PAULO VICTOR BARRETO PAES OLIVEIRA

Modelagem e Controle de uma Coluna de Destilação Didática Utilizando Redes Neurais e Aprendizado por Reforço

Campos dos Goytacazes, RJ

2020

PAULO VICTOR BARRETO PAES OLIVEIRA

Modelagem e Controle de uma Coluna de Destilação Didática Utilizando Redes Neurais e Aprendizado por Reforço

Trabalho de conclusão de curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia Fluminense como requisito parcial para conclusão do curso de Bacharelado em Engenharia de Controle e Automação.

Instituto Federal de Educação, Ciência e Tecnologia Fluminense – IFF

campus Campos Centro

Engenharia de Controle e Automação

Orientador: D.Sc. Adelson Siqueira Carvalho

Campos dos Goytacazes, RJ

2020

Biblioteca Anton Dakitsch
CIP - Catalogação na Publicação

O689m

Oliveira, Paulo Victor Barreto Paes
Modelagem e Controle de uma Coluna de Destilação Didática Utilizando Redes Neurais e Aprendizado por Reforço / Paulo Victor Barreto Paes Oliveira - 2020.
69 f.: il. color.

Orientador: Adelson Siqueira Carvalho

Trabalho de conclusão de curso (graduação) -- Instituto Federal de Educação, Ciência e Tecnologia Fluminense, Campus Campos Centro, Curso de Bacharelado em Engenharia de Controle e Automação, Campos dos Goytacazes, RJ, 2020.
Referências: f. 67 a 69.

1. Aprendizado por Reforço. 2. Redes Neurais Artificiais. 3. Coluna de Destilação. 4. Inteligência Artificial. 5. Controle Avançado. I. Carvalho, Adelson Siqueira, orient. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da Biblioteca Anton Dakitsch do IFF com os dados fornecidos pelo(a) autor(a).

PAULO VICTOR BARRETO PAES OLIVEIRA

Modelagem e Controle de uma Coluna de Destilação Didática Utilizando Redes Neurais e Aprendizado por Reforço

Trabalho de conclusão de curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia Fluminense como requisito parcial para conclusão do curso de Bacharelado em Engenharia de Controle e Automação.

Trabalho aprovado. Campos dos Goytacazes, RJ, 18 de fevereiro de 2020:

D.Sc. Adelson Siqueira Carvalho
INSTITUTO FEDERAL
FLUMINENSE
Orientador

Msc. Edson Simões dos Santos
INSTITUTO FEDERAL
FLUMINENSE
Convidado

Msc. Cleber de Medeiros Navarro
INSTITUTO FEDERAL
FLUMINENSE
Convidado

Campos dos Goytacazes, RJ
2020

"Eu não tenho medo de computadores. Tenho medo da falta deles"
(Isaac Asimov)

Agradecimentos

Principalmente à minha família, que me deu apoio e amor durante todos esses anos e que me fizeram chegar aonde estou.

À minha namorada que esteve ao meu lado durante boa parte do meu percurso acadêmico.

A todos os amigos que fiz durante a graduação, onde tivemos ótimos momentos de estudo e de diversão.

A todas as experiências extracurriculares que vivenciei, tais como: a Atlética Sheriff, a monitoria de Controle Clássico e a equipe de Mecatrônica. Onde pude participar de diversas competições e atividades que me fizeram evoluir como pessoa e como engenheiro.

Ao meu Orientador Adelson Siqueira Carvalho, pelas dicas, conselhos e orientações que me fez seguir pelo melhor caminho.

Ao Instituto Federal Fluminense, onde passei 5 anos da minha vida e que possibilitou essa grande conquista de me tornar um Engenheiro de Controle e Automação.

Resumo

Diante do grande aumento do poder computacional e da enorme quantidade de dados gerados ao longo das últimas décadas, tem se tornado comum a utilização de inteligência artificial (IA) em diversas áreas, como a de controle de processos industriais. Com base nisso, este trabalho teve dois objetivos principais, que foi a modelagem computacional da coluna de destilação didática presente no Instituto Federal Fluminense (IFF) por meio das redes neurais artificiais (RNA). E também o objetivo de desenvolver um controlador baseado em aprendizado por reforço, para realizar o controle da variável de temperatura de topo do modelo da planta criado. O trabalho foi desenvolvido por meio do programa MATLAB®, utilizando as ferramentas NNTool, Simulink e Reinforcement Learning Toolbox. Foram realizados testes que demonstraram resultados satisfatórios tendo como base o índice Mean Squared Error (MSE), que é o erro médio quadrático, no qual podem ser analisados por meio de gráficos e tabelas.

Palavras-chaves: Aprendizado por Reforço. Redes Neurais Artificiais. Coluna de Destilação. Inteligência Artificial. Controle Avançado.

Abstract

In view of the great increase in computational power and the huge amount of data generated over the last decades, the use of artificial intelligence (AI) in several areas, such as industrial process control, has become common. Based on this, this work had two main objectives, which was the computational modeling of the didactic distillation column present at the Federal Fluminense Institute through artificial neural networks (ANN). And also the objective of developing a controller based on reinforcement learning, to carry out the control of the top temperature variable of the created plant model. The work was developed using the MATLAB® software, using the tools NNTool, Simulink and Reinforcement Learning Toolbox. Tests were performed that showed satisfactory results based on the Mean Squared Error (MSE) index, in which they can be analyzed using graphs and tables.

Key words: Reinforcement Learning. Artificial Neural Networks. Distillation Column. Artificial Intelligence. Advanced Control.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

IA	Inteligência Artificial;
RNA	Rede Neural Artificial;
MATLAB®	Matrix Laboratory;
NNTool	Neural Network Toolbox;
TCC	Trabalho de Conclusão de Curso;
IFF	Instituto Federal Fluminense;
MSE	Mean Squared Error;
SISO	Single Input Single Output;
MIMO	Multiple Input Multiple Output;
PID	Proporcional Integral Derivativo;
LQR	Linear Quadratic Regulator;
DFI	Device Fieldbus Interface;
OPC	Ole for Process Control;
TCV	Válvula Controladora de Temperatura;
PDM	Processos de Decisão de Markov;
DT	Diferença Temporal;
$\sum$	Somatório;
∇	Gradiente;
DDPG	Deep Deterministic Policy Gradient;
DPG	Deterministic Policy Gradient;
DQN	Deep Q Network;
MPC	Model Predictive Control;
mA	Miliampère;

l/h	Litro por hora;
°C	Grau Celsius;
s	Segundo;
R	Medida de qualidade de ajuste do modelo;
CPU	Central Processing Unit;
GPU	Graphics Processing Unit.

Sumário

1 Introdução

Este capítulo realiza uma breve apresentação dos assuntos tratados na fundamentação teórica e do projeto proposto. Também define os objetivos deste trabalho e as justificativas para sua execução.

1.1 Contextualização

O crescimento na capacidade de processamento e armazenamento computacional das últimas décadas, tem tornado possível tecnologias que antes eram inviáveis de se utilizar, como é o caso da Inteligência Artificial (IA).

A IA faz parte de um dos campos da ciência da computação envolvida no projeto de sistemas que exibem características que são associadas com a inteligência do ser humano (BARR; FEIGENBAUM, 1981). Permitindo assim, que máquinas possam realizar tarefas complexas no lugar do ser humano, ou mesmo aumentando a eficiência do homem na interação com equipamentos sofisticados.

Ao mesmo tempo, diversos tipos de indústrias ao redor do mundo tem se automatizado cada vez mais, fazendo com que grandes quantidades de dados sejam coletados. Com a aquisição de dados se tornando mais fácil, barata e de boa qualidade, é possível desenvolver controladores ainda mais confiáveis e complexos atendendo necessidades de diferentes tipos de sistemas.

Levando em conta o cenário nacional, o Brasil é um dos países que apresenta maior desenvolvimento de tecnologias para o tratamento de biocombustíveis e processos de refino do petróleo. A separação dos componentes de uma mistura é um dos processos de maior importância na indústria química, petroquímica e de álcool, sendo a destilação o método de separação mais utilizado atualmente (PULIDO, 2011).

As colunas de destilação, além de possuirem grande importância na usinas de produção de álcool e nas refinarias de petróleo, são bons exemplos de sistemas complexos. Principalmente devido ao fato de possuirem diversas variáveis de entrada e saída do processo, e por apresentar um comportamento dinâmico não linear.

Encontra-se sistemas de controle em diversas indústrias de processos, seja regulando o nível de líquidos em reservatórios, concentrações químicas em tanques, até a espessura do material fabricado (NISE; SILVA, 2011). E o controlador é um dispositivo de vital importância no controle de um processo, onde necessita cada vez mais ser otimizada a variável controlada e se adaptar a dinâmica do sistema.

Sendo assim, e tendo em vista que no mundo atual as empresas buscam aumentar ainda mais a produtividade, a confiabilidade e a segurança dos processos, o uso de controladores com técnicas de inteligência artificial, como o aprendizado por reforço, é uma das opções existentes.

Pode-se dizer que o objetivo do aprendizado por reforço é o desenvolvimento de um agente (controlador) capaz de tomar decisões e interagir (atuar) com seu ambiente (planta) com base em suas observações do ambiente (sensor). A escolha da decisão a ser tomada a cada instante é feita de forma a garantir a maximização de uma certa função objetivo, que atribui recompensa a estados favoráveis para o agente.

1.2 Justificativa

Diante dos altos investimentos e aplicabilidades que o campo da inteligência artificial vem tendo nos últimos anos, e da alta demanda de controladores automáticos em sistemas cada vez mais complexos, como uma coluna de destilação, se faz de grande importância técnicas de controle avançado.

A utilização de um controlador baseado em aprendizado por reforço, um dos ramos da inteligência artificial, trás diversos benefícios como: o uso em sistemas não lineares e multivariáveis, otimização e controle adaptativo (SUTTON; BARTO, 2018).

1.3 Objetivo Geral

O objetivo geral deste trabalho é realizar a modelagem da coluna de destilação didática, presente no Instituto Federal Fluminense, por meio das redes neurais artificiais. E desenvolver o controlador baseado em aprendizado por reforço, para controlar o modelo computacional da planta.

1.4 Objetivos Específicos

- Realizar testes dinâmicos na planta física para coletar os dados;
- A partir dos dados coletados, criar um modelo computacional da torre de destilação utilizando redes neurais artificiais;
- Desenvolver o controlador baseado em aprendizado por reforço;
- Treinar o controlador e realizar o controle, via simulação, do modelo da planta previamente criado;

1.5 Estrutura do Trabalho

Este Trabalho de Conclusão de Curso (TCC) está dividido em seis capítulos, onde:

O primeiro capítulo trata de sua contextualização, justificativa e seus objetivos.

O segundo capítulo traz as fundamentações teóricas necessárias para o completo entendimento deste trabalho.

O terceiro capítulo apresenta a revisão bibliográfica de trabalhos correlatos e pesquisas sobre o tema.

O quarto capítulo aborda o desenvolvimento dos objetivos propostos, como a aquisição de dados, modelagem computacional da planta e o desenvolvimento do controlador.

O quinto capítulo apresenta os resultados obtidos das simulações e testes.

O sexto e último capítulo possui as considerações finais deste TCC, além da apresentação de propostas para trabalhos futuros.

2 Fundamentação Teórica

Este capítulo tem como objetivo explicar os conceitos fundamentais relacionados aos temas da teoria de Sistemas de Controle, Coluna de Destilação e Inteligência Artificial, para proporcionar um maior entendimento sobre o assunto.

2.1 Sistemas de Controle

Um sistema de controle pode ser compreendido como um conjunto de dispositivos que regulam o comportamento de outros sistemas, visando atingir algum objetivo preestabelecido.

Os sistemas de controle contribuem para muitos aspectos da sociedade moderna, podendo ser encontrado desde os lares como em fornos, geladeiras e aparelhos de vídeo, como também no meio industrial e científico como a condução de embarcações, aviões e o guiamento de mísseis e de ônibus espacial. Até mesmo representações de sistemas econômicos e psicológicos baseadas na teoria de sistemas de controle foram propostas (NISE; SILVA, 2011).

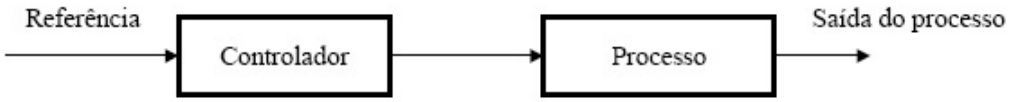
Quando o controle é realizado com pouca ou até mesmo nenhuma intervenção humana, é dito como controle automático. Entretanto, quando o sistema de controle necessitar de um operador humano constantemente modificando o sistema, diz-se que é do tipo manual (JR; YONEYAMA, 2000).

O primeiro exemplo significativo de um sistema de controle foi o regulador centrífugo construído por James Watt para o controle de velocidade de uma máquina a vapor, no final do século XVIII. Outros importantes trabalhos sobre a teoria de controle se devem a Minorsky, Hazen e Nyquist, no inicio do século XX, que trabalharam respectivamente com pilotagem de embarcações, estabilidade de sistemas e servomecanismos para controle de posição (OGATA; YANG, 2002).

Os sistemas de controle podem ser em malhar aberta, tambem conhecidos como sem realimentação, que consiste em um sistema que não se tem o elemento sensor para mostrar o valor da variável que esta sendo controlada, que no caso é a saída do processo. Em geral, eles são aplicados em sistemas mais simples e mais baratos, quando não há necessidade de um controlador mais complexo.

O diagrama de blocos básico de um sistema de controle em malha aberta é apresentado na Figura 1.

Figura 1 – Diagrama de blocos de controle em malha aberta



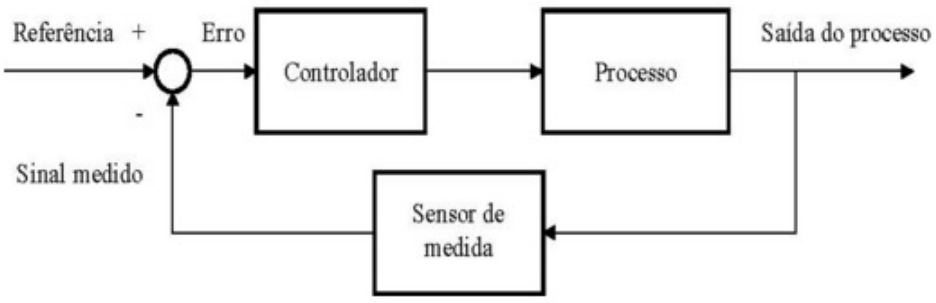
Fonte:(NOGY, 2015).

Já os sistemas de controle em malha fechada, ou com realimentação, são mais complexos e caros porém muito mais usados atualmente, devido ao fato de se conseguir medir a saída do processo. Com isso, a malha de controle consegue identificar se a saída da variável medida condiz com o valor desejado, fazendo com que o controlador atue de forma a corrigir a diferença entre eles, valor conhecido como erro (FRANKLIN; POWELL; EMAMI-NAEINI, 2013).

Fica evidente então, que a grande vantagem do controle em malha fechada é que ele consegue manter a variável controlada, tambem conhecida como variável do processo, no seu valor desejado. Sendo assim, ele consegue compensar possíveis pertubações externas ou não linearidades do sistema.

Na Figura 2, é apresentado o diagrama de blocos do sistema com realimentação, onde se pode identificar o sensor enviando o sinal medido, que então é comparado, por meio de um bloco somador, pelo sinal de referência. O erro gerado, vai ser a entrada do controlador e este por meio de um atuador vai influenciar a variável controlada do processo.

Figura 2 – Diagrama de blocos de controle em malha fechada



Fonte:(NOGY, 2015).

Portanto os sistemas de controle tem como objetivos principais o aumento da segurança dos operadores, o aumento da produção e da qualidade dos produtos ou serviços

executados e a diminuição dos custos operacionais.

2.1.1 Modelagem de um Sistema

Para desenvolver o controle de um sistema, muitas vezes, realiza-se a modelagem do processo a ser controlado, ou seja, obtem-se uma representação de como funciona a dinâmica do processo real.

Com o modelo da planta, o projetista tem algumas vantagens, tais como usá-lo como base para sistemas de controle baseados em modelo, e principalmente servir de base para simulações dos sistemas antes de testá-lo na planta física (SEBORG et al., 2010).

A modelagem do sistema pode ser feito basicamente de duas formas. Uma delas é a modelagem fenomenológica, ou matemática, que tem como base as leis físicas, equações diferenciais e a dinâmica entre os componentes do sistema, calculando-se então a função de transferência.

A outra forma é pela identificação do processo, conhecida também como modelo do tipo caixa preta, que é baseada em dados adquiridos do sistema. São modelos construídos a partir da determinação de uma relação matemática entre os dados de entrada e saída, no qual seus parâmetros não possuem significado físico (AGUIRRE, 2000). É recomendado quando já se possui o sistema físico para poder coletar os dados experimentais, e em sistemas mais complexos, já que se torna mais prático e viável a obtenção do modelo.

Existem diversas técnicas e algoritmos para obtenção do modelo pela identificação, das mais conhecidas estão o método da curva de reação, modelos auto-regressivos e redes neurais artificíais, sendo que este último será maior aprofundado posteriormente.

2.1.2 Controle Moderno

A teoria de controle é dividida em dois grandes grupos, que são o Controle Clássico e o Controle Moderno. No controle clássico, o objetivo é controlar processos de uma variável na entrada e uma na saída, conhecidos como sistemas monovariáveis ou SISO (do inglês *Single Input Single Output*), e aborda técnicas de projeto de controle tais como: método do lugar das raízes, método de resposta em frequência e os controladores proporcional integral derivativo (PID) (DORF; BISHOP, 2011).

No controle moderno, o enfoque é controlar sistemas que possuem muitas entradas e muitas saídas, conhecidos como multivariáveis ou MIMO (do ingles *Multiple Input Multiple Output*). As variáveis podem ser inter-relacionadas de maneira não linear entre elas.

Dentro da teoria de controle moderno existem diversos tipos de controladores, dependendo sempre do tipo da aplicação e do objetivo do projetista para utilizá-los. Alguns

dos tipos mais utilizados são o controle preditivo, o controle adaptativo e o controle ótimo, sendo que os dois últimos serão abordados nas subseções 2.1.3 e 2.1.4.

2.1.3 Controle Ótimo

A otimização de processos em tempo real tem como objetivo obter os setpoints das malhas de controle do processo de forma a maximizar os lucros (ou minimizar custos), obedecendo os limites operacionais da planta. Sendo necessário, para conseguir tal ponto ótimo de operação, um modelo que consiga descrever o processo com exatidão (NASCIMENTO, 2013).

Hoje, com o aumento computacional cada vez maior, o controle ótimo é muito visado no meio industrial, para obter uma maior produtividade e com menor custo possível. Com isso existem diversos tipos de algoritmos para a otimização do processo, alguns dos mais usados são o LQR (do inglês *Linear Quadratic Regulator*), algoritmos genéticos que são baseados na evolução biológica e algoritmos relacionados ao aprendizado por reforço.

2.1.4 Controle Adaptativo

A teoria de controle adaptativo se originou em meados do século XX, com uma gama de estudos sobre o controle em ambientes instáveis, principalmente o controle de aeronaves.

Um controlador adaptativo consegue modificar seu comportamento automaticamente, quando houver mudanças na estrutura dinâmica do sistema, ou distúrbios externos (ÅSTRÖM; WITTENMARK, 2013). Com isso ele consegue reduzir não apenas efeitos de distúrbios, bem como incertezas da dinâmica da planta, principalmente em sistemas não lineares e complexos.

Então, o controle adpatativo é uma boa solução para adequar os parâmetros do controlador, de forma a atender os requisistos do sistema desejado, fazendo com que seja possível projetar sistemas com melhor desempenho e funcionalidade.

2.2 Coluna de Destilação

A separação de misturas em seus componentes puros é de vital importância nas indústrias químicas e petroquímicas. Dentre os métodos de separação de misturas, a destilação é o mais utilizado, principalmente quando se deseja separar líquidos miscíveis em outros componentes. Sendo assim, as colunas de destilação são os equipamentos mais utilizados nas indústrias para realizar tal atividade.

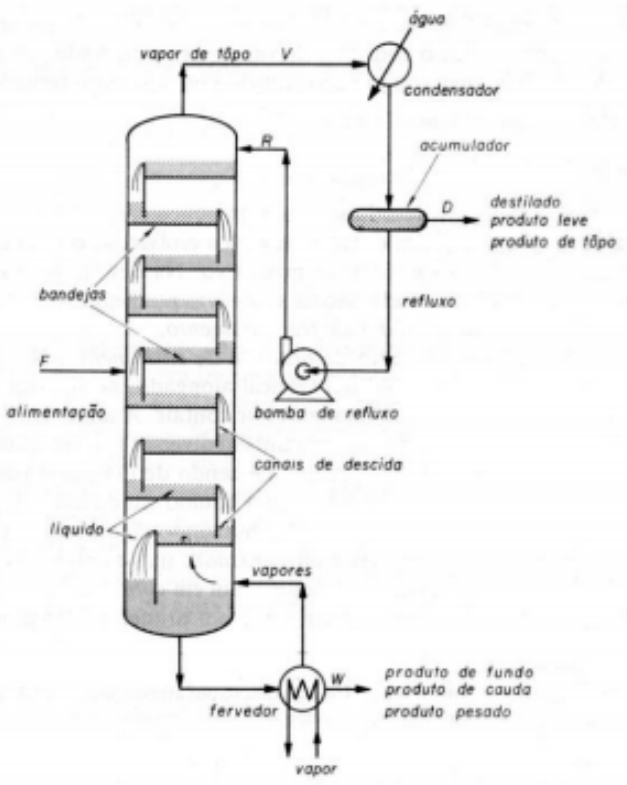
A destilação tem como princípio o fenômeno de fracionamento dos líquidos, onde os mais voláteis, ou seja, com pontos de ebulição menores, separam-se primeiro, seguidos

pelos outros componentes com suas respectivas volatilidades (RASOVSKY, 1973).

Entre as variáveis que são possíveis serem controladas, também conhecidas como variáveis de processo, em uma coluna de destilação, são o nível, a temperatura e a pressão. Destas o nível e a pressão são consideradas variáveis fundamentais para a operação da torre. Já a temperatura controla a qualidade do produto, pois indiretamente ela está associada com a composição da carga (CAMPOS; TEIXEIRA, 2006).

A vazão das válvulas de controle são consideradas como variáveis manipuladas, pois por meio delas o controlador atua na planta afetando as outras variáveis. Na Figura 3 é mostrado a configuração de uma coluna de destilação convencional.

Figura 3 – Configuração de uma Coluna de Destilação Convencional



Fonte:(RASOVSKY, 1973).

A alimentação é feita em um ponto intermediário, de forma que a mistura percorra a coluna descendo de prato em prato, e o vapor gerado na base da coluna, suba de prato em prato. Estabelecendo-se dois fluxos internos, um ascendente de forma a enriquecer a mistura do último prato (no topo da coluna), e outro descendente, de forma a empobrecer a mistura e concentrar os resíduos no primeiro prato (base da coluna), dessa forma, serão retirados do processo (MEDINA, 2008).

A configuração convencional de uma torre de destilação consiste em uma única

alimentação de carga e duas retiradas, uma localizada no topo (destilado) e a outra no fundo da torre (produto de fundo). O prato de alimentação separa a coluna em duas seções: seção de retificação (enriquecimento) que envolve o destilado e seção de esgotamento, englobando o produto de fundo (NASCIMENTO, 2013).

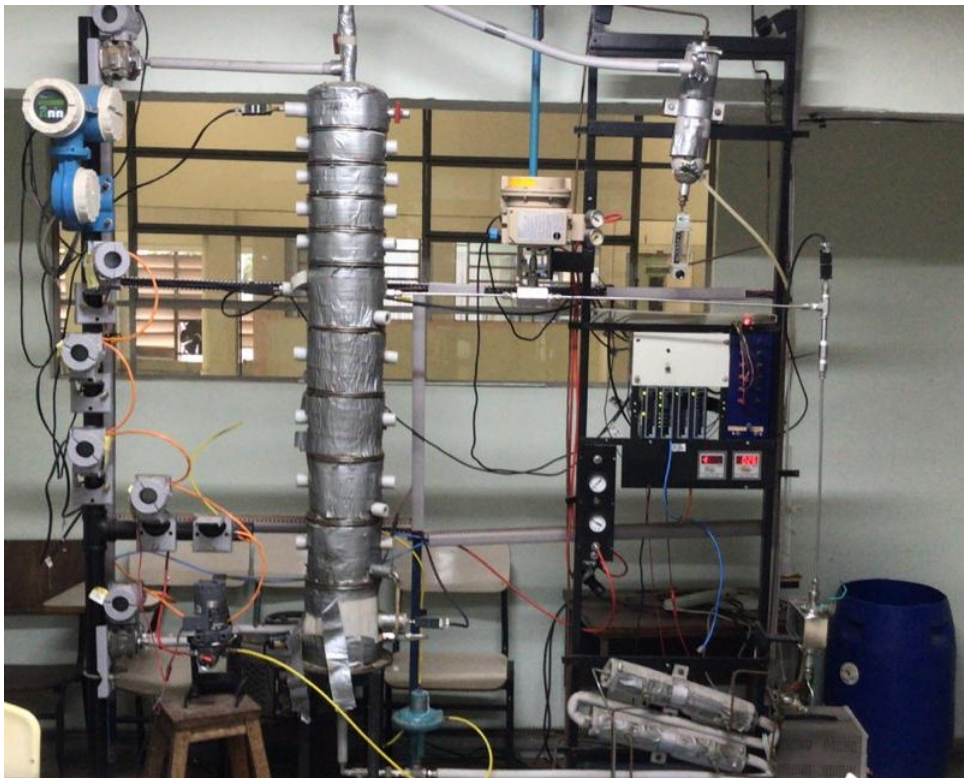
2.2.1 Planta Utilizada

A coluna de destilação didática utilizada no presente trabalho, localizada no IFF campus Campos Centro, não é convencional, pois as colunas A e B são conjugadas. Fora isso, não apresenta subsistemas como referverdor e refluxo de topo (NAEGELE, 2000).

Dentro da coluna existem 20 pratos perfurados sem calota, sendo que os 10 primeiros, de baixo para cima, fazem parte da coluna A ou coluna de destilação. E os outros 10 pratos constituem a coluna B ou coluna de retificação (SILVA; OTAL, 2010).

A fonte térmica é uma resistência elétrica de 1000W de potência em 127V de tensão, que se localiza no interior da base da coluna. Todo o destilado obtido se condensa e é coletado em uma proveta, fazendo com que a temperatura de topo seja controlada totalmente através da vazão de entrada da coluna, já que a resistência elétrica não varia (CARVALHO, 2008). Na Figura 4 é apresentada a planta utilizada no trabalho.

Figura 4 – Coluna de Destilação Didática do IFF

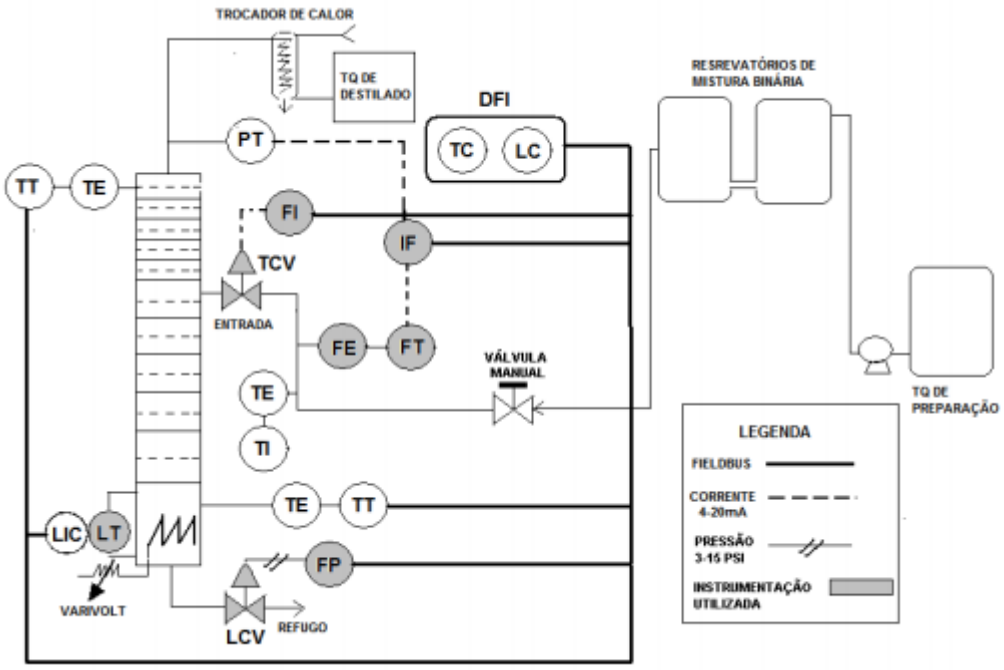


Fonte: Autor.

A planta também conta com um sistema de aquisição de dados de porte industrial, onde os instrumentos de medição das variáveis do processo são interligados através do protocolo Foundation Fieldbus. Por meio de uma DFI (*Device Fieldbus Interface*) a rede de instrumentos se integra a uma rede ponto-a-ponto com o computador, na qual as informações oriundas do processo ficam disponibilizadas no Syscon®, podendo ser monitoradas e modificadas (CARVALHO, 2008).

Percebe-se na Figura 5 , a presença de instrumentação eletrônica digital (Fieldbus Foundation), eletrônica analógica (4 mA a 20 mA) e pneumática analógica (3 PSI a 15 PSI).

Figura 5 – Fluxograma da Coluna de Destilação do IFF



Fonte: (MEDINA, 2008).

2.3 Inteligência Artificial

A Inteligência Artificial (IA) ou AI (do inglês *Artificial Intelligence*) surgiu pela tentativa de imitar a maneira de como os seres humanos pensam e resolvem problemas, criando métodos e introduzindo-as na programação de computadores (CAMPOS; SAITO, 2004).

Por se tratar de algo de difícil compreensão, já que a própria inteligência humana é suficientemente complexa de se definir, pode-se encontrar diferentes definições para o termo IA. Entre alguns exemplos de autores renomados na literatura especilizada, encontra-se que é o estudo das faculdades mentais através do uso de modelos computacionais (Charniak e McDermott, 1985 apud (JR; YONEYAMA, 2000)). Ou até mesmo, que a inteligência artificial é o estudo de como fazer os computadores realizarem tarefas que, no momento, são feitas melhor por pessoas (Rick, 1983 apud (JR; YONEYAMA, 2000)). E segundo (Nilsson, 1986 apud (JR; YONEYAMA, 2000)) a IA é o campo de conhecimento onde se estudam sistemas capazes de reproduzir algumas das atividades mentais humanas.

Os primeiros estudos e pesquisas sobre a área se deram no período da Segunda Guerra Mundial, início da década de 40 do século passado. Isso se deve pelo fato de que na época buscavam constantemente aumentar o poder bélico e tecnológico, para quebra de códigos encriptografados, cálculos balísticos e desenvolvimento de novas tecnologias e armas como a bomba atômica.

Um dos pioneiros tanto da área de inteligência artificial, quanto do campo da computação eletrônica como conhecemos hoje, foi Alan Turing, reconhecido por muitos como o pai da computação. Em um de seus artigos, publicado em 1950, ele propõe uma maneira de definir se uma máquina seria capaz de pensar, conhecido como O Jogo da Imitação (TURING, 1950).

Logo após a Segunda Guerra Mundial o computador não ficou restrito apenas ao âmbito militar e científico, começou a ser aos poucos utilizado em empresas, indústrias e principalmente em universidades. Com isso, nos Estados Unidos em 1956, John McCarthy reuniu em uma conferência proferida ao Darmouth College, na Universidade de New Hampshire, vários pesquisadores de renome para estudar o que foi denominado por Minsky, McCarthy, Newell e Simon de Inteligência Artificial (NEWELL, 1980).

Ao longo dos anos este campo enfrentou altos e baixos, tendo as décadas de 60 e 70 com pouquíssimas pesquisas e entusiasmos, voltando a surgir apenas na década de 80 com os trabalhos do físico e biólogo John Hopfield, sobre algorítmos de aprendizagem de redes neurais (HOPFIELD, 1982).

E um dos grandes marcos da inteligência artificial do século XX, foi a vitória do computador, criado pela IBM, Deep Blue contra o campeão mundial de xadrez da época

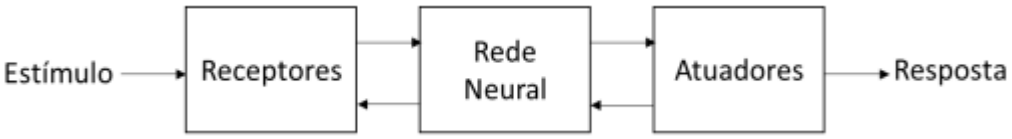
Garry Kasparov, em 1997 (NEWBORN, 2000). Este fato inédito, inspirou ainda mais o estudo e desenvolvimento da IA, e atualmente é uma das principais áreas tecnológicas, tanto no meio acadêmico e científico quanto no meio industrial no mundo todo.

2.3.1 Redes Neurais Artificiais

Um dos principais ramos da Inteligência Artificial atualmente, são as Redes Neurais Artificias (RNA). Podem ser definidas como estruturas que processam informação de forma paralela e distribuída e que consistem em unidades computacionais interconectadas por canais unidirecionais chamados de conexões (HECHT-NIELSEN, 1992). Ou também, como sendo uma técnica que simula, por meio de algorítmos de treinamento e modelos matemáticos, o funcionamento do cérebro humano (HAYKIN, 2007).

Como se pode perceber pelo nome, ela foi baseada no neurônio biológico humano, justamente por ser o principal componente do sistema nervoso. O cérebro é um computador muito complexo, não linear e paralelo, que tem a capacidade de organizar seus neurônios de forma a realizar processamento de forma muito rápida, reconhecer padrões e armazenar conhecimentos por experiências, entre outras capacidades superiores as máquinas atuais. Na Figura 6 é apresentado, por meio de diagrama de blocos, a representação do sistema nervoso.

Figura 6 – Representação em Diagrama de Blocos do Sistema Nervoso



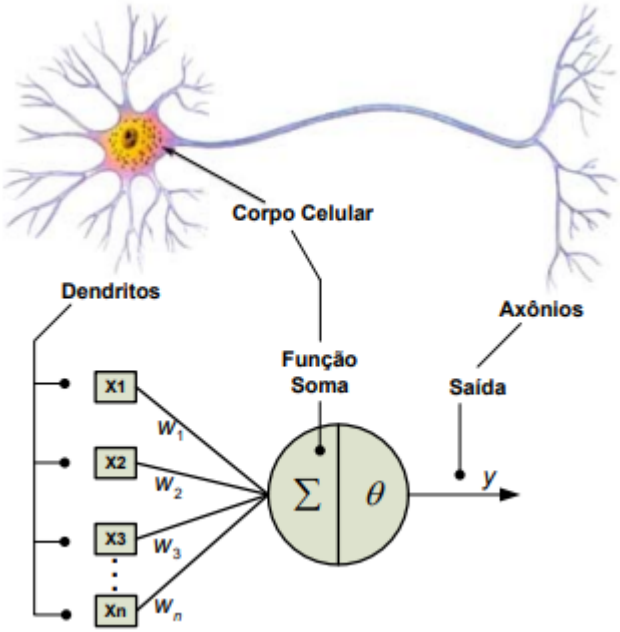
Fonte: (HAYKIN, 2007).

Vale notar que as RNA's são apenas inspiradas no nosso conhecimento sobre sistemas nervosos biológicos da natureza, e não buscam ser realísticas em todos os detalhes, ou seja, não é o ponto principal modelar o sistema nervoso biológico (JR; YONEYAMA, 2000).

O primeiro trabalho sobre a neuro computação é ainda no período da Segunda Guerra Mundial, em 1943 no artigo de McCulloch e Pitts. Nele sugeriam a construção de uma máquina baseada (ou inspirada) no cérebro humano. O neurofisiologista McCulloch e matemático Walter Pitts, apresentaram um trabalho que fazia uma analogia entre células vivas e processos eletrônicos, simulando o comportamento do neurônio natural, onde o neurônio possuia apenas uma saída, que era uma função de entrada (threshold) da soma do valor de suas diversas entradas (RUSSEL; NORVIG, 2004).

Na Figura 7 é apresentado uma comparação entre os elementos de um neurônio biológico e os de um neurônio artificial.

Figura 7 – Comparação de um Neurônio Artificial com um Neurônio Biológico



Fonte: (FERNANDES, 2009).

Um neurônio artificial é uma unidade de processamento de informação que é de vital importância para as redes neurais artificiais. As principais partes dos neurônios são: um conjunto de sinapses ou elos de conexão cada uma caracterizada por pesos próprios; a junção somadora para somar os sinais de entrada; e uma função de ativação para restringir a amplitude de saída do neurônio. A função de ativação pode ser de diferentes tipos, como: degrau, rampa, sigmóide, tangente hiperbólica (uma das mais usadas) (HAYKIN, 2007).

A arquitetura de uma RNA é relacionado, com o algoritmo de aprendizado utilizado para treinar a rede e também com a capacidade de resolução de determinados tipos de problemas.

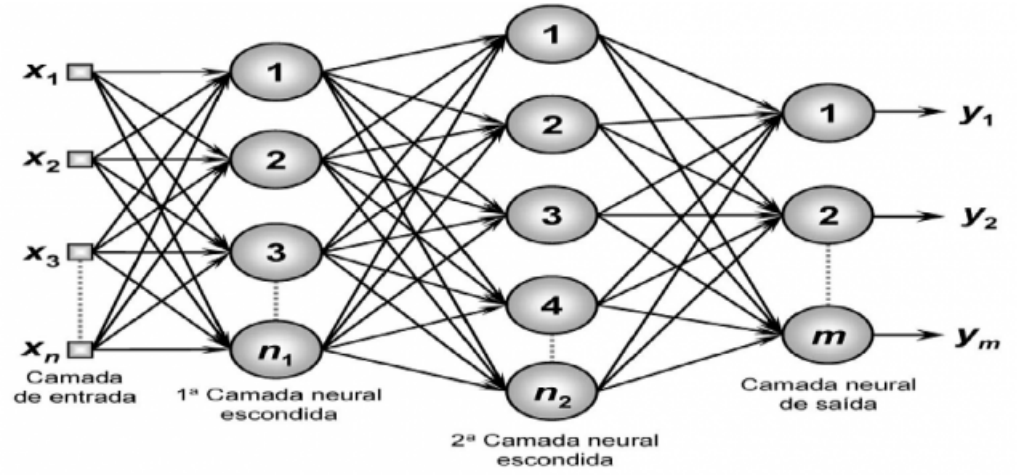
Entre os parâmetros mais importantes em uma rede estão:

- **Número de neurônios:** que precisam ser definidos nas três camadas que são a de entrada que recebem os dados de entrada da rede, a camada intermediárea (ou escondida) que processa os dados e aprendem a generalizá-lo, e a camada de saída responsável por apresentar uma saída conveniente para o problema.
- **Número de camadas:** especificamente das camadas intermediárias (ou escondidas) podendo ser uma única ou múltiplas camadas.

- **Tipo de conexão entre os neurônios:** que podem ser totalmente conectados ou apenas parcialmente conectados
- **Topologia da rede:** que podem ser principalmente divididas em FeedForward e Feedback.

Na Figura 8 é apresentado um exemplo de uma RNA com uma arquitetura FeedForward e com duas camadas intermediáreas.

Figura 8 – Arquitetura FeedForward de Duas Camadas Ocultas



Fonte: (FLAUZINO, 2010).

2.4 Aprendizado por Reforço

O Aprendizado de Máquina (do inglês *Machine Learning*) é uma das áreas mais importantes da Inteligência Artificial, e que vem alcançando muitas conquistas nos últimos anos. Arthur Samuel, um dos pioneiros da IA, definiu como um campo de estudo que garante aos computadores a capacidade de aprender sem serem explicitamente programados para determinada tarefa (SAMUEL, 1959).

Costuma-se dividir em três principais grupos de Aprendizado de Máquina, que são:

O Aprendizado Supervisionado, que é quando um instrutor disponibiliza tanto o conjunto de dados de entrada quanto o da saída desejada, para assim, poder aprender e generalizar para conjuntos não vistos em seu treinamento. Muito utilizado em tarefas de classificação, como por exemplo classificar se uma determinada imagem se refere a um cachorro ou gato, e também tarefas de ajuste de curvas, como será utilizado neste trabalho para identificar a dinâmica da coluna de destilação.

O Aprendizado Não-Supervisionado, que diferentemente do supervisionado, não lhe é fornecido o conjunto de dados de saída que se deseja, apenas os dados de entrada. Tem como objetivo encontrar similaridades ou padrões entre os dados para assim poder agrupa-los, tarefa conhecida como Clusterização.

E o Aprendizado por Reforço, que será o foco desta seção, no qual um agente interagindo com o ambiente deve tomar decisões para maximizar a recompensa (reforço) obtida, baseando-se em "tentativa e erro". Para maior entendimento serão apresentados detalhadamente seu histórico, elementos e algoritmos.

2.4.1 Histórico

O Aprendizado por Reforço se originou por meio de dois campos distintos, que trabalharam e cresceram independentes e que posteriormente começaram a se entrelaçar, para se tornar o moderno campo que é hoje.

O primeiro é o campo do aprendizado animal, no qual através de estudos experimentais se originou a ideia básica do reforço e da tentativa e erro. O principal trabalho é a Lei do Efeito de Thorndike, onde resumidamente diz que quanto maior for a satisfação ou o desconforto, maior será o reforço ou o enfraquecimento da ligação (Thorndike, 1911 apud (HAYKIN, 2007)).

Outro campo igualmente importante é o da teoria de Controle Ótimo, onde buscavam soluções ótimas para problemas de controle, assim como em Aprendizado por Reforço busca-se maximizar a recompensa obtida. Um dos principais expoentes na área foi Richard Bellman com seus trabalhos na década de 1950, como a Equação de Bellman e os Processos de Decisão de Markov (PDM).

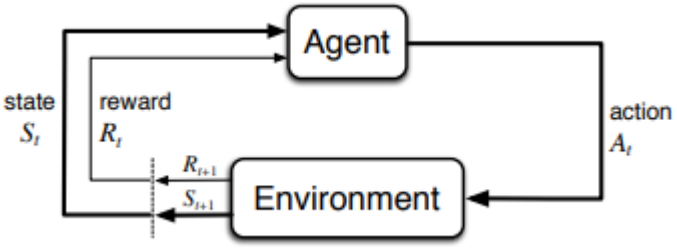
Atualmente o Aprendizado por Reforço deixou de ser algo restrito ao meio acadêmico e esta cada vez mais com aplicações em diferentes áreas como: jogos, carros autônomos, controle de processos industriais, robótica entre outros.

Um exemplo do grande sucesso ocorreu em 2017, quando o computador do Google chamado de AlphaGo, utilizando Aprendizado por Reforço, venceu o campeão mundial de Go. Para se ter uma noção da complexidade do jogo, ele possui 10^{171} posições possíveis, maior até que a quantidade de átomos no universo , em comparação , o xadrez tem 10^{50} posições (TECNOBLOG, 2017).

2.4.2 Elementos

Segundo Sutton e Barto (2018) e Russel e Norvig (2004) os principais elementos e subelementos de um sistema de Aprendizado por Reforço, que é aprensentada na Figura 9, são:

Figura 9 – Diagrama de Blocos do Aprendizado por Reforço



Fonte: (SUTTON; BARTO, 2018).

- **Ambiente:** é definido como o mundo com que o agente interage, podendo sofrer mudanças ao longo do tempo. Ele gera os estados e recompensas (saídas) que será sentido pelo agente, e recebe as ações do mesmo (entrada).

- **Agente:** utiliza as informações, presentes no estado atual do ambiente, para tomar decisões, com o intuito de maximizar a recompensa total a longo prazo. Ele sente o estado e a recompensa gerada pelo ambiente (entradas) e gera uma ação no ambiente (saída). O agente é capaz de alterar o ambiente ao interagir com o mesmo, e por isso espera-se que suas ações impactem nas observações e recompensas que passa a receber. Ele terá que balancear entre dois fatores, conhecido como dilema de *Exploitation vs Exploration*, onde a exploração *(exploration)* abre caminhos para aproveitar melhor a recompensa no futuro e ao mesmo tempo terá que aproveitar a situação atual *(exploitation)*. O Agente que só explora deixará de aproveitar as recompensas e não conseguirá alcançar o objetivo de maximização. Já o Agente que só aproveita o estado atual corre o risco de ficar preso num máximo local, em vez de procurar o máximo global do ambiente. Por isso, o balanceamento destes dois fatores é necessário para o bom funcionamento do agente.

- **Estado (s):** representa a situação do ambiente que o agente vai sentir e levará em conta para tomar decisão. Como exemplo a variável temperatura da coluna de destilação.

- **Recompensa (r):** define o objetivo em um problema de aprendizado por reforço. Em cada etapa de tempo, o ambiente envia para o agente um número escalar chamado de recompensa (ou reforço). Ele então que vai definir o que será um evento bom ou ruim para o agente. Seja Rt a função que representa o comportamento da recompensa total ao longo do tempo, onde $t + k$ representa o instante de tempo em que se adquire uma recompensa $rt + k$ e γ é um fator de desconto que representa o

quanto consideramos recompensas futuras ao selecionar ações, então:

$$Rt = (rt + 1) + (\gamma rt + 2) + ... = \sum_{K=0}^{n} \gamma^K rt + k + 1 \tag{2.1}$$

- **Ação (a):** é o que após a decisão do agente será feito para alterar o ambiente. Frequentemente, as ações do agente afetam não apenas o estado subsequente, mas continuam a ter impacto no futuro. Como exemplo abrir ou fechar a válvula de controle da coluna de destilação.

- **Política (π):** é o mapeamento de um estado particular e de uma ação que será feita nesse estado. Podendo ela ser determinística, onde se toma a mesma ação a cada intervalo de tempo:

$$A = \pi(s) \tag{2.2}$$

  Ou estocástica, onde por exemplo pode-se aplicar uma determinada ação na maior parte do tempo e outra ação na menor parte dele:

$$A = \pi(a|s) = P(A = a|S = s) \tag{2.3}$$

  Um algoritmo de aprendizagem por reforço tem como objetivo melhorar sua política conforme interage com o ambiente, de forma a obter recompensas melhores com o passar do tempo. Frequentemente, inicializa-se o agente com uma política aleatória e deixa-se que o agente aprenda do zero com suas próprias experiências.

- **Função de Valor:** é uma função que representa uma análise de desempenho da política adotada. Ela se difere do sinal de recompensa por avaliar o desempenho do agente a longo prazo, enquanto a recompensa é uma medida imediata da ação do agente. Tem-se a função de valor do estado, onde sabendo a recompensa esperada por estar em um estado, o agente pode avaliar o quão bom é encontrar-se naquele estado, e pode comparar o quão vantajoso é encontrar-se em diferentes estados, e utilizar estes valores como base para tomar suas decisões. Onde E é a expectativa de recompensa, a função de valor de estado é dado por:

$$V\pi(s) = E\pi(Rt|St^a = s) \tag{2.4}$$

  E também tem a função de valor de ação que, semelhante à função de valor do estado, determina o valor de recompensa esperada ao tomar uma ação a estando no estado s, caso siga-se uma política π. Esta função é mais útil para um agente de aprendizado por reforço, visto que este não pode controlar diretamente o estado que deseja atingir, e sim as ações que pode executar. Dado por:

$$Q\pi(s, a) = E\pi(Rt|St = s, At = a) \tag{2.5}$$

- **Modelo do Ambiente:** existem dois tipos de métodos de aprendizado por reforço relacionados com o modelo. O primeiro é o Livre de Modelo onde não se tem um modelo prévio do ambiente, e o agente vai aprendendo por pura experiência de tentativa e erro. E o segundo é o Baseado em Modelo onde se tem o modelo do ambiente, com isso poderá ser feito inferências de como ele irá se comportar ao longo do tempo.

2.4.3 Método de Aprendizagem

Muitos problemas em Aprendizado por Reforço possuem uma grande dimensionalidade do espaço de estados, não sendo possível tratar cada estado de maneira explícita e fazendo-se necessário o uso de aproximadores de funções para definir as funções de valor e política.

Os algoritmos em Aprendizado por Reforço podem ser divididos em 3 grupos, que são: o **Método Ator** que utilizam políticas parametrizadas e não possuem nenhuma forma de memória da função de valor; **Método Crítico** que utilizam uma função de valor espaço-estado sem uma função explícita para representar a política adotada; e o **Método Ator-Crítico** que utilizam duas estruturas distintas para realizarem o aprendizado (SUTTON; BARTO, 2018).

O método que será utilizado no presente trabalho será o do Ator-Crítico, que foi introduzido por (WITTEN, 1977). Ele é mais vantajoso de se utilizar quando se trabalha com espaços de ações contínuas, além de possuir a vantagem de conseguir aprender políticas estocásticas, obtendo as melhores probabilidades a partir da seleção de diversas ações.

A primeira estrutura do método é o Ator, utilizado para definir a política de ações aplicadas e a segunda estrutura é o Crítico, que estima a função de valor e avalia todas as ações executadas pelo ator.

Uma vez que o Ator possui uma política de ações explícita, calcular a ação a ser seguida exige pouco esforço, sendo possível trabalhar com um espaço contínuo de ações. O Crítico avalia as ações adotadas através do método de diferença temporal e utiliza o erro DT, para aprimorar sua estimação de valor e a política do crítico. Sendo V a função implementada pelo Crítico.

A função de valor V é a base para seleção da política de ações adotada, porém essa função é desconhecida pelo agente, sendo necessário aprender a estimá-la a partir das recompensas obtidas através das interações com o ambiente.

O erro de diferença temporal pode ser calculado através da Equação:

$$\delta TD = (rt + 1) + \gamma V(st + 1) - V(st) \tag{2.6}$$

Em que $st + 1$ representa o estado atingido após a execução de uma ação a partindo de um estado st, recebendo uma recompensa $rt + 1$ correspondente. Ao receber essa nova recompensa a função de valor V do novo estado é utilizada para atualizar a do estado anterior. E γ representa o fator de desconto limitado entre $0 < \gamma < 1$ (SUTTON; BARTO, 2018).

Na Figura 10 é apresentado a configuração do método Ator-Crítico.

Figura 10 – Configuração do Método Ator-Crítico

Fonte: (CALMON; PINHEIRO; FERREIRA, 2006).

O algoritmo de aprendizado por reforço que é usado neste trabalho foi o *Deep Deterministic Policy Gradient*, ou **DDPG**, que foi desenvolvido por (LILLICRAP et al., 2015) a partir de algoritmos de gradiente de política e do método Ator-Crítico.

Ele utiliza duas redes neurais profundas distintas, uma para representar a política da estrutura do Ator e a outra representando a do Crítico. Uma das grandes vantagens do DDPG é lidar com alta dimensão de estado contínuo e espaço de ação, que são necessários para resolver problemas de controle. A política do Crítico $Q(S, A)$, e a do Ator $\mu(S)$, são inicializadas com parâmetros aleatórios representados respectivamente por θq e $\theta\mu$. Além deles, também possui o Ator alvo $\mu'(S)$, que serve para melhorar a estabilidade da otimização, onde o agente atualiza periodicamente com base nos valores mais recentes dos parâmetros do Ator. E o Crítico alvo $Q'(S, A)$, que similarmente ao Ator alvo, também é atualizado pelo agente, entretanto com base nos valores mais recentes dos parâmetros do Crítico.

Para cada etapa de tempo o algoritmo funciona da seguinte forma:

- Para a observação atual S, é selecionado a ação $A = \mu(S) + N$, onde N é o ruído

estocástico do modelo. Executa a ação A e observa a recompensa R e a próxima observação S'.

- Guarda as observações de (S, A, R, S') dentro do buffer de experiência, e em seguida recolhe amostras aleatóriamente em um mini lote, chamado de M.

- Para calcular a recompensa cumulativa descrita pela equação 2.1, o agente primeiro calcula uma ação seguinte, passando a próxima observação Si' da experiência amostrada para o Ator alvo. O agente encontra a recompensa cumulativa passando a próxima ação ao Crítico alvo.

- Atualiza os parâmetros do Crítico, minimizando a perda L em todas as experiências amostradas.

$$L = \frac{1}{M}\sum_{i=1}^{M}(Ri - Q(Si, Ai|\theta q))^2 \tag{2.7}$$

- Atualiza os parâmetros do ator, usando o seguinte gradiente de política amostrado para maximizar a recompensa com desconto esperada:

$$Gai = \nabla AQ(Si, Ai|\theta q) \tag{2.8}$$

Onde: $A = \mu(Si|\theta\mu)$

$$G\mu i = \nabla\theta\mu(Si|\theta\mu) \tag{2.9}$$

Gai é o gradiente da saída do Crítico em relação à ação calculada pela rede do Ator, e $G\mu i$ é o gradiente da saída do Ator em relação aos parâmetros do Ator. Ambos os gradientes são avaliados para observação Si.

- Por fim, atualiza o Ator alvo e o Crítico alvo, dependendo do método de atualização suave ou periódica, mostrada respectivamente a seguir:

$$\begin{aligned} \theta q' &= \tau\theta q + (1-\tau)\theta q' \\ \theta\mu' &= \tau\theta\mu + (1-\tau)\theta\mu' \end{aligned} \tag{2.10}$$

$$\begin{aligned} \theta q' &= \theta q \\ \theta\mu' &= \theta\mu' \end{aligned} \tag{2.11}$$

3 Revisão Bibliográfica

Neste capítulo é apresentada uma revisão bibliográfica, também conhecida como estado da arte da pesquisa, no qual são apresentados artigos ciêntificos e trabalhos anteriores que estão relacionados com a presente monografia.

Costa e Filho (2013) realizaram a identificação, por meio de RNA, da mesma coluna de destilação que será usada neste trabalho. Foram coletados os dados da vazão de entrada e da temperatura de topo da coluna. O modelo criado foi validado de forma online, atráves de dados lidos em tempo real do sistema, evidenciando a eficácia das RNA's para registrar características dinâmicas do processo estudado.

Silver et al. (2014) expandiram o estudo sobre algoritmos para gradientes de política determinística, conhecidos como *Deterministic Policy Gradient Algorithms*, ou DPG. Esses gradientes podem ser estimados de forma mais eficiente do que suas contrapartes estocásticas, evitando uma integral problemática sobre o espaço de ação.

Na prática, o Ator-Crítico determinístico superou significativamente sua contraparte estocástica por várias ordens de magnitude, e resolveu um problema desafiador de aprendizagem por reforço com 20 dimensões de ação contínua e 50 dimensões de estado.

Lillicrap et al. (2015) apresentaram um algoritmo para um controlador livre de modelo, também pelo método Ator-Crítico, onde combinaram os avanços recentes em aprendizado profundo, como o *Deep Q Network (DQN)* e aprendizado por reforço, como o *Deterministic Policy Gradient (DPG)*.

O resultado disso foi um algoritmo que soluciona de forma robusta problemas em vários domínios com espaços de ação contínua, chamado de *Deep Deterministic Policy Gradient (DDPG)*, onde obtiveram sucesso em sua validação conseguindo resultados melhores que os outros algoritmos.

Gonalvez (2016) projetou um controlador adaptativo utilizando aprendizado por reforço para controle de nível em um processo de quatro tanques, configurando um sistema multivariável. O controlador utiliza o método Ator-Crítico com aproximação de funções por RNA de base radial e treinamento através do gradiente descendente do erro de diferença temporal.

Os resultados simulados demonstram um desempenho superior do controlador utilizando aprendizado por reforço quando comparado a um controlador PI tradicional. Os resultados obtidos pela aplicação no sistema real demonstram a posssibilidade de uso do algoritmo em um sistema de controle.

Spielberg, Gopaluni e Loewen (2017) estenderam o sucesso do aprendizado profundo e aprendizado por reforço para problemas de controle de processos, mostrando que, se as funções de recompensa forem formuladas corretamente, eles podem ser utilizados para o controle de processos industriais.

Os autores da pesquisa utilizaram as redes neurais profundas (do inglês *deep neural network*) para servir como aproximadores de função e assim aprender as políticas de controle. Uma vez treinada, a rede adquire uma política que mapeia a saída do sistema para controlar ações. Embora as políticas não sejam explicitamente especificadas, as redes neurais profundas são capazes de aprender políticas que são diferentes dos tradicionais controladores industriais. A abordagem foi avaliada tanto em sistemas SISO quanto em sistemas MIMO e testado em vários cenários distintos, porém apenas em sistemas lineares.

Spielberg, Gopaluni e Loewen (2018) compararam a eficácia do controlador por aprendizado por reforço, pelo método Ator-Crítico, em relação ao controlador predititivo por modelo (do inglês *Model Predictive Control - MPC*). Onde demostraram os benefícios do mesmo, simulando-os em sistemas SISO e em sistemas MIMO, e desta vez também testaram em sistemas não lineares e com distúrbios externos.

No Capítulo 4 será apresentado as etapas do desenvolvimento do presente trabalho, assim como as ferramentas e programas utilizados.

4 Projeto e Desenvolvimento

Neste capítulo é apresentado como ocorreu todo o desenvolvimento da pesquisa proposta, mostrando desde a aquisição dos dados da planta, passando pelo tratamento dos mesmos, e posteriormente utilizando-os para treinar as redes neurais artificiais. Também serão descritos os passos que foram necessários para o desenvolvimento do controlador e de seus algoritmos de aprendizagem.

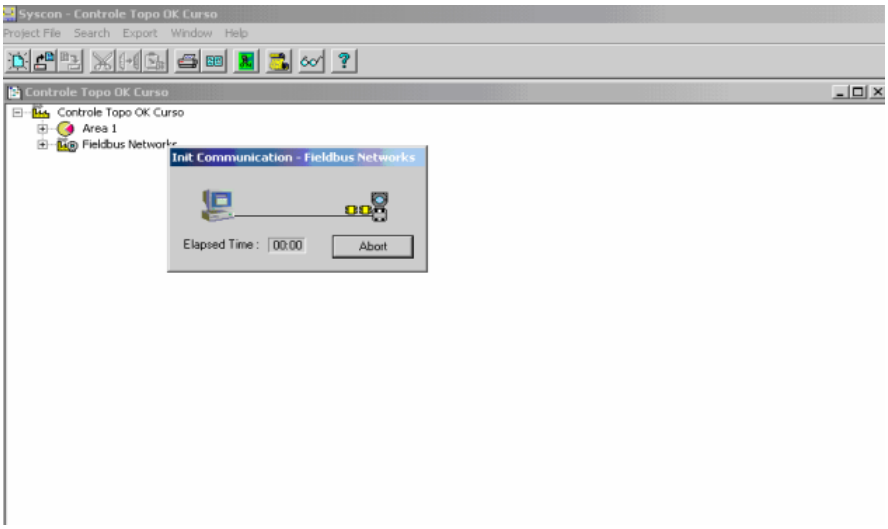
4.1 Aquisição de Dados da Planta

Entre os softwares que foram utilizados para a coleta de dados está o MATLAB®, acrônimo para *MATrix LABoratory*, que foi criado pela MathWorks. Ele teve a função de receber os dados da coluna de destilação, para poder vizualizar e posteriormente modificá-los. Além disso, o MATLAB® possui uma importante ferramente chamada de Simulink, que tem a função de simular sistemas dinâmicos em forma de diagramas de blocos.

O MATLAB® recebeu os dados da planta didática com o auxílio de uma ferramenta chamada OPCTool, que utiliza o método de comunicação OPC *(Ole for Process Control)*, com larga aceitação industrial e arquitetura aberta. Esta ferramenta permitiu a comunicação com o software Syscon®, que é o configurador de sistema desenvolvido pela SMAR para atender os produtos Foundation Fieldbus, que tem a função de realizar a configuração, manutenção e operação desses produtos em questão, bastando que haja um computador pessoal que tenha uma interface Fieldbus.

Na Figura 11 é apresentada a tela onde é inicializada a comunicação do software Syscon®.

Figura 11 – Tela do Syscon



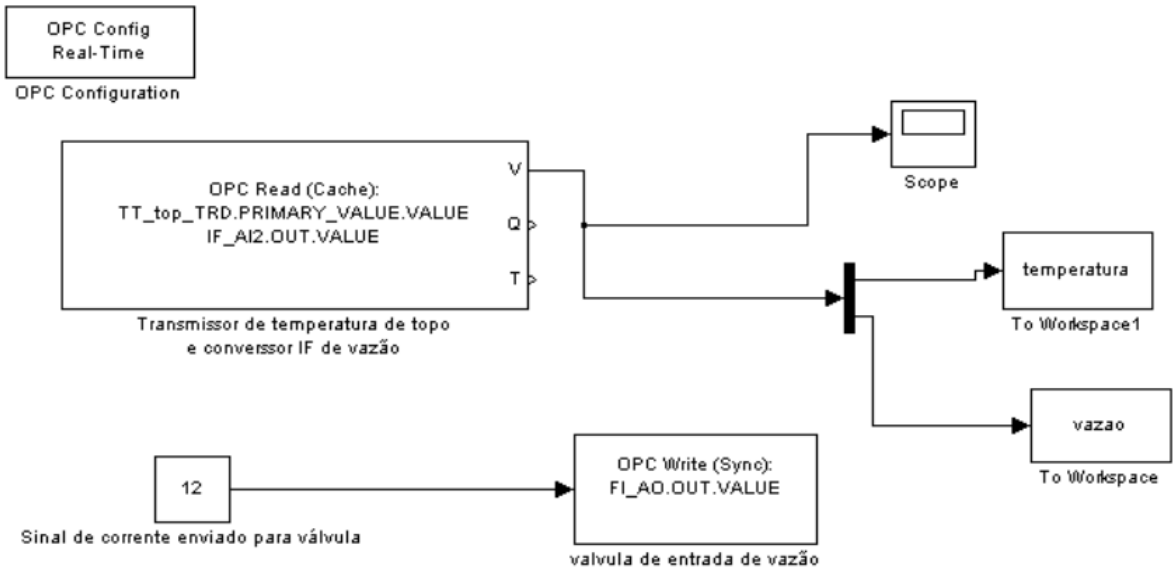
Fonte: Autor.

Para a coleta de dados que melhor representasse a dinâmica da coluna de destilação, foram realizados testes do tipo degrau. Nesse tipo de teste é provocada uma mudança de patamar na variável de entrada do sistema e é verificado o comportamento da variável de saída.

Nos testes aplicados, foram coletados os dados das variáveis de vazão de entrada da coluna, que é modificada pela abertura e fechamento da válvula controladora de temperatura (TCV), localizada no quinto prato da torre, e da temperatura de topo, que é influenciada pela vazão de entrada.

Na Figura 12 é apresentada a tela do Simulink, com os blocos OPC das variáveis da coluna de destilação.

Figura 12 – Tela do Simulink para Aquisição de Dados do Teste

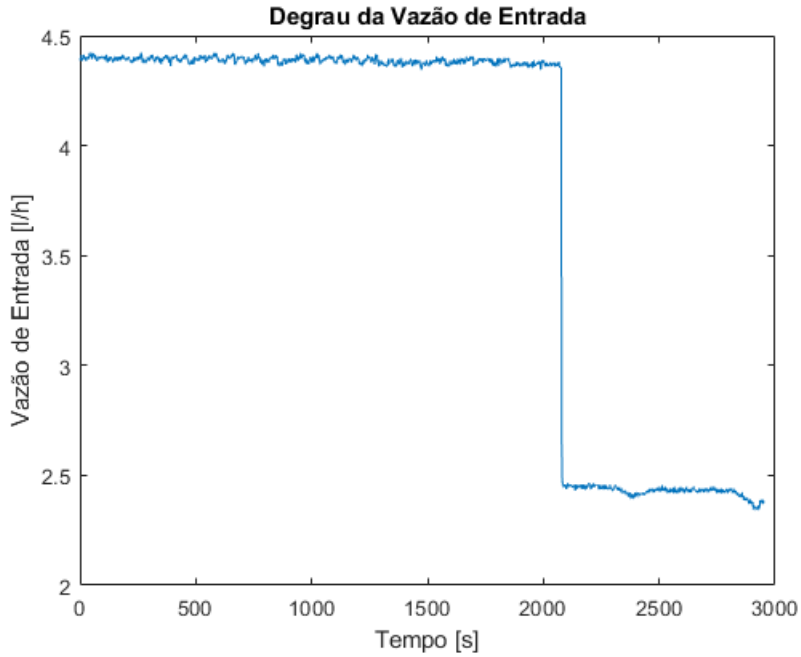


Fonte: Autor.

Na aplicação do degrau, foram utilizados dois valores para o sinal de abertura da válvula de controle, que varia de 4 a 20 mA. O primeiro foi de 14mA, que representava 4.4 l/h na vazão de entrada. Em seguida o sinal foi mudado para 12 mA, equivalente a 2.2 l/h de vazão.

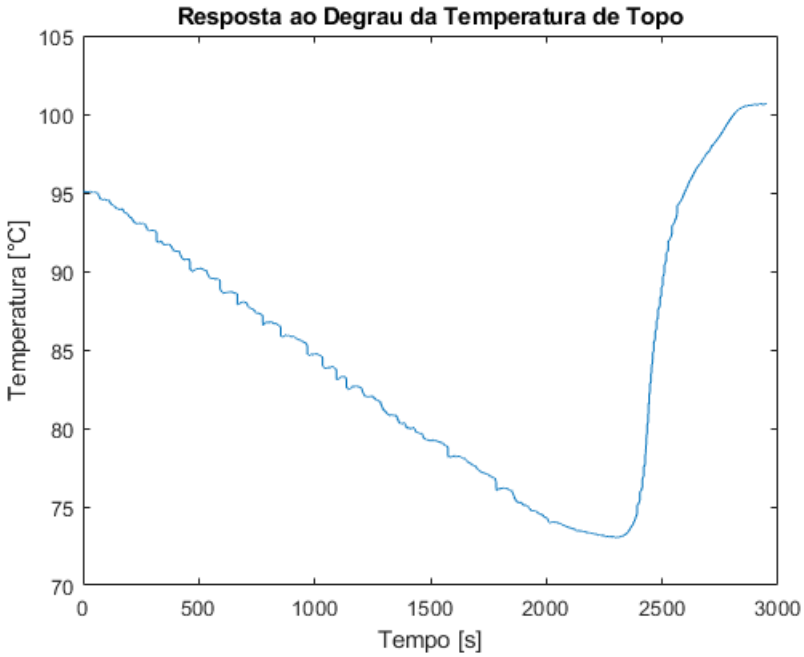
Com vistas a obter uma quantidade adequada de dados para realizar uma boa identificação da dinâmica da coluna, a duração do experimento foi de 50 minutos. As Figuras 13 e 14 apresentam os valores de vazão e temperatura coletadas durante a realização do experimento.

Figura 13 – Degrau da Vazão de Entrada do Teste



Fonte: Autor.

Figura 14 – Resposta ao Degrau da Temperatura de Topo do Teste



Fonte: Autor.

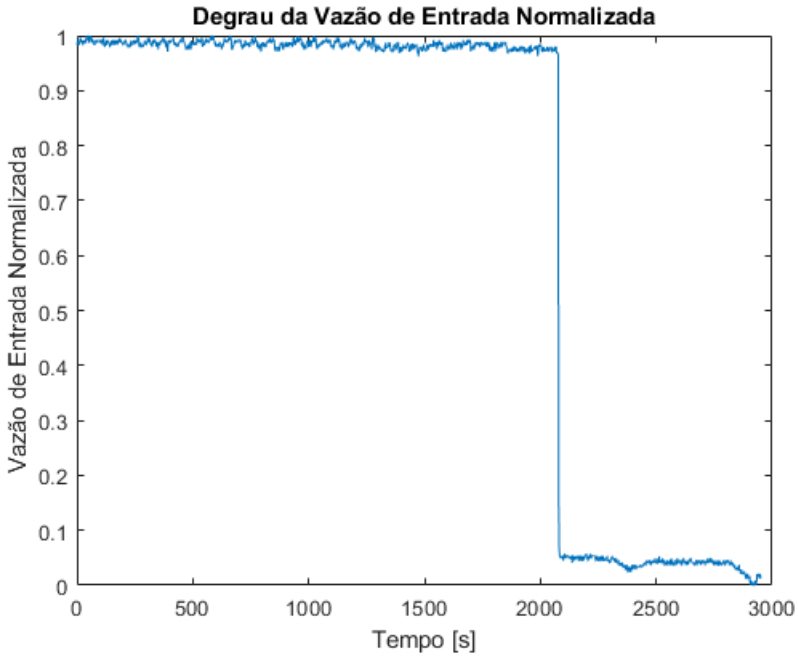
4.2 Tratamento dos Dados

Antes que os dados sejam utilizados para treinar as redes neurais, é preciso que eles sejam tratados. Logo, é mostrado como ocorreu cada passo de seu tratamento feito pelo MATLAB®.

A primeira etapa foi a **normalização**, que consiste em retirar a variável de sua escala real e colocá-la em uma escala representativa, mantendo as características originais da mesma. É um processo importante para diminuir possíveis divergências nos resultados obtidos nos testes e também para a adequação dos dados ao padrão estabelecido pela RNA. A escala representativa, deve ser normalizada de acordo com as funções de ativação que serão utilizadas nos neurônios da camada intermediária da RNA. A escala escolhida vai de 0 a 1.

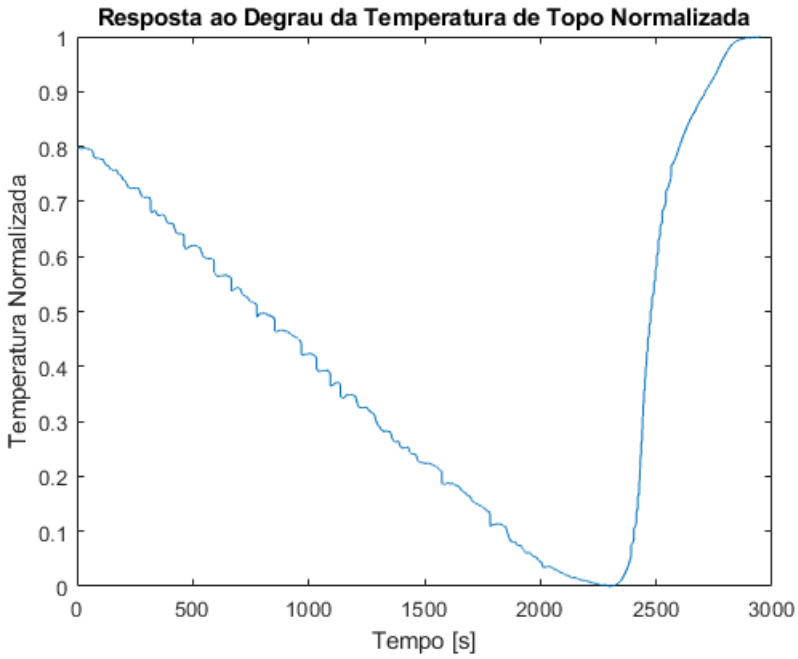
As Figuras 15 e 16 apresentam respectivamente a vazão de entrada e a temperatura de topo normalizadas.

Figura 15 – Vazão de Entrada Normalizada



Fonte: Autor.

Figura 16 – Temperatura de Topo Normalizada

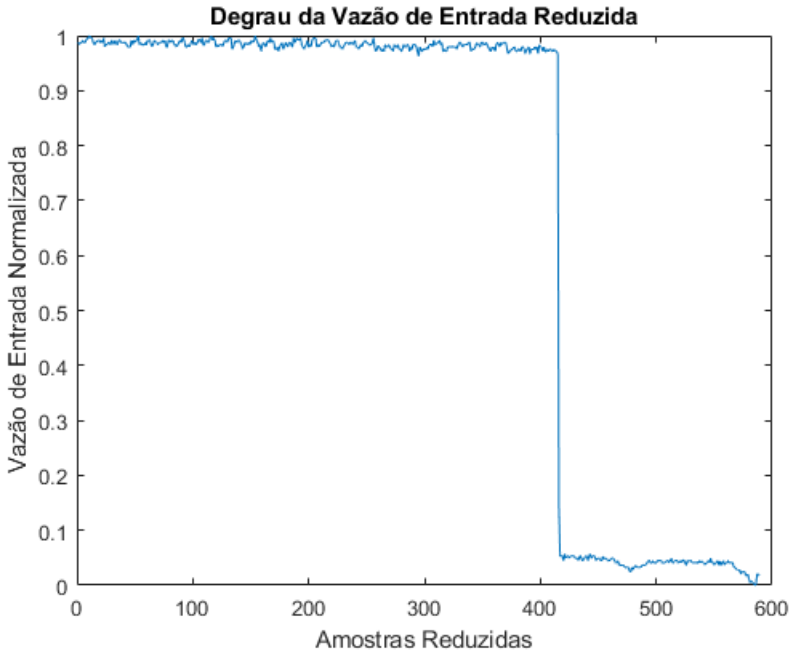


Fonte: Autor.

O próximo passo do tratamento é a **redução do número de amostras**, que similarmente a normalização, deve ocorrer de forma a manter as características originais do sistema. Com isso a matriz de dados pode ser representada pela matriz reduzida com um número menor de amostras, mas contendo a maior parte das informações importantes dos dados originais, facilitando assim o futuro treinamento da rede e preparando para o próximo passo de tratamento que é a criação dos atrasos nas variáveis.

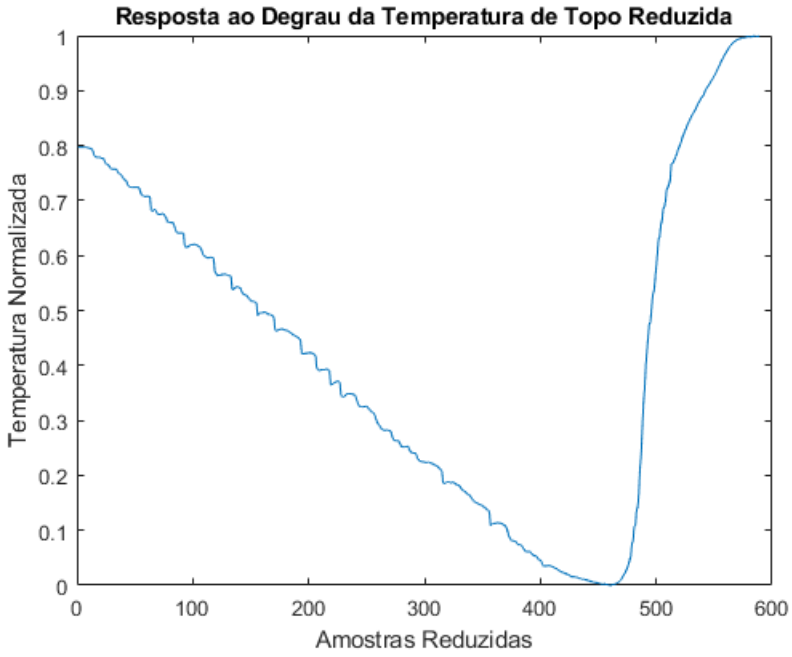
As Figuras 17 e 18 apresentam respectivamente o número de amostras da vazão de entrada e da temperatura de topo reduzidas de 5900 para 590 amostras.

Figura 17 – Vazão de Entrada Reduzida



Fonte: Autor.

Figura 18 – Temperatura de Topo Reduzida

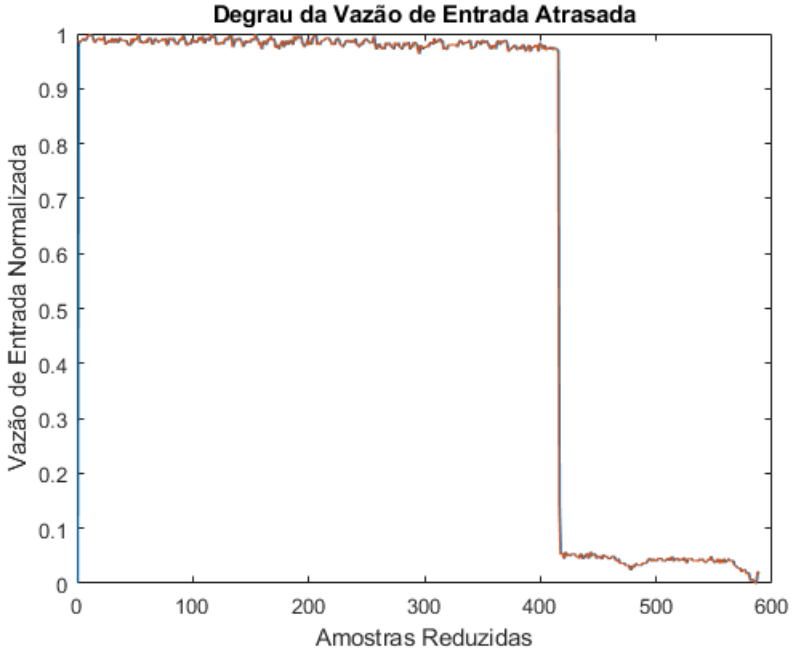


Fonte: Autor.

Por fim, foi realizado o **atraso das variáveis** que serve para gerar novas entradas para o treinamento da rede neural. Por ser um sistema de comportamento dinâmico e instável, o atraso das variáveis tem o objetivo de representar o comportamento real do sistema, pois, a vazão de entrada e a temperatura de topo em instantes anteriores influenciam no comportamento do sistema.

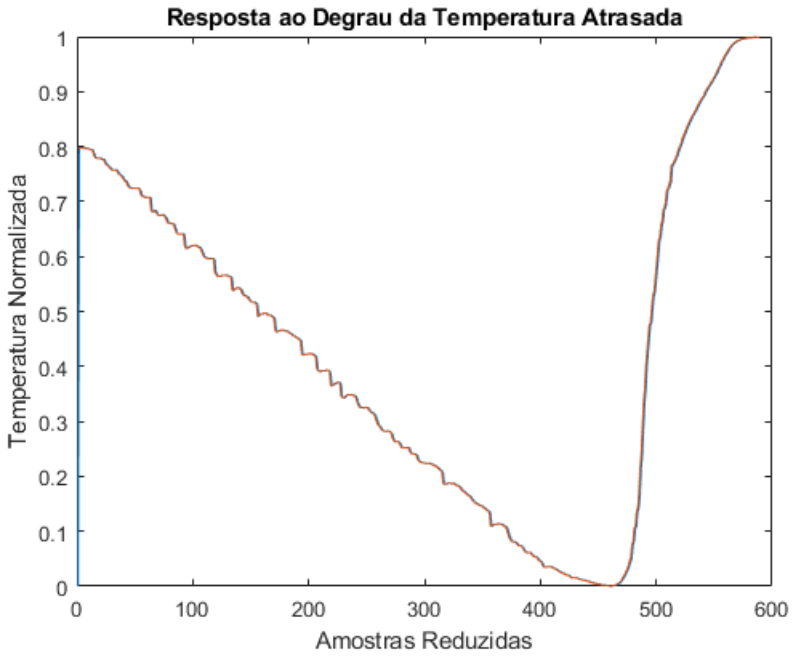
As Figuras 19 e 20 apresentam respectivamente a vazão de entrada e a temperatura de topo, que já haviam sido normalizadas e reduzidas, atrasadas.

Figura 19 – Vazão de Entrada Atrasada



Fonte: Autor.

Figura 20 – Temperatura de Topo Atrasada



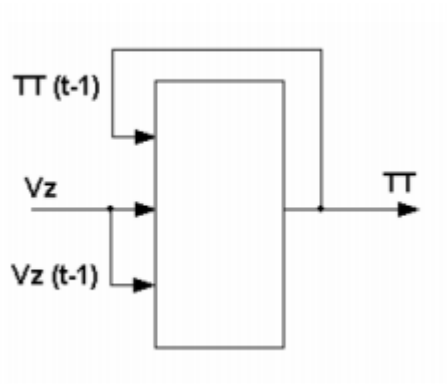
Fonte: Autor.

4.3 Treinamento das Redes Neurais Artificiais

Após o tratamento do conjunto de dados coletados, foi realizado o treinamento das redes neurais artificiais, através da ferramenta do MATLAB chamada de NNTool, abreviação para Neural Network Toolbox. Nela foram importados como entrada do sistema as variáveis de vazão de entrada, vazão de entrada atrasada, e temperatura de topo atrasada, todas em um único vetor de três dimensões. E como saída do sistema, conhecido como o dado alvo da rede, foi importada a variável de temperatura de topo. Todas as variáveis que foram importadas já estavam devidamente normalizadas e reduzidas.

A Figura 21 apresenta a configuração das varíaveis de entrada e saída que foram importadas para o NNTool.

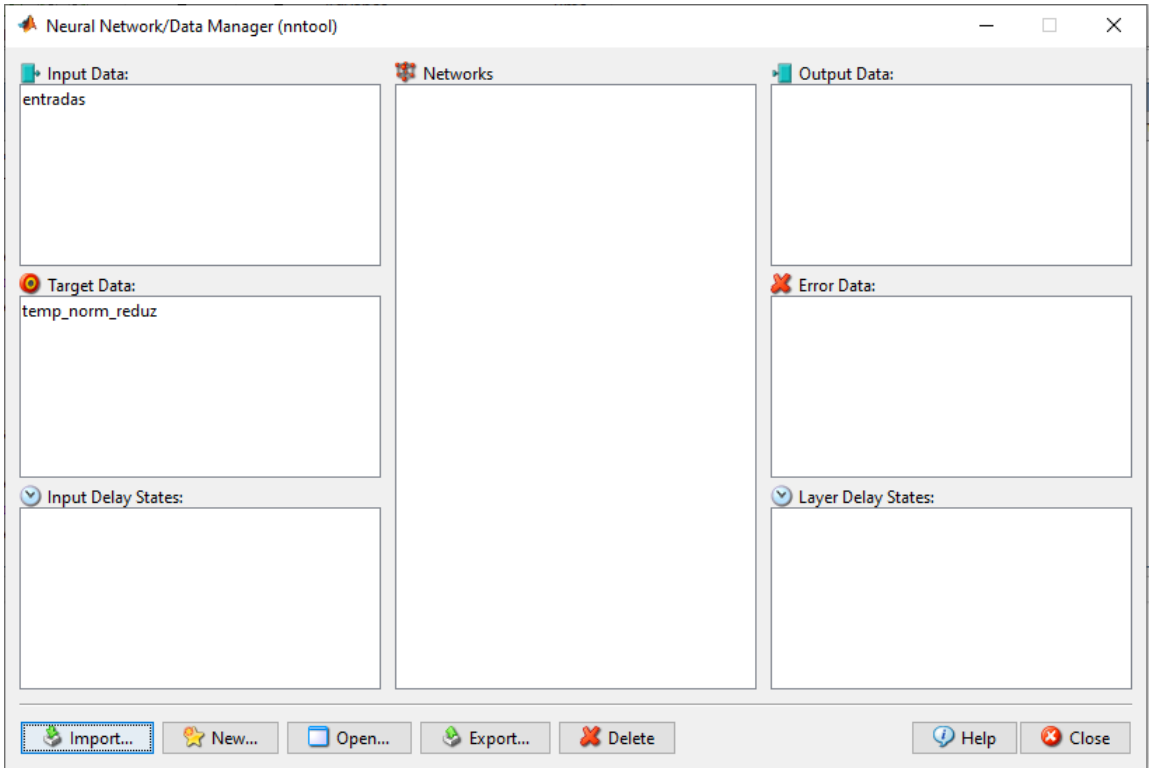
Figura 21 – Configuração das Variáveis Importadas para o NNTool



Fonte:(CARVALHO, 2004).

Na Figura 22 é apresentada a tela inicial da ferramenta NNtool, no qual foram importados os dados das entradas como *Input Data* e da saída como *Target Data*.

Figura 22 – Tela Inicial do NNTool



Fonte: Autor.

Com os dados já importados na ferramenta, foi criado a RNA. Em sua criação é necessário definir diversos parâmetros como: o tipo de arquitetura da rede, o algoritmo de treinamento, a função de treinamento, o número de camadas ocultas, o número de neurônios em cada camada, suas funções de ativação e o critério de avaliação do erro.

No Quadro 1 é apresentada as configurações dos parâmetros da RNA, no qual foram escolhidos baseados em trabalhos anteriores de identificação da coluna de destilação.

Quadro 1 - Configuração dos Parâmetros da RNA

Parâmetro	**Configuração**
Arquitetura da Rede	Feed-Forward
Algoritimo de Aprendizagem	Back-propagation
Função de Treinamento	Levenberg-Marquardt
Função de Ativação	Logarítmica Sigmóide (camada intermediária) e Purelin (camada de saída)
Critério de Desempenho	Mean Squared Error (MSE)

Fonte: Autor.

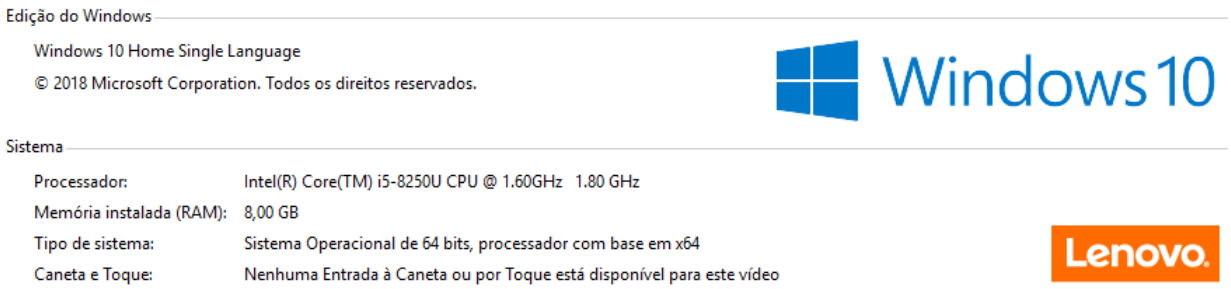
O número de camadas ocultas e de neurônios em cada camada, foi testado em busca de uma configuração que retornasse o melhor ajuste de RNA aos dados.

No Capítulo 5 será apresentado os resultados da identificação da coluna de destilação por RNA.

4.4 Desenvolvimento do Controlador usando Aprendizado por Reforço

Para criar o controlador, foi nessário utilizar a versão mais recente do MATLAB que é a 2019b, onde possui as ferramentas necessárias para o aprendizado por reforço. A configuração do computador utilizado para o desenvolvimento e treinamento do agente é apresentada na Figura 23.

Figura 23 – Configuração do Computador



Fonte: Autor.

Primeiro foi necessário criar o código no editor do MATLAB, que iria gerar o ambiente de treinamento do agente no Simulink. Este pode ser visto no Apêndice A.

Nele foram definidas as observações do ambiente junto com os seus limites inferiores e superiores, e as ações do agente também com seus limites. Além disso foram criadas duas RNA, a primeira foi a do Crítico, que recebeu como entradas as observações e as ações, e teve como parâmetros os neurônios, camadas ocultas e as funções de ativação. Em seguida foi criado a rede neural do Ator que é a responsável por gerar as ações, a qual recebeu como entrada apenas as observações. Em ambas as redes, um importante parâmetro é a da taxa de aprendizagem, que se for muito pequeno irá demorar muito para o agente aprender, e se for grande poderá convergir para um mínimo local.

Foram especificadas as opções para criar o agente por DDPG (*Deep Deterministic Policy Gradient*), que entre os diversos parâmetros os principais são o tempo de amostragem e a variância do ruído que ajudam na exploração do agente. Então, a partir das especificações, do Crítico e do Ator é criado o agente.

Em seguida, foram definidas as opções do treinamento do agente, que é uma das principais etapas do processo. Foi especificado o número máximo de episódios, o número máximo de tempo por episódio e o critério de parada do treinamento.

Por fim, foi criada uma outra função que teve o intuito de resetar o valor de referência da temperatura, e com isso gerar novos valores aleatórios. Para isso foi estipulado

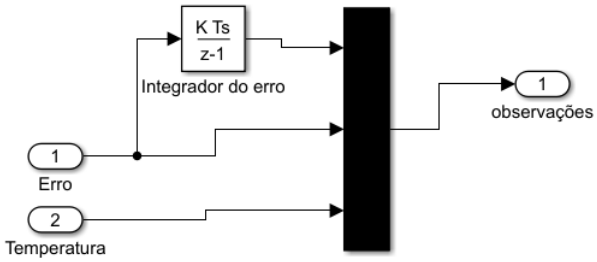
um valor que variava de 70 a 95, pois assim não seriam gerados valores nem muito baixos e nem muito altos para a planta, e com isso o agente pode treinar com diferentes valores de temperatura. Este código pode ser visto no Apêndice B.

Com os códigos prontos, foi então desenvolvido o ambiente no Simulink, onde foi colocado o bloco do subsistema do agente que acaba de ser criado e o subsistema da rede neural representando o modelo da torre de destilação, que foi gerado pelo comando *gensim*.

Além deles, também foram criados outros três subsistemas. O primeiro foi o que gera as observações do ambiente, estas que foram definidas como a temperatura de topo, o erro (diferença da temperatura atual e a da referência) e a integral do erro.

Na Figura 24 é apresentado o subsistema das observações, contendo as entradas, um bloco do multiplexador e a saída.

Figura 24 – Subsistema que Gera as Observações

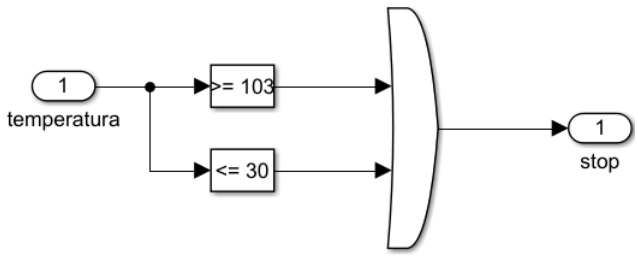


Fonte: Autor.

O segundo subsistema desenvolvido, foi o que determina a parada de um episódio do treinamento do agente. Ele é responsável por garantir que a variável não ultrapasse certos limites impostos, como no caso foi escolhido para quando a temperatura for maior ou igual a 103, ou menor ou igual a 30, a simulação do episódio terminava.

Na Figura 25 apresenta o subsistema de parada do episódio, contendo a entrada da temperatura, dois blocos de comparação de valores, um bloco lógico OU e a saída.

Figura 25 – Subsistema de Parada do Episódio

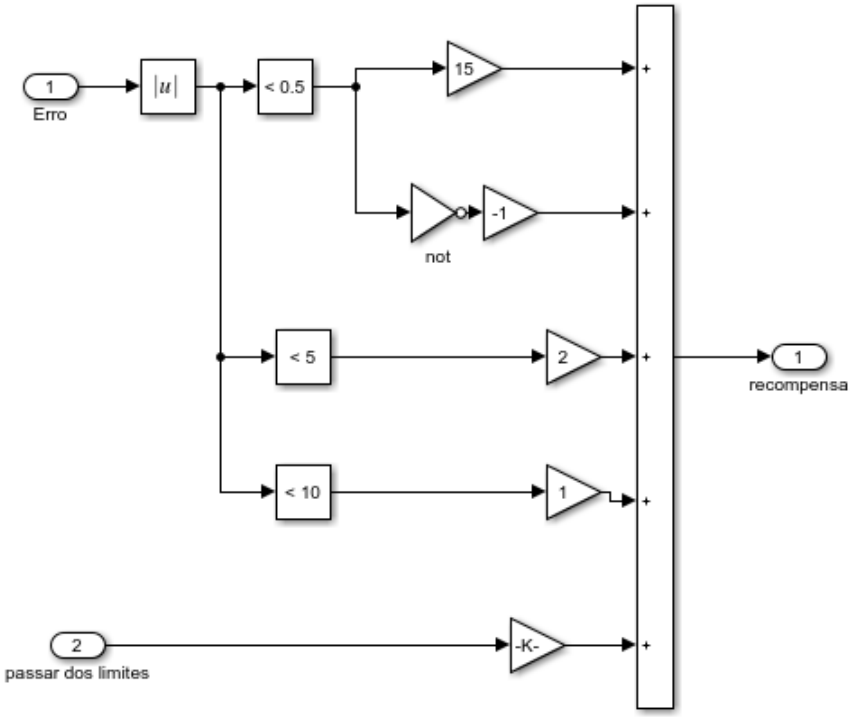


Fonte: Autor.

Por último, o subsistema onde é calculada a recompensa que o agente irá obter ao longo do treinamento. Este bloco, diferente dos demais, é constantemente alterado, pois ao decorrer dos treinamentos é possível avaliar melhor se as recompensas que foram calculadas estão de acordo com o agente e o ambiente ou precisa ser melhorada em algum ponto.

Pode-se observar na Figura 26, que é gerada recompensa positiva a partir do módulo do erro, e quanto menor for este erro maior será a recompensa que o agente receberá. Por outro lado, quando a temperatura passar dos limites do bloco de parada, ou enquanto ele não chegar no erro menor que 0.5 ele receberá uma recompensa negativa, chamada de penalização. As penalizações impostas foram de -1 para quando não alcançasse erro menor do que 0.5 e de -100 quando ultrapassasse os limites.

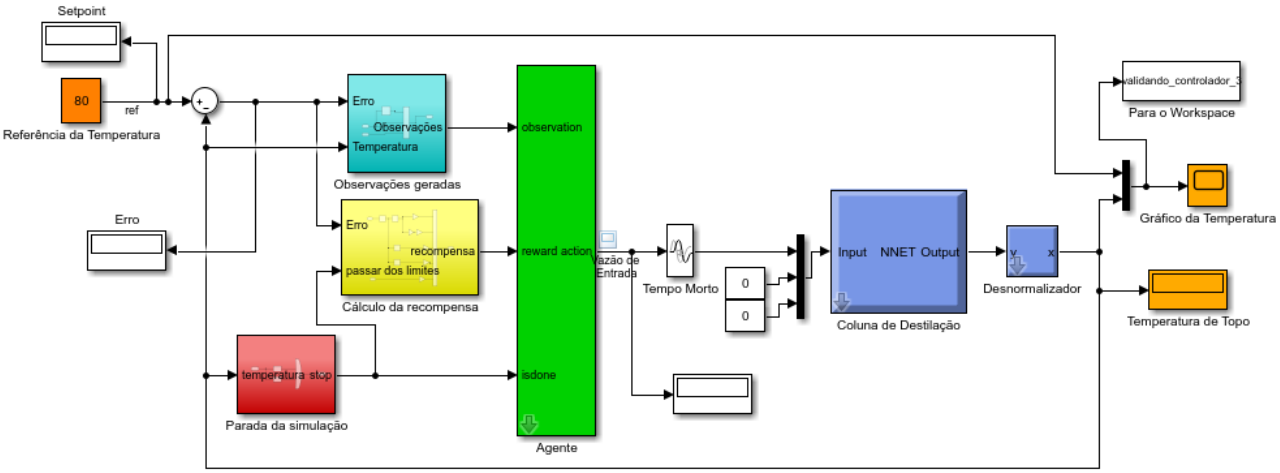
Figura 26 – Subsistema que Calcula a Recompensa



Fonte: Autor.

Além dos subsistemas, pode-se observar na Figura 27 o modelo completo do Simulink, contendo a referência na qual é resetada a cada episódio, um bloco de soma para comparar o sinal de referência com o da temperatura atual, atraso de transporte para simular o tempo morto, um bloco para desnormalizar a temperatura da saída da rede neural criada, displays e gráficos para acompanhar os sinais, e o bloco que envia os dados para o *Workspace*.

Figura 27 – Modelo do Simulink Completo



Fonte: Autor.

No Capítulo 5 serão apresentados os resultados dos treinamentos do agente e de seu desempenho.

5 Resultados

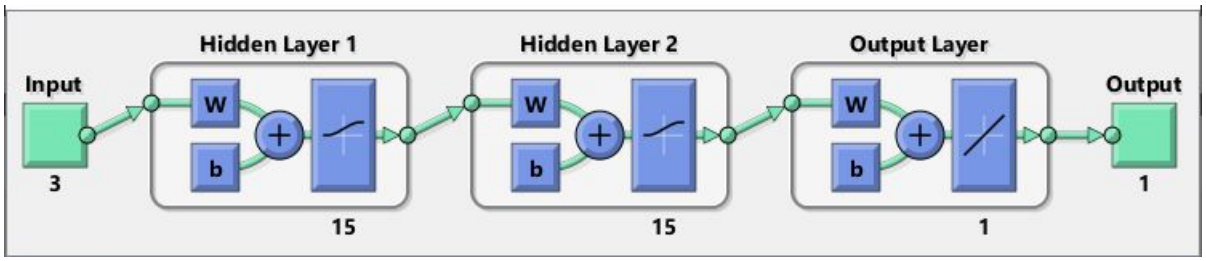
A partir do desenvolvimento demonstrado no capítulo 4, são apresentados aqui os resultados da rede neural criada, do treinamento do agente e do controle do modelo da planta.

5.1 Modelo da Planta

Para encontrar a melhor RNA que representasse o modelo, foram alteradas apenas o número de camadas ocultas e de neurônios. Começando com uma camada até 3 camadas, e 1 neurônio em cada camada até 20 neurônios.

A RNA que obteve maior êxito foi a de duas camadas intermediarias, ambas com a função Logarítmica Sigmóide como função de ativação, e 15 neurônios em cada camada, como pode-se observar na Figura 28.

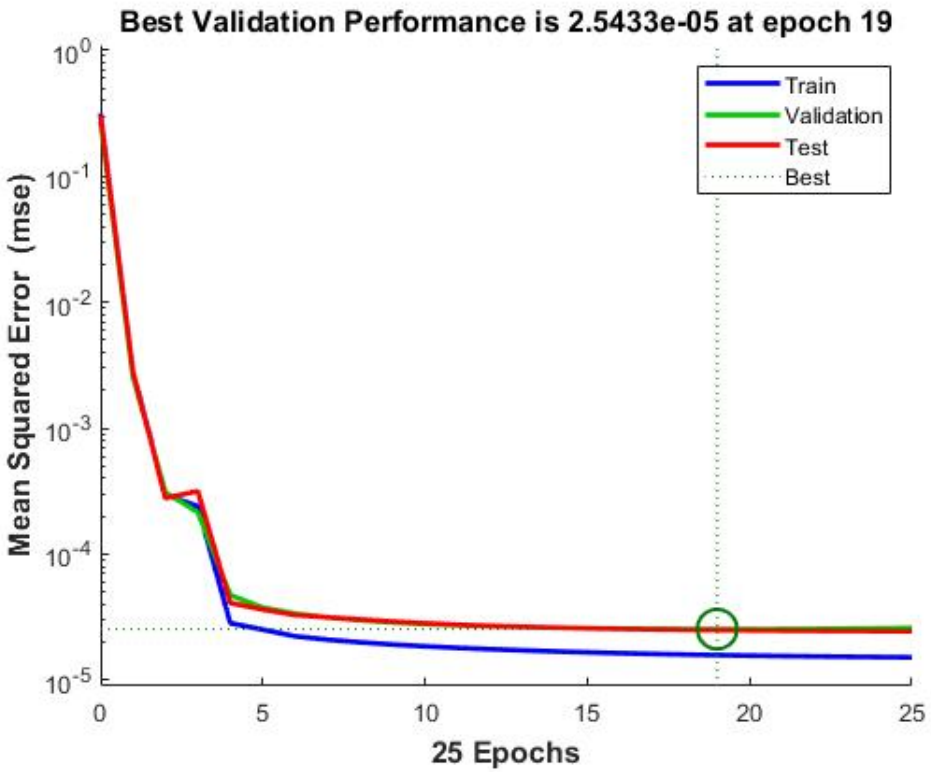
Figura 28 – Topologia da RNA



Fonte: Autor.

Foram necessárias apenas 19 iterações para atingir um valor de erro próximo de tolerância, avaliado pelo índice MSE, que é apresentada na Figura 29.

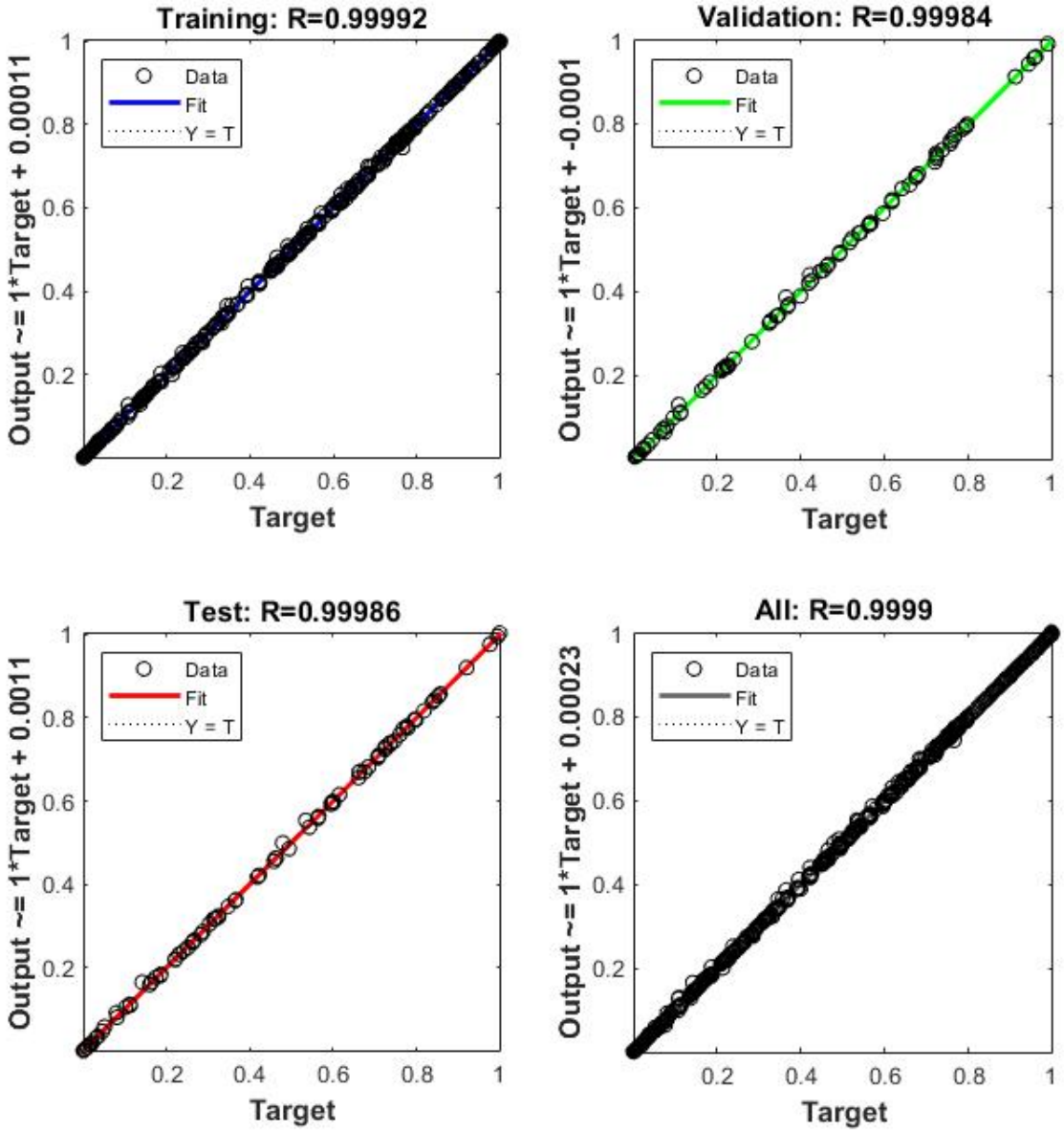
Figura 29 – Performance



Fonte: Autor.

Após o treinamento, foi realizado uma análise por meio da regressão linear. Pode-se observar na Figura 30 que foi separado o conjunto de dados de treinamento, teste e validação, e no final foi realizado a regressão sobre todo o conjunto. O R é uma medida de qualidade do ajuste do modelo em relação à sua capacidade de estimar corretamente os valores de saída, e quanto mais próximo de 1, melhor será o modelo.

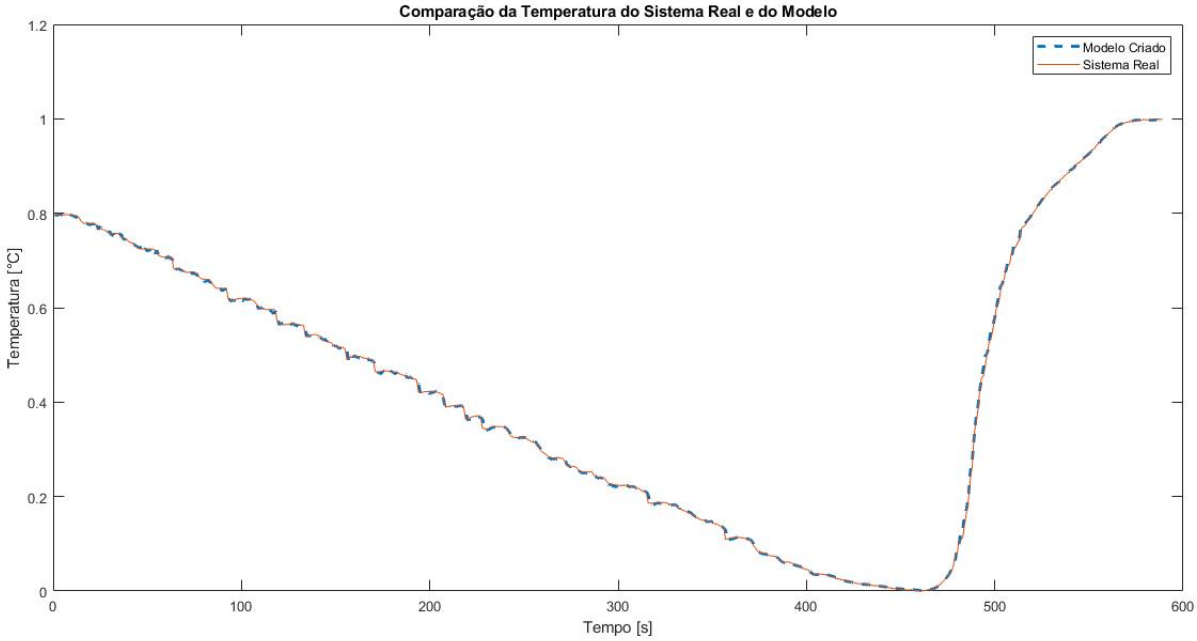
Figura 30 – Gráficos de Regressão Linear



Fonte: Autor.

Com a rede neural devidamente treinada, ela foi exportada junto com seus sinais de saída e de erro para o *Workspace*. Lá, os dados de saída da RNA foram comparadas com os dados de temperatura medidos na coluna de destilação. Na Figura 31 pode-se observar que a RNA consegue descrever com muita exatidão o conjunto de dados da temperatura de topo medida, obtendo um MSE de $1.8633e-05$.

Figura 31 – Comparação da Saída da RNA com a Temperatura Medida



Fonte: Autor.

5.2 Controlador Desenvolvido

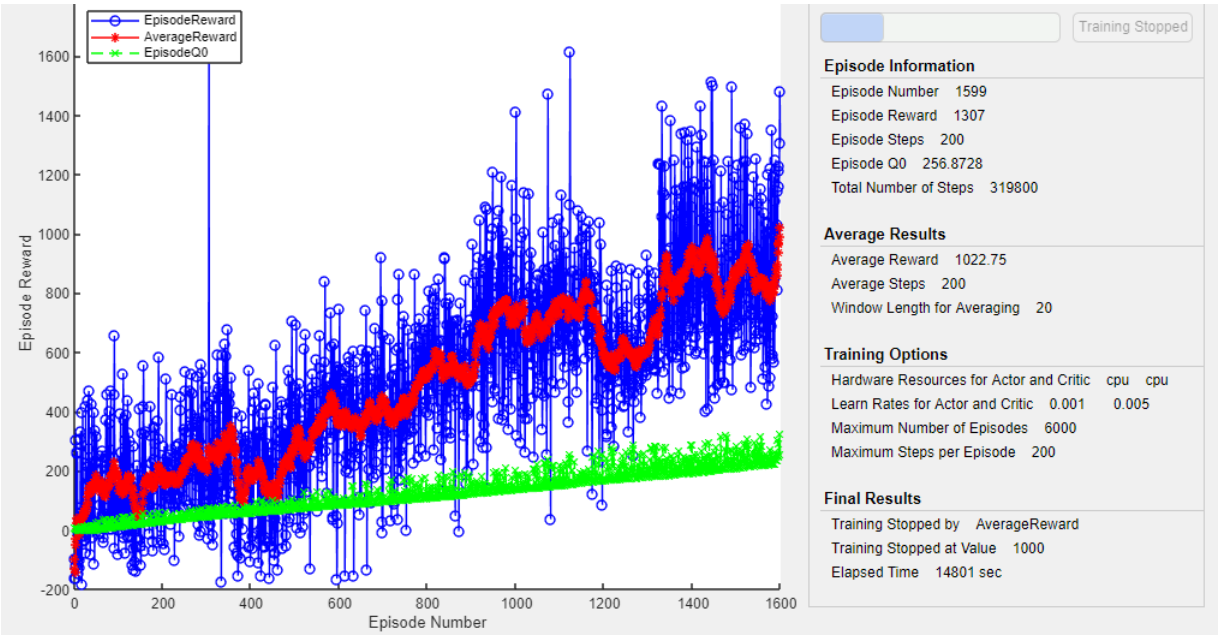
Com o modelo da coluna criado, foi então possível treinar o agente do aprendizado por reforço, que será o controlador do sistema. Esta foi a parte do trabalho que mais consumiu o tempo, devido ao prolongado tempo de treinamento do agente.

Entre muitos treinamentos, diversos parâmetros tiveram que ser alterados, tais como o tempo de simulação por episódio, o critério de parada para quando o agente estiver bem treinado, a taxa de aprendizagem das estruturas do Ator e do Crítico, junto com seus neurônios e camadas e alterar o cálculo da recompensa para que não demorasse muito para que o agente ganhasse recompensa e nem que ele ganhasse recompensa rapidamente.

O treinamento que obteve melhor êxito levou aproximadamente 4 horas e 11 minutos para ser finalizado. Além disso precisou de 1599 episódios, que foi o necessário para alcançar a média da recompensa estipulada que era de 1000.

No gráfico da Figura 32, podem ser vistos três informações, a primeira em azul é a quantidade de recompensa por episódio, a segunda em vermelho é a média dessa recompensa ao longo do treinamento, e a em verde é o *EpisodeQ0* que faz parte da estrutura do Crítico onde é estimado a recompensa a longo prazo do agente.

Figura 32 – Treinamento do melhor Agente de Aprendizado por Reforço

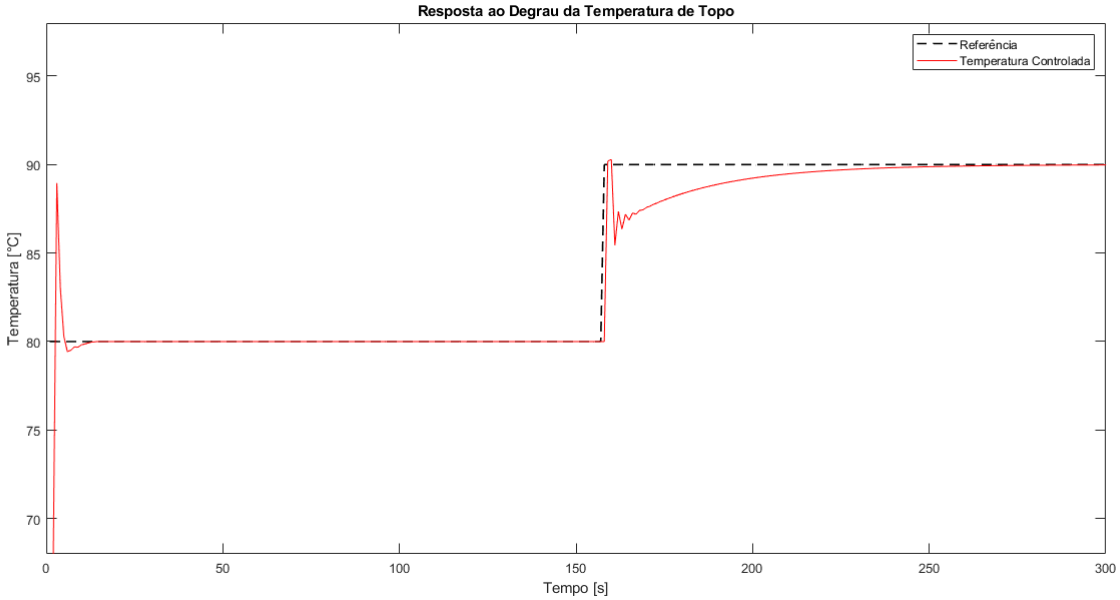


Fonte: Autor.

Uma vez que o agente foi treinado, é realizada a sua validação por meio de testes em degrau em certos sinais de referência. Todos os testes tiverem o mesmo tempo de simulação de 300 segundos, ou 5 minutos, e foram avaliados pela comparação do MSE.

A Figura 33 apresenta o Teste 1, no qual foi aplicado degrau com valor de referência de 80 no início da simulação, e outro degrau com valor de 90 aos 150 segundos. O agente treinado conseguiu controlar a variável do processo nos dois patamares, obtendo sobressinal e rápida resposta no primeiro degrau e sem sobressinal porém com uma resposta mais lenta no segundo degrau.

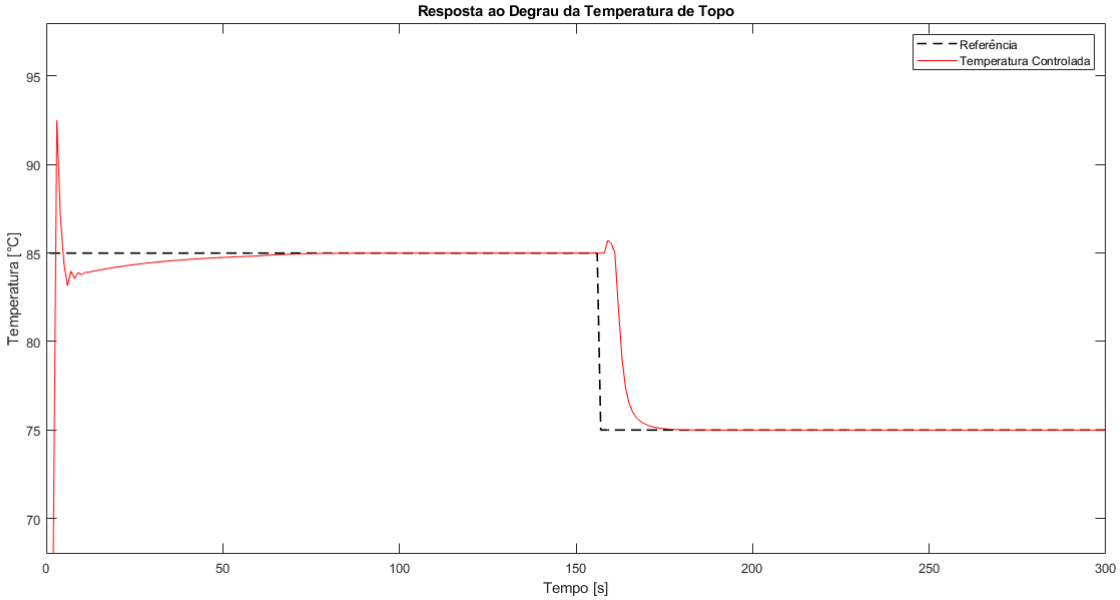
Figura 33 – Validação do Agente - Teste 1



Fonte: Autor.

A Figura 34 apresenta o Teste 2, no qual foi aplicado degrau com valor de referência de 85 no início da simulação, e outro degrau com valor de 75 aos 150 segundos. O agente treinado conseguiu controlar a variável do processo nos dois patamares, obtendo sobressinais pequenos e rápidas respostas.

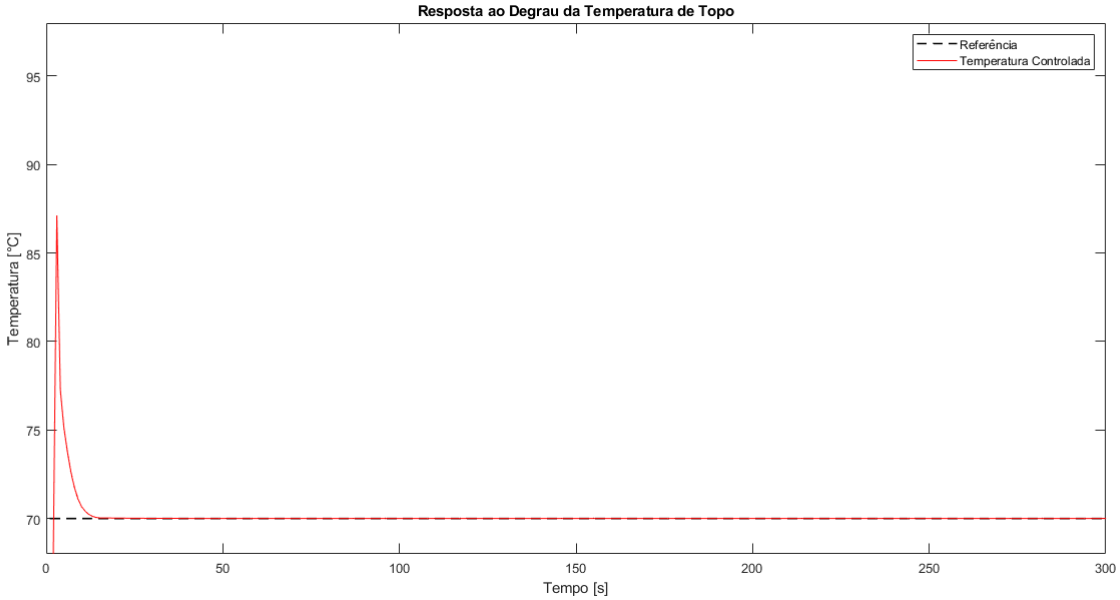
Figura 34 – Validação do Agente - Teste 2



Fonte: Autor.

A Figura 35 apresenta o Teste 3, no qual foi aplicado apenas um degrau no valor de referência do limite inferior que o agente que havia sido treinado, que era de 70. O agente treinado obteve um sobressinal elevado em comparação aos outros teste, entretanto alcançou rapidamente o valor desejado da variável do processo.

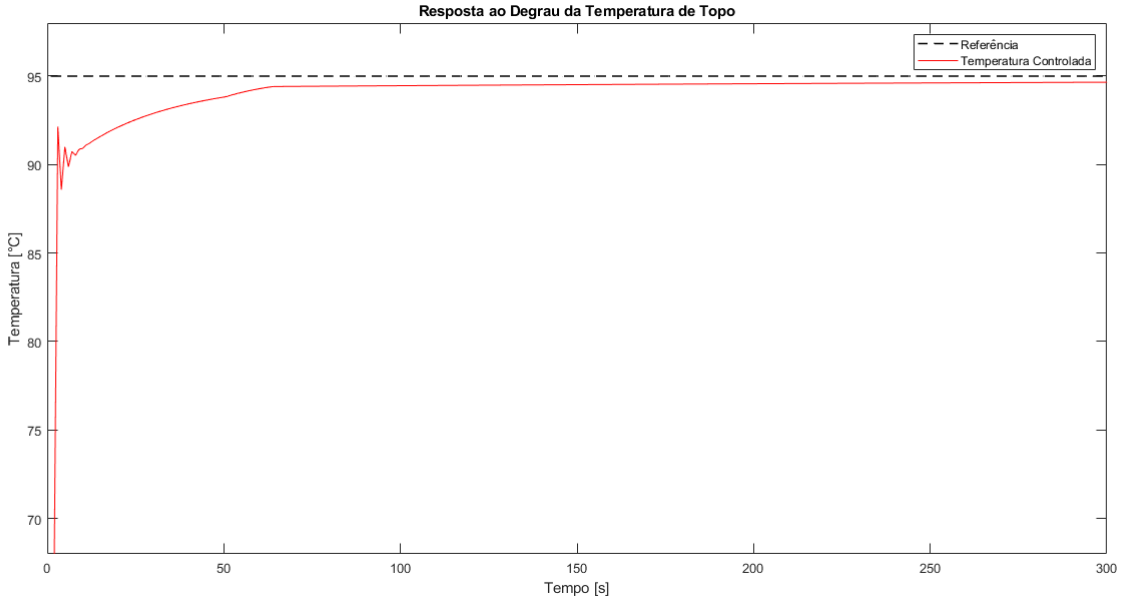
Figura 35 – Validação do Agente - Teste 3



Fonte: Autor.

Por último, a Figura 36 apresenta o Teste 4, no qual também foi aplicado apenas um degrau, porém no valor de referência do limite superior que o agente que havia sido treinado, que era de 95. Este teve um pouco mais de dificuldade para controlar, precisando de maior tempo para alcançar o valor de referência, entretanto não obteve sobressinal.

Figura 36 – Validação do Agente - Teste 4



Fonte: Autor.

Na Tabela 1, é mostrado o desempenho de cada teste de validação do agente, avaliado pelo índice MSE.

Tabela 1 – Desempenho dos Testes de Validação do Agente

Teste	**Degrau**	**MSE**
1	80 –> 90	$7.2102e+3$
2	85 –> 75	$6.4049e+3$
3	70	$4.9091e+3$
4	95	$8.9342e+3$

Fonte: Autor.

6 Conclusão

Este Trabalho de Conclusão de Curso alcançou com êxito os objetivos propostos, visto que conseguiu criar o modelo da coluna de destilação didática, desenvolver o controlador baseado em aprendizado por reforço e controlar o modelo computacional da planta. De acordo com os valores do índice MSE obtidos do modelo criado e do controle realizado, os resultados foram satisfatórios.

O controlador desenvolvido pelo aprendizado por reforço se mostrou de grande aplicabilidade, visto que conseguiu controlar o modelo da planta em diversos valores e com pouca dificuldade. Entretanto, o tempo de treinamento do agente e os inúmeros parâmetros que podem ser modificados no treinamento do aprendizado por reforço, podem deixar o desenvolvimento do controlador bastante demorado. Para amenizar tal problema no treinamento, sugere-se a utilização de computação paralela com o CPU (*Central Processing Unit*) ou GPU (*Graphics Processing Unit*), algo que não foi realizado neste trabalho.

Também é notório, que o uso das RNA em sistemas de controle é de grande ajuda. Visto que foram utilizadas tanto na modelagem de um sistema complexo como a torre de destilação, como no algoritmo de treinamento do controlador.

Para trabalhos futuros recomenda-se realizar o controle da coluna de destilação Física, utilizando o controlador criado. Além disso, pode ser expandido o controle para as outras variáveis da coluna, tais como o nível da base e a pressão do topo.

A Código do aprendizado por reforço

```
% Treinando o controlador do aprendizado por reforco no modelo neural da
    torre de destilacao
% Utilizando o modelo com a vazao de entrada e temperatura de topo
clc

% Criando o ambiente do Controle de Temperatura
obsInfo = rlNumericSpec([3 1],...
    'LowerLimit',[-inf -inf -inf ]',...
    'UpperLimit',[ inf  inf inf ]');
obsInfo.Name = 'Observa  es';
obsInfo.Description = 'Integral do erro; Erro; Temperatura medida';

% Gerando as acoes do agente
actInfo = rlNumericSpec([1 1],'LowerLimit',[0],'UpperLimit',[1]);
actInfo.Name = 'Vaz o de Entrada';
numActions = actInfo.Dimension(1);

% Definindo o ambiente
modelo = 'controlando_modelo_temp_usando_rl';
agente = 'controlando_modelo_temp_usando_rl/Agente';

env = rlSimulinkEnv(modelo, agente, obsInfo, actInfo);

% Funcao que reseta e gera valores aleatorios para os Setpoints
 env.ResetFcn = @(in)localResetFcn(in);

% Especificando o tempo de simulacao Tf e o tempo de amostragem do agente
    Ts
Tf = 20;
Ts = 0.1;

% Fixando o seed do gerador de aleatoriedade.
rng(10)

% Criando a rede neural do Critico, com 2 camadas de entradas, as
    observacoes e as acoes, e uma saida.
statePath = [
    imageInputLayer([3 1 1],'Normalization','none','Name','State')
    fullyConnectedLayer(150,'Name','CriticStateFC1')
    reluLayer('Name','CriticRelu1')
    fullyConnectedLayer(150,'Name','CriticStateFC2')
    reluLayer('Name','CriticRelu2')
    fullyConnectedLayer(150,'Name','CriticStateFC3')];
```

```
41
42 actionPath = [
43     imageInputLayer([numActions 1 1],'Normalization','none','Name','Action'
       )
44     fullyConnectedLayer(150,'Name','CriticActionFC1')];
45
46 commonPath = [
47     additionLayer(2,'Name','add')
48     reluLayer('Name','CriticCommonRelu')
49     fullyConnectedLayer(1,'Name','CriticOutput')];
50
51 criticNetwork = layerGraph();
52 criticNetwork = addLayers(criticNetwork,statePath);
53 criticNetwork = addLayers(criticNetwork,actionPath);
54 criticNetwork = addLayers(criticNetwork,commonPath);
55 criticNetwork = connectLayers(criticNetwork,'CriticStateFC3','add/in1');
56 criticNetwork = connectLayers(criticNetwork,'CriticActionFC1','add/in2');
57
58
59 % Especificando as opcoes para a representacao do Critico
60 criticOpts = rlRepresentationOptions('LearnRate',5e-03,'GradientThreshold'
       ,1);
61 critic = rlRepresentation(criticNetwork,obsInfo,actInfo,'Observation',{'
       State'},'Action',{'Action'},criticOpts);
62
63
64 % Construindo o Ator, usando as observacoes como camada de entrada
65 actorNetwork = [
66     imageInputLayer([3 1 1],'Normalization','none','Name','observation')
67     fullyConnectedLayer(150, 'Name','actorFC1')
68     reluLayer('Name','ActorRelu1')
69     fullyConnectedLayer(150, 'Name','actorFC2')
70     reluLayer('Name','ActorRelu2')
71     fullyConnectedLayer(1, 'Name','actorFC3')
72     tanhLayer('Name','actorTanh')
73     scalingLayer('Name','ActorScaling1','Scale',0.7,'Bias',0.3)];
74
75 actorOptions = rlRepresentationOptions('LearnRate',1e-3,'GradientThreshold'
       ,1,'L2RegularizationFactor',1e-4);
76 actor = rlRepresentation(actorNetwork,obsInfo,actInfo,...
77     'Observation',{'observation'},'Action',{'ActorScaling1'},actorOptions);
78
79 % Especificando as opcoes para criar o agente por DDPG (Deep Deterministic
       Policy Gradient)
80 agentOpts = rlDDPGAgentOptions(...
81     'SampleTime',Ts,...
82     'TargetSmoothFactor',1e-3,...
```

```
83      'DiscountFactor',0.99, ...
84      'MiniBatchSize',64, ...
85      'ExperienceBufferLength',1e6);
86  agentOpts.NoiseOptions.Variance = 0.2;
87  agentOpts.NoiseOptions.VarianceDecayRate = 1e-5;
88
89  % Criando o agente usando as representacoes do Ator, do Critico e as opcoes
        do agente
90  agent = rlDDPGAgent(actor,critic,agentOpts);
91
92  % Treinando o Agente
93  % Especificando as funcoes de treinamento
94  maxepisodes = 6000;
95  maxsteps = ceil(Tf/Ts);
96  trainOpts = rlTrainingOptions(...
97      'MaxEpisodes',maxepisodes, ...
98      'MaxStepsPerEpisode',maxsteps, ...
99      'ScoreAveragingWindowLength',20, ...
100     'Verbose',false, ...
101     'Plots','training-progress',...
102     'StopTrainingCriteria','AverageReward',...
103     'StopTrainingValue',1000);
104
105 doTraining = false;    % true se quiser treinar um novo agente, e false para
        carregar um agente ja treinado
106 if doTraining
107     % Treinando novo agente
108     trainingStats = train(agent,env,trainOpts);
109     save("agente_modelo_temp_18.mat",'agent')
110 else
111     % Carregando um agente ja treinado
112     load('C:\Users\paulo\Documents\PV\TCC\Desenvolvimento\Controlador\
        Controlador Temperatura\Imagens do Simulink do Melhor Controlador 2\
        agente_modelo_temp_16','agent')
113 end
```

B Função que gera novos valores de referência

```
function in = localResetFcn(in)
% Gerando Sinais de Setpoints Aleatorios

% Setpoint da Temperatura - Variando de 70 - 95
blk = sprintf('controlando_modelo_temp_usando_rl/SP da Temperatura');
h = 25*rand + 70
while h <= 27 || h >= 103
    h = 25*rand + 70;
end
in = setBlockParameter(in,blk,'Value',num2str(h));


end
```

Referências

AGUIRRE, L. A. Introdução a identificacão de sistemas: Técnicas lineares e não-lineares aplicadas a sistemas reais. 2000. Citado na página 19.

ÅSTRÖM, K. J.; WITTENMARK, B. *Adaptive control*. [S.l.]: Courier Corporation, 2013. Citado na página 20.

BARR, A.; FEIGENBAUM, E. A. *The handbook of artificial intelligence*. [S.l.]: Butterworth-Heinemann, 1981. v. 2. Citado na página 14.

CALMON, A.; PINHEIRO, N. C.; FERREIRA, R. Desenvolvimento de um robô-cachorro comportamental : percepção e modelagem comportamental. 12 2006. Citado na página 32.

CAMPOS, M. C. M. M. de; TEIXEIRA, H. C. *Controles típicos de equipamentos e processos industriais*. [S.l.]: Edgard Blücher, 2006. Citado na página 21.

CAMPOS, M. M. de; SAITO, K. *Sistemas inteligentes em controle e automação de processos*. [S.l.]: Ciência Moderna, 2004. Citado na página 24.

CARVALHO, A. S. *Identificação por RNA da relação vazão de alimentação por temperatura de topo de uma coluna de destilação didática*. [S.l.]: Monografia Tecnólogo de Automação Industrial, IFF, 2004. Citado na página 45.

CARVALHO, A. S. *Modelagem de Colunas de Destilação Através de Modelos Auto-Regressivos*. [S.l.]: Dissertação de Mestrado apresentada ao Centro de Ciências e Tecnologia, UENF, 2008. Citado 2 vezes nas páginas 22 e 23.

COSTA, A.; FILHO, J. *Identificação Offline e Validação Online de Modelo Computacional para uma Coluna de Destilação*. [S.l.]: Trabalho de Conclusão de Curso, IFF, 2013. Citado na página 34.

DORF, R. C.; BISHOP, R. H. *Modern control systems*. [S.l.]: Pearson, 2011. Citado na página 19.

FERNANDES, R. A. S. *Identificação de Fontes de Correntes Harmônicas por Redes Neurais Artificiais*. [S.l.]: Dissertação de Mestrado de Engenharia Elétrica, USP, 2009. Citado na página 26.

FLAUZINO, S. S. *Redes neurais artificiais para engenharia e ciências aplicadas*. [S.l.]: São Paulo: Artliber Editora, 2010. Citado na página 27.

FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. *Sistemas de controle para engenharia*. [S.l.]: Bookman Editora, 2013. Citado na página 18.

GONALVEZ, D. V. *Controle Adaptativo de Processo de Nível Utilizando Aprendizado por Reforço Ator-crítico*. [S.l.]: Trabalho de Conclusão de Curso, UNB, 2016. Citado na página 34.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2007. Citado 3 vezes nas páginas 25, 26 e 28.

HECHT-NIELSEN, R. Theory of the backpropagation neural network. In: *Neural networks for perception*. [S.l.]: Elsevier, 1992. p. 65–93. Citado na página 25.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, National Acad Sciences, v. 79, n. 8, p. 2554–2558, 1982. Citado na página 24.

JR, C. L. N.; YONEYAMA, T. Inteligência artificial em controle e automação. *Editora Edgard Blücher Ltda*, 2000. Citado 3 vezes nas páginas 17, 24 e 25.

LILLICRAP, T. P. et al. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. Citado 2 vezes nas páginas 32 e 34.

MEDINA, L. *Modelagem e Identificação dos Dispositivos da Malha de Nível de Uma Coluna de Destilação*. [S.l.]: CURSO SUPERIOR DE TECNOLOGIA EM AUTOMAÇÃO INDUSTRIAL, 2008. Citado 2 vezes nas páginas 21 e 23.

NAEGELE, E. F. *Proposta de Controle para uma Coluna de Destilação Didática*. [S.l.]: Dissertação de Mestrado apresentada ao Departamento de Automação, UFES, 2000. Citado na página 22.

NASCIMENTO, A. J. V. *Análise de Modelos Reduzidos de Colunas de Destilação para Aplicações em Tempo Real*. [S.l.]: PÓS-GRADUAÇÃO DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, 2013. Citado 2 vezes nas páginas 20 e 22.

NEWBORN, M. Deep blue's contribution to ai. *Annals of Mathematics and Artificial Intelligence*, Springer, v. 28, n. 1-4, p. 27–30, 2000. Citado na página 25.

NEWELL, A. Physical symbol systems. *Cognitive science*, Elsevier, v. 4, n. 2, p. 135–183, 1980. Citado na página 24.

NISE, N. S.; SILVA, F. R. da. *Engenharia de sistemas de controle*. [S.l.]: LTC, 2011. v. 6. Citado 2 vezes nas páginas 14 e 17.

NOGY, R. *Projeto Parte 5: Sistema de Controle*. 2015. Disponível em: <http://blogareadeteste.blogspot.com.br/2015/07/projeto-5-sistema-de-controle.html>. Citado na página 18.

OGATA, K.; YANG, Y. *Modern control engineering*. [S.l.]: London, 2002. v. 4. Citado na página 17.

PULIDO, J. L. *Estudo de um Novo Conceito de Coluna de Destilação: Coluna de Destilação com Integração Interna de Calor*. [S.l.]: Dissertação de Mestrado de Engenharia Química da Universidade Estadual de Campinas, 2011. Citado na página 14.

RASOVSKY, E. M. Alcool; destilarías. Instituto do Acúcar e do Alcool. Río. BR, 1973. Citado na página 21.

RUSSEL, S.; NORVIG, P. *Inteligência Artificial. 2ª. Edição*. 2004. Citado 2 vezes nas páginas 25 e 28.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, IBM, v. 3, n. 3, p. 210–229, 1959. Citado na página 27.

SEBORG, D. E. et al. *Process dynamics and control*. [S.l.]: John Wiley & Sons, 2010. Citado na página 19.

SILVA, H.; OTAL, L. *Um Estudo Comparativo entre RNA's e Modelos Auto-Regressivos para Identificação de uma Coluna de Destilação*. [S.l.]: Trabalho de Conclusão de Curso do IFF-Campos, 2010. Citado na página 22.

SILVER, D. et al. Deterministic policy gradient algorithms. In: . [S.l.: s.n.], 2014. Citado na página 34.

SPIELBERG, S.; GOPALUNI, B.; LOEWEN, P. Deep reinforcement learning approaches for process control. In: . [S.l.: s.n.], 2017. p. 201–206. Citado na página 35.

SPIELBERG, S.; GOPALUNI, B.; LOEWEN, P. *Process Control using Deep Reinforcement Learning*. 2018. Citado na página 35.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018. Citado 5 vezes nas páginas 15, 28, 29, 31 e 32.

TECNOBLOG. *Computador do Google venceu um campeão mundial neste jogo chinês*. 2017. Disponível em: <https://tecnoblog.net/192604/computador-google-vence-campeao-go/>. Citado na página 28.

TURING, A. M. Computing machinery and intelligence. In: *Parsing the Turing Test*. [S.l.]: Springer, 1950. p. 23–65. Citado na página 24.

WITTEN, I. H. An adaptive optimal controller for discrete-time markov environments. *Information and control*, Elsevier, v. 34, n. 4, p. 286–295, 1977. Citado na página 31.