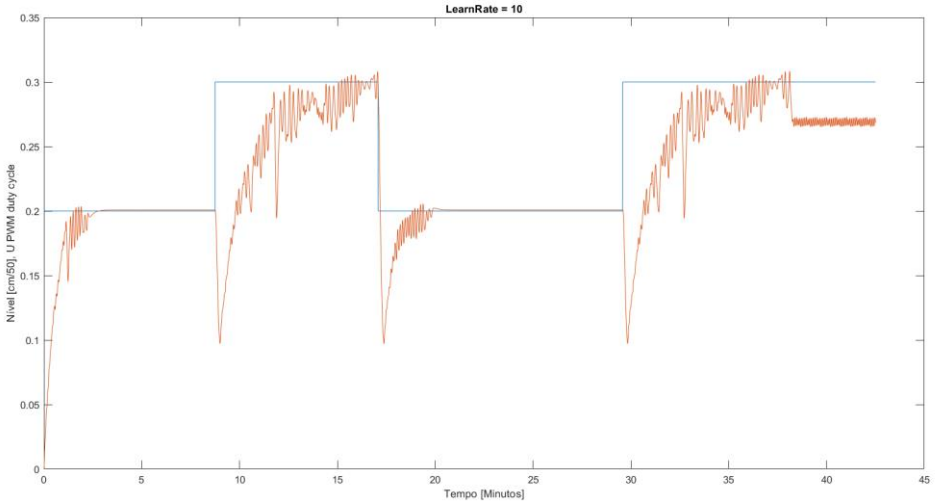


Figura 5.8 Resultado LearnRate = 10 varios pontos de operação.

Para o uso de diferentes referências, como pode ser observado na figura 5.8, deve-se zerar a tabela Q e começar o treinamento do zero.

Na tabela 5.1 pode-se ver o resumo dos resultados apresentados, na primeira coluna encontram-se as variações feitas no parâmetros taxa de aprendizado e função de recompensa.

Tabela 5.1 Resultados para diferentes taxas de aprendizado - Q-Learning.

Parâmetros	Erro máximo*	Tempo TPAO**	RMSE
Lr = 0,05, linear	6%	11 minutos	0,0192
Lr = 0,99, linear	7,3%	4,7 minutos	0,0189
Lr = 0,99, quadrático	9,17%	7 minutos	0,0181
Lr = 10, linear	Próximo à 0%	2,3 minutos	0,0368

*Erro máximo após tempo para encontrar ações ótimas
**Tempo para encontrar ações ótimas

Observando-se os testes apresentados e resultados, é possível concluir que existe uma forte relação entre a taxa de aprendizado e o tempo que o algoritmo leva para chegar ao estado onde ele para de experimentar novas ações por já ter encontrado a ação ou o conjunto de ações ótimos para maximizar a recompensa ganha.

Também é possível observar que a função de recompensa tem grande impacto no desempenho do algoritmo e que o erro linear é superior ao erro quadrático como função de recompensa para a planta de nível de líquidos nas condições estudadas.

O algoritmo se tornou extremamente difícil de definir parâmetros que não o deixavam com comportamento indesejável para taxas de aprendizado maiores que 1, o algoritmo pode seguir apenas uma faixa de referência muito limitada, como pode ser visto na figura 5.8. Entretanto, como foi mostrado no último teste, existem parâmetros que fazem com que o sistema aprenda uma única ação ótima que faz com que a saída do sistema se torne tão próxima quanto possível da referência.

Uma característica evidente desse algoritmo é que os estados e ações precisam ser discretizados para que seja possível compor a tabela Q, isso pode gerar 2 tipos de problemas principais, o primeiro é o erro inerente a qualquer discretização, para usar a tabela Q, será preciso fazer uma interpolação para descobrir qual o estado mais próximo da medida realizada, além disso, a discretização das ações pode causar erro uma vez que a ação que

levaria a referência, pode não estar no conjunto de ações possíveis, por isso, discretizações são fonte de erros. Por outro lado, uma discretização muito fina torna o sistema irrealizável computacionalmente, como exemplo, com a discretização das ações em 100 possíveis níveis de ciclo de trabalho PWM, exigir-se-ia 76 GB de memória RAM para rodar o algoritmo, mantendo a quantidade de estados constante, e essa quantidade de memória RAM não é encontrada em computadores de mesa normalmente.

5.2. RESULTADOS NARMA-L2

Para efeito de comparação, um controlador proporcional integral (Ganho do canal proporcional Kp = 20, Ganho do canal integral Ki = 2) e NARMA-L2, foram implementados.

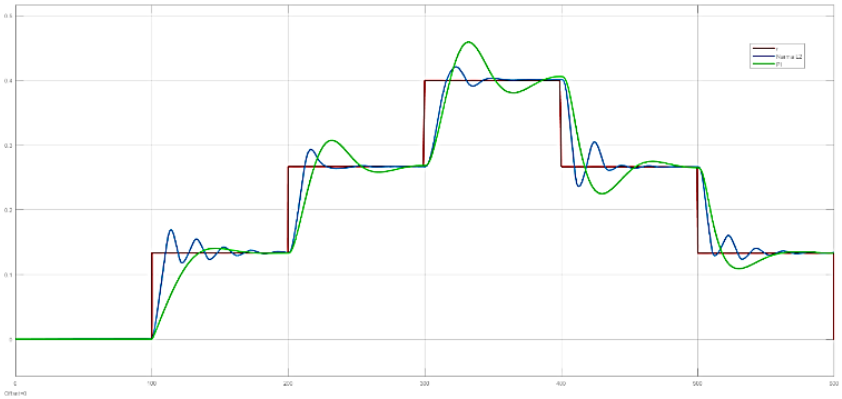


Figura 5.9 Comparativo NARMA-L2 e PI.

Como pode ser visto pela simulação, o controlador NARMA-L2(em azul) possui desempenho superior ao PI(em verde), como o sistema não é linear, têm-se diferentes respostas para cada referência, mesmo sendo a proposta do controlador NARMA-L2 de cancelar as não linearidades do sistema, entretanto, o maior tempo de assentamento para o sistema com o NARMA-L2 foi de 55 segundos, enquanto que para o PI foi de 150 segundos.

Tabela 5.2 Resumo de desempenho dos controladores NARMA-L2 e PI

Controlador	Sobrepasso	Tempo acomodação
NARMA-L2	7%	55 segundos
PI	15%	150 segundos

5.3. APRENDIZADO POR REFORÇO ATOR-CRITICO

Inicializando-se pesos e centros das redes para o ator e para o crítico de forma aleatória, com taxa de aprendizado 0,05 e tempo de amostragem de 0,5 segundos, usando os passos citados no algoritmo 3, obteve-se:

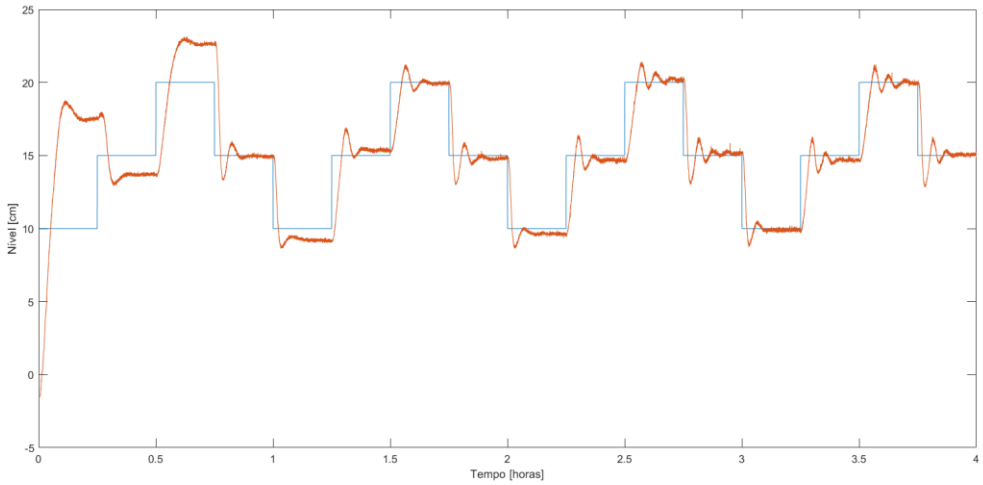


Figura 5.10 Resposta do sistema T=0,5 LR=0,05.

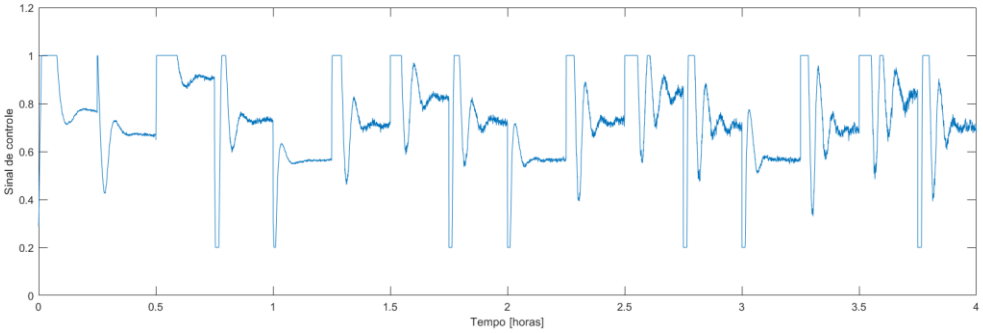


Figura 5.11 Sinal de controle.

O algoritmo possui saturação na saída, o sinal de saída, PWM, tem ciclo de trabalho que varia de 0,2 a 1, portanto, qualquer ação do ator menor que 0,2 de saída das redes será cortada devido a zona morta no motor, e qualquer sinal superior a 1 será cortado devido a característica do próprio sinal PWM, é importante ter em mente que a ação do ator será traduzida em um ciclo de trabalho PWM e esse sinal será usado para excitar o motor-bomba.

Uma forma de lidar com saturações comum, é o uso da função sigmoide, que será colocada na saída da RBN do ator.

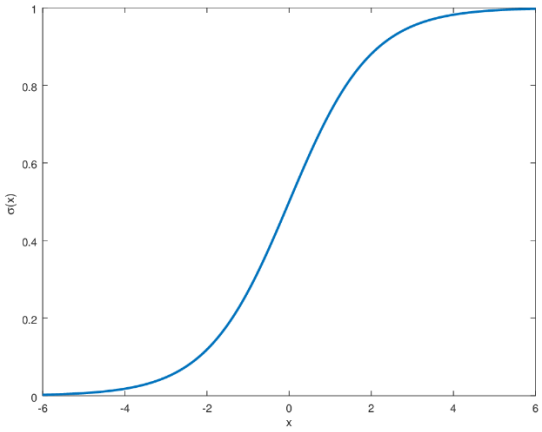


Figura 5.12 Sigmoide usada para eliminar saturação.

A função sigmoide é muito adequada para o uso no algoritmo, uma vez que, como explicado anteriormente, o sinal de controle será um sinal PWM, que varia de 0 a 1.

Com a ação do ator passando por uma função sigmoide ao invés de saturação, obteve-se a seguinte resposta na planta:

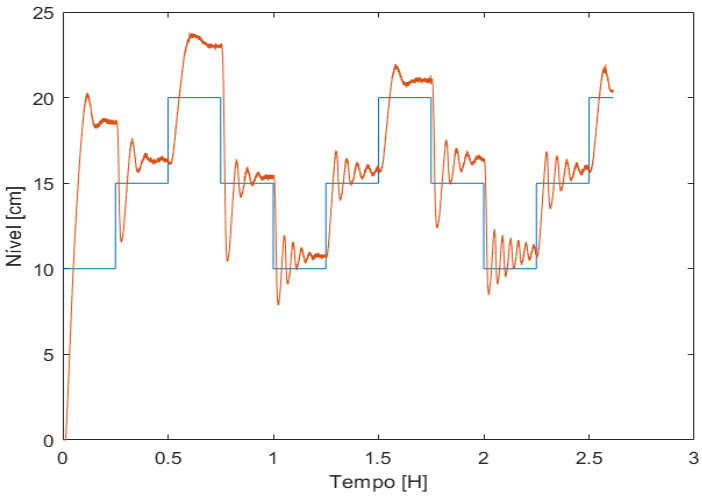
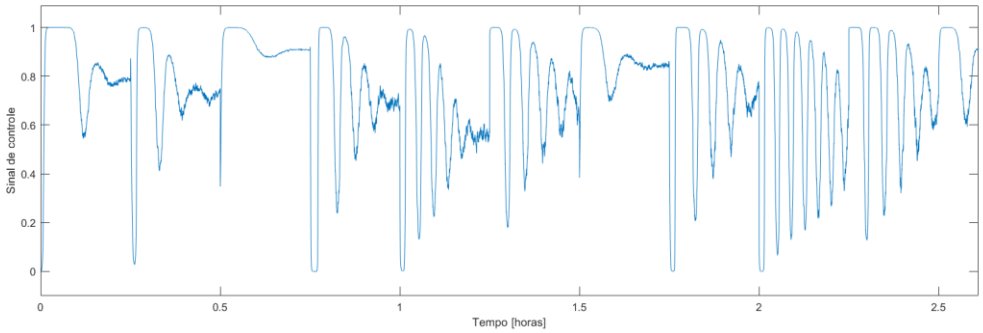


Figura 5.13 Resposta do sistema a eliminação de saturação.



5.14 Sinal de controle.

Como pode ser visto no resultado, houve uma piora na resposta do sistema, se comparado ao resultado anterior, após a variação de parâmetros na sigmoide, os resultados foram consistentemente piores, portanto não convém eliminar saturação nessas condições.

Deseja-se conhecer como o algoritmo se comportaria em algumas condições, deseja-se saber se as redes treinadas para o ator e para o crítico terão boa capacidade de generalização, ou seja, se essas redes terão capacidade de gerar sinais de controle que causarão um erro tolerável mesmo após a parada de atualização dessas redes.

Outra característica que desejava-se conhecer, era a capacidade do algoritmo de lidar com variações no ambiente, ou seja, se o algoritmo seria capaz de se adaptar às mudanças ocorridas na planta. Essa característica é muito desejável em sistemas onde há evolução dos parâmetros com o tempo. Esse tipo de comportamento pode ser visto em alguns sistemas, em foguetes, por exemplo, com o passar do tempo o combustível vai sendo consumido, e por isso o empuxo necessário para causar a mesma aceleração muda com o tempo.

Na planta descrita no capítulo 3, esse comportamento pode ser facilmente estudado variando-se as posições das gavetas no engaste, com isso pode-se obter diferentes vazões entre 2 tanques conservando-se os mesmos estados para os outros tanques.

Na simulação abaixo feita com o software Simulink, a atualização das redes do ator e do crítico param após 2 horas, nesse momento a referência vigente é 10 centímetros e em 4 momentos distintos as posições das válvulas são mudadas, em aproximadamente 40 minutos, a posição da válvula entre os tanques 3 e 4 é mudada do engaste 2 para o 3, como a abertura fica maior, uma maior vazão começa a fluir do tanque 3 para o tanque 4 e o nível deste começa a subir.

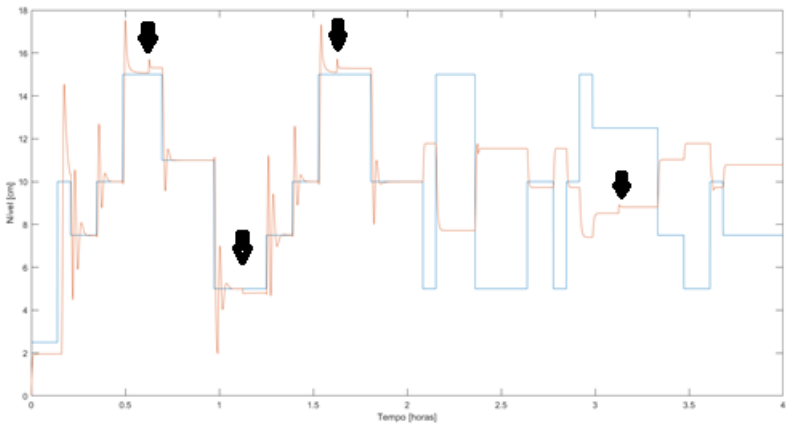


Figura 5.15 Treinamento para em 2 horas, setas indicam mudança de posição das válvulas.

Essas permutações na posição da válvula ocorrem em outros 3 momentos e estão marcados por uma seta preta na figura 5.13. Após duas horas, a atualização das redes do ator e do crítico param, e como pode ser visto, o controlador não consegue fazer com que o sistema siga referencias além da última em que foi treinado, esse fenômeno pode ser consequência do que é conhecido como *Over Fitting*, em que as redes perdem a capacidade de generalização devido a qualidade dos dados apresentados a elas durante o treinamento.

O sistema tem constante de tempo alta, por isso as redes são treinadas durante muito tempo em cada ponto de referência, e isso pode estar causando baixa capacidade de generalização dessas redes. Na simulação é possível observar que quando as referências mudam, as redes geram um sinal de saída que causa ao sistema chegar a um nível diferente da referência, ou seja, as redes perderam a capacidade controlar o sistema nesses pontos. Entretanto, uma vez que a referência volte a ser próxima aos 10 centímetros, o controlador volta a ser capaz de fazer o sistema seguir a referência.

Uma técnica usada para melhorar a capacidade de generalização de redes neurais comumente utilizada quando há *over fitting*, é chamada de *data augumentation*, essa técnica é muito utilizada nos campos de *Deep Learning*, *Machine Learning e Data Science*, e seu objetivo é melhorar os dados que são apresentados para as redes, uma forma de fazer isso no contexto desse trabalho, é parar o treinamento das redes quando a saída do sistema estiver próxima a referência, ou seja, realizar o treinamento apenas quando o erro for maior

que uma dada quantidade, esse procedimento foi adotado e os resultados das simulações podem ser vistos na figura 5.16.

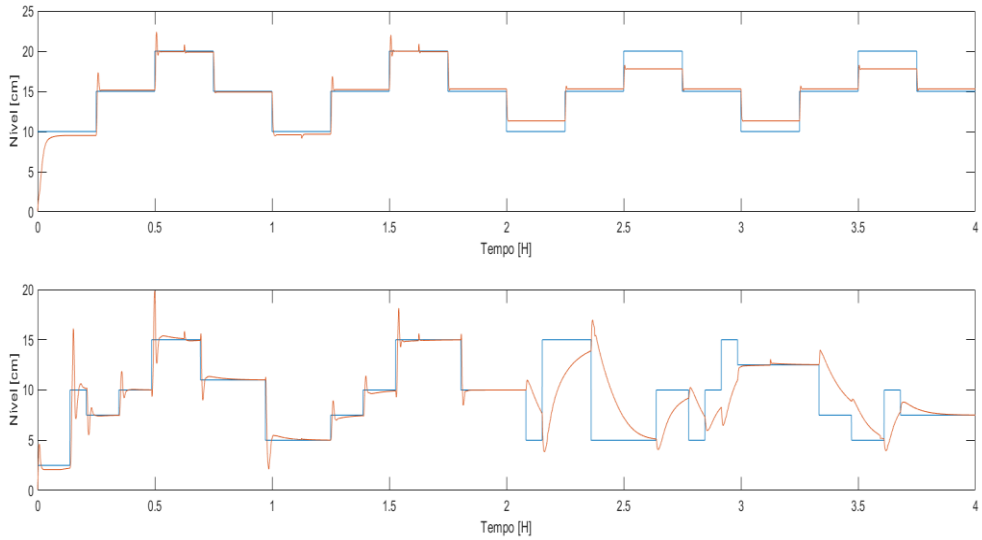


Figura 5.16 Resposta do sistema, na figura superior, com canal integral, na figura inferior, sem PI.

Além da atualização das redes ocorrer apenas quando houver um erro substancial, foi adicionado um canal PI ao controlador para zerar o erro causado em uma das simulações, o resultado pode ser visto no segundo gráfico da figura 5.16.

O canal PI não pode ter um ganho alto, porque ele irá interferir no treinamento das redes, além disso, será adicionado um polo ao sistema, se visto no domínio de Laplace, isso pode trazer complicações quanto a estabilidade do sistema, o *Reinforcement Learning* pode ser visto como um controlador proporcional com ganho variável, por isso, o PI aumenta as chances de o sistema entrar em uma região de instabilidade no LGR.

Nos testes feitos, para ganhos superiores a 0,5 no canal integral com o RL, o sistema fica instável. Na simulação o ganho usado para o canal integral foi 0,1, o que foi o bastante para zerar o erro lentamente.

Ao final das duas horas de treinamento, quando o treinamento acaba, é nítido a perca de desempenho do controlador sem o treinamento, o sistema tende a seguir a referência tão lentamente que o controlador se torna impraticável.

Nos resultados do primeiro gráfico da figura 5.16, sem o canal PI e sem atualização das redes quando o sistema está seguindo a referência, é possível notar uma melhora na resposta com relação a simulação onde o treinamento é feito continuamente durante as primeiras 2 horas, figura 5.13. Essa pequena melhora é provavelmente devido ao *data augumentation*, mas não foi o bastante para obter-se uma resposta desejável.

Outro problema que pode estar causando a dificuldade de generalização das redes do ator e do crítico, é o fato de que quando sistema começa a seguir a referência, a única informação que a rede pode identificar é o ganho da planta, ou seja, a única característica visível do sistema no estado estacionário, é o ganho. Uma forma de contornar esse problema, é adicionar uma onda senoidal com amplitude baixa em relação à referência, dessa forma, o sistema estará sempre oscilando em pequenas amplitudes e suas características fundamentais ficam visíveis para o controlador, os resultados podem ser visto na figura 5.17.

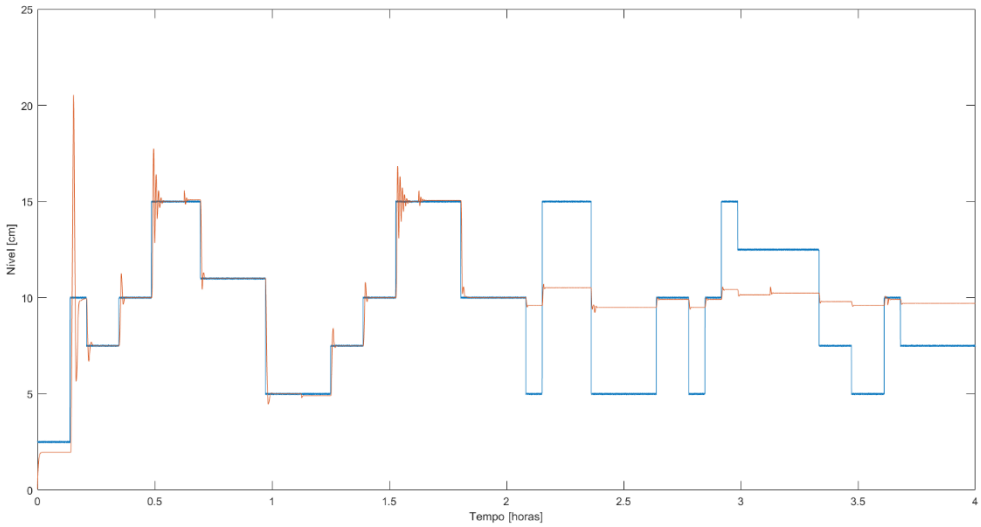


Figura 5.17 Treinamento para em 2 horas, há mudança de posição das válvulas.

A onda senoidal adicionada a referência na simulação acima, tem amplitude de 0,05 centímetros, é visível que a resposta do sistema passou a oscilar mais durante o treinamento, é possível notar que apesar de a rede prever corretamente o sentido em que a ação de controle deve ser tomada, ela falha em acertar o ganho.

Como dito anteriormente, o PI tem o potencial de zerar o erro estacionário em sistemas controlados por algoritmos de aprendizado por reforço, entretanto, efeitos indesejáveis como perca de estabilidade podem ocorrer, para ilustrar isso, considere o diagrama de lugar geométrico das raízes feito com a identificação da função de transferência linearizada da planta em torno do ponto de operação 10 centímetros (h4).

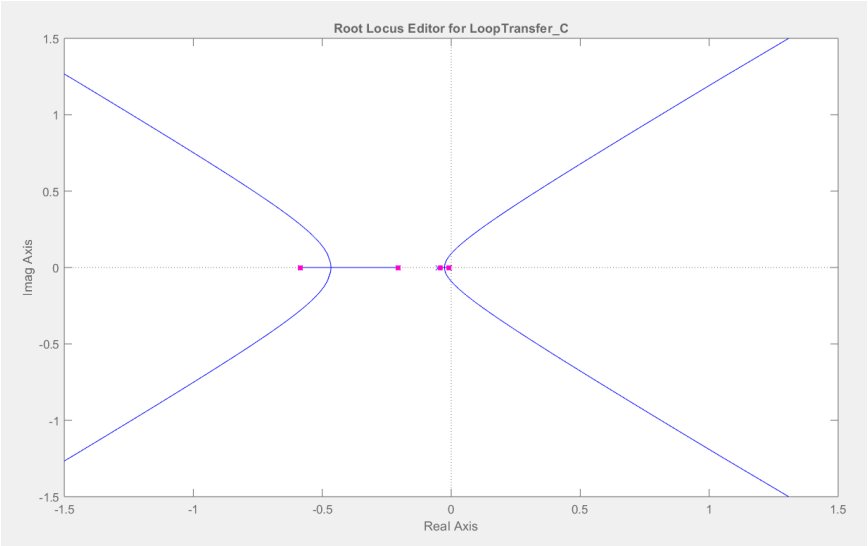


Figura 5.18 LGR sem PI.

O sistema ficará instável com um ganho superior a 53.3. Entretanto, após adicionar o integrador o LGR como o da figura 5.17.

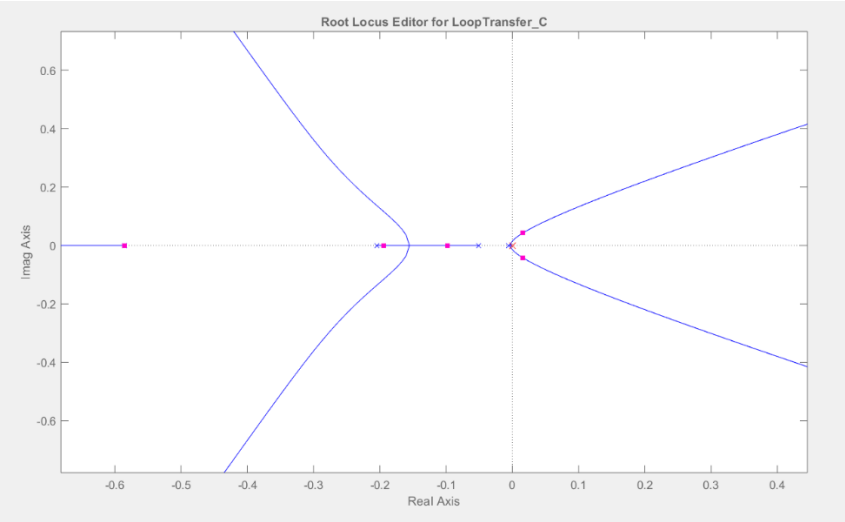


Figura 5.19 LGR após adição de PI.

O sistema fica instável com ganhos superiores a 0,055 , ou seja, a faixa de estabilidade do sistema ficou quase 100 vezes menor, portanto, há de se ter cuidado ao usar integradores para zerar o erro junto ao uso de algoritmos de aprendizado por reforço para controle de sistemas.

Como o algoritmo evolui com o tempo, o ganho relacionado às suas ações deve mudar a cada iteração e como a região de estabilidade estará reduzida, será mais fácil de o sistema se tornar instável.

5.4. PROPOSTA DE MELHORIA PARA O ALGORITMO DE APRENDIZADO POR REFORÇO ATOR-CRÍTICO

5.4.1. AUMENTAR AS REDES NEURAIS ARTIFICIAS DO ATOR E DO CRÍTICO

A rede de base radial do crítico deve aproximar a função de valor, a função valor deve retornar um valor para cada par estado-ação do sistema, logo, o papel do crítico é análogo ao papel da tabela Q no algoritmo Q-Learning apresentado.

No Q-Learning a ação com maior valor para o estado atual do sistema na tabela Q é escolhida para ser tomada, esse papel é desempenhado pelo crítico no algoritmo de aprendizado por reforço ator-crítico, os 2 algoritmos possuem muito em comum.

No algoritmo desenvolvido por Matos [10], implementou como entradas para a rede do crítico o sinal de referência, a medida do nível e o sinal de controle. Para a entrada da rede do ator, apenas os sinais de referência e o nível no quarto tanque. Entretanto, como foi mostrado na descrição da planta, o nível do quarto tanque depende do nível do terceiro tanque, fato que foi utilizado para controle com o algoritmo *Q-Learning*. Então, ao se escolher a melhor ação para atuar no nível do quarto taque, deve-se também considerar qual o nível do terceiro tanque, entretanto, nível do terceiro tanque também depende do nível do segundo e do quarto tanque, e por fim, o nível do segundo tanque depende do nível primeiro e do terceiro, algo natural, uma vez que todos os tanques estão conectados.

Portanto, a melhor escolha de ação para controlar o nível do quarto tanque deverá considerar o nível de todos os 4 tanques a cada instante, com isso o ator poderia tomar uma decisão “mais informada”, no controle com *Q-Learning,* apenas os níveis do quarto e do terceiro tanque são considerados, isso porque a tabela Q ficaria 10000 vezes maior ao se considerar o estado dos 4 tanques, tornando o algoritmo muito exigente de memória RAM, problema que não existe com o aprendizado por reforço ator-crítico pelo uso de RNA para aproximar funções.

Logo, espera-se que seja possível melhorar a estimação do par estado-ação se os quatro níveis forem considerados, e também espera-se que o ator escolha uma melhor ação para controlar o nível do quarto tanque se considerar o nível dos quatro tanques. Para implementar o que foi discutido, deve-se aumentar as redes do ator e do crítico para comportar as novas entradas.

O resultado da simulação com as mudanças descritas pode ser visto na figura 5.18.

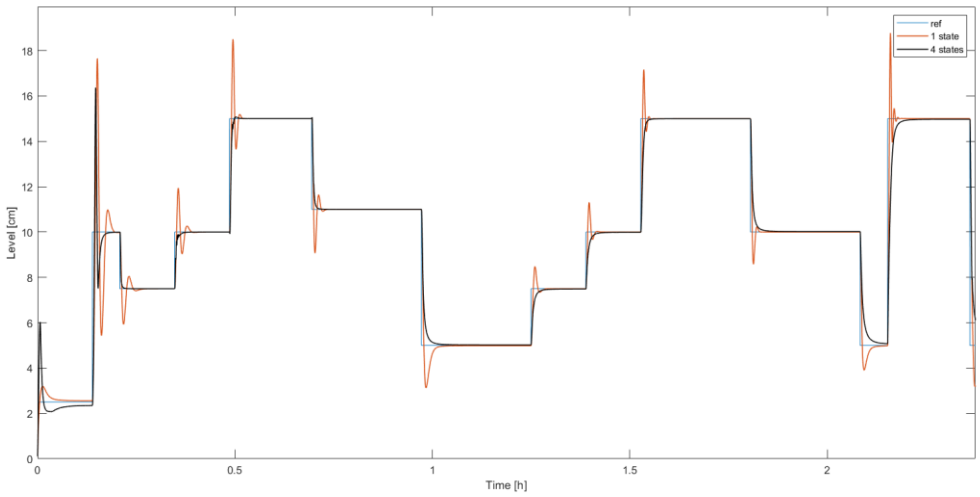


Figura 5.20 Respostas do controlador Ator-Crítico observando 4 estados(preto) e apenas 1 estado(laranja).

Como pode ser observado na figura 5.18 e nos resultados apresentados na tabela 5.3, houve uma melhora na resposta durante o regime transiente do sistema, o algoritmo foi mais

rápido para seguir a referência, isso pode ser consequência da maior quantidade de informação que é apresentada a cada iteração para o treinamento do ator e do crítico.

Outra grande diferença está no sobre-passo que ocorre a cada mudança de referência, após 1 hora e meia de treinamento o sobrepasso na nova referência é 20% para o controlador com 1 estado e próximo a zero para o controlador com 4 estados.

Tabela 5.3 Comparação para algoritmos com 1 e 4 estados.

Estados observados	Sobrepasso*	Tempo acomodação	RMSE
1 estado(H4)	20%	41 segundos	0,0149
4 estados	0%	45 segundos	0,0148

*sobrepasso após 1 hora e meia de treinamento

Além disso, o algoritmo é capaz de controlar a planta de nível em condições mais severas, como exemplo, ao se apertar muito a válvula entre o primeiro e o segundo tanques, ou seja, colocar a gaveta no primeiro engaste, o sistema tendia a se tornar oscilatório observando-se apenas 1 estado, como pode ser visto na figura 5.19.

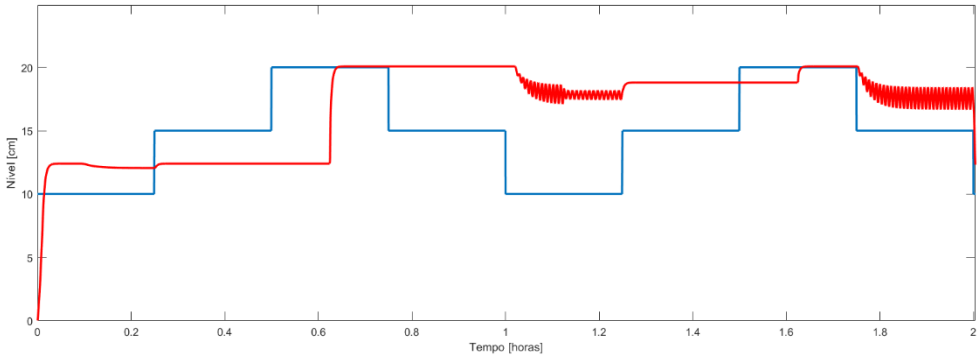


Figura 5.21 Resposta do sistema com válvulas em posição baixa(apertadas) para RL observando um estado.

Como pode ser observado, com apenas um estado o algoritmo é incapaz de fazer o controle da planta, entretanto, observando os 4 estados, o algoritmo é capaz de fazer o controle do nível com bom desempenho.

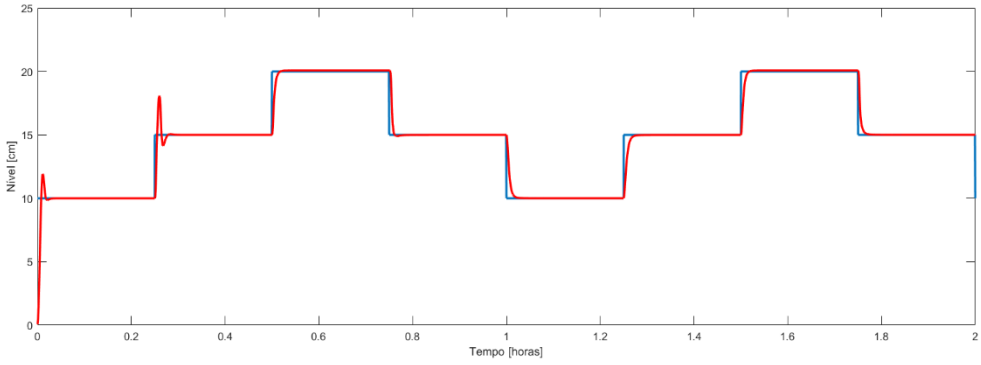


Figura 5.22 Resposta do sistema com válvulas em posição baixa(apertadas) para RL com 4 estados.

Oliveira [7] Implementou controladores da teoria de controle clássica para controle da planta de nível de líquidos estudada neste trabalho, entre eles, Controlador proporcional integral, controlador no espaço de estados, e controle em espaço de estados com integrador. Os melhores resultados foram obtidos com controladores em espaço de estados e sua variante com integrador.

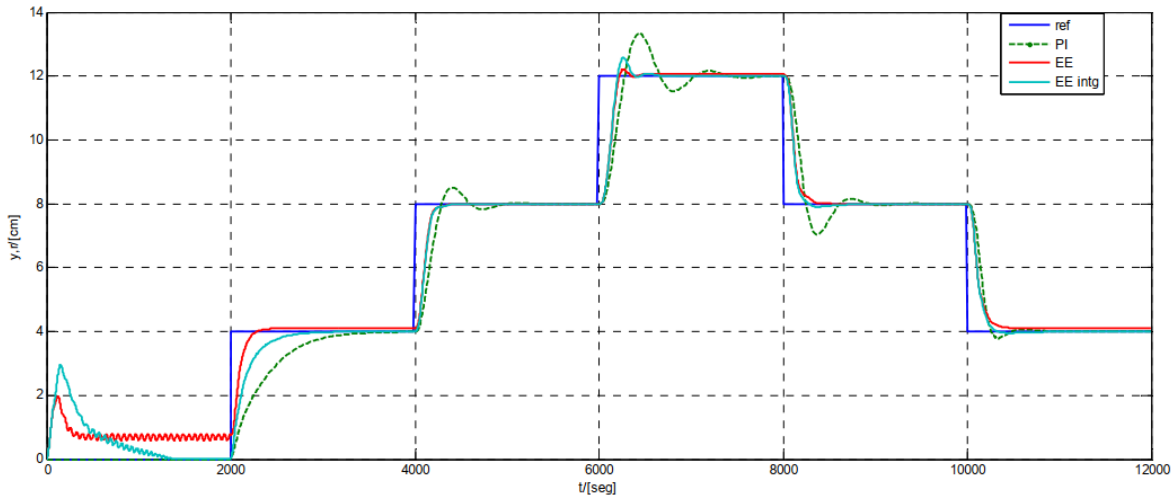


Figura 5.23 Desempenho de controladores da teoria clássica no controle da planta de nível de líquidos, extraído de [7].

Um controlador em espaço de estados pode considerar o nível de todos os 4 tanques para gerar o sinal de controle e por isso tem desempenho superior a um PI que considera apenas o nível do tanque a ser controlado, por isso, espera-se um melhor desempenho para estes controladores, o mesmo foi observado com os controladores de aprendizado por reforço ator-crítico.

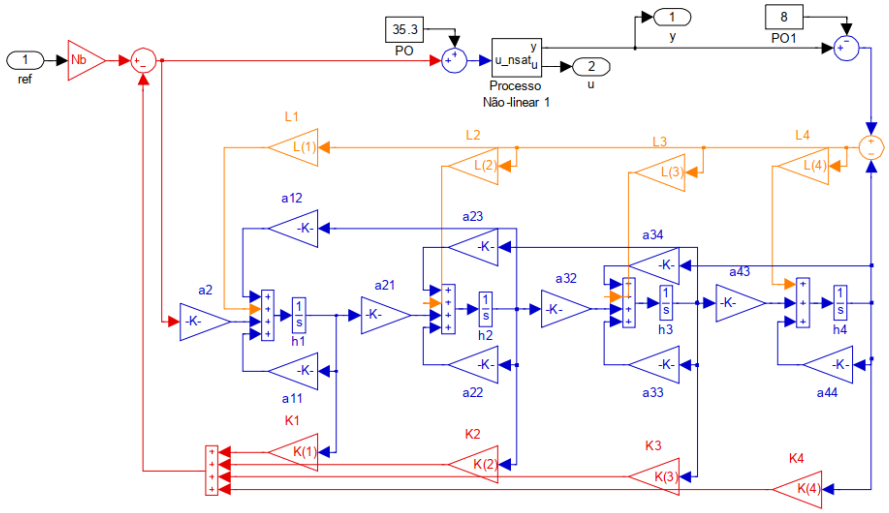


Figura 5.24 Diagrama de blocos de controlador em espaço de estados, observador de estados é usado para estimar nível nos tanques, extraído de [7].

Para avaliar o efeito da taxa de aprendizado após as mudanças feitas nas redes do ator e do crítico, assim como foi feito com o algoritmo Q-Learning, várias simulações foram feitas, nessas simulações, as posições das válvulas permanecem sempre as mesmas, o único parâmetro variado será a taxa de aprendizado.

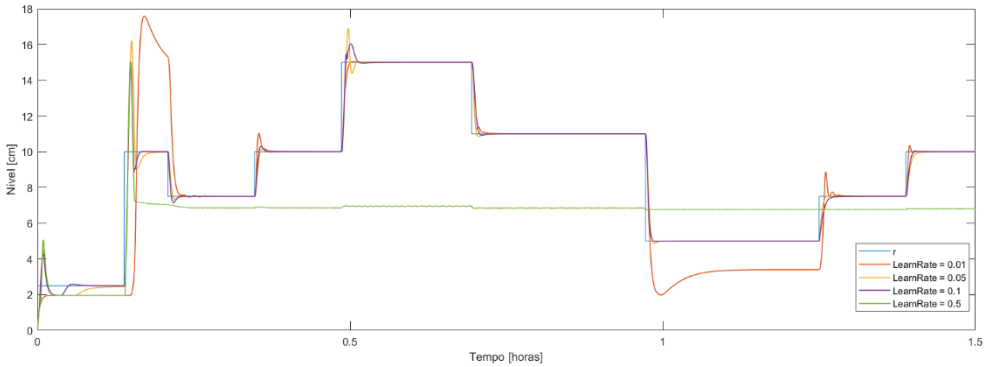


Figura 5.25 Comportamento do algoritmo para diferentes taxas de aprendizado.

Os resultados resumidos são apresentados na tabela 5.4.

Tabela 5.4 RMSE para diferentes taxas de aprendizado RL observando 4 estados

Taxa de aprendizado	RMSE
0,01	0,0354
0,05	0,0144
0,1	0,0133
0,5	0,0777

Com a tabela 5.4 é possível concluir que a melhor escolha para taxa de aprendizado para reduzir o erro quadrático médio é a taxa 0,1, entretanto, desempenho muito próximo a esse é obtido usando uma taxa de aprendizado de 0,05. Com a taxa de aprendizado 0,5 o algoritmo é incapaz de seguir a referência, a taxa de aprendizado é muito alta para o controlador nessas condições, logo o algoritmo encontra um mínimo local na função de recompensa e o treinamento para de ter proveitoso. Para uma taxa de aprendizado de 0,01, o algoritmo falha em seguir a referência em alguns pontos. Com uma taxa de aprendizado tão baixa, o algoritmo tem dificuldades em atualizar as redes do ator e do crítico a tempo.

5.4.2. A TAXA DE DECAIMENTO DA TAXA DE APRENDIZADO EM SISTEMAS VARIANTES COM O TEMPO

Outro ponto que pode ser melhorado no algoritmo de aprendizado por reforço ator-crítico para o controle da planta de nível de líquidos estudada, é o decaimento da taxa de aprendizado. Esse decaimento pressupõe que as melhores ações para cada par estado-ação serão encontrados com o passar do tempo, e por isso, a taxa de aprendizado poderia ser decrementada com o tempo, essa estratégia poderia mitigar problemas como *over fitting*, entretanto, para sistemas variantes no tempo essa abordagem pode trazer prejuízos.

Na figura 5.13 pode ser visto que ao se trocar a posição das gavetas no engaste, o algoritmo reage no sentido de seguir a referência novamente, entretanto, fica evidente que haverá erro estacionário, esse erro acorre devido a incapacidade do algoritmo de treinar novamente as redes do ator e do crítico para se adaptarem ao novo ambiente.

A taxa de aprendizado para o ator e para o crítico é determinada na primeira iteração, deve ser um valor entre 0 e 1. Após cada iteração a taxa de aprendizado recebe um pequeno desconto e toda vez que a referência muda, a taxa de aprendizado é restaurada. Logo, se o sistema tiver uma mudança brusca em seus parâmetros após algum tempo, a taxa de aprendizado poderá estar baixa e o sistema não terá capacidade de treinar as redes do ator e do crítico para voltar a seguir a referência.

A seguir são aprensetados os resultados de simulações com diferentes valores de taxa de decaimento da taxa de aprendizado, que é o parâmetro que determina qual o tamanho do decremento da taxa de aprendizado a cada iteração, sendo que uma taxa 0 determina que a

taxa de aprendizado vai a zero na primeira iteração, e uma taxa 1 determina que não haverá decaimento da taxa de aprendizado.

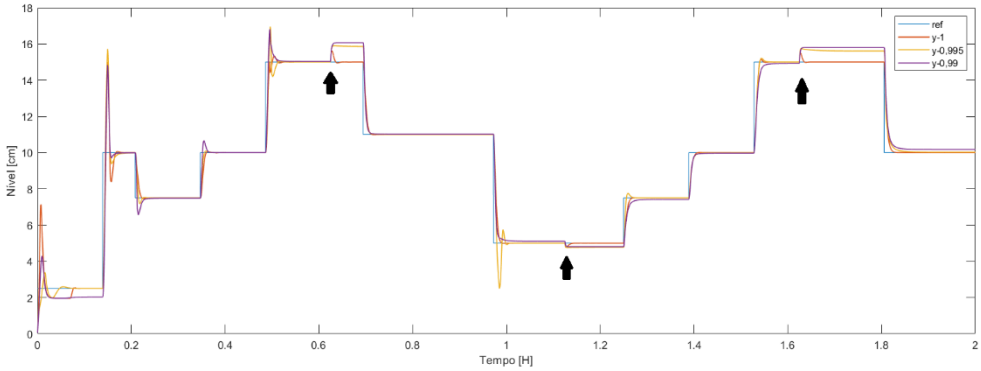


Figura 5.24 Resposta do sistema para diferentes taxas de decaimento da taxa de aprendizado.

Como pode ser visto na figura 5.24, mantendo-se a taxa de decaimento em 0,995, o algoritmo com 4 estados possui desempenho inferior para lidar com variações na planta em relação ao algoritmo com apenas um estado (figura 5.13). Isso pode ser consequência do maior tamanho das redes do ator e do crítico para o algoritmo de 4 estados. Entretanto, para a taxa de decaimento 1, ou seja, não há decaimento, o algoritmo com 4 estados possui desempenho satisfatório para lidar com as variações na posição da válvula entre o terceiro e quarto tanques.

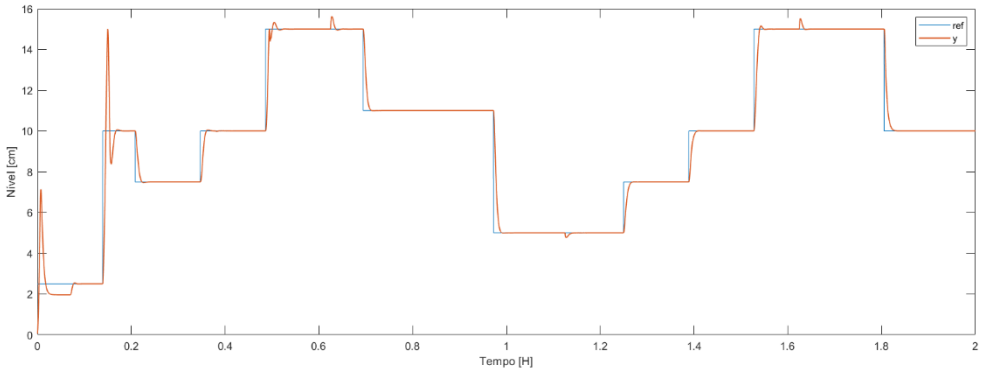


Figura 5.25 Resposta do sistema sem decaimento da taxa de aprendizado.

Portanto, no controle de sistemas onde se espera variações com o tempo, a taxa de decaimento da taxa de aprendizado pode diminuir a capacidade do controlador de lidar com essas variações e seguir a referência. Na figura 5.25 é possível notar que, sem o decaimento da taxa de aprendizado, o algoritmo se torna capaz de seguir a referência mesmo após a mudança na posição das válvulas, isso ocorre porque no momento em que a posição das válvulas é mudada, a taxa de aprendizado continua sendo alta o suficiente para treinar as redes neurais aritificias do ator e do crítico e seguir a referência.

6. CONCLUSÕES E PROPOSTAS PARA TRABALHOS FUTUROS

Como foi apresentado, o controle de sistemas dinâmicos pode ser feito através do algoritmo *Q-Learning* em alguns casos, a depender dos requisitos de projeto de precisão. Para uma taxa de aprendizado de 0,99, o erro máximo foi de 7,1%, e para muitos projetos isso pode ser intolerável, principalmente onde exige-se precisão elevada.

O Q-Learning é comumente apresentado para resolver problemas de encontrar a saída de labirintos evitando obstáculos e jogos com poucas ações e poucos estados possíveis, entretanto, como foi demonstrado neste trabalho, o algoritmo pode ser também usado para controle de um sistema relativamente complexo como a planta de 4 tanques interligados.

Também foi apresentado que o algoritmo de aprendizado por reforço ator-crítico não irá seguir referencias além da qual ele acabou de ser treinado para seguir, caso o treinamento cesse, como ficou evidente nos vários experimentos apresentados. Além disso, o uso de integradores para zerar o erro estacionário mas fará com que o sistema se torne instável mais facilmente.

Na última sessão do trabalho, ficou claro que uma melhoria da resposta do controlador de aprendizado por reforço ator-crítico poderia ser obtida ao se alimentar as redes do ator e do crítico com os estados, ou níveis, de todos os 4 tanques. A situação é análoga entre o desempenho de controladores PI e em espaço de estados. Oliveira [7] mostrou que para o sistema estudado neste trabalho, controladores em espaço de estados tem desempenho superior, isso é consequência de um controlador em espaço de estados poder considerar o nível de todos os 4 tanques para gerar o sinal de controle, em constraste com PI, PD e PID's que consideram apenas o nível do tanque a ser controlado, houve uma melhora considerável no desempenho do controlador e foi possível aumentar a taxa de aprendizado do algoritmo, o que resultou em um menor erro quadrático médio, além disso, o algoritmo foi capaz de controlar o nível da planta em condições mais severas, o que não era possível no algoritmo com apenas 1 estado.

O aumento das redes do ator e do crítico causou um maior erro estacionário ao se mudar a posição das válvulas da planta se comparado ao algoritmo com apenas 1 estado. Esse erro era consequência do decaimento da taxa de aprendizado, parâmetro em RL comum para evitar problemas com *overfitting,* entretanto, em sistemas onde se espera variação com o tempo não convém reduzir a taxa de aprendizado com o tempo.

Tabela 6.1 Comparativo entre principais controladores abordados durante o trabalho.

Controlador	Sobrepasso	Tempo acomodação	RMSE
PI	15%	150 segundos	0,1054
NARMA-L2	7%	55 segundos	0,1039
RL - 1 estado(H4)	20%	45 segundos	0,0149
RL - 4 estados	0%	41 segundos	0,0148

Como pode ser evidenciado através da tabela 6.1 e dos resultados obtidos durante este trabalho, o controlador com melhor desempenho no controle do nível na planta de nível de líquidos estudada, foi o algoritmo de aprendizado por reforço ator-crítico observando 4 estados, sua implementação na planta poderia ser feita usando observadores de estados, com isso seria desnecessário a instalação de novos sensores de nível.

Além disso, com uma taxa de decaimento da taxa de aprendizado unitária, o algoritmo se torna adequado para controle do sistemas quando há mudança na posição das válvulas em tempo de execução, sem erro estacionário.

Como sugestão de trabalhos futuros, seria interessante estudar como o algoritmo de aprendizado por reforço ator-crítico se comportaria em sistemas de controle de várias saídas, como exemplo, uma planta de nível com 2 bombas injetando água em tanques diferentes.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Bauchspiess, A. (2016). “Notas de aula da disciplina Controle Dinâmico”, UnB.

[2] Bauchspiess, A. (2018). “Notas de aula da disciplina Introdução ao controle inteligente numérico”, UnB.

[3] Dreyfus, S. (1973). "The computational solution of optimal control problems with time lag". IEEE Transactions on Automatic Control. 18 (4): 383–385.

[4] Gonçalves, D. V. (2016). Controle Adaptativo De Processo De Nível Utilizando Aprendizado Por Reforço Ator-crítico. Trabalho de Graduação em Engenharia de Controle e Automação, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 61p.

[5] Haykin, S. S. (2009). Neural Networks and Learning Machines. Prentice Hall.

[6] Harnad, Steven (2008). The Annotation Game: On Turing (1950) on Computing, Machinery, and Intelligence. in Epstein, Robert; Peters, Grace: Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer: Springer. pp. 23–66.

[7] Oliveira, J. C. P. (2009). Avaliação de Controle Neural a um Processo de Quatro Tanques Acoplados. Tese de Doutorado em Engenharia Elétrica Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 115p

[8] McCulloch, W. S.; Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics 5(4), 115–133.

[9] Minski, M. L.; Papert, S. A. (1969). Perceptrons: an introduction to computational geometry. MA: MIT Press, Cambridge.

[10] Matos, L. G. (2018). Control and Identification of Fourth Order Fluid Level System Using Neural Networks and Reinforcement Learning. Master’s Thesis. Publicação 690/2017. Departamento de Engenharia Elétrica. Universidade de Brasília. Brasília.

[11] Narendra, K.; Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks 1(1), 4–27.

[12] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65(6), 386–408.

[13] Rumelhart, D .E., Hinton, G. E. and Williams, R.J., “Parallel Distributed Processing.” Cambridge, MA., 1986

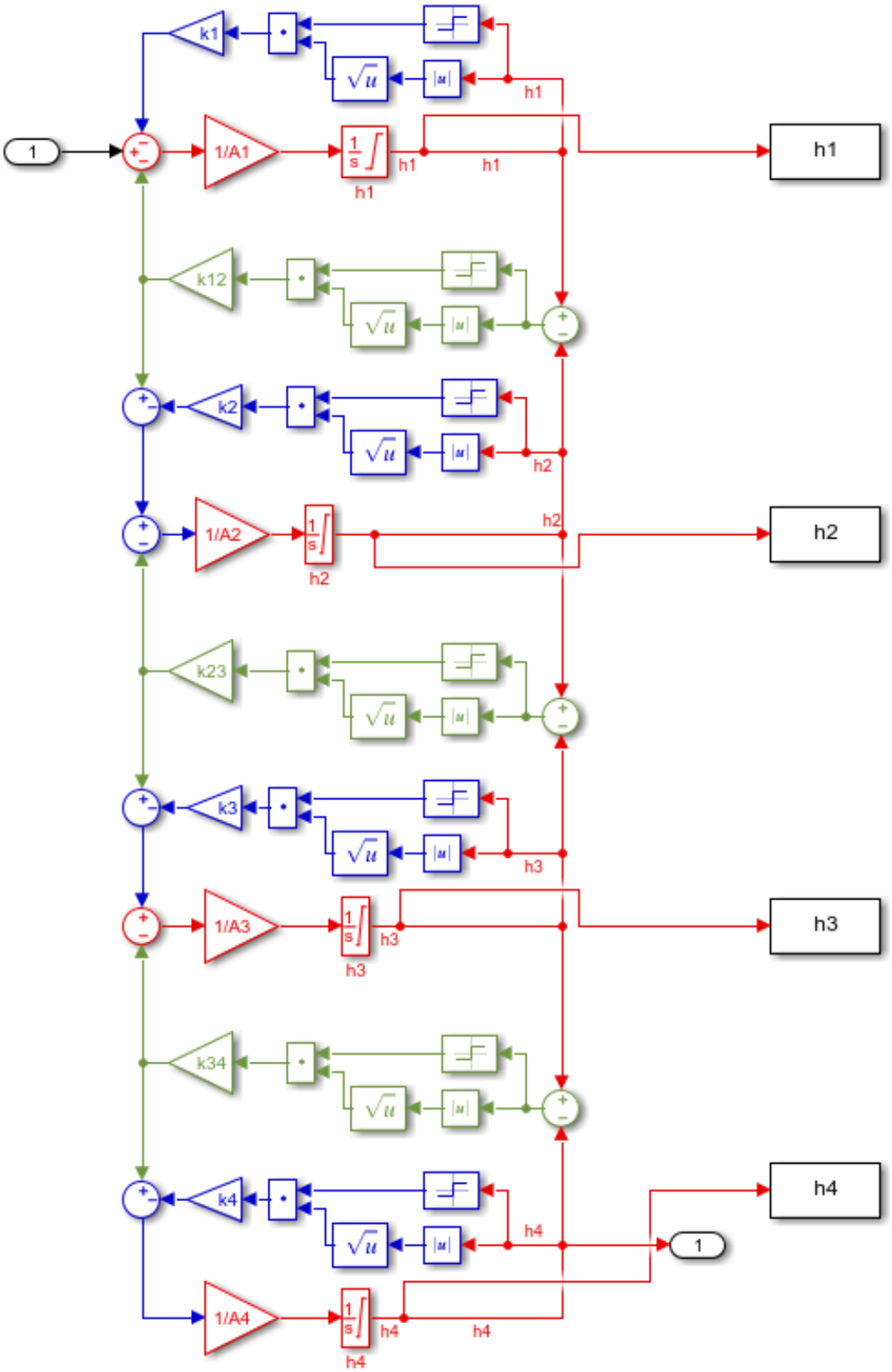
[14] Russell, S.; Norvig, P. (2003). Artificial Intelligence: A Modern Approach 2 ed. [S.l.]: Prentice Hall.

[15] Sutton, R. S.; Barto, A. G. (2017). Reinforcement learning: An introduction. MIT press Cambridge.

[16] Werbos, P. J. (1975). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Harvard University.

[17] M. Mitchell, Tom (1997). Machine Learning. McGraw Hill.

APÊNDICE 1 – DIAGRAMA DE BLOCOS SIMULINK



APÊNDICE 2 – CÓDIGO DE CONTROLE – APRENDIZADO POR REFORÇO Q-LEARNING

```
%% Simulação da planta de 4 tanques - controle com
aprendizado por reforço - Q-Learning
% Autor: Álvaro Queiroz
% Modelo Simulink da planta de Adolfo Bauchspiess
% Parâmetros de simulação Júlio Cézar


clc;
close all;
clear

% Process Parameters
d=6;       % reservoirs depth
A1=d*9.9; % cm2   - transversal section of the reservoirs
A2=d*10;  % cm2
A3=d*10;  % cm2
A4=d*10;  % cm2


% valve positions 4-3-3;
k1=0;
k3= 0;
k12=12.18;

k23=11.89;
%k23=4.6461;

%k34=5.2618;
k34=10.09;

k2=4.313;
k4=4.9667;


% initial conditions
h1=0;
h2=0;
h3=0;
h4=0;

hmax=1;  % reservoirs heigth 50 cm
qmax=60;
qmin=0;


% Sampling rate

T = 0.5;  % seconds
```

```
H4       = zeros(100,1);
entrada = zeros(10201,1);
saida = zeros(10201,1);
%% DEFINE Q-LEARNING PARAMS
% Reward function
rewardFunc = @(x,xdot)(-abs(x-xdot));
% Learning rate
learnRate = 0.99; % How is new value estimate weighted
against the old (0-1). 1 means all new and is ok for no
noise situations.

%%% Exploration vs. exploitation
% Probability of picking random action vs estimated best
action
epsilon = 0.1; % Initial value
epsilonDecay = 0.95; % Decay factor per iteration.

%%% Future vs present value
discount = 0.98; % When assessing the value of a state &
action, how important is the value of the future states?


actions = linspace(0.2,1,20); % actions the controller may
take



%% INITIALIZE REF, REWARDS, STATES AND Q MATRIX

ref(1:2100) = 0.2;
ref(2101:4100) = 0.2;
ref(4101:6100) = 0.2;
ref(6101:8201) = 0.2;
ref(8101:10201) = 0.2;
ref = ref';

% State1 is H4,
x1 = 0.0:0.01:1;
x2 = 0.0:0.01:1;



%Generate a state list
states=zeros(length(x1)*length(x2),2); % 2 Column matrix
of all possible combinations of the discretized state.
index=1;
for j=1:length(x1)
    for k = 1:length(x2)
        states(index,1)=x1(j);
        states(index,2)=x2(k);
```

```
        index=index+1;
    end
end


R = rewardFunc(states(:,1),ref); % cost function
Q = repmat(R,[1,length(actions)]); % Q is length(x1) x
length(x2) x length(actions)
% We should save Q to reset Q table in case of new ref
Q0 = Q;


U = 1;
H4(1) = 0;


for k=2:10201

    %% CHOOSE ACTION, OBSERVE STATE
    if abs(ref(k)-ref(k-1))>0
        Q = Q0;
    end


        % Interpolate the state within our discretization
(ONLY for choosing
        % the action. We do not actually change the state
by doing this!)
    [~,sIdx] = min(sum((states -
repmat([H4(1),h3(end)],[size(states,1),1])).^2,2));

        % Choose an action:
        % EITHER 1) pick the best action according the Q
matrix (EXPLOITATION). OR
        % 2) Pick a random action (EXPLORATION)
    if (rand() > epsilon) % Pick according to the Q-matrix
it's the last episode or we succeed with the
rand()>epsilon check. Fail the check if our action doesn't
succeed (i.e. simulating noise)
        [~,aIdx] = max(Q(sIdx,:)); % Pick the action the Q
matrix thinks is best!
    else
        aIdx = randi(length(actions),1); % Random action!
    end

    U = actions(aIdx);
    %take action, observe output
    sim('modelo_simulink');
    H4(1) = h4(end);
```

```
    ERRO = H4(1)-ref(k);
 %% UPDATE Q-MATRIX

    % As the system get close to ref, it's actions receive
bonus ever
    % increasing
    if abs(ERRO) < 0.01
        bonus = 0.01;
    else
    if abs(ERRO) < 0.008
        bonus = 0.02;
    else
    if abs(ERRO) < 0.005
        bonus = 0.04;
    else
        bonus = -0.01;
    end
    end
    end



    [~,snewIdx] = min(sum((states -
repmat([H4(1),h3(end)],[size(states,1),1])).^2,2)); %
Interpolate again to find the new state the system is
closest to.

    if k < 10201 % On the last iteration, stop learning
and just execute. Otherwise...
        % Update Q
        %Q(sIdx,aIdx) = Q(sIdx,aIdx) + learnRate * (
R(snewIdx) + discount*max(Q(snewIdx,:)) - Q(sIdx,aIdx) +
bonus );
        Q(sIdx,aIdx) = Q(sIdx,aIdx) + learnRate *
(R(snewIdx) + discount*max(Q(snewIdx,:)) - Q(sIdx,aIdx) +
bonus*20 );

    end

    % Decay the odds of picking a random action vs picking
the
    % estimated "best" action. I.e. we're becoming more
confident in
    % our learned Q.
    epsilon = epsilon*epsilonDecay;

    %save states of simulation for the next

    h1=h1(end);
```

```
    h2=h2(end);
    h3=h3(end);
    h4=h4(end);

    %save input output

    entrada(k)= U;
    saida(k)= H4(1);

    %display in command windows ref H4 U

    [ref(k); H4(1); U]


end
```

APÊNDICE 3 – CÓDIGO DE CONTROLE – APRENDIZADO POR REFORÇO ATOR-CRÍTICO 4 ESTADOS

```
%% Simulação da planta de 4 tanques - controle com
aprendizado por reforço
% Modelo Simulink de Adolfo Bauchspiess
% Parâmetros de simulação Júlio Cézar
% código de controle feito a partir de alterações no
código de Lucas
% Guilhem

% Autor: Álvaro Queiroz

clc;
close all;
clear;


% Process Parameters
d=6;       % reservoirs depth
A1=d*9.9; % cm2   - transversal section of the reservoirs
A2=d*10;  % cm2
A3=d*10;  % cm2
A4=d*10;  % cm2


% Valores obtidos da tese de Julio Cezar - posições 4-3-3;
k1=0;
k3= 0;

%k12=12.18;
k12=9.06;
%k12 = 16.18;
k23=11.89;
%23=4.6461;
%k12 = 20.62;

%k34=5.2618;
k34=10.09;
%k34 = 16.65;
k2=4.313;
k4=4.9667;


% condicoes iniciais
h1=0;
h2=0;
h3=0;
h4=0;
```

```
hmax=1;  % reservoirs heigth 50 cm - gain of 0.02
qmax=60;
qmin=0;

% Taxa de Amostagem e Delay

T = 0.5;  %tempo em segundos

delay = round(20/T); %o delay é em número de amostragens
%delay = 0;
%
for i=1:9
    b(i) = (10-i)/10;
end

% dfening RNA - RBF
% Variâncias das funções de ativação das redes neurais.
Globais porque permanecem inalteradas.

num_ent_actor   = 5;
num_ent_critic  = 6;

%Número de Neurons nas camadas escondidas do Actor e do
Critic. repetições
%experimentais me fizeram crer que são necessários poucos
neurons prara o
%funcionamento completo do algoritmo. muitos neurons deixa
o algoritmo
%lento e traz nenhum ganho ou atrapalha o processo.

cam1            = 4^num_ent_actor;
cam2            = 4^num_ent_critic;

% Auxiliares. São usados nas funções "Actor" e "Critic"
para depois atualizar os pesos.
aux             = zeros(cam1,1);
aux2            = zeros(cam2,1);

% Taxads de Aprendizado do Actor, do Critic e dos centros
eta_actor       = 0.1;
% Taxa de Aprendizado dos pesos do Actor
eta_critic      = 0.1;
% Taxa de Aprendizado dos pesos do Critic
eta_mi          = 0.1;
epsilon         = 0.02;
% Equivale a 1 cm
LearnRateDecay = 1;
eta_actor0 = eta_actor;
eta_critic0 = eta_critic;
```

```
% Variâncias das funções de Ativação do Actor e do Critic.
sigma_actor     = 0.2;
sigma_critic    = 0.2;

%% Inicialização/Condições Iniciais

% Pesos Actor e Critic Respectivamente
w = rand(1,cam1)* 1-0.5;
v = rand(1,cam2)*1-0.5;


% Centros
mi_actor = [];
% Centros uniformemente distribuidos
mi_actor(1,:) = repelem(linspace(0,1,64),16);
mi_actor(2,:) = repelem(linspace(0,1,64),16);
mi_actor(3,:) = repelem(linspace(0,1,64),16);
mi_actor(4,:) = repelem(linspace(0,1,64),16);
mi_actor(5,:) = repelem(linspace(0,1,64),16);


mi_critic = [];
mi_critic(1,:) = repmat(linspace(0,1,64),1,64);
mi_critic(2,:) = repmat(linspace(0,1,64),1,64);
mi_critic(3,:) = repmat(linspace(0,1,64),1,64);
mi_critic(4,:) = repmat(linspace(0,1,64),1,64);
mi_critic(5,:) = repmat(linspace(0,1,64),1,64);
mi_critic(6,:) = repmat(linspace(0,1,64),1,64);

 for i=1:num_ent_actor
%Pesos Aleatoriamente distribuidos
    mi_actor(i,:) = rand(1,cam1);
 end

 for i=1:num_ent_critic
    mi_critic(i,:) = rand(1,cam2);
 end

gama = 1;

H4       = zeros(100,1);
% As ultimas 100 saidas do sistema para poder filtrar e
usar no estimador
H4F      = 0;
% As ultimas 100 saidas filtradas do sistema
H4_EST   = zeros(100,1);
% As ultimas 100 saidas estimadas, pelo mesmo motivo da
saida medida.
```

```
D         = 900/T;
% Número de segundos que cada ponto de referência ficará
vigtente  valor original:900/T
ref       = repelem([0.2 0.3 0.4 0.3 0.2 0.3 0.4 0.3 0.2
0.3 0.4 0.3 0.2 0.3 0.4 0.3 0.2 0.3 0.4 0.3 0.2 0.3 0.4
0.3],D);                          % Referências
%ref = idinput([28800,1,3],'prbs',[0 0.005],[0,0.2]);
%ref = refFunc();



%ard       = arduino;
% Objeto Arduino
entrada = zeros(16*D,1);
saida = zeros(16*D,1);
eta_critic_r = zeros(16*D,1);
eta_actor_r = zeros(16*D,1);


%% PASSO 1 - INICIALIZAR O SISTEMA E DEFINIR A0 E S0
%  CALCULAR U1 E Q1

    ERRO1                         = ref(1) - H4F;
    X_actor                       =
[ref(1);H4F;h1(end);h2(end);h3(end)];
    for i = 1:cam1
       aux(i,1)= exp(-0.5*(1/sigma_actor^2)*(norm(X_actor-
mi_actor(:,i))^2));
    end
   a1 = aux;
   U = w*a1;

    if U > 1
        U = 1;
    else
        if U<0
            U=0;
        end
    end

    U = 0.2 + U*0.8;
% saida da função actor.

    X_critic                  = [ref(1);H4F;U;h1;h2;h3];
    for i = 1:cam2
       aux2(i,1)= exp(-
0.5*(1/sigma_critic^2)*(norm(X_critic-mi_critic(:,i))^2));
    end
```

```
    c1 = aux2;
    Q1 = v*c1;
% saida da função Critic


ref_ref = 0;

%tic
for k=1:16*D

    %% PASSO 2 - OBSERVAR O NOV ESTADO (PREDIÇÃO) E
CALCULAR A RECOMPENSA

    H4                          = circshift(H4,1);

    sim('modelo_simulink');
    H4(1) = h4(end);


    H4F                         = b*H4(1:9)/sum(b);

    ERRO2                       = ERRO1;
    ERRO1                       = ref(k) - H4F;
    erro = ref(k) - H4(1);


    [ref(k); H4F; U]

    r = 3*ERRO1 - 0*ERRO2 ;
% Pode se usar o ERRO 2 como uma função de derivativo..
mas na prática não demonstrou muita diferença.

    %% PASSO 3 - CALCULAR U2 - ACTOR DO ESTADO NOVO

    X_actor                     =
[ref(k);H4F;h1(end);h2(end);h3(end)];
%O estado foi definido como o nível atual e a referência.
POnde estou e quão longe do Objetivo
    for i = 1:cam1
       aux(i,1)= exp(-0.5*(1/sigma_actor^2)*(norm(X_actor-
mi_actor(:,i))^2));
    end

    a2 = aux;
    U = w*a2;

    if U > 1
        U = 1;
    else
        if U<0
```

```
            U=0;
        end
    end

    U = 0.2 + U*0.8;
    %% PASSO 4 - CALCULAR Q2 - CRITIC DO ESTADO NOVO

    X_critic                    =
[ref(1);H4F;U;h1(end);h2(end);h3(end)];
%As entradas do estimador de valor de estado são o estado
e a ação selecionada nesse estado
%     [~,Q2]                      = critic(X_critic2,v);
%Calcula o valor do par estado-ação Atual.
    for i = 1:cam2
       aux2(i,1)= exp(-
0.5*(1/sigma_critic^2)*(norm(X_critic-mi_critic(:,i))^2));
    end



    c2 = aux2;
    Q2 = v*c2;

    %% PASSO 5 - CALCULAR O ERRO DE DIFERENÇA TEMPORAL

    td_error                    = r + gama*Q2 - Q1;

    %% PASSO 6 - ATUALIZAR AS REDES NEURAIS


    if k < 16*D

        if abs((abs(ref(k))- abs(ref_ref)))> epsilon
            ref_ref = ref(k);
            eta_actor = eta_actor0;
            eta_critic = eta_critic0;
        else
            eta_critic = eta_critic*LearnRateDecay;
            eta_actor = eta_actor*LearnRateDecay;
        end

    w = w + eta_actor*td_error*a1';

    v  = v  + eta_critic*td_error*c1';
    end



    %% PASSO 7 - S<-S' e A<-A'OU SEJA U1<-U2 E Q1<-Q2
```

```
    entrada(k)                  = U;
    saida(k)                    = H4(1);
    eta_critic_r(k)             = eta_critic;
    eta_actor_r(k)              = eta_actor;

    h1=h1(end);
    h2=h2(end);
    h3=h3(end);
    h4=h4(end);

    Q1                          = Q2;
    a1                          = a2;
    c1                          = c2;

    % change in valve position will take place in pre-
defined times

    if k==2*D+D/2
        k23=20.62;
    end

    if k==4*D+D/2
        k23=11.89;
    end

    if k==6*D+D/2
        k23=20.62;
    end
    if k==9*D+D/2
        k23=11.89;
    end
    if k==12*D+D/2
        k23=20.62;
    end
    if k==14*D+D/2
        k23=11.89;
    end
%pause(k*T - toc);
end
```