UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
ENGENHARIA MECATRÔNICA

LUÍS EDUARDO FERNANDES COSTA LIMA

APRIMORAMENTO DO CONTROLADOR PID EM PROCESSOS ROLL-TO-ROLL UTILIZANDO APRENDIZADO POR REFORÇO

Joinville
2024

LUÍS EDUARDO FERNANDES COSTA LIMA

APRIMORAMENTO DO CONTROLADOR PID EM PROCESSOS ROLL-TO-ROLL UTILIZANDO APRENDIZADO POR REFORÇO

Trabalho apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia Mecatrônica, no Centro Tecnológico de Joinville, da Universidade Federal de Santa Catarina.

Orientador: Dr. Ricardo José Pfitscher

Joinville
2024

Dedico este trabalho de conclusão de curso aos meus pais, ao meu irmão e demais familiares, que me ensinaram valiosas lições sobre perseverança e me forneceram o suporte financeiro e emocional necessário para que eu pudesse realizar o sonho de me tornar engenheiro. Agradeço também aos meus amigos pelo constante apoio durante essa jornada.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me conceder a vida e por me permitir ultrapassar todos os obstáculos encontrados ao longo da realização deste trabalho.

Ao Prof. Dr. Ricardo José Pftischer, meu orientador, pela sua dedicação e amizade ao desempenhar essa função, proporcionando orientação essencial para a conclusão deste trabalho.

Agradeço ao Instituto de Produção de Tecnologia Fraunhofer, pelo fornecimento de dados e materiais que foram fundamentais para o desenvolvimento da pesquisa.

Aos meus pais e irmão, pelo incentivo nos momentos difíceis e pela compreensão durante minha ausência enquanto me dedicava à realização deste trabalho.

Aos amigos, pela amizade e pelo apoio demonstrado ao longo de todo o período em que me dediquei a este trabalho.

A todos que, de alguma forma, contribuíram para a realização deste trabalho.

“ *Se sua fraqueza o dominar, aqueça seu coração, cerre os dentes e siga em frente.* ”

Kyojuro Rengoku

RESUMO

Atualmente, cerca de 90% das malhas de controle dependem de controladores PID, tornando essa estratégia predominante em processos que requerem uma malha fechada. Em tais contextos, o controle preciso de variáveis específicas é essencial para assegurar a qualidade e eficiência dos produtos resultantes. Um exemplo desses processos é a manufatura *roll-to-roll* (R2R), uma prática que remonta à primeira revolução industrial e continua sendo essencial na produção de diversas superfícies funcionais nas áreas de saúde, energia, automobilística, entre outras. Além disso, na manufatura R2R é necessário utilizar controladores que cumpram requisitos específicos, uma vez que o controle da tensão do substrato tem impacto na qualidade das superfícies produzidas. Tradicionalmente, a definição de parâmetros PID é feita por meio de métodos analíticos clássicos os quais exigem um profundo conhecimento das plantas, que muitas vezes são complexas e apresentam não-linearidades frequentemente simplificadas, o que pode comprometer a eficácia do modelo. Em adição, esse processo de obtenção dos parâmetros pode consumir muito tempo na indústria. Com os avanços no poder computacional, no entanto, a obtenção de um controlador por meio de métodos de ajuste tem se destacado como uma abordagem vantajosa. Esses métodos são mais rápidos, automatizados e capazes de considerar as dinâmicas do sistema, sem a necessidade de simplificações. Considerando isso, este trabalho explora o uso de inteligência artificial, especificamente o aprendizado por reforço, como método de ajuste de controlador, visando aprimorar o controle de tensão durante um processo R2R. Como resultado, obteve-se um controlador com um valor de custo 65.1% menor em relação ao controlador definido empiricamente pelo operador. Os objetivos deste estudo incluem a coleta e o tratamento de dados, simulação do processo, desenvolvimento de um algoritmo de aprendizado por reforço para ajustar os parâmetros do controlador PID e avaliação do seu desempenho. A base teórica desta pesquisa abrange desde conceitos de engenharia de controle até aprendizado de máquina, destacando o papel desses elementos na criação de soluções viáveis para enfrentar desafios presentes em contextos industriais.

Palavra-chave: Inteligencia artificial; Manufatura R2R; Controle PID.

ABSTRACT

Currently, about 90% of control loops employ PID controllers, making this strategy predominant in processes that require a closed loop. In these contexts, precise control of specific variables is essential to ensure the quality and efficiency of the resulting products. An example of these processes is roll-to-roll (R2R) manufacturing, a practice that dates back to the first industrial revolution and remains crucial in the production of various functional surfaces in healthcare, energy, automotive, and other sectors. However, in R2R manufacturing, it is necessary to use controllers that meet specific requirements, as the control of substrate tension impacts the quality of the produced surfaces. Traditionally, the definition of PID parameters is done through analytical methods, which requires in-depth knowledge of plants, often complex and presenting nonlinearities that are frequently simplified, potentially compromising the model's effectiveness. Additionally, this parameter-obtaining process can be time-consuming in the industry. However, with advances in computational power, obtaining a controller through tuning methods has emerged as an advantageous approach. These methods are faster, automated, and capable of considering the system dynamics without the need for simplifications. Therefore, this paper explores the use of artificial intelligence, specifically reinforcement learning, as a method for controller tuning, aiming to improve tension control during an R2R process. As a result, a controller with a cost value 65.1% lower than the empirically defined controller by the operator was obtained. The objectives of this study include data collection and processing, process simulation, development of a reinforcement learning algorithm to tune the PID controller parameters, and evaluation of its performance. The theoretical basis of this research encompasses concepts from control engineering to machine learning, highlighting the role of these elements in creating viable solutions to address challenges present in industrial contexts.

Keywords: Artifical intelligence; Process simulation; PID control.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Durante toda a história da humanidade, houve tentativas de controlar o mundo em que vivemos para que o resultado seja o desejado pelo homem. Essas tentativas de controle, as quais pode-se denominar de sistemas de controle, têm como característica a habilidade fundamental de medir a saída do sistema - ou seja, de verificar o estado atual - e tomar uma ação para aproximar essa saída de um valor de interesse (Burns, 2001). O próprio ser humano pode ser visto como um sistema de controle sofisticado (Kuo, 1985). Por exemplo, ao dirigir um carro, que representa o sistema a ser controlado, a visão é utilizada como o principal meio de medir a saída e, então, com os braços e pernas, a pessoa toma uma ação a partir dessa saída.

Atualmente, os sistemas de controle são essenciais em diversas áreas da indústria, sendo empregados no controle de qualidade, na linha de montagem, nos sistemas de transporte, na eletrônica de potência, na tecnologia espacial e na robótica. Para atender aos objetivos específicos de cada área, surgiram novas tecnologias a partir da década de 1960, aproveitando o avanço nas capacidades computacionais dos computadores. Essas tecnologias incluem o controle preditivo, o controle adaptativo, o controle robusto, entre outros (Faccin, 2004).

Dentre esses controladores, o Proporcional Integral Derivativo (PID) é o método de controle mais utilizado na indústria, representando até 90% de todas malhas de controle, e a permanência desse controlador nos setores industriais, justifica-se pelo fato de esses controladores fornecerem soluções estáveis, robustas e simples para problemas de controle (Knospe, 2006). Na sua versão mais utilizada, controladores PID tem 3 parâmetros que os tornam simples e, portanto, os fazem uma escolha mais comum e apropriada para um grande número de problemas reais do que controladores mais complexos (Åström, 2002).

Um dos processos que utiliza desse sistema de controle é a produção de superfícies funcionais a partir da manufatura R2R que tornou-se um elemento importante na engenharia (Lee *et al.*, 2020). Um dos principais motivos dessa manufatura ser tão atrativa para diversas áreas da industria é que ela tem bom desempenho com processos de grande volume e que precisam de alto rendimento. Isso faz com que esse processo seja adequado a produção de superfícies funcionais e que represente uma solução rentável, já que os processos da manufatura são, em geral, realizadas por uma única maquina, o que diminui o custo com operadores (Greender *et al.*, 2018). Como é essencial para a manufatura de superfícies funcionais, aprimorar os processos R2R gera um impacto direto na melhoria dos outros processos, ampliando consideravelmente as possibilidades de fabricação (Lee *et al.*, 2020).

Exemplos de superfícies funcionais que utilizam o sistema R2R incluem o de-

senvolvimento de filme adesivo antimicrobiano contra germes hospitalares, essencial na manutenção da higiene em ambientes hospitalares (Kosel, 2018), e a fabricação de placas bipolares metálicas para a geração de energia com hidrogênio (Reimer *et al.*, 2021).

O sistema R2R é composto por um conjunto de enroladores e unidades de fixação, como mostra a Figura 1. No início do processo em cadeia, pelo menos um enrolador, chamado de desbobinador, marcado na Figura 1 com a letra D, contém a membrana que passa pela manufatura. O papel do desbobinador vai além do fornecimento de material; ele também é responsável por controlar a tensão na membrana que está antes da unidade de fixação, marcado na Figura 1 com a letra F(Lee *et al.*, 2020).

Figura 1 – Sistema R2R

Fonte: Adaptada de Fraunhofer IPT (2023).

No final do processo, há pelo menos um enrolador, chamado de rebobinador, marcado na Figura 1 com a letra R, que armazena a superfície funcional e é o responsável por controlar a tensão sobre a membrana após a unidade de fixação (Lee *et al.*, 2020). Em adição, como já foi citada, é necessária pelo menos uma unidade de fixação entre o desbobinador e o rebobinador, a qual determina a velocidade com que o material é alimentado para os demais módulos da cadeia (Lee *et al.*, 2020).

Além disso, o controle preciso da tensão na membrana durante a manufatura é crucial para garantir a eficiência e a qualidade das superfícies funcionais (Dehui *et al.*, 2014). Diversos fatores, como temperatura, elasticidade do material e processos

físicos e químicos envolvidos, têm impacto direto sobre a tensão na membrana. Assim, torna-se essencial a implementação de um sistema de controle capaz de acomodar todas as dinâmicas do sistema que afetam essa variável crítica (Lee *et al.*, 2020).

Uma solução comum para manter a tensão estável ao longo do processo é a implementação do controlador PID clássico (Lee *et al.*, 2020). Em circunstâncias ideais, a abordagem analítica de definição do controlador PID, quando utiliza parâmetros adequados para o sistema dinâmico, alcança resultados positivos e uma resposta próxima ao valor alvo (Bansal *et al.*, 2012). No entanto, a obtenção desses parâmetros ideais depende de um conhecimento matemático das dinâmicas do processo e também que esse sistema não seja influenciado por grandes ruídos externos (Lee; Jang, 2021).

Ademais, ao lidar com variações no sistema e particularidades de cada processo - como no caso dos sistemas R2R a posição dos eixos, as propriedades do material e as flutuações de temperatura - a obtenção desses parâmetros pode tornar-se uma tarefa difícil e, em alguns casos, obtendo controladores insatisfatórios ou até instável (Lee; Jang, 2021). Isso pode ser atribuído ao fato que, na construção do controlador PID, o sistema dinâmico é linearizado, então essas linearizações e ruídos do sistema fazem com que na prática o sistema se afaste do ideal (Gouda *et al.*, 2000). Então, uma alternativa ao método analítico clássico de se obter os parâmetros PID é o uso de algoritmos de otimização para ajuste de controladores, que permite obter parâmetros que tem resposta tão rápidas quanto possível com menor sobressinal e erro em estado estacionário próximo do zero (Ribeiro *et al.*, 2017).

A Indústria 4.0 introduziu a *Internet of Things* (IoT), uma rede de dispositivos conectados que possibilitam a comunicação de dados em tempo real, criando ambientes de fabricação dinâmicos, conectados e complexos (Sacomano *et al.*, 2018). Isso implica o constante envio e armazenamento de grandes volumes de dados, na ordem de *terabytes* (Marjani *et al.*, 2017). Essa extensa quantidade de dados armazenados viabiliza o uso da Inteligência Artificial (IA) industrial, já que sua eficácia depende de um volume substancial de dados para prover análises preditivas e facilitar a implementação de outras tecnologias nesse contexto complexo (Peres *et al.*, 2020a). Contudo, apesar dessas condições favoráveis, a implementação generalizada de IA na indústria ainda tem desafios para sua implementação em ambientes reais, como garantir que dados e modelos para IA industrial sejam facilmente localizáveis, acessíveis e reutilizáveis, além disso, é preciso criar as infraestruturas adequadas para garantir o nível de qualidade, segurança e confiabilidade (Peres *et al.*, 2020a).

Tendo em vista o potencial da inteligência artificial industrial, este trabalho explora uma alternativa baseada em aprendizado de máquina (ML) para ajustar os parâmetros do controlador PID responsável pelo controle da tensão sobre a membrana em um processo R2R. Em contraste com a abordagem analítica, que se baseia

na linearização do sistema, este trabalho apresenta a construção de um controlador baseado em aprendizado por reforço (RL), e explora contornar a dificuldade de usar um equipamento físico para treinamento, dividindo o treinamento em *offline*, que é feito com base em uma simulação da máquina obtida por métodos de regressão, e a validação online, no momento em que o treinamento está mais próximo de convergir, são selecionados os melhores controladores segundo o agente e são testados no sistema dinâmico real. Dado que algoritmos inteligentes têm a capacidade de convergir e acomodar uma ampla gama de dinâmicas do sistema (Peres *et al.*, 2020a), a abordagem proposta pode vir a ser uma solução para superar os desafios apresentados pelas variações complexas no processo de criação de superfícies funcionais com R2R e pela alta complexidade matemática desses sistema, que, como dito, faz com que o processo de implementação de um controlador para um sistema R2R seja um trabalho difícil e, consequentemente, tome muito tempo no contexto industrial.

1.1 OBJETIVOS

1.1.1 Objetivo geral

Desenvolver um agente de RL o qual ajusta um controlador, com objetivo de melhorar o controle atual da tensão no processo de manufatura R2R.

1.1.2 Objetivos Específicos

- Analisar dados da tensão durante a manufatura;
- Simular a tensão sobre a membrana em um processo R2R;
- Desenvolver um agente de RL para gerar os parâmetros de um controlador PID;
- Avaliar o desempenho do controlador durante o processo de manufatura;

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentadas as bases fundamentais necessárias para a compreensão do tema abordado neste trabalho. Esses fundamentos teóricos possibilitarão entender a modelagem e o funcionamento de uma máquina R2R, bem como a estrutura e operação de um controlador PID, envolvendo, assim, todo o processo de manufatura de superfícies funcionais. Na sequência, serão discutidas a origem e a arquitetura das Redes Neurais Artificiais (RNA), além dos modelos matemáticos utilizados para o treinamento e obtenção de resultados por meio dessas RNA, bem como modelos de predição de valores que utilizam essas RNAs ou outros modelos matemáticos, então, por fim, será discutido diferentes algoritmos de RL e suas diferentes características. Isso proporcionará uma compreensão mais ampla sobre a aplicação e a expansão da Inteligência Artificial na indústria.

2.1 MANUFATURA ROLL-TO-ROLL

As origens da tecnologia de produção R2R se estendem até o final da Revolução Industrial, na segunda metade do século XIX. Este período foi marcado pelo surgimento das indústrias de impressão e fotografia, que impulsionaram inovações e avanços na tecnologia de produção. Um marco importante na evolução da impressão em larga escala foi a criação da impressora rotativa por Richard Hoe na década de 1840. Essa inovação, junto com os desenvolvimentos anteriores em papel enrolado e máquinas a vapor, permitiu a impressão contínua em grandes volumes de papel de forma econômica, estabelecendo a base para a adoção dos processos R2R (Greender *et al.*, 2018).

Na indústria contemporânea, o processo de manufatura R2R é extensivamente aplicado em uma variedade de produtos e aplicações, cobrindo vários segmentos industriais. Métodos de impressão como *inkjet*, flexografia e serigrafia são frequentemente utilizados na produção de superfícies funcionais em operações R2R (Greender *et al.*, 2018).

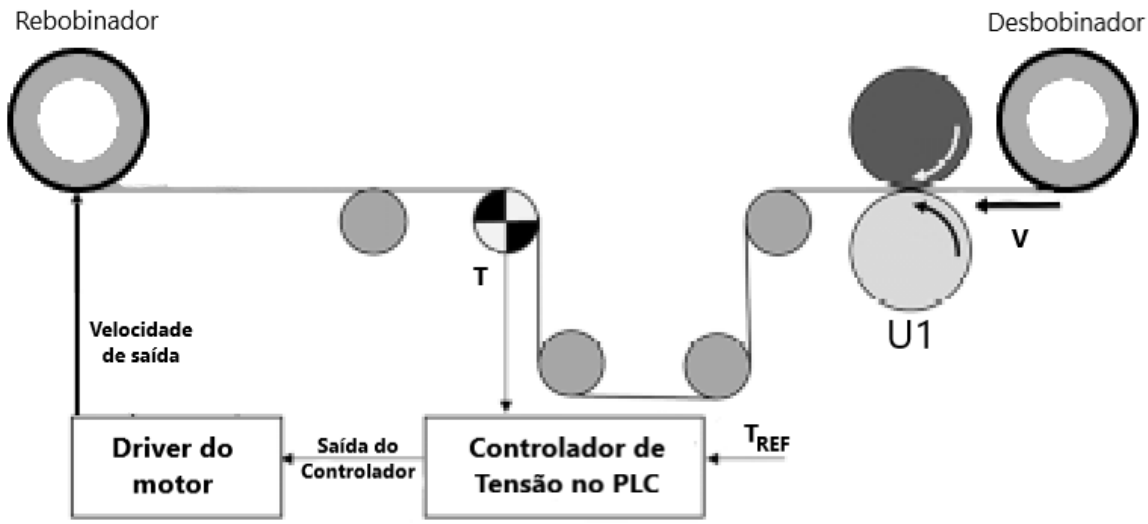
Uma característica comum em todas essas diversas operações de manufatura é o processamento contínuo de materiais bidimensionais (2D), como filmes finos e planos, que são movimentados em uma velocidade constante entre dois ou mais rolos rotativos. Estas superfícies apresentam propriedades físicas ou químicas específicas, que lhes conferem usos especiais, justificando sua produção através deste método. A variedade de usos é extensa, refletindo a ampla gama de aplicações possíveis com a manufatura R2R (Greender *et al.*, 2018).

2.1.1 Operação de uma máquina roll-to-roll

As operações R2R abrangem uma ampla gama de mercados e produtos, mas compartilham princípios operacionais comuns. A natureza modular dessas operações permite a adição ou remoção de etapas de conversão (módulos) conforme os requisitos do produto filme, tornando esse processo versátil (Greender *et al.*, 2018).

Na Figura 2, uma superfície é conduzida entre dois rolos, o desbobinador e o rebobinador, passando pela unidade de compressão U1 que é responsável por definir a velocidade da superfície no sistema. Nessa esquemática, o substrato bruto ou parcialmente processado é desbobinado de um rolo de fornecimento e introduzido na máquina R2R. Em seguida, o substrato bruto percorre uma série de etapas consecutivas de processo, ocultadas nessa figura, enquanto é transportado a uma velocidade e tensão controlada pela máquina R2R (Lee *et al.*, 2020).

Figura 2 – Esquemática de um sistema R2R



Fonte: Adaptado de Lee *et al.* (2020).

Essa tensão no substrato é controlada pelo bobinador para garantir a planaridade e o processamento do substrato livre de defeitos durante toda a operação (Greender *et al.*, 2018). É importante observar que uma única linha R2R pode ter mais de um desbobinador se a operação incluir etapas de laminação ou intercalação. No entanto, geralmente a linha é finalizada por um único rebobinador, que armazena a superfície final daquela processo R2R (Greender *et al.*, 2018).

O processamento constante do substrato através dos módulos de conversão requer a seleção cuidadosa de uma velocidade que atenda a todos os elementos do processo, garantindo assim uma operação robusta em toda a linha, portanto, esse acoplamento entre as etapas do processo significa que a velocidade final da linha é

limitada pela etapa mais lenta (Greender *et al.*, 2018). No entanto, ao contrário da velocidade, as tensões podem variar em diferentes pontos devido ao uso de unidades de fixação que podem ter tensões distintas na entrada e na saída, então cada módulo de conversão pode trabalhar com tensões otimizadas para seu processo (Lee *et al.*, 2020).

As velocidades de linha nas operações R2R variam consideravelmente dependendo do tipo de filme, dos requisitos de qualidade e do número de etapas de conversão ao longo da linha. Em aplicações mais especializadas e com tecnologias avançadas que envolvem produtos de filme com estruturas de camadas complexas e requisitos rígidos de registro, são comumente utilizadas velocidades inferiores a 10m/-min (Greender *et al.*, 2018).

Como dito anteriormente, um fator importante para a operação R2R é a tensão na superfície, que é controlada ao longo da linha pelos bobinadores. Uma tensão inadequada ou não uniforme pode resultar em vários defeitos, como enrugamento, linhas de revestimento, não uniformidade na espessura da camada revestida e deformação no substrato e nas camadas depositadas, tornando fundamental o monitoramento e controle em linha dessa variável para garantir a produção de filmes livres de defeitos e com desempenho ideal (Greender *et al.*, 2018). Isso não apenas reduz o desperdício, mas também melhora os rendimentos e reduz os custos de fabricação. Além disso, é importante controlar as pressões e tensões de enrolamento dentro do rebobinador, pois níveis excessivamente altos podem danificar as camadas funcionais, resultando em produtos defeituosos (Greender *et al.*, 2018).

Uma solução viável é minimizar a tensão de enrolamento e reduzir o tempo de armazenamento dos rolos. No entanto, uma tensão excessivamente baixa pode causar instabilidade no rolo, resultando em problemas como o telescópio, onde as camadas se deslizam para fora perto do centro. Portanto, a escolha da tensão de enrolamento deve ser ponderada para garantir a qualidade do produto final. Assim, identificar os intervalos que atingem os requisitos é importante para evitar tais problemas (Lee *et al.*, 2020).

2.2 SISTEMA DE CONTROLE

Na engenharia e na ciência, diversos processos demandam um controle preciso de variáveis específicas para atingir metas bem definidas. Nesse contexto, a engenharia de controle desempenha um papel fundamental em áreas como o controle de temperatura, posição, nível, velocidade e, no caso deste trabalho, a tensão sobre uma superfície funcional na manufatura R2R (Garcia, 2021).

Esses processos podem ser descritos por sistemas complexos, com diversas entradas e saídas que interagem de forma não linear. Para modelar as características

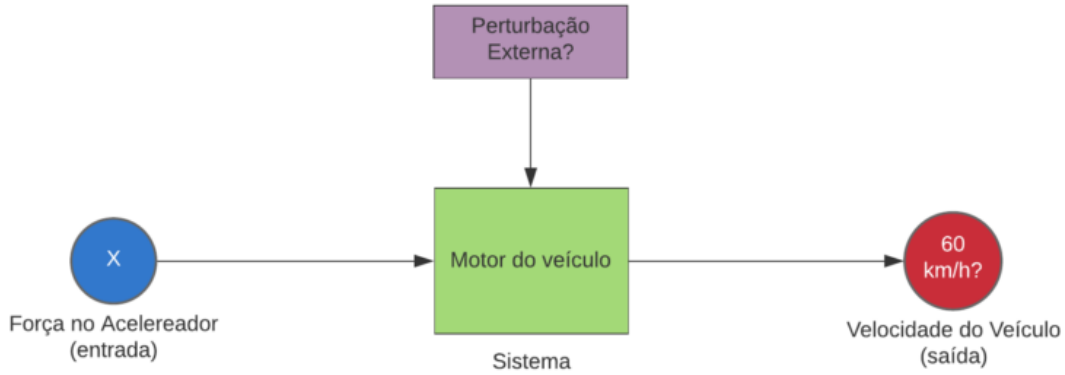
dinâmicas desses sistemas, é essencial simplificar sua complexidade matemática para projetar um sistema de controle baseado na teoria analítica clássica (Ogata, 1999).

Desde a década de 1920, com o surgimento de novas tecnologias que impulsionaram o uso de máquinas na indústria, a produção teve um rápido aumento, gerando a necessidade de métodos de controle automático (Bennett, 1993). A partir dos anos 1970, com a introdução de conceitos de realimentação, como margem de estabilidade e sensibilidade, a teoria de projeto de sistemas de controle teve um grande impacto. A arquitetura de malha fechada possibilitou melhorias significativas no controle automático em comparação com os sistemas de malha aberta (Borjas; Demetrio, 2009), portanto, para compreender os métodos de controle é necessário ter conhecimento das diferentes arquiteturas de controle.

2.2.1 Malha aberta e malha fechada

Os sistemas em malha aberta, consistem em um sinal de entrada enviada ao sistema dinâmico, que gera uma saída. Na Figura 3 é mostrado um exemplo de malha aberta do controle de velocidade de um veículo, onde a entrada do sistema é a força no acelerador e a saída é a velocidade do veículo. Uma desvantagem evidente desse sistema é a ausência de sensoriamento da saída, tornando-o vulnerável a perturbações externas na entrada e a variações no sistema dinâmico não previstas.

Figura 3 – Representação de um sistema em malha aberta



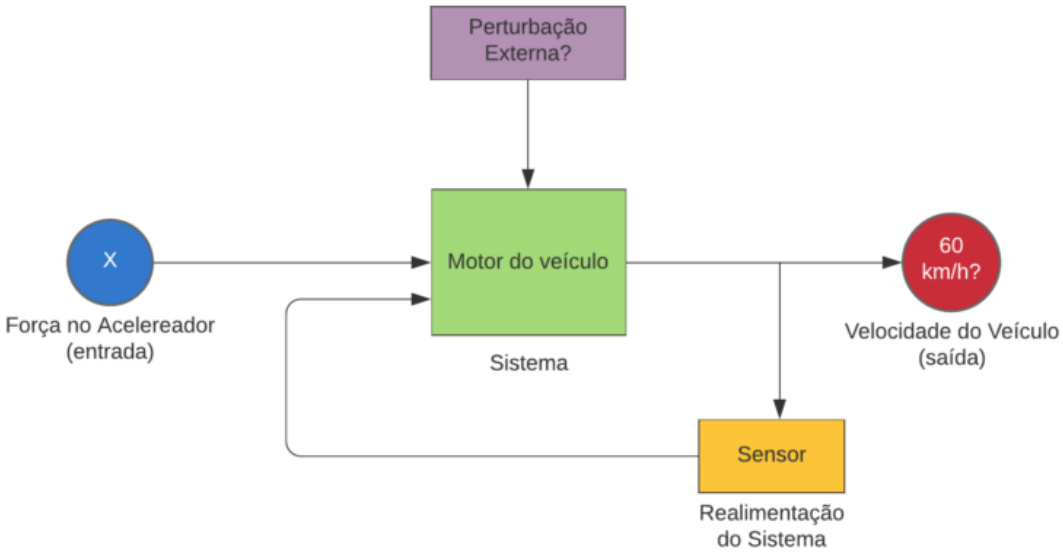
Fonte: IEEE (2020)

A Figura 3 mostra que mesmo fornecendo a força no acelerador o sistema de controle não tem informação sobre a saída do sistema e, então, não pode ajustar a sua entrada de acordo com esse valor. Portanto, essa falta de realimentação impede que o sistema de controle seja ajustado às mudanças no ambiente, comprometendo seu desempenho, capacidade de correção de erros e tornando-se incapaz de se adaptar a diferentes cenários do sistema (Borjas; Demetrio, 2009).

Nos sistemas em malha fechada, a saída é continuamente monitorada por sensores e, em seguida, realimentada negativamente para a entrada do controlador.

A Figura 4 mostra o mesmo exemplo da malha aberta, porém agora com realimentação, o que permite que o sistema de controle conheça o estado atual que depende da entrada fornecida, do sistema e das pertubações externas. Portanto, essa realimentação possibilita o ajuste contínuo do sistema de controle com base nas saídas do sistema dinâmico, calculando a saída do controlador em relação ao valor desejado. Isso resulta em um controle mais preciso e adaptativo (Borjas; Demetrio, 2009).

Figura 4 – Representação de um sistema em malha fechada



Fonte: IEEE (2020)

2.2.2 Controle PID

O controlador PID é uma combinação linear das ações proporcional (P), integral (I) e diferencial (D) sobre os erros em um sistema de malha fechada (Tao *et al.*, 1998), esse erro é calculado pela Equação 1

$$e(t) = d(t) - y(t) \tag{1}$$

na qual d é o valor desejado e y é o valor real do sistema obtido pela realimentação negativa do sistema em malhada fechada.

O controle proporcional altera a amplitude do sinal de entrada sem impactar na fase do mesmo. Essa ação é a mais significativa dentro do controlador PID e varia de forma proporcional ao erro atual. Já o controle integral registra o histórico de estados do sistema e, então, toma ação de acordo com esse histórico acumulado do erro do sistema, ou seja, atua sobre a integral do erro no tempo. Por outro lado, o controle diferencial determina a derivada do sinal do erro, refletindo a taxa de variação em um sistema. Esse controle é um modo de regulação preditiva que prevê variações do

sistema, aumenta o amortecimento e melhora a margem de fase, consequentemente aprimorando o desempenho do sistema. No entanto, essa ação é muito sensível a ruídos, pois esses podem alterar a taxa de variação do sinal e impactar na predição do sistema. Por essa razão, seu ganho é, em geral, menor do que o ganho proporcional (Ogata, 1999).

2.2.2.1 Leis de controle

A equação de um controlador PID no domínio do tempo pode ser expressa pela Equação 2.

$$s(t) = K_p * (e(t) + \frac{1}{T_i} * \int_0^t e(t) * \, dx + T_d * \frac{de(t)}{dt}) \tag{2}$$

onde $e(t)$ é a entrada do controlador, ou seja, o erro; K_p é o ganho proporcional; T_i é a constante de tempo integral e T_d é a constante de tempo diferencial. Nessa equação, nota-se que o único valor que atua no erro instantâneo é o valor de K_p. Já as outra constantes atuam em diferentes fases do erro, o ganho integral é dado por $K_i = K_p/T_i$ e o ganho diferencial é dado por $K_d = K_p * T_d$.

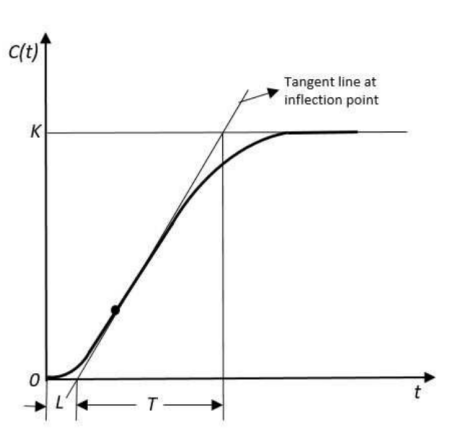
2.2.2.2 Sintonia de controladores PID

A sintonia de controladores PID é um processo fundamental na área de sistemas de controle, pois visa ajustar os parâmetros do controlador PID para garantir um bom desempenho do sistema controlado, alcançando uma resposta rápida e estável do sistema e garantindo que ele responda de forma eficaz às mudanças nas condições de operação e às perturbações externas (Euzébio *et al.*, 2015). Ao ajustar os parâmetros do controlador PID, busca-se alcançar um equilíbrio entre a resposta rápida do sistema (tempo de resposta) e a estabilidade (baixa oscilação ou sobressinal) em torno do ponto de operação desejado (Euzébio *et al.*, 2015). Para essa tarefa, a literatura já tem alguns métodos explorados, e alguns desses métodos são descritos nos próximos parágrafos.

Um método amplamente utilizado na sintonia de controladores PID é o método de Ziegler-Nichols. Este método é notável por sua independência em relação ao conhecimento detalhado da planta do sistema, utilizando dados experimentais para obter os parâmetros do controlador (Júnior, 2006). O procedimento padrão para ajustar os parâmetros envolve abrir a malha de controle para eliminar a realimentação e, em seguida, observar a resposta do sistema a uma entrada degrau de amplitude M. É essencial que a resposta seja estável e apresente a característica de forma em "S", como mostra a Figura 5, caso contrário, o método não é aplicável (Patel, 2020).

Onde o eixo x representa o tempo e o eixo y o valor da saída do sistema.

Figura 5 – Resposta esperada à um degrau para aplicação do método



Fonte: Patel (2020)

Uma vez obtida a resposta, é possível extrair duas constantes importantes: o atraso L e a constante de tempo T. Essas constantes são determinadas ao traçar uma tangente pelo ponto de inflexão da curva. Nos pontos em que a tangente intercepta o eixo das abscissas e a linha horizontal com ordenada K, obtemos L e T, respectivamente. O valor K representa o ganho do sistema. Com esses valores determinados, pode-se recorrer ao Quadro 1 para obter os parâmetros do controlador PID necessários para a sintonia adequada do sistema.

Quadro 1 – Quadro do método da sensibilidade limite

Controlador	Ganho do Controlador	Tempo Integral	Tempo Derivativo
P	$K_p = MT/(KL)$	—	—
PI	$K_p = 0.9MT/(KL)$	$T_i = 3.33L$	—
PID	$K_p = 1.2MT/(KL)$	$T_i = 2L$	$T_d = L/2$

Fonte: Adaptado de Berto *et al.* (2004).

Outro método sintonia é o lugar geométrico das raízes, o qual é uma técnica analítica empregada para determinar todas as raízes da equação diferencial que governa o sistema dinâmico e definem o seu estado. Essa abordagem permite uma síntese conveniente das respostas transitórias ou de frequência desejadas (Gomes, 2009). Um gráfico, traçado no plano complexo, é construído com base nos zeros, raízes do numerador, e polos, raízes do denominador, da função de transferência no plano S da malha aberta, os quais estão prontamente disponíveis. Conforme o ganho da função varia de zero a infinito, o lugar das raízes assume uma função de transferência igual a -1. Embora o gráfico possa ser estabelecido aproximadamente por meio de inspeção visual, os segmentos significativos do lugar podem ser calculados com precisão e rapidez utilizando uma ferramenta simples.

2.3 INTELIGENCIA ARTIFICIAL E INDÚSTRIA 4.0

A Inteligência Artificial (IA) é um campo que busca entender e replicar as capacidades cognitivas humanas em computadores. Pode-se defini-lo como o estudo de métodos para que computadores realizem tarefas nas quais os humanos se destacam no momento (Rich, 1985), ou, defini-lo como uma subárea fundamental da ciência da computação, que lida com o desenvolvimento de sistemas capazes de processar dados e executar funções associadas à inteligência humana (Pizard; Vallespir, 2017). Nos últimos 50 anos, os pesquisadores de IA têm explorado uma ampla gama de técnicas e algoritmos para capacitar os computadores a realizar uma variedade de tarefas anteriormente possíveis apenas para humanos (Ertel, 2018).

Embora os computadores sejam excelentes em realizar cálculos em um curto período de tempo, os humanos possuem habilidades superiores em outras áreas, como reconhecer objetos ou faces em um instante. Uma das características mais impressionantes da inteligência humana é a adaptabilidade, nossa capacidade de nos ajustar a diferentes ambientes e modificar nosso comportamento com base na experiência e no aprendizado (Ertel, 2018). É precisamente nesse aspecto que se destaca a diferença entre o aprendizado humano e a capacidade computacional. O ML, uma subárea central da IA, surge com o objetivo de desenvolver algoritmos que permitam aos computadores aprender com dados e experiências, aproximando-se assim da capacidade de adaptação humana (Peres *et al.*, 2020b).

Além disso, a Indústria 4.0 está gerando uma necessidade cada vez maior de trazer agilidade, produtividade e sustentabilidade à manufatura. A IA tem sido considerada uma das tecnologias mais importantes para atingir essas características e alterar a maneira como os processos de manufatura são conduzidos atualmente (Huang *et al.*, 2021). Então, no âmbito industrial, o ML tem se destacado devido à sua capacidade de perceber o ambiente, processar os dados recebidos e resolver problemas complexos enquanto aprendem com suas experiências (Peres *et al.*, 2020b). Então as próximas sessões exploram conceitos fundamentais para o entendimento do uso da IA nesse trabalho.

2.3.1 Redes Neurais Artificiais

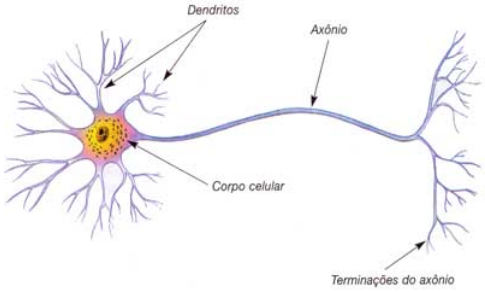
A forma como os seres humanos aprendem e se adaptam a diferentes ambientes possibilitou a expansão da espécie por todos os continentes do planeta. Essa habilidade é objeto de estudo desde os primórdios da medicina. Na década de 1940, com o surgimento dos primeiros computadores eletrônicos, iniciou-se a busca por métodos que replicassem essa capacidade de maneira artificial (Carvalho *et al.*, 2011). Foi nesse contexto que surgiram, por meio de algoritmos matemáticos e computacionais, os primeiros modelos de aprendizado, como as Redes Neurais Artificiais (Carvalho *et*

al., 2011). Essas redes são baseadas no funcionamento do cérebro humano e em sua capacidade de aprendizado e adaptação a novos ambientes, portanto, compreender esse algoritmo requer uma compreensão prévia do funcionamento do sistema nervoso humano.

2.3.1.1 Sistema nervoso humano

O sistema nervoso é composto por células altamente especializadas que se formam principalmente antes do nascimento (Goodman *et al.*, 1981). Os principais componentes desse sistema são os neurônios, representados na Figura 6, os quais são constituídos por três partes fundamentais.

Figura 6 – Representação de um neurônio



Fonte: SUPERA (2023)

Os dendritos são responsáveis por receber estímulos nervosos de outros neurônios e encaminhá-los para o corpo celular. Esse, por sua vez, processa esses diversos estímulos e gera um impulso nervoso que é enviado para o axônio. O axônio é especializado em conduzir o impulso nervoso gerado pelo corpo celular para os dendritos de outros neurônios, fechando assim o ciclo de comunicação dentro de um neurônio. No processo de aprendizagem humana, novas conexões, conhecidas como sinapses, são formadas entre diferentes neurônios, podendo ser excitatórias ou inibitórias. Essas novas sinapses alteram a maneira como a informação é processada (Xu *et al.*, 2009).

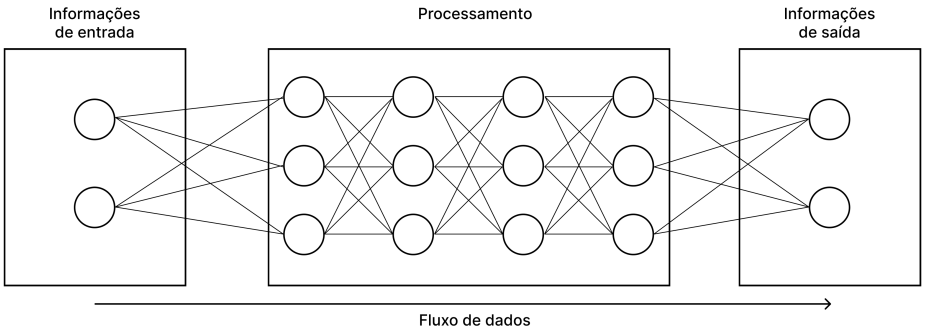
2.3.1.2 Arquitetura das Redes Neurais Artificiais

Assim como no sistema nervoso, as RNA são compostas por pequenas unidades de processamento que recebem, processam e encaminham informações, sendo semelhantes aos neurônios. Por isso, essas unidades são chamadas de neurônios artificiais. Essas estruturas se agrupam em camadas, onde os neurônios de uma camada se conectam e transmitem dados aos neurônios da camada seguinte (Carvalho *et al.*, 2011), conforme ilustrado na Figura 7.

A sinapse entre os neurônios artificiais depende dos pesos associados a cada conexão, conferindo maior ou menor impacto a determinado neurônio. Assim como no

processo de aprendizagem do sistema nervoso humano, esses pesos representam a formação e a modificação das conexões entre os neurônios, sendo responsáveis por armazenar o conhecimento e a experiência adquiridos pela RNA. Ajustar esses pesos é um passo importante para que o modelo alcance resultados satisfatórios (Rauber, 2005).

Figura 7 – Rede neural artificial



Fonte: Autoria própria (2024).

O processamento dessas informações ocorre pela ponderação de cada entrada do neurônio com o peso associado. A entrada E do neurônio é calculada como a soma desses valores, conforme expresso na Equação 3.

$$E = \sum_{i=1}^{n} x_i * p_i \tag{3}$$

Onde x_i é o valor recebido pela sinapse i, p_i é o peso dessa conexão e n é o número de conexões desse neurônio.

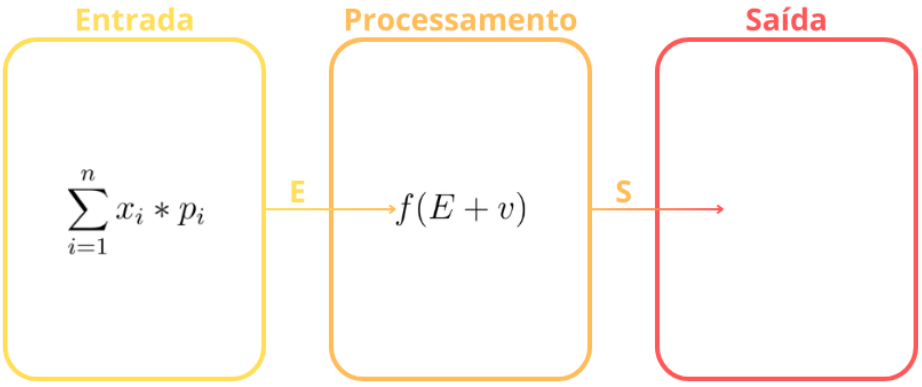
Em seguida, a entrada E é passada para a próxima estrutura, análoga ao corpo celular, na qual um valor v, denominado viés, é somado à entrada. Então, esse novo valor é encaminhado para uma função matemática, denominada função de ativação, que realizará o processamento dessa informação e encaminhará a saída S, definida pela Equação 4.

$$S = f(E + v) \tag{4}$$

Onde f é a função de ativação e v é o valor do viés.

A função do viés é facilitar ou dificultar a ativação desse neurônio. Valores com módulo muito grande e sinal negativo farão com que o neurônio só seja ativado para valores de E muito grandes. Já um viés com valores grandes e sinal positivo fará com que esse neurônio possa ser ativado mesmo para valores de E muito negativos. A Figura 8 representa um neurônio artificial dividido em entrada (análoga ao dendrito), processamento (análoga ao corpo celular) e saída (análoga ao axônio).

Figura 8 – Neurônio Artificial



Fonte: Autoria própria (2024).

2.3.1.3 Processo de aprendizagem e Backpropagation

O processo de aprendizagem em RNA é o passo fundamental para o desenvolvimento e adaptação desses sistemas, assim como ocorre no sistema nervoso humano. Esse processo é caracterizado pela capacidade da RNA de ajustar seus parâmetros internos, especialmente os pesos das conexões entre neurônios, com base nas entradas fornecidas e nas saídas desejadas. O algoritmo *Backpropagation* é comumente utilizado no processo de aprendizagem de RNAs, ele é responsável por propagar o erro, diferença entre saída da RNA e a saída desejada, de volta para as camadas internas da RNA, ajustando os pesos, mostrados na Equação 3. Esse processo de retroalimentação permite que a RNA corrija suas representações internas para o valor de sua saída se aproximar dos valores desejados (Wythoff, 1993).

2.3.1.4 Função de Custo

A avaliação do desempenho da RNA durante o treinamento é realizada por meio de uma função de custo, que quantifica a diferença entre as saídas previstas e as saídas desejadas. O objetivo é minimizar essa função de custo, indicando que a RNA está se aproximando da correta representação dos padrões de entrada e saída (Ngatched; Woungang, 2022).

2.3.1.5 Épocas e *Batch*

O treinamento da RNA é realizado por meio de épocas, onde cada época consiste em uma passagem completa pelo conjunto de treinamento. Além disso, o conceito de *batch* refere-se à quantidade de dados de treinamento usada em uma única atualização dos pesos da RNA. O ajuste de épocas e tamanho de *batch* é crucial para evitar *overfitting*, que é o aprendizado em excesso ao conjunto de treinamento, o que faz a rede neural aprender bem esse conjunto, porém, não conseguir generalizar

esse aprendizado para outros dados, tendo, por consequência, um desempenho pior para os dados de testes (Brownlee, 2018).

2.3.2 Predição de valores

Uma tecnologia que tem se tornado mais comum com avanços da computação são os algoritmos de previsão de valores com precisão, esses algoritmos desempenham um papel fundamental em várias áreas, como medicina (Rath *et al.*, 2020) e meteorologia (Sreehari; Srivastava, 2018). Os algoritmos de predição têm o propósito de analisar dados históricos e identificar padrões, permitindo fazer previsões sobre eventos futuros ou valores desconhecidos (Osborne, 2019). Ao aplicar métodos estatísticos e técnicas de aprendizado de máquina, os algoritmos de predição têm o potencial de fornecer informações valiosos e melhorar a tomada de decisões em uma ampla gama de setores (Bakar; Tahir, 2009). Alguns algoritmos comuns na previsão de valores são apresentados nas seções a seguir.

2.3.2.1 Baseada em Redes Neurais Artificiais

As informações presentes nesta seção são baseadas em (SILVA *et al.*,), para simplificar a leitura essa citação será ocultada dos próximos parágrafos.

No âmbito da previsão de valores, o aprendizado de máquina supervisionado para regressão se destaca por meio de algoritmos baseados em redes neurais artificiais. Esses algoritmos trabalham ao treinar um modelo com um conjunto de dados contendo entradas e suas respectivas saídas correspondentes. Durante o treinamento, o modelo ajusta seus parâmetros de acordo com os exemplos fornecidos, visando minimizar uma função de custo que quantifica a discrepância entre as previsões do modelo e os valores reais. Este processo de ajuste é frequentemente realizado utilizando o método de otimização gradiente descendente, onde os pesos da rede são atualizados iterativamente para reduzir o erro de previsão.

Uma vez treinado, o modelo de rede neural está apto a fazer previsões para novas entradas. Com base nos padrões aprendidos durante o treinamento, a rede neural pode estimar valores para novas situações não vistas anteriormente. Esse poder de generalização é especialmente útil em cenários onde precisamos prever valores futuros com base em dados históricos, como previsão de séries temporais, previsão de preços de ativos financeiros e previsão de demanda de produtos.

As arquiteturas de redes neurais para regressão, como redes neurais *feed-forward*, redes neurais convolucionais ou redes neurais recorrentes, são amplamente aplicados em diversas áreas devido à sua capacidade de capturar padrões complexos nos dados e fornecer previsões precisas e flexíveis. Eles desempenham um papel crucial na análise preditiva, impulsionando a tomada de decisões informadas em uma variedade de contextos industriais e empresariais.

2.3.2.2 Modelo de regressão linear múltipla

Regressão linear múltipla é uma técnica para modelar a relação linear entre 2 ou mais variáveis (Bakar; Tahir, 2009). O modelo de regressão linear geral, com termos de erro normal, em termos das variáveis X é mostrado na Equação 5.

$$Y_i = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \theta_{n-1} X_{n-1} \tag{5}$$

Onde Y_i é o valor predito, $\theta_0, \theta_1, ..., \theta_{n-1}$ são os parâmetros do modelo linear e $X_0, X_1, ..., X_{n-1}$ são as variáveis de entrada do sistema.

Para encontrar os parâmetros que trazem melhores resultados para os dados do sistema é utilizado em geral o critério dos mínimos quadrado (Weisberg, 2005). Esse critério toma a soma dos quadrados das diferenças entre os valores reais e os valores previstos pela linha de regressão e é descrito pela Equação 6.

$$J = \sum_{i=1}^{n} (y_i - h_i)^2 \tag{6}$$

Onde n é o número de observações, y_i é os i-ésimo valor observado e h_i é o i-ésimo valor previsto.

O objetivo é encontrar os parâmetros da Equação 5 que minimizam essa soma de quadrados. O ajuste desses parâmetros é, em geral, feito utilizando método de otimização, como o gradiente descendente (Weisberg, 2005).

A ideia principal do gradiente descendente é ajustar iterativamente os parâmetros de um modelo para minimizar uma função de custo. O processo começa com uma estimativa inicial dos parâmetros e, em seguida, ajusta esses parâmetros em direção ao mínimo da função de custo (Weisberg, 2005). Esse ajuste é feito utilizando a Equação 7.

$$\theta_i = \theta_i - \alpha \nabla J \tag{7}$$

Onde θ_i são os parâmetros da Equação 5, α é conhecido como taxa de aprendizado, que varia entre 0 e 1 e ∇J é a derivada da Equação 6.

Este método calcula os coeficientes que minimizam a soma dos quadrados dos resíduos, levando à linha de regressão que melhor se ajusta aos dados (Weisberg, 2005).

2.3.3 Algoritmos RL

O RL é uma subcategoria do ML que capacita os computadores a aprenderem interagindo com o ambiente. Em suma, o RL é formada por dois componentes: um ambiente e um agente. O agente interage com o ambiente executando ações específicas e recebendo retornos do ambiente. Esse retorno é utilizado para calcular a

"recompensa"do agente. O agente aprende ao melhorar seu desempenho buscando obter mais recompensas positivas do ambiente. Esse processo de aprendizado forma um ciclo de recompensa entre o ambiente e o agente, orientando o aprimoramento do agente por meio de algoritmos de RL(Ding *et al.*, 2020).

2.3.3.1 *Deep* RL

A Deep Reinforcement Learning (DRL) combina as vantagens das Redes Neurais Profundas (RNP), que são RNAs com múltiplas camadas ocultas, e do RL para criar sistemas de inteligência artificial. A principal razão para incorporar o aprendizado profundo no RL é aproveitar a escalabilidade das RNP em espaços de alta dimensão (Ding *et al.*, 2020). Nesse algoritmo temos o sistema de interesse que tem os estados s envolvidos dentro de um conjunto S. A cada instante t, o agente observa o estado s_t do sistema e responde aplicando uma ação a_t, escolhida dentro de um conjunto A. Então, a dinâmica do sistema produz um novo estado s_{t+1} e o agente recebe o valor escalar r_t, chamado de recompensa. Esse ciclo se repete e a recompensa total do agente no tempo t é

$$\sum_{t=1}^{\infty} \gamma^{t-1} r_t$$

onde γ é o fator fixado de desconto.

O valor escalar da recompensa é calculado por uma função fixa $r_t = r(s_t, a_t)$, conhecida como heurística. E, as ações do agente são tomadas de acordo com uma política π, que é dependente do estado s_t e gera uma ação dentro do conjunto A. Então, o objetivo do agente é de maximizar sua recompensa, escolhendo a melhor política π.

Para conquistar esse objetivo, o agente interage de forma cíclica com o sistema e toma o conjunto ação a_t, estado s_t e recompensa r_t para escolher uma política π que maximize a recompensa acumulada do agente.

2.3.3.2 *Asynchronous Advantage Actor-Critic*

O algoritmo *Asynchronous Advantage Actor-Critic* (A3C), introduzido por (Mnih *et al.*, 2016), é uma método de aprendizado por reforço que combina métodos baseados em política e em valor. Ele se destaca pela utilização de múltiplos agentes que treinam de forma assíncrona, cada um com uma cópia separada do ambiente. Essa abordagem contrasta com os métodos tradicionais que dependem de um único agente e ambiente centralizados. A execução assíncrona permite que os agentes explorem diferentes partes do espaço de estado de maneira mais eficiente e robusta, acelerando o processo de aprendizado e melhorando a estabilidade do treinamento (Mnih *et al.*, 2016).

A estrutura do A3C integra o método ator-crítico, onde o ator ajusta a política para maximizar a recompensa e o crítico estima o valor da política atual. O algoritmo introduz a vantagem de utilizar múltiplas *threads* ou processos para atualizar os parâmetros do modelo, mitigando problemas como correlações entre amostras consecutivas e a necessidade de grande capacidade de memória para armazenar transições de estado. Isso resulta em um treinamento mais eficiente e com menor uso de recursos computacionais (Mnih *et al.*, 2016).

Os experimentos conduzidos por (Mnih *et al.*, 2016). demonstraram que o A3C tem um bom desempenho em várias tarefas, incluindo jogos Atari, controle de robôs e treinamento de veículos autônomos. O A3C mostrou ser capaz de lidar com ambientes altamente dinâmicos e de grande complexidade, confirmando sua eficácia e versatilidade. Em resumo, o A3C representa um avanço significativo na aplicação de aprendizado por reforço, permitindo treinar redes neurais de forma estável e eficiente em uma ampla gama de cenários (Mnih *et al.*, 2016).

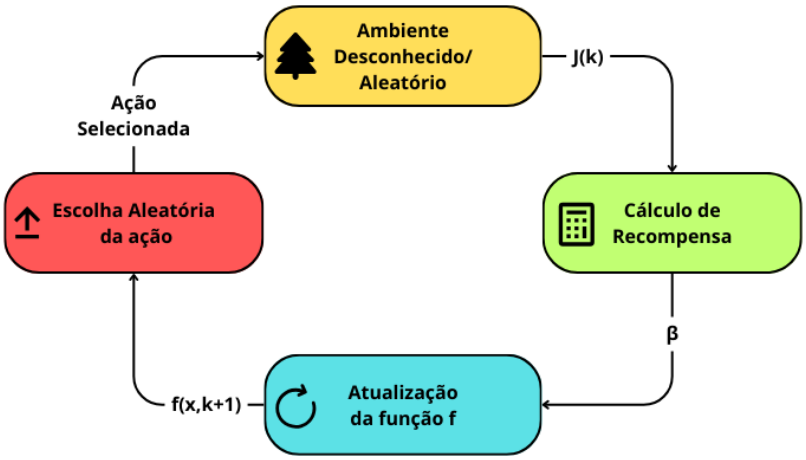
2.3.3.3 CARLA

O algoritmo *Continuous Action Reinforcement Learning Automata* (CARLA), introduzido por(Howell *et al.*, 1997), é uma abordagem que opera por meio de interações com um ambiente aleatório ou desconhecido, selecionando ações em um processo estocástico de tentativa e erro. Em cenários que envolvem parâmetros contínuos que podem ser alterados com segurança no ambiente, o CARLA é considerado uma abordagem simples e eficiente. Uma de suas vantagens é fornecer informações adicionais sobre a convergência por meio das funções de densidade de probabilidade. Esse algoritmo tem sido aplicado com sucesso em diversas áreas, como no controle de suspensão ativa (Howell *et al.*, 1997) e no projeto de filtros digitais (Howell; Gordon, 1998). As informações presentes nesta seção são baseadas em (Howell; Best, 2000), para simplificar a leitura essa citação será ocultada dos próximos parágrafos.

O esquema do CARLA, representado na Figura 9, apresenta uma abordagem na qual um autômato utiliza uma função de densidade probabilística f para guiar a seleção da próxima ação. Essa função f prioriza as ações com maiores valores, o que significa que as ações mais promissoras têm uma probabilidade maior de serem escolhidas. Quando uma ação selecionada resulta em uma melhora no desempenho do sistema, o sistema de aprendizado aumenta a probabilidade dessa ação ser selecionada novamente. Esse ajuste é feito modificando a função f com uma função gaussiana centrada na ação que gerou o melhor desempenho. Assim, não apenas a própria ação é considerada, mas também as ações próximas a ela têm suas probabilidades aumentadas.

À medida que o sistema continua aprendendo, a função de densidade probabilística converge gradualmente para uma distribuição em torno do valor desejado do

Figura 9 – Algoritmo CARLA



Fonte: Autoria própria (2024).

parâmetro. Esse processo de aprendizado contínuo permite que o CARLA refine suas decisões ao longo do tempo, adaptando-se às mudanças no ambiente e buscando alcançar o objetivo desejado de forma mais eficiente. A capacidade de ajustar dinamicamente as probabilidades das ações com base no desempenho passado é uma característica fundamental do CARLA, que o torna uma ferramenta poderosa para lidar com problemas complexos em ambientes dinâmicos e incertos.

Sendo x o parâmetro de saída do CARLA, definido em uma faixa específica $\{x_{min}, x_{max}\}$, a função de densidade probabilística $f(x,k)$, onde k é o número de iterações, é inicialmente definida como:

$$f(x,1) = \begin{cases} \frac{1}{x(max)-x(min)}, & \text{se } x(min) \leq x \leq x(max) \\ 0, & \text{senão} \end{cases} \tag{8}$$

E a ação é tomada através da Equação:

$$z(k) = \int_{x_{max}}^{x_{min}} f(x,k) * dx \tag{9}$$

Onde $z(k)$ é um valor aleatório selecionado a cada nova interação de maneira uniforme entre a faixa $\{0,1\}$.

Então, após a ação ser tomada, ela é aplicada no ambiente por um período definido e, em seguida, um valor escalar $J(k)$ é calculado de acordo com uma função de custo predefinida. Com o valor do custo estabelecido, o algoritmo emprega a regra de recompensa/punição, na qual, se a ação resultar em um custo maior que a média, nenhuma alteração é feita ($\beta = 0$); entretanto, se a ação resultar em um custo menor que a média, f é ajustado, com o valor da recompensa limitado em $\beta = 1$. Esse comportamento é definido pela Equação 10.

$$\beta(k) = min(max(0, \frac{J_{med} - J(k)}{J_{med} - J_{min}}), 1) \tag{10}$$

Após o cálculo da recompensa β a função de densidade probabilística é atualizada de acordo com a Equação 11.

$$f(x, k+1) = \begin{cases} \alpha(k) * (f(x,k) + \beta * H(x,r)), & \text{se } x(min) \leq x \leq x(max) \\ 0, & \text{senão} \end{cases} \tag{11}$$

onde H é função de vizinhança gaussiana simétrica e r é a ação escolhida nessa interação:

$$H(x, r) = (\frac{g_h}{x_{max} - x_{min}}) * \exp{(-\frac{(x-r)^2}{2(g_w(x_{max} - x_{min})^2})} \tag{12}$$

Onde g_w e g_h são parâmetros livres que determinam a velocidade e a resolução do aprendizado, ajustando a altura e a largura de H. Os valores padrões para esses parâmetros são $g_w = 0.2$ e $g_h = 0.3$. Além disso, o parâmetro α da Equação 13 é utilizado para normalizar a função de densidade probabilística e é definido como:

$$\alpha(k) = \frac{1}{\int_{x_{max}}^{x_{min}} (f(x,k) + \beta(k)H(x,r))dx} \tag{13}$$

Onde a integral na parte inferior representa a área sob a função de distribuição de probabilidade após os ajustes de acordo com a recompensa. Então, como α é o inverso desse valor, temos que esse parâmetro é o normalizador desta função, para que o valor da integral da função $f(x, k+1)$ seja sempre 1, ao multiplicar a nova função f com α como mostra a Equação 11.

2.4 TRABALHOS RELACIONADOS

Com os conceitos e conhecimentos que serão a base para este trabalho introduzidos nas seções passadas, essa seção tem objetivo de apresentar trabalhos relacionados e como esses desenvolveram as suas soluções para seus problemas.

O artigo "Reinforcement learning approach to autonomous PID tuning" (Dogru *et al.*, 2022) desenvolveu um novo método para ajuste de controladores PID ao formular o problema de ajuste de PID como um RL limitado, essas limitações garantem a segurança da operação durante a e exploração do ambiente. O primeiro passo no desenvolvimento foi a modelagem inicial do sistema que foi feita identificando uma resposta aproximada a um passo e, então, foi construído o modelo *offline*. Em seguida, o agente DRL aprendeu as dinâmicas do modelo *offline*. Quando o agente atingiu um desempenho satisfatório no passo anterior, ele foi submetido ao ambiente *online* para realizar o ajuste fino do que aprendeu nas dinâmicas reais do processo. Por

fim, esquema proposto foi testado, obtendo sucesso em um experimento em escala piloto. A proposta dos autores utiliza um agente DRL, o qual consegue se adaptar as não-linearidades do sistema. Assim, inspirado no trabalho de (Dogru *et al.*, 2022), esse TCC também divide a fase de aprendizado em simulada (*offline*) e real (*online*), tornando o processo mais seguro e menos custoso.

O artigo "On-line PID tuning for engine idle-speed control using continuous action reinforcement learning automata" (Howell; Best, 2000) apresenta uma abordagem para a automação do ajuste de controladores PID utilizando CARLA. O foco do estudo foi otimizar os parâmetros do controlador para o controle da velocidade de um motor ocioso, que se baseia em ajustar o ângulo da embreagem regulando a velocidade do motor, para minimizar a função custo definida. Após a implementação do algoritmo e a criação de três autômatos - uma para cada parâmetro PID - foram escolhidos os parâmetros do método de Ziegler-Nichols para estado inicial de aprendizado , então o agente foi testado com um modelo ideal de motor, com objetivo de minimizar o critério de custo. Ao final do aprendizado no modelo simulado, o agente foi submetido a um teste de um motor real. As conclusões foram que o controlador PID ajustado primeiro no ambiente simulado e depois no ambiente real obteve ganhos de controle significativos e benefícios em termos de custos, confirmando a efetividade do algoritmo CARLA. Assim como a implementação proposta no capítulo seguinte, esse TCC utiliza o algoritmo CARLA para o ajuste dos parâmetros PID, que se destaca devido a sua simplicidade e transparência, porém diferente do sistema dinâmico utilizado por (Howell; Best, 2000), as máquinas R2R, objeto deste estudo, são muito diversas e um único modelo ideal poderia não representar com fidelidade o ambiente real, então a implementação proposta no capítulo seguinte apresenta uma abordagem diferente para a obtenção do ambiente simulado.

O artigo "Deep reinforcement learning with shallow controllers: An experimental application to PID tuning" (Lawrence *et al.*, 2022) demonstra a implementação de um algoritmo de DRL usando o controlador PID como uma política treinável no ambiente dinâmico real. Esse trabalho foca na avaliação do DRL para um ajuste automático do PID de modo *model-free*, ou seja, sem utilizar ou estimar as dinâmicas do ambiente. O primeiro passo foi a utilização de um PLC comum para a implementação do controlador PID, evitando a necessidade de adição de *hardware*. Em seguida, foi inicializado a lei de controle em uma região segura dos parâmetros de controla para garantir a segurança durante os estágios iniciais de aprendizado. Então foi conduzido o treinando em tempo real diretamente no sistema dinâmico real, sem o uso de pré-treinamento em modelos de simulação ou banco de dados coletados. Após o treinamento, foi avaliado o desempenho da estratégia de ajuste e do algoritmo de DRL de acordo com um critérios específicos. Em conclusão, essa abordagem demonstrou a viabilidade de implementar um algoritmo de DRL em um sistema dinâmico real uti-

lizando um controlador PID, já que se obteve um controlador PID bem ajustado que os profissionais podem facilmente compreender e utilizar com confiança. Diferente da implementação proposta no próximo capítulo e dos outros trabalhos expostos anteriormente nessa seção, esse trabalho não utiliza de um ambiente simulado, então o número de épocas de treinamento pode ser limitada devido ao custo de utilização da máquina e é necessário limitar os parâmetros do controlador, como esse TCC fez, afim de manter a segurança do processo no ambiente, assim como adicionar gatilhos caso esse processo entre em uma região de funcionamento não segura.

3 METODOLOGIA

Neste capítulo são descritos os modelos do sistema e métodos de avaliação empregados neste trabalho para implementação do algoritmo de ajuste de controlador com base em RL e para avaliar o desempenho do mesmo. A Figura 10 demonstra as etapas deste trabalho.

Nessa figura, está representada a divisão do trabalho em duas principais etapas que foram trabalhadas de forma paralela. O ambiente de simulação, o qual tem como objetivo construir, com base em dados coletados, um ambiente que simule as dinâmicas da máquina e o algoritmo de ajuste, o qual tem como objetivo implementar um agente que seja capaz de interagir com o ambiente, seja esse real ou simulado, e com base nessas iterações ajustar o controlador PID para parâmetros que diminuam a função custo, que é definida dentro dessa etapa.

Os tópicos a seguir explicarão o desenvolvimento de cada uma das etapas da Figura 10.

3.1 PROJETO DO SISTEMA

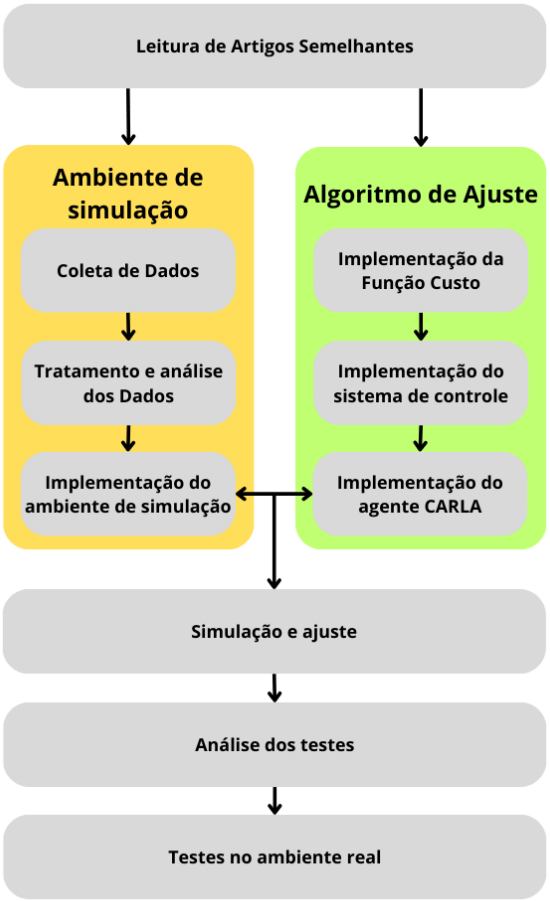
Espera-se que o algoritmo desenvolvido durante este trabalho seja generalizado para o uso de ajuste de controladores PID em sistemas de manufatura. Então, apesar do trabalho manter o foco na modelagem e ajuste de controlador para sistema R2R, um esboço do sistema genérico que pode ser utilizado em qualquer malha fechada que utiliza controladores PID está representado na Figura 11.

O sistema proposto é composto pelo sistema dinâmico que tem a implementação do controlador PID e, também, sensores que permitem não só o funcionamento da malha fechada, mas também a criação de um ambiente de simulação que posteriormente será utilizado para o treinando do agente de RL.

O próximo componente é o ambiente de simulação, este tem um papel fundamental no sistema pois permite que sejam feitos testes e iterações sobre uma simulação do sistema dinâmico. Com isso, é possível evitar custos financeiros associados ao uso do ambiente real, como gastos com energia elétrica, uso de equipamentos e materiais, entre outros. Ao simular o ambiente, pode-se reduzir significativamente os custos operacionais.

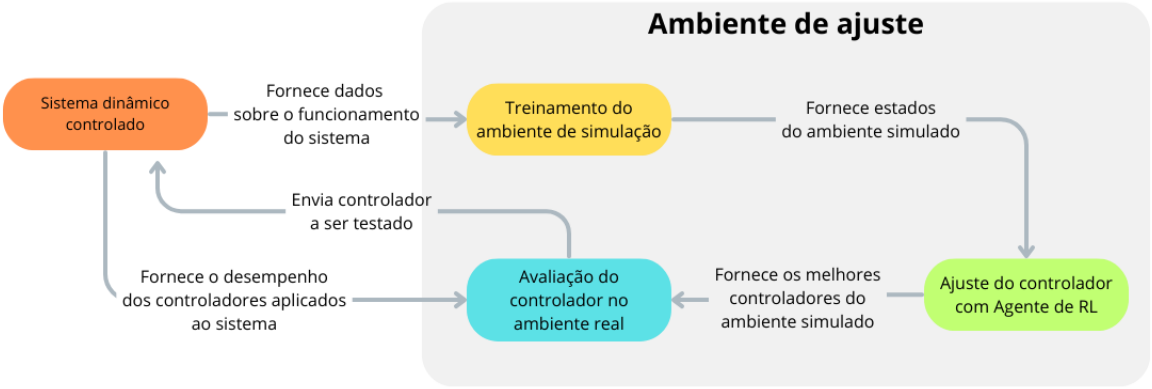
Além disso, a utilização de um ambiente simulado permite uma execução mais rápida dos experimentos de treinamento do agente, pois não estamos limitados pelas restrições de tempo do ambiente real. Isso significa que pode-se realizar um maior número de tentativas e iterações em um menor período de tempo, acelerando o processo de aprendizado do agente.

Figura 10 – Fluxograma das etapas do trabalho



Fonte: Autoria própria (2024).

Figura 11 – Sistema Proposto



Fonte: Autoria própria (2024).

Outra vantagem é a segurança da máquina durante o processo de exploração do ambiente pelo agente. Em um ambiente real, especialmente em casos onde o agente está interagindo com equipamentos ou dispositivos físicos, pode haver riscos de danos ou acidentes. Com a simulação, pode-se evitar esses riscos e garantir a integridade da máquina e a segurança dos operadores.

Além desses motivos, a simulação também oferece maior controle sobre as condições do ambiente, permitindo a manipulação de variáveis específicas para testar diferentes cenários e condições de operação. Isso proporciona uma maior compreensão do comportamento do agente em diferentes contextos e permite o refinamento do algoritmo de RL de forma mais eficaz.

Então, com o ambiente de simulação implementado, o próximo componente é o agente de RL que tem o objetivo de ajustar o controlador para se reduzir a função custo citada na Seção 2.3.1.4. Para tanto, o agente utiliza o ambiente de simulação para avaliar a seleção dos parâmetros do controlador PID e seus valores internos de acordo com o resultado.

Após atingir um resultado esperado ou o número de interações limite, o agente fornece o melhor controlador do ambiente de simulação para o componente responsável por avaliar este controlador no ambiente real. Este componente envia o controlador para o ambiente real e então observa seu desempenho, a avaliação é dada pelo valor da função custo desse controlador, sendo comparado ao desempenho de outros controladores.

Neste trabalho, o sistema dinâmico é uma máquina R2R operada pela ferramenta TwinCat provida pela Beckhoff, essa ferramenta é responsável pela comunicação com os módulos da máquina e pela coleta de dados da mesma, nessa seção essa ferramenta será referenciada apenas como PLC e o ambiente de ajuste é um máquina virtual com sistema operacional Ubuntu 22.04, processador com 8 cores virtuais, 16GB de memória primária. O algoritmo de RL escolhido foi o CARLA devido à sua simplicidade. Embora algoritmos que utilizam RNA, como o A3C, fossem também uma possível solução, como discutido na Seção 2.4, optou-se pelo CARLA. Como será detalhado na Seção 4.5, a complexidade de implementar o A3C em um PLC, que opera com uma linguagem de baixo nível, levou à decisão de descartar esse algoritmo considerando que as futuras etapas desse trabalho incluem a implementação do agente diretamente no sistema real para o refinamento do controlador após as etapas de treinamento em ambiente simulado.

Esse sistema dinâmico é caracterizado principalmente pelas variáveis do Quadro 2. A tensão no ciclo atual sobre o substrato, obtida usando um sensor de tensão, que é um parâmetro importante para o desempenho da manufatura R2R; a penúltima e a antepenúltima tensão, que são valores da tensão atual dos ciclos passados e mantidos em nível lógico dentro do PLC, sendo importantes para o controle dessa variável;

a saída do controlador, que representa uma ação sobre a velocidade dos enroladores com o objetivo de manter a tensão sobre o substrato igual ao *set point*; o *set point*, que é uma variável existente apenas no PLC e representa o valor definido pelo operador para o qual a tensão deve ser mantida; a aceleração do enrolador que contém o substrato; a velocidade da unidade de compressão, que representa a velocidade na qual o substrato está passando pela manufatura; o diâmetro do substrato no enrolador; e a velocidade de sincronização do enrolador. Essas variáveis são monitoradas e controladas para garantir a eficiência e a qualidade do processo de manufatura e, nesse trabalho, para a implementação do ambiente simulado.

Quadro 2 – Variáveis do sitema dinâmico

Nome
Tensão atual sobre a membrana
Penúltima tensão sobre a membrana
Antepenúltima tensão sobre a membrana
Aceleração do enrolador
Set point
Velocidade da unidade de compressão
Diâmetro do substrato no enrolador
Velocidade de sincronização do enrolador

Fonte: Autoria própria (2024).

3.2 COLETA DE DADOS

A coleta de dados é uma etapa inicial em qualquer análise de processos industriais, fornecendo informações essenciais para compreender o funcionamento e o desempenho de máquinas e sistemas. Neste estudo, foram coletados os dados diretamente do ambiente real, visando capturar uma visão detalhada e precisa das operações do sistema dinâmico.

Durante um período de uma hora, foram realizadas medições das variáveis relevantes a cada intervalo de 1 milissegundo. Essa frequência de amostragem foi escolhida para garantir a captura de dados em alta resolução, permitindo uma análise minuciosa do comportamento da máquina ao longo do tempo. Além disso, durante esse período, foram introduzidas mudanças aleatórias no valor do *set point* e nos parâmetros do controlador a cada 30 segundos.

Essas mudanças foram projetadas para simular uma variedade de cenários operacionais e testar a resposta da máquina a diferentes condições de operação, fazendo com que o banco de dados produzido englobe uma quantidade maior de dinâmicas da máquina. Essas mudanças incluem alterações no *set point* e no controlador PID para avaliar a capacidade de resposta do sistema às mudanças na tensão do

substrato, bem como desligamentos temporários do controlador para examinar seu impacto no desempenho da máquina. Apesar de ser selecionada de forma aleatória, essas mudanças foram definidas dentro de um intervalo para que não entre em nenhum estado que possa danificar a máquina. O valor do *set point* foi definido entre 0 e 100 Newtons e os controladores foram selecionados da Tabela 1, apesar de nenhum desses controladores terem um desempenho satisfatório em todos os casos eles são estáveis, então podem ser utilizados de forma segura.

Tabela 1 – Controladores usados para a coleta de dados do ambiente real

K_p	T_n	T_v
0	1	50
0	1	100
10	1	100
5	50	50
2,5	75	25
7.5	25	75
5	20	80
0.7	40	10

Fonte: Autoria própria (2024).

As variáveis coletadas incluem o *set point* da tensão, a aceleração do enrolador, a velocidade da unidade de compressão, o diâmetro do substrato no enrolador, a velocidade de sincronização do enrolador, a saída do controlador, a tensão atual, a penúltima tensão e a antepenúltima tensão sobre a membrana. Essa seleção de uma gama de variáveis foi essencial para capturar todos os aspectos relevantes do processo R2R e permitir uma análise e seleção de algumas para serem utilizadas nas próximas etapas.

Essa abordagem para coleta de dados, permitiu ter informações sobre a dinâmica da máquina. Os dados coletados servirão como base para análises e simulações futuras e o desenvolvimento de um algoritmo de ajuste de controlador que visa aumentar a eficiência e a qualidade da produção.

3.3 TRATAMENTO E ANÁLISE DOS DADOS

O tratamento dos dados coletados é uma etapa essencial para extrair informações significativas e prepará-las para análise. Nesta seção, descrevemos o processo pelo qual os dados brutos foram manipulados e preparados para análise, bem como os métodos utilizados para analisar esses dados e, então, selecionar quais são relevantes para a próxima etapa.

Inicialmente, os dados do ambiente real foram recebidos em formato de tabela como descrito na seção anterior. Para facilitar a análise, esses dados foram converti-

dos para o formato CSV (Comma Separated Values), que é amplamente utilizado para armazenar dados tabulares. Durante essa conversão, o cabeçalho da tabela foi mantido, preservando os nomes das variáveis coletadas mencionadas na seção anterior.

Após a conversão para o formato CSV, a coluna "Name", que representava o índice de cada aquisição de dados a partir de 0 e que é adicionada pela ferramenta de coleta de dados do PLC, foi removida. Como esse valor não está relacionado à dinâmica da máquina e não contribui para a análise, sua remoção simplifica a estrutura dos dados e elimina informações irrelevantes.

Na fase de análise, os dados tratados foram carregados em um ambiente de programação em Python[1], utilizando a biblioteca Pandas, uma ferramenta para manipulação e análise de dados tabulares. Com o objetivo de correlacionar de forma quantitativa a tensão atual sobre a membrana e as demais variáveis, foi feito, com auxílio do Pandas, o cálculo do coeficiente de correlação de Pearson, descrito pela Equação 14, para a variável de tensão atual sobre a membrana em relação as demais variáveis, exceto o *set point*.

$$\rho = \frac{\sum_{i=1}^{n}(x_i - x_{med})(y_i - y_{med})}{\sqrt{\sum_{i=1}^{n}(x_i - x_{med})^2 * \sum_{i=1}^{n}(y_i - y_{med})^2}} \tag{14}$$

Onde x e y são as valores coletados das variáveis que esta sendo calculado o coeficiente e n é o número de coletas.

A tensão atual sobre a membrana é importante, pois representa o valor sobre o qual o controlador PID atuará no futuro. Portanto, é fundamental entender como essa variável se relaciona com outras variáveis do sistema. A correlação de Pearson fornece uma medida da força e direção dessa relação linear entre duas variáveis, ajudando a identificar possíveis influências ou dependências.

Então, com os coeficientes obtidos, como mostra a Tabela 2, podemos notar que os valores de penúltima tensão e antepenúltima tensão têm um alto coeficiente e os valores da saída do controlador e aceleração do enrolador tem um menor valor mas ainda são valores que mostram uma influência, e como no sistema dinâmico a aceleração é calculada utilizando a saída do controlador, podemos então tomar apenas este valor para a etapa de construção do ambiente de simulação.

3.4 IMPLEMENTAÇÃO DO AMBIENTE DE SIMULAÇÃO

Conforme delineado na Seção 3.1, a implementação de um ambiente de simulação é importante para a realização de experimentos seguros e eficientes na exploração do sistema dinâmico por parte do agente responsável por ajustar os parâmetros PID sem a necessidade de intervenção direta no sistema real. Nesta seção, aborda-

[1] GitHub com o Código desenvolvido: https://github.com/LuisLima2002/CarlaPidTuning

Tabela 2 – Coeficiente de Pearson das variáveis coletadas em relação à tensão atual

Variável coletada	Coeficiente de Pearson
Penúltima tensão	0.9998
Antepenúltima tensão	0.9995
Saída do controlador	-0.0355
Aceleração do enrolador	0.05807
Velocidade da unidade de compressão	0.0037
Diâmetro do substrato no enrolador	-0.0146
Velocidade de sincronização do enrolador	0.0093

Fonte: Autoria própria (2024).

remos os aspectos práticos da implementação do ambiente de simulação, detalhando suas características e funcionalidades essenciais.

O ambiente de simulação é composto por um controlador e um preditor, os quais desempenham papéis distintos e interagem entre si para reproduzir o comportamento do ambiente dinâmico real. O controlador recebe os estados fornecidos pelo preditor e utiliza essa informação para tomar uma ação, seguindo uma lógica semelhante à lógica do ambiente real, este controlador é o alvo de ajuste do agente implementado na próxima seção. Por outro lado, o preditor utiliza os estados anteriores e a saída do controlador para gerar a estimativa da tensão atual. Essa tensão estimada é então realimentada para o próximo passo da simulação, formando um ciclo contínuo de interação entre o controlador e o preditor. Esse ciclo de realimentação permite que o sistema simule a dinâmica do ambiente real de forma iterativa e progressiva, possibilitando a avaliação do desempenho do controlador em diferentes cenários e condições operacionais.

Então, visando simplificar o processo de treinamento do modelo que será utilizado como preditor, foi empregada a ferramenta AutoGluon, que é uma framework de AutoML(Automatic Machine Learning). Esta ferramenta automatiza o processo de seleção e treinamento dos melhores modelos de aprendizado de máquina para um conjunto de dados específico, buscando encontrar a configuração ideal dos algoritmos e hiper-parâmetros com ênfase em empilhamento automatizado de modelos, ele também tem tratamento automatizado dos dados, identificando o seu tipo e lidando com ausência e balanceamento das entradas (AUTOGLUON, 2022). Para a modelagem da regressão, foram utilizadas como entradas a saída do controlador, o penúltimo valor de tensão e o antepenúltimo valor de tensão, enquanto a saída desse modelo foi configurada para prever a tensão atual e então foi utilizados os dados processados para o treinamento do modelo, como mostra o Código 3.1.

Para o desenvolvimento do segundo candidato a ser o modelo de regressão, foi implementado uma RNA utilizando a biblioteca PyTorch. Diferentemente da abordagem anterior com AutoGluon, a implementação de uma RNA é mais complexa e de-

```
train_data = TabularDataset('./trainData.csv')
test_data = TabularDataset('./testData.csv')
predictor = TabularPredictor(label="lrMeasuredForce",eval_metric="mae",
                             path=f"../medium_quality-10min")
predictor.fit(train_data,
              presets="medium_quality",
              time_limit=10*60)
test_data_nolab = test_data.drop(columns=["lrMeasuredForce"])
y_test = test_data.get("lrMeasuredForce")
y_pred = predictor.predict(test_data_nolab)
perf = predictor.evaluate_predictions(y_true=y_test,
                                      y_pred=y_pred,
                                      auxiliary_metrics=True)
print(str(perf))
```

Código 3.1 – Treinamento do preditor AutoML

```
model = nn.Sequential(nn.Linear(len(X[0]),64),nn.ReLU()
                      ,nn.Linear(64,32),nn.ReLU()
                      ,nn.Linear(32,1))
loss_fn=nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
epochs=100
loss_history=[]
test_loss_history=[]
loader = dataUtil.DataLoader(dataUtil.TensorDataset(X_train, y_train)
                             , shuffle=True, batch_size=int(len(X_train/2)))
for epoch in range(epochs):
    model.train()
    for X_batch, y_batch in loader:
        y_pred = model(X_batch)
        loss = loss_fn(y_pred, y_batch)
        loss_history.append(loss.to('cpu').detach().numpy())
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    if epoch % 10 != 0: continue
    model.eval()
    with torch.no_grad():
        y_pred = model(X_test)
        test_rmse = np.sqrt(loss_fn(y_pred, y_test))
    print("Epoch %d: test RMSE %.4f" % (epoch, test_rmse))
plt.plot(loss_history)
plt.show()
```

Código 3.2 – Treinamento do preditor PyTorch

manda maior expertise em programação e conhecimento sobre redes neurais. Nesse caso, a rede neural foi configurada para receber as mesmas variáveis de entrada utilizadas no AutoGluon: a saída do controlador, o penúltimo valor de tensão e o antepenúltimo valor de tensão. Para a arquitetura da rede, optou-se por duas camadas internas, a primeira com 64 neurônios e a segunda com 32 neurônios, com a função de ativação ReLU aplicada entre elas, como mostra o Código 3.2.

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(X.detach().numpy(), y.detach().numpy())
print(reg.score(X, y))
y_predicted=reg.predict(X)
print(reg.coef_)
```

Código 3.3 – Treinamento do preditor Regressão linear

O terceiro candidato a ser desenvolvido como modelo para o preditor do ambiente de simulação visa simplificar e tornar o modelo mais previsível e alinhado com o comportamento esperado, optou-se por utilizar um modelo de regressão linear como preditor. Esse modelo foi configurado para utilizar as mesmas variáveis de entrada dos modelos anteriores, ou seja, tem como entradas os valores de penúltima e antepenúltima tensão, assim como os valores de saída do controlador e tem como saída a tensão atual sobre a membrana. Para treinamento desse modelo foi utilizado a biblioteca "sklearn", como mostra o Código 3.3

Os resultados obtidos por cada modelo, seus respectivas desempenhos e suas implicações no ambiente de simulação serão discutidos no Capítulo 4.

3.5 IMPLEMENTAÇÃO DO MODELO DE AJUSTE

Com o ambiente de simulação pronto, conforme descrito na última seção, o foco agora se volta para a implementação do agente responsável por ajustar o controlador nesse ambiente. A função principal desse agente é observar o ambiente a cada iteração e, com base nessa observação, gerar um custo que reflita o desempenho do controlador atual. Em seguida, o agente tem como objetivo minimizar essa função de custo, buscando encontrar o melhor controlador para o ambiente em questão. O Quadro 3 apresenta os principais parâmetros envolvidos na implementação do modelo de ajuste e a faixa de valores desses parâmetros nesse trabalho. Para compreender a implementação desse processo, é necessário iniciar pela definição e entendimento da função de custo.

3.5.1 Função de custo

Para avaliar o desempenho de um sistema de controle, existem vários pontos que podem ser considerados na avaliação. Um deles é o tempo de resposta, que se refere ao tempo que o sistema leva para atingir uma determinada porcentagem do valor de regime estabelecido após uma perturbação. Quanto menor o tempo de resposta, mais rápido o sistema é em ajustar-se às mudanças no ambiente.

Outro aspecto importante é o sobressinal, que é a quantidade pela qual a resposta excede o valor de regime antes de estabilizar. Esse parâmetro indica a quantidade de oscilação inicial do sistema e é desejável mantê-lo o mais próximo possível de

Tabela 3 – Principais parâmetros da implementação do modelo de ajuste

Nome	Faixa de valores
Função custo	[0,$\inf$]
K_p	[0, 1.5]
T_n	[20 ms, 80 ms]
T_v	[0 ms, 60 ms]
T_d	0 ms
P_{Kp}	[0,$\inf$]
P_{Tn}	[0,$\inf$]
P_{Tv}	[0,$\inf$]
g_w	0.2
g_h	0.3
β	[0,1]
Δt	1 ms

Fonte: Autoria própria (2024).

zero para evitar oscilações indesejadas. Quando esse valor é muito alto, pode indicar a ocorrência de oscilações excessivas que, além de afetar a estabilidade do sistema, podem causar danos à máquina ou ao processo controlado.

O erro em estado estacionário é outro critério relevante, representando a diferença entre a saída desejada e a saída real após o sistema ter alcançado o equilíbrio. Idealmente, esse erro deve ser minimizado para garantir um desempenho preciso do sistema.

Além disso, as saídas limites do controlador também devem ser consideradas. Se o controlador atingir seus limites de saturação, isso pode indicar que o sistema não está respondendo conforme o esperado ou que o controlador está sobrecarregado, o que pode levar a um desempenho ruim do sistema ou, assim como no pico da resposta muito alto, a danos à máquina ou ao processo controlado .

Em relação à heurística escolhida neste trabalho, optou-se pelo soma do módulo do erro entre a tensão real e o *set point* de cada estado. Essa heurística foi selecionada para simplificar o sistema e testar a capacidade dos algoritmos de RL em minimizar a função custo nesse ambiente. Nesse contexto, o objetivo do agente é manter a tensão o mais próximo possível do *set point*, diminuindo o erro em estado estacionário e visando alcançar um bom desempenho do sistema de controle.

3.5.2 Sistema de controle

Para garantir que o sistema de controle utilizado no ambiente de simulação seja o mais próximo possível do sistema dinâmico real controlado pelo PLC, foi implementada uma comunicação direta com uma instância desse PLC. Essa instância do PLC replica exatamente a estrutura do controlador real, fornecendo uma representação fiel do ambiente de controle. Isso significa que as mesmas funções e algoritmos

utilizados pelo PLC no ambiente real estão disponíveis para o sistema de ajuste no ambiente de simulação. Essa abordagem permite que o agente de ajuste interaja diretamente com o controlador virtual, experimentando as mesmas condições que encontraria no sistema real. Dessa forma, qualquer controlador definido pelo agente e testada no ambiente de simulação terá o mesmo comportamento quando aplicada ao sistema real controlado pelo PLC.

Esse controlador implementado dentro do PLC recebe o valor do *set point*, que é mantido constante, o valor atual da variável a ser controlada, os parâmetros do controlador PID e o tempo de amostragem Δt, que é igual a 1 milissegundo. E, com esses valores, o controlador retorna a saída a ser aplicada no sistema dinâmico e o estado do mesmo, que caso seja de erro significa que o mesmo não está mais agindo sobre o sistema. A Função de transferência desse controlador está descrita na Equação 15.

$$G_{PID}(s) = K_p * (1 + \frac{1}{T_n s} + \frac{T_v s}{1 + T_d s}) \tag{15}$$

Onde K_p é o ganho do sistema, T_n é o parâmetro integral, que tem mais impacto quanto menor for seu valor, T_v é o parâmetro derivativo e T_d é o parâmetro de amortecimento, que neste trabalho é fixado em 0, ou seja, não age sobre o sistema.

3.5.3 Agente de busca excessiva

Nessa seção, é apresentado o desenvolvimento de um agente que busca o melhor controlador dentro da granularidade dos testes, ou seja, aquele que, dentre todas as possíveis combinações, minimizou a função de custo definida na Seção 3.5.1.

Para a implementação desse agente, o primeiro passo é definir a faixa de valores de cada um dos parâmetros do controlador PID, assim como a amostragem nessa faixa. Considerando que o ganho do sistema K_p tem P_{Kp} possibilidades, o parâmetro integral T_n tem P_{Tn} possibilidades, e o parâmetro derivativo T_v tem P_{Tv} possibilidades, esse agente terá $P_{Kp} * P_{Tn} * P_{Tv}$ ciclos no ambiente de simulação.

O próximo passo na implementação do agente é a definição dos pontos iniciais e do *set point* que serão usados nos testes de cada controlador. Para garantir que esses parâmetros escolhidos representem bem o ambiente real, é importante selecionar pontos comuns ao funcionamento típico do sistema.

A fase seguinte é o desenvolvimento do Código 3.4. Esse laço tem a função de realizar todas as combinações dos parâmetros PID definidas anteriormente e, então, inseri-las no ambiente de simulação. Cada combinação gerará um valor para a função custo, e o agente manterá a combinação que obteve o menor valor na função custo.

Essa abordagem sistemática visa garantir que o agente explore todas as possíveis configurações do controlador PID, identificando a configuração que oferece o melhor desempenho de acordo com os critérios definidos. A capacidade de realizar

```
soe_lowest = np.inf
bestConstants=[0,0,0]
i=0
for kp in kps:
    print(str(i))
    for tn in tns:
        for tv in tvs:
            controller = PID(kp,tn,tv)
            history= response(predictor,controller,u,initialState)
            soe = np.sum(np.abs(u - history))
            if soe< soe_lowest:
                soe_lowest=soe
                bestConstants=[kp,tn,tv]
                print("Best constante now:", bestConstants)
    i+=1
```

Código 3.4 – Laços de interação com o ambiente de simulação

esses testes em um ambiente de simulação permite uma avaliação abrangente e segura, sem riscos para o sistema real. Assim, ao final do processo, o melhor controlador identificado no sistema de simulação estará pronto para ser testado no ambiente real, afim de validar o seu desempenho real.

3.5.4 CARLA

O código desenvolvido nessa seção está disponível na integra no GitHub[2]. Para a implementação do algoritmo CARLA, o primeiro passo foi a criação da classe "Automata" que representa o agente responsável por encontrar o valor que minimiza a função custo. O construtor dessa classe, mostrado no Código 3.5, recebe os valores de mínimo e máximo que limitam a saída do autômato, o número de amostras que a função de distribuição de probabilidade deve manter, quanto maior esse valor, maior é a resolução dessa função, porém mais memória e mais tempo são consumidos, e os valores de g_h e g_w, como descrito na Seção 2.3.3.3.Esses parâmetros são essenciais para definir o comportamento e a eficácia do agente, pois eles determinam a velocidade e a resolução do aprendizado. Valores maiores de g_h e g_w podem levar a um aprendizado mais preciso, mas também aumentam o tempo de processamento e o uso de memória primária. Portanto, é importante encontrar um equilíbrio entre precisão e eficiência computacional ao definir esses parâmetros.

A implementação da função *step*, mostrada no Código 3.6, é responsável por selecionar um ponto dentro dos limites predefinidos durante a criação do autômato, conforme especificado pela Equação 9. Esse ponto escolhido representa a próxima ação a ser tomada pelo agente no ambiente de simulação.

Além disso, a função *step* também é responsável por salvar essa última escolha, permitindo que nos próximos passos o algoritmo atualize a função de distribuição

[2] GitHub com o Código desenvolvido: https://github.com/LuisLima2002/CarlaPidTuning

```
def __init__(self,min,max,gw,gh,probabilityStoreSize=1000):
    self.min = min # minimum value posible for Automata output
    self.max = max # maximum value posible for Automata output
    self.gh = gh # Learning ratio
    self.gw = gw # How large is the affect area
    self.probabilityStoreSize = probabilityStoreSize
    self.k=0 # number of epochs (inner parameter)
    self.lastAction=0 # (inner parameter)
    self.R=[] # store of J


    self.probabilityDistribution=[]  # function f(xi,k)
    self.historyProbabilityDistribution=[]
    initialProb = 1/(self.max-self.min)
    for i in range(self.probabilityStoreSize):
        x=self.min+i*(self.max-self.min)/(self.probabilityStoreSize-1)
        self.probabilityDistribution.append([x,initialProb])
```

Código 3.5 – Construtor do Automata

```
def step(self):
    # random between 0-1, action will be the point where
    # the area bellow is equal to z
    z = random.random()

    area=0 # holds currently area ( numeric integral )
    for i in range(self.probabilityStoreSize-1):
        start = self.probabilityDistribution[i][0]
        startValue = self.probabilityDistribution[i][1]
        end = self.probabilityDistribution[i+1][0]
        endValue = self.probabilityDistribution[i+1][1]
        space = np.linspace(start,end,100)
        a = (startValue-endValue)/(start-end)
        b = startValue-start*a
        for j in range(len(space)-1):
            x1 = space[j]
            x2 = space[j+1]
            probability=a*x1+b
            area+=probability*(x2-x1)
            if(abs(area - z)<0.01):
                self.lastAction=x1
                return x1
    raise Exception("Not found for z")
```

Código 3.6 – Função *step*

de probabilidade com base nas ações anteriores e em seu desempenho.

A implementação eficiente e precisa da função *step* é essencial para garantir que o algoritmo CARLA possa explorar efetivamente o espaço de ações disponíveis e encontrar a melhor estratégia para minimizar a função custo. Portanto, é importante testar e ajustar essa função durante o desenvolvimento do algoritmo, garantindo que ela seja capaz de fornecer resultados confiáveis e consistentes em diferentes cenários de simulação.

```
1 def explore(self):
2     x =  random.random()*(self.max-self.min)+self.min
3     self.lastAction=x
4     return x
```

Código 3.7 – Função *explore*

A implementação da função *explore*, mostrada no Código 3.7 tem objetivo de melhorar o processo de aprendizado do agente. Ao contrário da função *step*, a função *explore* seleciona um valor dentro da faixa definida com uma probabilidade distribuída de forma uniforme. Essa abordagem permite que o agente explore efetivamente todo o espaço de ações disponíveis, garantindo uma exploração abrangente e completa.

Assim como na função *step*, a função *explore* também salva a ação selecionada. Esse passo possibilita que o agente ajuste sua distribuição de probabilidade ao longo do tempo. Ao registrar as ações tomadas durante a exploração, o agente pode usar essas informações para atualizar sua compreensão do ambiente e adaptar sua estratégia de seleção de ações de acordo com as experiências passadas. Portanto, a implementação da função *explore* permite uma exploração abrangente e adaptativa do ambiente de simulação.

A última função implementada é utilizada para o processo de adaptação e melhoria contínua do agente CARLA. Conhecida como função *update*, mostrada no Código 3.8, ela é responsável por receber o valor da função custo, que reflete o desempenho do controlador no ambiente. Com base nesse valor, a função *update* utiliza a Equação 10 para calcular o valor de β, porém, com o objetivo de limitar o uso dos recursos computacionais, o valor J_{med} e J_{min} são calculados utilizando apenas os últimos 500 valores de custo obtidos pelo agente.

Após calcular o valor de β, a função *update* utiliza os parâmetros g_h e g_w, definidos previamente no construtor da classe "Automata", para ajustar a distribuição de probabilidade do agente. Essa atualização é realizada de acordo com a Equação 11, que permite ao agente adaptar sua estratégia de seleção de ações com base nas experiências passadas e na função custo calculada de acordo com o desempenho no ambiente.

Portanto, a implementação cuidadosa e eficaz da função *update* é fundamental para o sucesso do algoritmo CARLA. Ao permitir que o agente ajuste dinamicamente sua distribuição de probabilidade com base no desempenho passado, essa função possibilita uma melhoria contínua no desempenho do controlador, permitindo que ele se adapte de forma mais eficaz às demandas do ambiente de simulação.

```
def update(self,J):
    self.R.append(J)
    if(len(self.R)<3): return
    if(len(self.R)>500): self.R=self.R[-500:]
    self.Jmed= np.mean(self.R) # medium cost
    self.Jmin= np.min(self.R) # minimum cost
    self.k+=1

    # 0<B<1, if cost is greater than medium it will have a positive
    # impact on the probability, else no impact
    B=np.min([np.max([0.0,(self.Jmed-J)/(self.Jmed-self.Jmin)]),1.0])

    newprobDistri=[]
    a=0
    for i in range(len(self.probabilityDistribution)):
        x,probability = self.probabilityDistribution[i]

        gh_weighted = (self.gh/(self.max-self.min))

        exp_divider = (2*np.power(self.gw*(self.max-self.min),2))

        H=gh_weighted*np.exp(-np.power(x-self.lastAction,2)/exp_divider)
        propNN=probability+B*H
        newprobDistri.append(propNN)

        deltaX =(self.probabilityDistribution[i+1][0]-x)

        deltaY = (self.probabilityDistribution[i+1][1]-propNN)
        if(i+1<len(self.probabilityDistribution)):
            a+=propNN*deltaX+deltaY*deltaX/2
    a=1/a # normalise the function

    aux = []
    for pair in self.probabilityDistribution:
        aux.append(pair.copy())
    self.historyProbabilityDistribution.append(aux)

    for i in range(len(newprobDistri)):
        self.probabilityDistribution[i][1]=newprobDistri[i]*a
```

Código 3.8 – Função *update*

3.6 SIMULAÇÕES E AJUSTE

O ambiente de simulação é composto por um preditor e um controlador, conforme descrito na Seção 3.4. Então, para as simulações descritas a seguir, utilizou-se o agente CARLA conectado ao controlador do ambiente. Três autômatos CARLA foram criados e funcionam em conjunto como o agente de ajuste do controlador PID, com cada autômato vinculado a um parâmetro específico do controlador (P, I, D).

O processo de simulação e ajuste foi estruturado em duas fases principais: a fase exploratória e a fase de aprendizado, essas fases são explicadas nas próximas seções.

3.6.1 Fase exploratória

Inicialmente, realizou-se um ciclo de iterações exploratórias utilizando a função "explorar" de cada autômato. Este ciclo é importante no início do aprendizado para garantir que o ambiente seja amplamente conhecido e para evitar que um estado inicial crie um viés significativo durante o aprendizado. Durante cada iteração exploratória, os parâmetros são selecionados de forma totalmente aleatória, e o controlador selecionado é submetido ao ambiente de simulação. O preditor, então, retorna o conjunto de estados a cada passo da simulação, que é utilizado para calcular a função custo e, então, atualizar a função de distribuição de probabilidade de acordo com o desempenho do controlador selecionado pelo agente. Importante destacar que a função custo é uniforme para todos os três autômatos, durante todo o processo de simulação o mesmo valor de função custo é utilizado para todo o agente. Após o ciclo exploratório, a função de distribuição de probabilidade de cada autômato já não é mais uniforme, indicando o início do aprendizado sobre o ambiente.

3.6.2 Fase de aprendizado

Na fase de aprendizado, utilizou-se a função *step* de cada autômato, onde os parâmetros são escolhidos com base na função de distribuição de probabilidade atualizada durante a fase exploratória. Cada autômato, portanto, começa a convergir para um ou mais possibilidades para os parâmetros do controlador PID que diminuem a função custo. Se, ao final desse ciclo, um autômato convergir para mais de dois valores, isso pode indicar que o ciclo de aprendizado foi muito curto ou que existem inconsistências no ambiente de simulação. Este diagnóstico é importante para as mudanças subsequentes no processo de treinamento ou no próprio modelo de simulação.

Ao concluir o ciclo de aprendizado, os valores específicos para os quais cada autômato convergiu são apresentados, e, então, serão testados no ambiente real para validar o agente de ajuste e o ambiente de simulação. A correspondência entre os bons resultados obtidos tanto no ambiente simulado quanto no real é um forte indica-

tivo de que tanto o agente de ajuste quanto o ambiente de simulação estão adequadamente configurados e funcionando de maneira eficiente. Essa análise é fundamental para garantir a robustez e a confiabilidade do sistema antes de sua implementação prática.

Essa abordagem iterativa e baseada em simulação permite um ajuste fino dos parâmetros do controlador PID de forma segura e eficiente, minimizando a necessidade de intervenções diretas no sistema real e reduzindo o risco de danos ou ineficiências durante o processo de ajuste.

3.7 CONSIDERAÇÕES PARCIAIS

Nesta seção, é apresentado as considerações e decisões tomadas durante a implementação do ambiente de simulação e do agente de RL.

Primeiramente, para a construção do ambiente de simulação, selecionamos as variáveis de entrada do modelo: a penúltima tensão, a antepenúltima tensão e a saída do controlador. Essas variáveis foram escolhidas com base na correlação de Pearson, demonstrando sua importância para prever a tensão atual. Para o modelo preditivo, foram implementados o modelo de regressão linear, o modelo baseado no AutoGluon e o modelo baseado no PyTorch.

Na implementação do agente RL, utilizou-se o algoritmo CARLA por sua simplicidade, o que facilita a implementação em ambientes de baixo nível como o PLC. A função custo a ser minimizada foi definida como a soma do módulo do erro entre a tensão real e o *set point* em cada estado. Assim, o objetivo do agente é manter a tensão real o mais próximo possível do *set point*.

Para garantir a similaridade com o ambiente de controle real, foi implementada uma comunicação direta com uma instância do PLC. Esta instância replica exatamente a estrutura do controlador do sistema real, proporcionando um ambiente de simulação que reflete fielmente o sistema dinâmico.

Por fim, foram criadas duas fases de treinamento para o agente RL: a exploratória e a de aprendizado. A fase exploratória toma decisões aleatórias, sem considerar as experiências passadas, permitindo a exploração completa do ambiente. A fase de aprendizado, por sua vez, utiliza as experiências de cada passo anterior para escolher ações que têm maior probabilidade de minimizar a função custo, aprimorando continuamente o desempenho do agente.

4 RESULTADOS

Neste capítulo serão mostrados a execução dos modelos de preditor usados no ambiente de simulação, do agente de busca excessiva que visa testar todas possibilidades de controladores no ambiente de simulação e os controladores definidos pelo agente CARLA após o treinamento, o qual foi descrito em detalhes no capítulo anterior. Assim como, os resultados desses controladores no ambiente real e comparações com outros controladores definidos previamente.

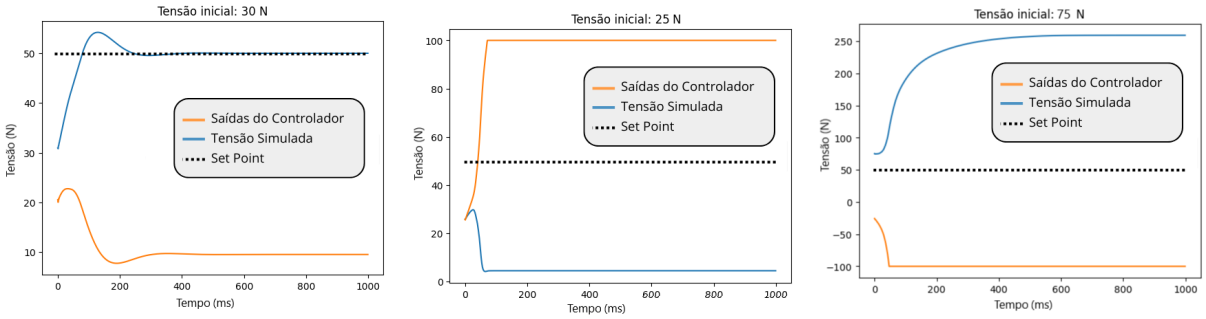
4.1 MODELOS DE PREDITOR

O modelo baseado no AutoGluon, após a conclusão do treinamento, obteve um desempenho satisfatório na predição dos dados presentes no banco de dados de teste, evidenciado pelo coeficiente de determinação R^2 igual a 0.99992, indicando uma boa capacidade de prever o estado do sistema. No entanto, durante a fase de validação, observou-se que as predições geradas pelo modelo obtido pelo AutoGluon exigiam um tempo considerável para serem calculadas, cerca de 0.945 segundos por predição. Essa demora na realização das predições tornaria inviável o uso desse modelo no ambiente de simulação, onde é necessário processar um grande volume de previsões em tempo hábil para garantir uma simulação eficiente e em tempo real.

O modelo treinado com PyTorch, após o treinamento, apresentou um desempenho satisfatório, obtendo um alto coeficiente de determinação R^2, 0.9994 , e um tempo de resposta adequado durante a fase de validação, cerca de 0.0002 segundos. No entanto, ao ser testado no ambiente de simulação, com um controlador que era estável no ambiente real com saídas limitadas em 100 e -100, assim como no sistema real, e um *set point* de 50 N, surgiram instabilidades imprevisíveis, como mostra a Figura 12.

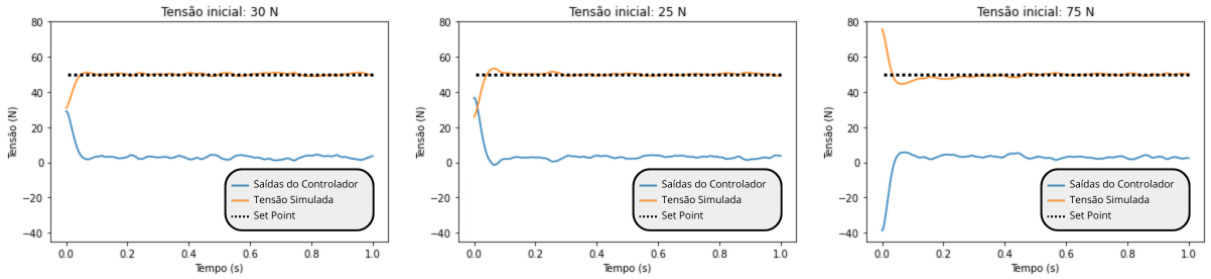
Pode-se notar que, a simulação mostrada na Figura 12a tem um bom desem-

Figura 12 – Simulações usando RNA treinada com biblioteca PyTorch



(a) Tensão inicial 30 N (b) Tensão inicial 25 N (c) Tensão inicial 75 N

Figura 13 – Simulações usando modelo de regressão linear



(a) Tensão inicial 30 N (b) Tensão inicial 25 N (c) Tensão inicial 75 N

penho e parece englobar as dinâmicas do sistema real, mas então nas simulações seguintes, uma instabilidade leva o preditor a um estado desconhecido em relação aos estados presentes no banco de dados utilizado no treinamento e, então, todo o sistema se torna incoerente, mesmo que o controlador tente diminuir a tensão atual, como mostrado na Figura 12c, ou aumenta-lá, como mostrado na Figura 12b.

As previsões incoerentes geradas pela RNA acabam por comprometer a simulação, levando-a a divergir do ambiente real em determinadas situações, e essa instabilidade foi produzida alterando a tensão inicial da simulação em apenas 5 N, mantendo ainda em um valor inicial dentro dos dados coletados. Essa falta de robustez e incapacidade de generalização do modelo de RNA, o tornam inutilizável.

O modelo de regressão linear, após o treinamento, alcançou valores de coeficiente de determinação R^2 tão satisfatórios quanto os obtidos pelos modelos anteriores, cerca de 0.9992. Para tornar esse modelo mais realista em relação ao sistema dinâmico, o qual tem pertubações externas de alta frequência que não são englobadas nesse modelo, e evitar que controladores com ação derivativa alta tenham menor função de custo devido a essa falta desses ruídos na simulação, foi introduzido um ruído virtual de até 0.5%, simulando as variações e interferências presentes na operação da máquina que não são capturadas pela regressão linear tradicional. Essa abordagem visa melhorar a capacidade do modelo de regressão linear em lidar com as nuances e imprecisões do ambiente de simulação, contribuindo para uma maior robustez e eficácia na realização de previsões.

Além disso, apresentou a vantagem de ser mais previsível e estável. Durante a execução dos mesmos testes realizados no modelo treinado com PyTorch, o modelo de regressão linear obteve resultados promissores, como mostra a Figura 13, sendo capaz de realizar previsões dentro do esperado e com tempos de processamento adequados.

Pode-se notar pelas Figura 13a, Figura 13b e Figura 13c que esse modelo tem um bom desempenho na simulação e parece representar de forma simplificada as dinâmicas do sistema real. Em adição, os testes realizados não demonstraram as instabilidades presentes no modelo treinado com PyTorch.

A escolha do modelo de regressão linear como preditor final é justificada pela sua combinação de precisão, baixo tempo de cálculo, previsibilidade e estabilidade. Apesar de outros modelos terem apresentado coeficientes de determinação ligeiramente superiores, o tempo de resposta inadequado do modelo baseado no AutoGluon e a instabilidade do modelo baseado no PyTorch tornaram-os inviáveis para o uso no ambiente de simulação. O modelo de regressão linear não apenas alcançou um desempenho comparável em termos de precisão, mas também foi capaz de lidar com as variações do ambiente de simulação de forma consistente, garantindo previsões confiáveis e eficientes.

4.2 AGENTE DE BUSCA EXCESSIVA

Para o teste do agente de busca excessiva, foi selecionada valores para os parâmetros PID com um baixa granularidade, sendo uma faixa de 0.0001 até 1.5 para K_p, com 50 amostras, uma faixa de 20 até 80 para T_n, com 61 amostras, e uma faixa de 0 até 60 para T_v, com 61 amostras. Portanto, todas as combinações possíveis totalizaram 152500, resultando em um número de ciclos equivalente no ambiente de simulação.

Caso esse algoritmo seja executado, cada ciclo tomaria cerca de 20 segundos para ser concluído. Isso significa que o tempo total necessário para a execução completa desse algoritmo seria de aproximadamente 52 horas. Este tempo elevado tornou o algoritmo inadequado para uso em ambientes industriais.

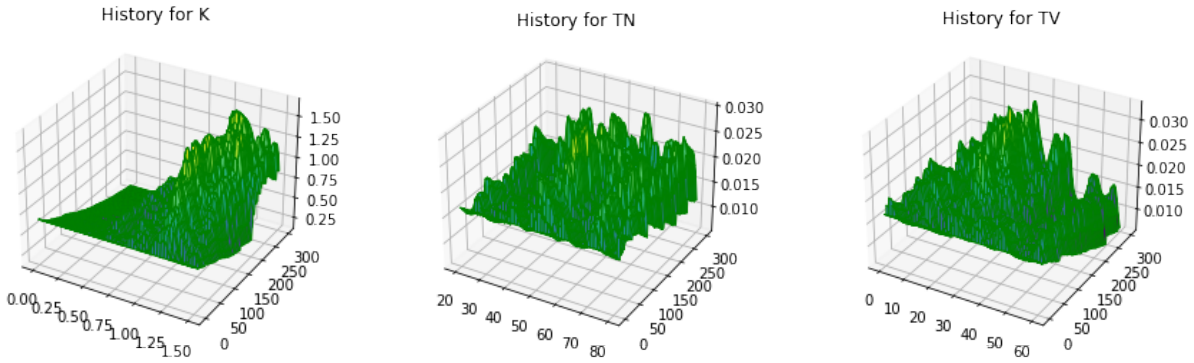
Dessa forma, esse agente foi descartado devido à sua inutilidade prática no contexto industrial, demonstrando a necessidade de abordagens mais eficientes e menos custosas em termos de tempo de processamento.

4.3 APRENDIZADO DO AGENTE CARLA NO AMBIENTE SIMULADO

Como descrito na Seção 3.6, o primeiro ciclo do ajuste é o de exploração. Foi estabelecido que este ciclo teria 300 épocas, um número suficiente para que o agente conheça todo o ambiente no qual vai convergir. E no final desse ciclo, após aproximadamente 1.67 horas de processamento, a função de distribuição de probabilidade de cada autômato é mostrada na Figura 14.

Onde o eixo x representa a faixa de ações que cada autômato pode tomar, o eixo z representa as épocas e o eixo y representa a função de distribuição de probabilidade. Então, com base na Figura 14, pode-se notar que o eixo y está uniformemente distribuído para época igual a 1, porém, ao decorrer das épocas (eixo z), começam a se formar picos nos locais onde o controlador teve um desempenho melhor que a média de desempenhos até o momento. Logo, ao observar que na última época o eixo

Figura 14 – Desenvolvimento da função de distribuição de probabilidade na fase exploratória

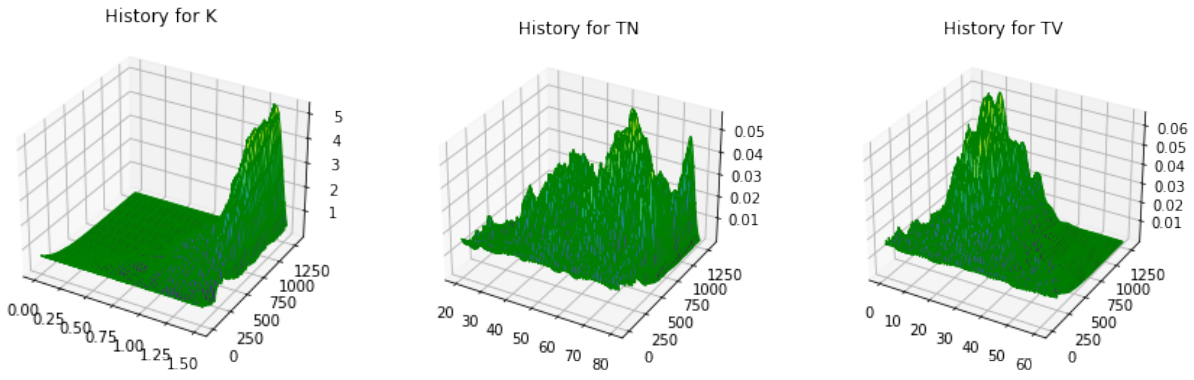


Fonte: Autoria própria (2024).

y é formado por diferentes picos, pode-se concluir que o agente conseguiu conhecer toda a faixa do sistema na fase exploratória.

Após isso, começa a fase de aprendizado, para esse ciclo foram definidas inicialmente 1000 épocas. Então, partindo dos conhecimentos adquiridos na fase exploratória, cada autômato utiliza sua função de distribuição de probabilidade para tomar um ação e, após o resultado do sistema, toma o conjunto de estados da simulação para o aprendizado. A Figura 15 mostram a função de distribuição de probabilidade de cada autômato ao final das 1000 épocas, após aproximadamente 5.56 horas de processamento:

Figura 15 – Desenvolvimento da função de distribuição de probabilidade na fase de aprendizado

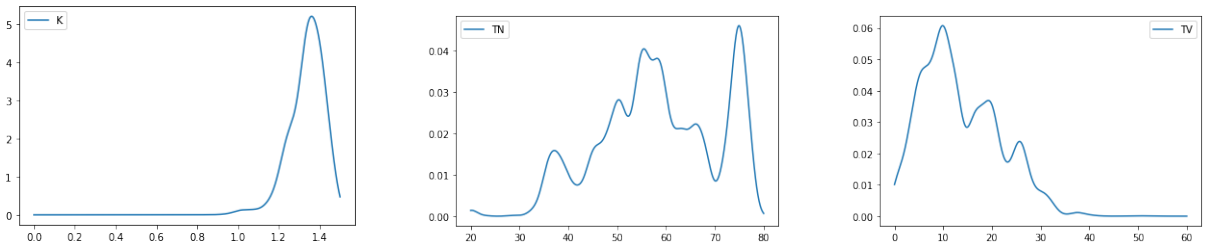


Fonte: Autoria própria (2024).

Os eixos x,z e y tem o mesmo significado dos explicados na Figura 14. Com base nessa figura, podemos ver que na época igual a 1 as funções de probabilidade tem o mesmo formato da última época da fase exploratória de cada autômato, mostrada na Figura 14, e ao decorrer das épocas os valores que tem um bom desempenho tem sua probabilidade aumentada e, logo, tem maior chance de serem selecionados de novo, isso faz com que seja formado um pico cada vez maior sobre esse bons

valores e, diferente da fase exploratória, os autômatos tendem a convergir para valores específicos e muito próximos um do outros. A Figura 16 mostram a função de distribuição de probabilidade na última época de cada autômato.

Figura 16 – Função de distribuição de probabilidade ao final da fase de aprendizado

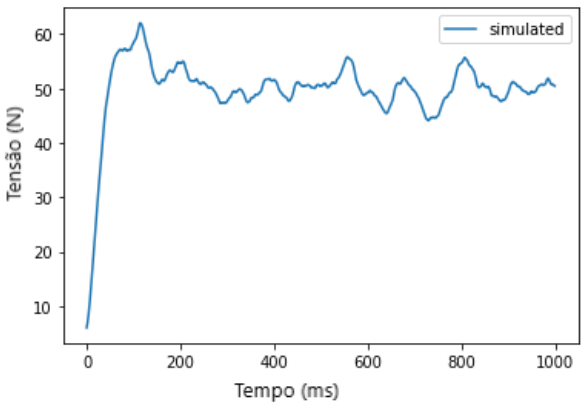


Fonte: Autoria própria (2024).

Onde o eixo x representa a faixa de ações que cada autômato pode tomar e o eixo y representa a função de distribuição de probabilidade. Portanto, pode-se notar que o parâmetro K_p convergiu para apenas um valor, enquanto os parâmetros T_n e T_v convergiram para mais de um valor, e realizando uma análise da Figura 16 pode-se supor que o agente ainda estava aprendendo e, caso tivesse mais épocas, iria convergir, porém como o ambiente simulado não representa o ambiente real com total fidelidade, aumentar o número de épocas pode causar um *overfitting* ao ambiente simulado. Então para os testes da próxima seção foram escolhidos os 2 maiores valores dos autômatos conectadas aos parâmetros T_n e T_v.

A Figura 17 mostra o resultado do controlador formado pelos parâmetros com maior valor em cada autômato no ambiente simulado.

Figura 17 – Desempenho do controlador convergido no ambiente simulado



Fonte: Autoria própria (2024).

Com base nesse resultado, é possível notar que, em ambiente simulado, o agente foi capaz de convergir para valores que diminuíram o função custo no ambiente simulado, ou seja, mantiveram o valor da tensão próximo ao *set point* durante toda a simulação.

4.4 VALIDAÇÃO NO AMBIENTE REAL

Para os testes realizados nesta seção foi utilizado a máquina real, contendo um substrato plástico e nenhum processo foi realizado sobre o substrato. Cada controlador foi testado utilizando o mesmos critérios, a tensão foi inicialmente configurada para 5 N, que representa 10% do *set point*, utilizando um controlador conhecido que, apesar de não ter o desempenho desejado é estável e tem um baixo erro em estado estacionário, e, então o controlador a ser testado foi carregado no PLC e o *set point* foi configurado para 50 N e foi iniciado a coleta dos dados, para a posterior análise do desempenho, por 5 segundos, após este tempo a coleta parou e esse processo foi reiniciado para que o próximo controlador possa ser testado.

Foram selecionados 4 controladores providos do agente CARLA, todos tendo o mesmo valor de K_p, já que este convergiu para um valor único, e os parâmetros T_n e T_v dos controladores são todas as possíveis combinações dado que havia duas opções para cada um desses parâmetros. Em adição, foram testados 4 controladores previamente selecionados de forma iterativa pelo operador da máquina, buscando alcançar estabilidade nos controladores, totalizando 8 controladores testados. A Tabela 4 representa os controladores testados e o valor do custo em cada um dos testes:

Tabela 4 – Desempenho dos controladores testados no ambiente real

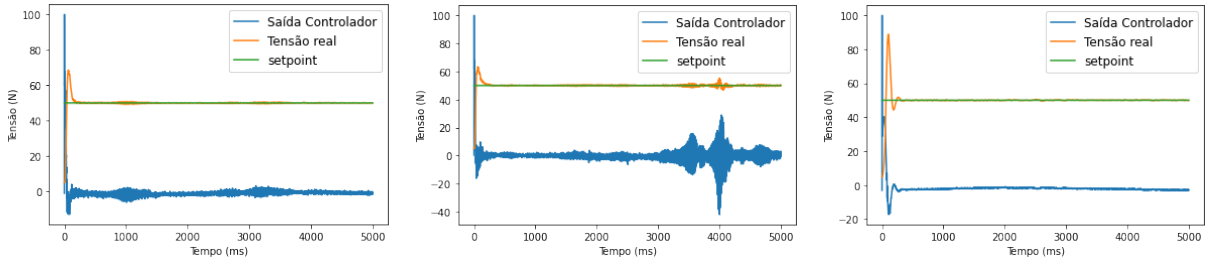
Classificação	**Origem**	K_p	T_n	T_v	**Função custo**	**Diferença percentual**
1°	Agente CARLA	1.36	57	10	3136	0%
2°	Agente CARLA	1.36	75	10	3254	3.7%
3°	Operador	0.7	50	5	5179	65.1%
4°	Operador	1	40	0	6976	122.4%
5°	Operador	0.5	35	10	7862	150.7%
6°	Agente CARLA	1.36	57	20	12716	305.5%
7°	Agente CARLA	1.36	75	20	15616	398.0%
8°	Operador	1.5	20	30	37071	1082.11%

Fonte: Autoria própria (2024).

Pelos valores de custo exibidos na Tabela 4, podemos notar que o controlador que obteve uma menor função custo, ou seja, o melhor controlador para a heurística definida foi um controlador definido pelo agente CARLA, com parâmetros igual a 1.36, 57 e 10, para K_p,T_n e T_v, respectivamente, e o segundo melhor controlador é outro controlador definido pelo agente, com parâmetros igual a 1.36, 75 e 10, para K_p,T_n e T_v, respectivamente. Em seguida, com um diferença percentual no valor da função custo de 65.1% em relação ao melhor controlador, é um controlador definido pelo operador, com parâmetros igual a 0.7, 40 e 5, para K_p,T_n e T_v, respectivamente. Além disso, pode-se notar pelas diferenças percentuais entre cada controlador e o melhor controlador que a partir do terceiro controlador esse percentual já representa

uma diferença notável no desempenho dos controladores. A Figura 18 mostra em detalhes como foi o desempenho dos 3 melhores controladores no ambiente real, e no Apêndice A está o desempenho dos outros 5 controladores.

Figura 18 – Desempenho dos três melhores controladores no ambiente real



(a) Melhor controlador (b) 2° melhor controlador (c) 3° melhor controlador

Na Figura 18a nota-se que o melhor controlador manteve a tensão muito próxima do *set point* com um pico de sobressinal perto de 40%, e também nota-se que a tensão varia com uma grande frequência, mas baixa amplitude em torno do *set point*, essa variação pode também ser percebida com a variação de alta frequência da saída do controlador. Essas grandes variações são consequência dos altos valores dos parâmetros PID, principalmente do parâmetro derivativo.

Ademais, na Figura 18b podemos ver que o 2° melhor controlador tem um desempenho muito parecido com o anterior, porém tem um menor pico de sobressinal, cerca de 25%. Entretanto, mesmo com um menor pico de sobressinal a alta variação da tensão com uma maior amplitude fez com que esse controlador obtivesse uma maior função custo que o anterior.

Já, na Figura 18c podemos ver que o 3° melhor controlador tem um desempenho diferente dos anteriores, com um pico de sobressinal com cerca de 90% e um maior tempo de acomodação. Porém, ao se acomodar esse controlador tem pouca variação em torno do *set point* e tem saídas de controlador bem baixas em comparação com os casos anteriores. O maior pico de sobressinal e o maior tempo de acomodação faz com que esse controlador tenha um maior valor na função de custo do que os anteriores, mesmo com uma menor variação da tensão após a acomodação.

4.5 LIMITAÇÕES E TRABALHOS FUTUROS

Essa seção visa lidar com as limitações presentes na aplicação atualmente e nas futuras alterações para a melhoria da mesma. Embora a implementação atual tenha sido capaz de gerar um controlador que tem menor custo que os demais, ela ainda não está adequada para o uso em ambiente industrial, como será discutidos nos próximos parágrafos.

A limitação que mais impacta os objetivos de médio prazo deste projeto é a escolha da função de custo. A soma do módulo do erro entre a tensão real e o *set point* de cada estado, função escolhida nesse trabalho, é simples e tem o objetivo claro de manter a tensão o mais próximo possível do *set point*, porém não avalia outros pontos importantes para o uso do controlador no ambiente industrial. Fatores como o pico de sobressinal, tempo de acomodação, variação em torno do *set point* e módulo da saída do controlador são importantes para a manufatura R2R e também devem ser consideradas na escolha do controlador. Então, apesar do controlador, o qual o desempenho está na Figura 18a, ser o melhor avaliado pela função de custo ser proveniente do agente CARLA, o mais adequado para uso industrial é o 3° melhor controlador, o qual o desempenho é mostrado na Figura 18c. O motivo disso é que apesar do alto tempo de acomodação e um maior pico de sobressinal, após a acomodação a membrana mantém a tensão com poucas vibrações e muito próxima do *set point*, o que impacta positivamente na qualidade das superfícies funcionais produzidas no processo R2R.

Outra limitação da implementação desse trabalho é o tempo que o agente toma para ajustar o controlador, que tem gargalos causados pela comunicação com o sistema de controle do PLC, apesar de ainda ser mais rápido que os métodos analíticos de definição de um controlador. A criação de uma instância de um PLC que tem a mesma implementação do controlador da máquina real permite que o ambiente de simulação utilize a mesma implementação do sistema real, que é interna ao PLC e não pode ser acessada. Porém a comunicação entre o ambiente de treinamento e a instância do PLC, dois processos distintos, cria um gargalo no treinamento e faz com que o sistema demore mais tempo.

Apesar dos resultados positivos com o agente CARLA minimizando a função custo, a busca por um refino de suas capacidades é fundamental. Então, a próxima etapa para a melhoria desse trabalho seria a implementação desse agente no sistema ambiente real após a o treinamento no ambiente simulado, e, devido a simplicidade do CARLA comparado a algoritmos como A3C, essa etapa é factível, visto que após a fase de treinamento no ambiente de simulação, o agente já está mais apto a tomar ações que não vão causar danos a máquina, como mostra os desempenhos na Tabela 4. Então, com menos ciclos que a fase de aprendizado no ambiente de simulação para evitar custos de manter o sistema real funcionando, o agente pode refinar o controlador diretamente no sistema, considerando todas as dinâmicas que foram simplificas na criação do ambiente simulado. Dessa forma o resultado do final do agente será mais adequado ao sistema.

5 CONCLUSÃO

Este trabalho tratou-se do controle da variáveis no ambiente industrial ao abordar a definição de controladores com a implementação um agente de RL, especificamente o algoritmo CARLA, que fornece um controlador PID ajustado para o sistema dinâmico. Para cumprir esse objetivo foi primeiro feita a análise e a implementação de um ambiente simulado, que visava possibilitar que o agente interaja com o sistema sem ter os riscos e custos de utilizar a máquina real. A principal função desse ambiente simulado foi que possibilitasse a criação de um controlador que pudesse ser testado e ajustado nele e também fosse eficiente no ambiente real.

Inicialmente, a coleta de dados foi realizada diretamente em uma máquina R2R. Então foi feito o tratamento e análise das variáveis encontradas em relação a tensão sobre a membrana e foram selecionados os dados que tinham maior correlação com a tensão, atendendo ao objetivo específico "Analisar dados da tensão durante a manufatura".

Esses dados foram utilizados para a criação de um modelo de regressão linear que previa a tensão atual a partir da penúltima e antepenúltima tensão, bem como a saída do controlador. Apesar de serem considerados modelos mais complexos para essa função, como redes neurais artificiais treinadas com o AutoGluon e PyTorch, optou-se pelo modelo de regressão linear devido à sua simplicidade, previsibilidade, eficiência computacional e um bom valor de coeficiente de determinação R^2 de 0.9992, atendendo ao objetivo específico "Simular a tensão sobre a membrana em um processo R2R".

No ambiente de simulação foi implementado uma comunicação com uma instância do PLC que contia o controlador PID, assegurando que o controlador utilizado no ambiente simulado fosse igual ao utilizado no ambiente real. Essa abordagem garantiu que o controlador ajustado pelo agente CARLA tivesse mesmo comportamento em ambos ambientes. Porém, essa comunicação também gerou um gargalo na fase de treinamento, já que a comunicação entre o processo do PLC e do treinamento consome tempo e acontecia muitas vezes a cada época.

Para o ajuste do controlador, foi utilizado o algoritmo CARLA, escolhido por sua simplicidade e eficiência na implementação em ambientes de baixo nível, como o PLC. A função custo definida foi a soma do módulo do erro entre a tensão real e o *set point*, simplificando o objetivo do agente para manter a tensão o mais próxima possível do valor desejado.

O agente foi implementado com duas funções principais: a função *step*, responsável por escolher ações baseadas na distribuição de probabilidade, e a função *explore*, que toma ações de forma uniforme para explorar o ambiente. Adicionalmente,

a função *update* foi utilizada para ajustar a distribuição de probabilidade com base nos resultados das ações tomadas, atendendo ao objetivo específico "Desenvolver um agente de RL para gerar os parâmetros de um controlador PID".

Após executar o agente no ambiente de simulação foram gerados 4 controladores que tinham um bom desempenho e, então, junto a 4 controladores definidos pelo operador foram testados em uma máquina R2R, contendo um substrato plástico e nenhum processo sendo realizado sobre o substrato. Dentre esses 4 controladores definidos pelo agente, o melhor deles obteve um valor de custo 65.1% menor em relação ao melhor controlador definido pelo operador no processo de tentativa e erro. Dessa forma o objetivo específico "Avaliar o desempenho do controlador durante o processo de manufatura" foi atendido de forma parcial, pois a máquina R2R em que o controlador foi testado não estava manufaturando uma superfície funcional.

Apesar dos bons resultados obtidos, algumas limitações foram identificadas. A função custo simples utilizada não considera outros aspectos importantes do controle industrial, como pico de sobressinal, tempo de acomodação e variações em torno do *set point*. Esses fatores são importantes para a qualidade do produto final e a estabilidade do processo de manufatura.

Logo, para aprimorar o sistema, a implementação de uma função custo mais abrangente que considere múltiplos critérios de desempenho é importante, pois isso permitirá que o controlador se torne mais eficiente para o processo de manufatura R2R e adequado para uso industrial.

Outro trabalho futuro promissor é a implementação direta do agente CARLA no sistema real. Dessa forma, o agente seria primeiro submetido a fase exploratória e de aprendizado no ambiente simulado, como é feito nesse trabalho, e resultaria em uma função de distribuição de probabilidade em que bons controladores no ambiente simulado são priorizados. Então, após esse período inicial de treinamento, o agente seria submetido a uma nova fase de aprendizado no ambiente real. Na qual o agente seria capaz de refinar esses controladores estáveis definidos na etapa anterior utilizando todas as dinâmicas que foram simplificadas na simulação, resultando em um controlador ainda mais afinado e eficiente.

O desenvolvimento deste trabalho mostra a viabilidade e as vantagens de utilizar algoritmos de RL para o ajuste de controladores em sistemas dinâmicos. Logo, pode-se concluir que o algoritmo CARLA em conjunto com o ambiente de simulação baseado em um modelo de regressão linear, que visa simular a tensão sobre a membrana de uma máquina R2R, conseguiu ajustar os parâmetros do controlador PID para diminuir o valor da função custo definida, conseguindo um melhor desempenho nos testes no sistema R2R real que os controladores PID escolhidos de forma empírica.

REFERÊNCIAS

ÅSTRÖM, K. J. Control system design lecture notes for me 155a. Department of Mechanical and Environmental Engineering University of California Santa Barbara, v. 333, 2002.

AUTOGLUON. **Predicting Columns in a Table - In Depth**. 2022. Disponível em: https://auto.gluon.ai/0.4.0/tutorials/tabular_prediction/tabular-indepth.html#sec-tabularadvanced. Acesso em: 24 mai. 2024.

BAKAR, N. M. A.; TAHIR, I. M. Applying multiple linear regression and neural network to predict bank performance. International Business Research, v. 2, n. 4, p. 176–183, 2009.

BANSAL, H. O.; SHARMA, R.; SHREERAMAN, P. Pid controller tuning techniques: a review. Journal of control engineering and technology, Arastirmax Scientific Publication Index, v. 2, n. 4, 2012.

BENNETT, S. **A history of control engineering, 1930-1955**. [S.l.]: IET, 1993.

BERTO, M. I.; SÁ, F. R. d.; SILVEIRA JR, V. Avaliação de controles pid adaptativos para um sistema de aquecimento resistivo de água. Food Science and Technology, SciELO Brasil, v. 24, p. 478–485, 2004.

BORJAS, M.; DEMETRIO, S. **Estudo da identificação por subespaços em malha aberta e fechada e proposta de novos algoritmos.** 2009. Tese (Doutorado) — Universidade de São Paulo, 2009.

BROWNLEE, J. What is the difference between a batch and an epoch in a neural network. Machine learning mastery, v. 20, 2018.

BURNS, R. **Advanced control engineering**. [S.l.]: Elsevier, 2001.

CARVALHO, A. *et al.* Inteligência artificial–uma abordagem de aprendizado de máquina. Rio de Janeiro: LTC, v. 2, p. 45, 2011.

DEHUI, W. *et al.* Optimization of taper winding tension in roll-to-roll web systems. Textile Research Journal, v. 84, n. 20, p. 2175–2183, 2014. Disponível em: https://doi.org/10.1177/0040517514538697.

DING, Z. *et al.* Introduction to reinforcement learning. Deep reinforcement learning: fundamentals, research and applications, Springer, p. 47–123, 2020.

DOGRU, O. *et al.* Reinforcement learning approach to autonomous pid tuning. Computers Chemical Engineering, v. 161, p. 107760, 2022. ISSN 0098-1354. Disponível em: https://www.sciencedirect.com/science/article/pii/S0098135422001016.

ERTEL, W. **Introduction to artificial intelligence**. [S.l.]: Springer, 2018.

EUZÉBIO, T. A. M. *et al.* Sintonia ótima de controlador pid descentralizado para processos mimo. Universidade Federal de Campina Grande, 2015.

FACCIN, F. Abordagem inovadora no projeto de controladores pid. 2004.

FRAUNHOFER-INSTITUT FüR PRODUKTIONSTECHNOLOGIE. **Roll-to-roll module**. 2023. Disponível em: https://www.ipt.fraunhofer.de/de/angebot/sondermaschinen/rolle-zu-rolle/jcr:content/contentPar/gallery2.vimg.3col.jpg/ipt/de/images/Kompetenzen/Bildergalerien/200_modulbaukasten_r2r/grossbilder/2r2-lamination.jpg. Acesso em: 13 nov. 2023.

GARCIA, C. **Controle de processos industriais: estratégias convencionais**. [S.l.]: Editora Blucher, 2021. v. 1.

GOMES, S. C. P. Lugar geométrico das raízes incremental e sua aplicação na sintonia de controlares pid. 2009.

GOODMAN, C. S.; BATE, M.; SPITZER, N. C. Embryonic development of identified neurons: origin and transformation of the h cell. Journal of Neuroscience, Soc Neuroscience, v. 1, n. 1, p. 94–102, 1981.

GOUDA, M.; DANAHER, S.; UNDERWOOD, C. Fuzzy logic control versus conventional pid control for controlling indoor temperature of a building space. IFAC Proceedings Volumes, v. 33, n. 24, p. 249–254, 2000. ISSN 1474-6670. 8th IFAC Symposium on Computer Aided Control Systems Design (CACSD 2000), Salford, UK, 11-13 September 2000. Disponível em: https://www.sciencedirect.com/science/article/pii/S1474667017369008.

GREENDER, J.; PEARSON, G.; CAKMAK, M. **Roll-to-Roll Manufacturing**. [S.l.: s.n.], 2018. https://books.google.de/books?hl=pt-BR&lr=&id=qD9NDwAAQBAJ&oi=fnd&pg=PP13&dq=roll-to-roll+importance&ots=ob1XRkG3fb&sig=1qV-Ab1j5BY7QZYgIxDA5B5yj3A&redir_esc=y#v=onepage&q&f=false.

HOWELL, M.; GORDON, T. Continuous learning automata and adaptive digital filter lksign, control 98, swansea, 1-4 september 1998. Exeter, England, 1998.

HOWELL, M. N.; BEST, M. C. On-line pid tuning for engine idle-speed control using continuous action reinforcement learning automata. Control Engineering Practice, Elsevier, v. 8, n. 2, p. 147–154, 2000.

HOWELL, M. N. *et al.* Continuous action reinforcement learning applied to vehicle suspension control. Mechatronics, Elsevier, v. 7, n. 3, p. 263–276, 1997.

HUANG, Z. *et al.* A survey on ai-driven digital twins in industry 4.0: Smart manufacturing and advanced robotics. Sensors, MDPI, v. 21, n. 19, p. 6340, 2021.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **Conceitos básicos de controle: malha aberta X malha fechada**. 2020. https://edu.ieee.org/br-ufcgras/conceitos-basicos-de-controle-malha-aberta-x-malha-fechada/#:~:text=Ainda%20assim%2C%20a%20malha%20aberta,tornar%20o%20sistema%20mais%20responsivo. Accessed on 2023-11-14.

FERNANDES JÚNIOR, F. G. **Metodologia para re-sintonia de controladores pid industriais**. 2006. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2006.

KNOSPE, C. Pid control. IEEE Control Systems Magazine, v. 26, n. 1, p. 30–31, 2006.

KOSEL, M. Flexpol-antimicrobial flexible polymers for its use in hospital environments-h2020. Impact, v. 2018, n. 10, p. 69–71, 2018. ISSN 2398-7073. Disponível em: https://www.ingentaconnect.com/content/sil/impact/2018/00002018/00000010/art00024.

KUO, B. **Sistemas de controle automatico**. Prentice-Hall do Brasil, 1985. ISBN 9788570540164. Disponível em: https://books.google.de/books?id=M6LlZwEACAAJ.

LAWRENCE, N. P. *et al.* Deep reinforcement learning with shallow controllers: An experimental application to pid tuning. Control Engineering Practice, v. 121, p. 105046, 2022. ISSN 0967-0661. Disponível em: https://www.sciencedirect.com/science/article/pii/S0967066121002963.

LEE, J.; BYEON, J.; LEE, C. Theories and control technologies for web handling in the roll-to-roll manufacturing process. International Journal of Precision Engineering and Manufacturing-Green Technology, v. 7, n. 2, p. 525–544, Mar 2020. ISSN 2198-0810. Disponível em: https://doi.org/10.1007/s40684-019-00185-3.

LEE, Y.-S.; JANG, D.-W. Optimization of neural network-based self-tuning pid controllers for second order mechanical systems. Applied Sciences, v. 11, n. 17, 2021. ISSN 2076-3417. Disponível em: https://www.mdpi.com/2076-3417/11/17/8002.

MARJANI, M. *et al.* Big iot data analytics: Architecture, opportunities, and open research challenges. IEEE Access, v. 5, p. 5247–5261, 2017.

MNIH, V. *et al.* Asynchronous methods for deep reinforcement learning. *In*: PMLR. **International conference on machine learning**. [S.l.], 2016. p. 1928–1937.

NGATCHED, T. M. N.; WOUNGANG, I. **Pan-African Artificial Intelligence and Smart Systems: First International Conference, PAAISS 2021, Windhoek, Namibia, September 6-8, 2021, Proceedings**. [S.l.]: Springer Nature, 2022. v. 405.

OGATA, K. Modern control engineering. Book Reviews, v. 35, n. 1181, p. 1184, 1999.

OSBORNE, J. W. Prediction in multiple regression. Practical Assessment, Research, and Evaluation, v. 7, n. 1, p. 2, 2019.

PATEL, V. V. Ziegler-nichols tuning method: Understanding the pid controller. Resonance, Springer, v. 25, n. 10, p. 1385–1397, 2020.

PERES, R. S. *et al.* Industrial artificial intelligence in industry 4.0 - systematic review, challenges and outlook. IEEE Access, v. 8, p. 220121–220139, 2020.

PERES, R. S. *et al.* Industrial artificial intelligence in industry 4.0 - systematic review, challenges and outlook. IEEE Access, v. 8, p. 220121–220139, 2020.

PIZARD, S.; VALLESPIR, D. Towards a controlled vocabulary on software engineering education. European Journal of Engineering Education, Taylor & Francis, v. 42, n. 6, p. 927–943, 2017.

RATH, S.; TRIPATHY, A.; TRIPATHY, A. R. Prediction of new active cases of coronavirus disease (covid-19) pandemic using multiple linear regression model. Diabetes & metabolic syndrome: clinical research & reviews, Elsevier, v. 14, n. 5, p. 1467–1474, 2020.

RAUBER, T. W. Redes neurais artificiais. Universidade Federal do Espírito Santo, v. 29, 2005.

REIMER, U. *et al.* Design and modeling of metallic bipolar plates for a fuel cell range extender. Energies, v. 14, n. 17, 2021. ISSN 1996-1073. Disponível em: https://www.mdpi.com/1996-1073/14/17/5484.

RIBEIRO, J. M. S. *et al.* Comparison of pid controller tuning methods: analytical/-classical techniques versus optimization algorithms. *In*: **2017 18th International Carpathian Control Conference (ICCC)**. [S.l.: s.n.], 2017. p. 533–538.

RICH, E. Artificial intelligence and the humanities. Computers and the Humanities, JSTOR, v. 19, n. 2, p. 117–122, 1985.

SACOMANO, J. B. *et al.* **Indústria 4.0**. [S.l.]: Editora Blucher, 2018.

SILVA, M. V. C. *et al.* **Uso de redes neurais artificiais e modelos de regressão para estimar volume de espécies nativas em portel, pará-Brasil. Kurú [online]. 2020, vol. 17, n. 40**. [S.l.]: ISSN.

SREEHARI, E.; SRIVASTAVA, S. Prediction of climate variable using multiple linear regression. *In*: IEEE. **2018 4th International Conference on Computing Communication and Automation (ICCCA)**. [S.l.], 2018. p. 1–4.

SUPERA. **Células nervosas**. Aachen, 2023. Disponível em: https://metodosupera.com.br/neuronios-glossario-do-cerebro/. Acesso em: 14 nov. 2023.

TAO, Y.; YIN, Y.; GE, L. **New PID control and application**. [S.l.]: China Machine Press, Beijing 2002, 1998.

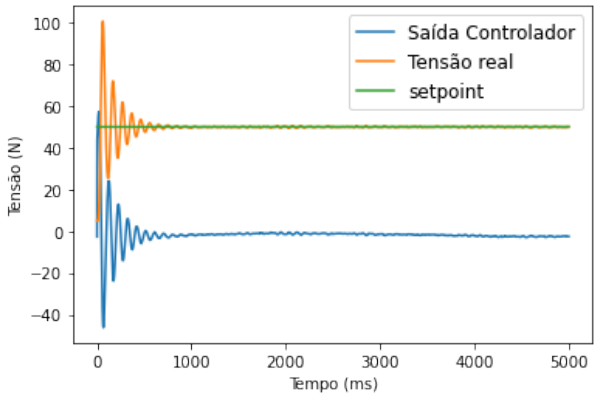
WEISBERG, S. **Applied linear regression**. [S.l.]: John Wiley & Sons, 2005. v. 528.

WYTHOFF, B. J. Backpropagation neural networks: a tutorial. Chemometrics and Intelligent Laboratory Systems, Elsevier, v. 18, n. 2, p. 115–155, 1993.

XU, T. *et al.* Rapid formation and selective stabilization of synapses for enduring motor memories. Nature, Nature Publishing Group UK London, v. 462, n. 7275, p. 915–919, 2009.

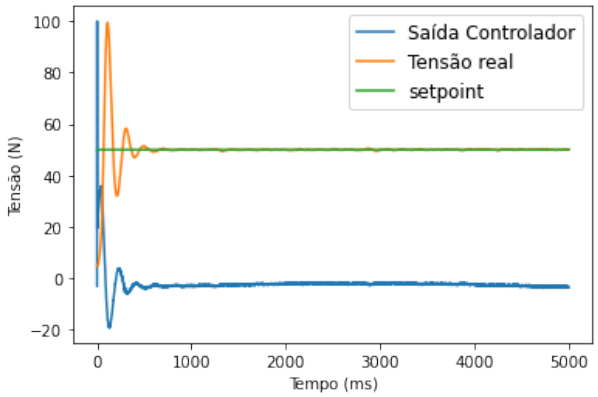
APÊNDICE A – DESEMPENHO DOS CONTROLADORES NO AMBIENTE REAL

Figura 19 – Desempenho do 4° melhor controlador no ambiente real



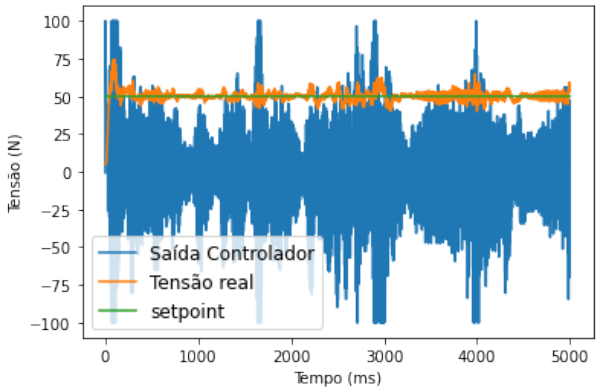
Fonte: Autoria própria (2024).

Figura 20 – Desempenho do 5° melhor controlador no ambiente real



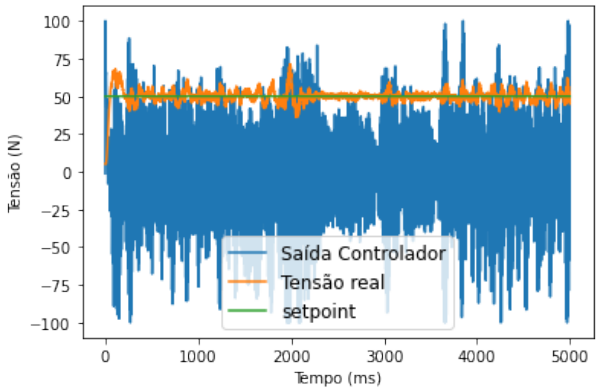
Fonte: Autoria própria (2024).

Figura 21 – Desempenho do 6° melhor controlador no ambiente real



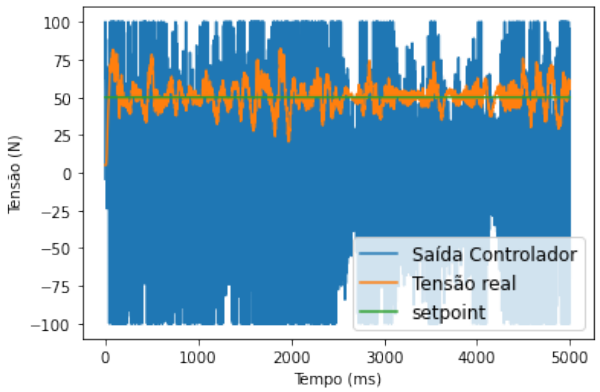
Fonte: Autoria própria (2024).

Figura 22 – Desempenho do 7° melhor controlador no ambiente real



Fonte: Autoria própria (2024).

Figura 23 – Desempenho do 8° melhor controlador no ambiente real



Fonte: Autoria própria (2024).