Improving the PID Controllers of Roll-to-Roll Processes using Reinforcement Learning

Luís Eduardo Fernandes Costa Lima[1], **Ricardo José Pfitscher**[1]

[1]Universidade Federal de Santa Catarina (UFSC)
Joinville – SC – Brasil

luisedufernandes2002@gmail.com,ricardo.pfitscher@ufsc.br

Abstract. *Approximately 90% of control loops in industrial systems utilize Proportional Integral Derivative (PID) controllers, which are essential for ensuring product quality in roll-to-roll (R2R) manufacturing processes. These processes, which involve the continuous handling of materials on rolls, require precise control, especially of substrate tension, as it is important to control because this variable is directly connected to the quality of the final product. Traditional tuning of PID parameters can be complex, as it requires the mathematical formulation of all the dynamics of the process, making it laborious to define using the traditional approach. However, advances in computational techniques have facilitated the development of automated tuning methods for PID controllers. This article investigates the application of reinforcement learning, an artificial intelligence technique, to optimize the tuning of PID controllers. The proposed methodology occurs in two stages: first, we create a simulation environment to prevent damage to real R2R machines; then, we use the CARLA reinforcement learning algorithm to adjust the PID parameters. The results indicate a 65.1% reduction in costs compared to traditional empirical tuning methods, demonstrating a significant improvement in process efficiency.*

1. Introdução

Os sistemas de controle são essenciais em diversas áreas da indústria, incluindo controle de qualidade, linhas de montagem, sistemas de transporte, eletrônica de potência, tecnologia espacial e robótica. Com os avanços tecnológicos desde os anos 1960, tem-se desenvolvido métodos de controle cada vez mais sofisticados, incluindo o controle preditivo e adaptativo. Dentre esses, o controlador PID se destaca, sendo usado em até 90% das malhas de controle industriais. Isto se deve à sua simplicidade, robustez e eficácia na resolução de problemas de controle [Knospe 2006].

Um dos processos que utiliza PID para controle é a produção de materiais ou dispositivos a partir da manufatura R2R, que se caracteriza por realizar deposição, montagem ou processamento de camadas de forma contínua em um substrato flexível, que normalmente é armazenado na forma de rolo. [Lee et al. 2020]. Este tipo de manufatura é atrativo para diversas áreas da indústria, pois apresenta bom desempenho em processos de grande volume que necessitam de alto rendimento. Além disso, representa uma solução rentável, uma vez que os processos de manufatura são, em geral, realizados por uma única máquina, o que reduz os custos com operadores [Greener 2018]. Como os processos R2R são essenciais para a manufatura de dispositivos e materiais flexíveis,

aprimorá-los gera um impacto direto na melhoria desses produtos, ampliando consideravelmente as possibilidades de fabricação [Lee et al. 2020].

O sistema R2R é composto por um conjunto de enroladores e unidades de fixação. No início do processo em cadeia, pelo menos um dos enroladores, marcado com a letra D na Figura 1 e chamado de desbobinador, contém a membrana que será processada durante a 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. No final do processo, há pelo menos um enrolador, marcado na Figura 1 com a letra R e chamado de rebobinador, que armazena a superfície funcional e é o responsável por controlar a tensão sobre a membrana após a unidade de fixação. 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 e está marcada na Figura 1 com a letra F [Lee et al. 2020].

Figura 1. Sistema R2R

O controle preciso da tensão na membrana durante a manufatura é crucial para garantir a eficiência e a qualidade do produto final [Dehui et al. 2014]. Porém, 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 de parâmetros de um controlador PID que satisfaça os requisitos pode se tornar uma tarefa difícil [Lee and 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].

Tendo em vista o potencial da inteligência artificial (IA) na indústria, este trabalho explora uma alternativa baseada em aprendizado de máquina (ML , do inglês *Machine Learning*) 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, este trabalho apresenta a construção de um controlador baseado em aprendizado por reforço (RL, do inglês *Reinforcement Learning*), e explora contornar a dificuldade de usar um equipamento físico para treinamento, dividindo o treinamento em duas fases, *offline* e *online*. Na primeira fase, é realizada uma simulação da máquina R2R através de métodos de regressão. A segunda fase, de validação, ocorre no momento em que o treinamento

está mais próximo de convergir, e são selecionados os melhores controladores segundo o agente para serem testados na máquina real. Os resultados mostram uma redução de 65,1% na função custo com parâmetros definidos pelo controlador de RL quando comparado à parâmetros definidos empiricamente.

O restante do texto está organizado da seguinte forma. A Seção 2 conceitua controladores PID e discute os trabalhos relacionados. Em seguida, a Seção 3 apresenta a proposta de sistema de ajuste de parâmetros do controlador PID. Os resultados de avaliação são apresentados na Seção 4, incluindo uma discussão das principais limitações. Por fim, as conclusões desse esforço de pesquisa são apresentados na Seção 5.

2. Fundamentação e Trabalhos relacionados

A presente seção discute aspectos fundamentais para entendimento do contexto da aplicação do método de aprendizado e apresenta os trabalhos relacionados.

2.1. 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

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

em que, e é o valor do erro; d é o valor desejado; y é o valor real do sistema obtido pela realimentação negativa do sistema em malhada fechada e t é o tempo.

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 erros do sistema e, então, toma ação de acordo com esse histórico, 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].

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

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

em que, s é a saída do controlador; e é a entrada do controlador, ou seja, o erro; K_p é o ganho proporcional que é um número não negativo; T_n é a constante de tempo integral que é um número maior que zero; T_v é a constante de tempo diferencial que é um número não negativo e t é o tempo. 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 $K_i = K_p/T_n$ e o ganho diferencial é dado por $K_d = K_p * T_v$.

2.2. Trabalhos relacionados

Howell e Best, em [Howell and Best 2000], propuseram uma abordagem para automatizar o ajuste de controladores PID usando o método CARLA. O estudo focou em otimizar os parâmetros PID para controlar a velocidade de um motor ocioso, ajustando o ângulo da embreagem para minimizar uma função custo. Utilizando os parâmetros iniciais de Ziegler-Nichols, foram criados três autômatos, um para cada parâmetro PID (K_p, T_n e T_v). O agente foi treinado em um modelo simulado do motor e, após o aprendizado, testado em um motor real. Os resultados mostraram que o controlador PID ajustado no ambiente simulado proporcionou ganhos significativos no controle e reduziu os custos, confirmando a eficiência do algoritmo CARLA.

[Dogru et al. 2022] desenvolveram um 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*. Quando o agente atingiu um desempenho satisfatório no passo *offline*, 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 de aprendizagem por reforço profundo (DRL, do inglês *deep reinforcement learning*), o qual consegue se adaptar a não-linearidades.

Em [Lawrence et al. 2022], os autores implementaram um algoritmo de Deep Reinforcement Learning (DRL) usando um controlador PID como uma política treinável em um ambiente dinâmico real, adotando uma abordagem *model-free*, ou seja, sem necessidade de modelar ou estimar as dinâmicas do sistema. O controlador PID foi integrado em um Controlador Lógico Programável (CLP) comum, eliminando a necessidade de *hardware* adicional. Inicialmente, os parâmetros do controlador foram ajustados em uma região segura para garantir proteção durante as fases iniciais de aprendizado. O treinamento ocorreu em tempo real, diretamente no sistema dinâmico, sem pré-treinamento em simulações ou uso de dados preexistentes. Após o término do treinamento, a estratégia de ajuste do PID e o desempenho do algoritmo de DRL foram avaliados de acordo com critérios específicos. A abordagem demonstrou que é viável aplicar DRL em um ambiente real para ajustar automaticamente controladores PID, resultando em um controlador bem ajustado, compreensível e de fácil uso para os profissionais.

A presente proposta agrega as contribuições isoladas da literatura. Assim como em [Howell and Best 2000] e [Dogru et al. 2022], e diferentemente de [Lawrence et al. 2022], a implementação segue uma abordagem de duas fases de aprendizado, *offline* e *online*, avançando nos modelos de simulação ao inserir modelos de regressão baseados em AutoML. A escolha por não fazer a implementação diretamente no CLP como em [Lawrence et al. 2022], se deu para evitar danos ao equipamento controlado na fase de aprendizado. Da mesma forma que em [Howell and Best 2000], o trabalho utilizou uma implementação baseada no método CARLA para a fase *online*, uma vez que o modelo é mais simples que abordagens mais modernas de RL e se mostrou eficiente para os ajustes do controlador. A principal diferença ocorre na fase *offline*, onde o modelo de simulação foi treinado com diferentes combinações de comportamento forçadas na máquina, com fins de generalização.

3. Ajuste de parâmetros PID com RL

A proposta deste artigo é que o sistema de ajuste seja generalizado para controladores PID em sistemas de manufatura de propósito geral. Então, apesar de a modelagem e ajuste de controlador ser conduzida no contexto de sistemas R2R, uma arquitetura genérica é proposta na Figura 2.

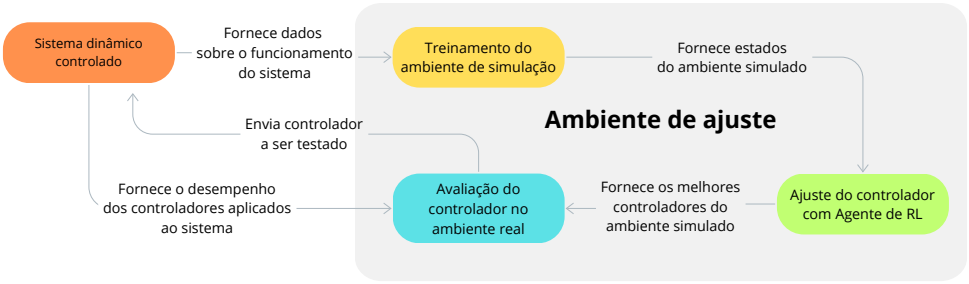


Figura 2. Arquitetura de aprendizagem para ajuste de parâmetros PID

A arquitetura é composta 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 treinamento do agente de RL. Nesse artigo, com fins de validação, o sistema dinâmico utilizado é uma máquina de manufatura R2R operada pela ferramenta TwinCat provida pela Beckhoff, que tem apenas um substrato plástico o qual não passa por nenhum processo físico ou químico.

3.1. Ambiente de simulação

O ambiente de simulação permite testar e iterar parâmetros do sistema controlado sem os custos financeiros do ambiente real, como energia, equipamentos e materiais. Além disso, ele acelera o treinamento do agente, possibilitando mais tentativas em menos tempo e garantindo a segurança dos equipamentos, evitando riscos de danos que poderiam ocorrer em interações diretas com dispositivos físicos. Para o treinamento do modelo, foram utilizadas como entradas a saída do controlador, o penúltimo valor de tensão e o antepenúltimo valor de tensão, a variável predita é a tensão atual resultante.

Para implementação do sistema de simulação foram avaliados três modelos como candidatos a preditor. O primeiro modelo foi implementado com a ferramenta AutoGluon [Aut 2022], que é uma framework de AutoML (Automated Machine Learning). Esta ferramenta automatiza o processo de seleção e treinamento dos melhores modelos de aprendizado de máquina para um conjunto de dados, buscando encontrar a configuração ideal dos algoritmos e hiper-parâmetros, ele também tem tratamento automatizado dos dados, identificando o seu tipo e lidando com ausência e balanceamento das entradas.

Para o segundo candidato foi implementado uma RNA utilizando a biblioteca PyTorch, do Python. Diferentemente da abordagem com AutoGluon, a implementação de uma RNA é mais complexa e demanda maior conhecimento em programação e redes neurais. 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. Esta configuração foi a melhor obtida após uma busca em grade, que avaliou dife-

rentes combinações com entre uma e três camadas ocultas, e tamanho variando entre 2 e 64 neurônios (em potências de base 2).

O terceiro candidato para o preditor do ambiente de simulação visa simplificar e tornar o modelo mais explicável, para tanto, 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 e foi introduzido um ruído virtual de até 0.5% à saída, simulando as variações e interferências presentes na operação da máquina que não são capturadas pela regressão linear.

3.2. Ambiente de aprendizagem

Uma vez implementado o ambiente de simulação, o próximo passo é executar o agente de RL. Esse componente tem o objetivo de ajustar o controlador para se reduzir a função custo. Para tanto, o agente utiliza o ambiente de simulação para avaliar a seleção dos parâmetros do controlador PID (K_p, T_n e T_v da Equação 2) e seus valores internos de acordo com o resultado. Após atingir um valor alvo ou o limite de interações, 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.

No contexto desse artigo, optou-se pela soma do módulo do erro entre a tensão real e o *set point* de cada estado como a função custo. 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.

Em termos de algoritmo de RL, foi implementado o *Continuous Action Reinforcement Learning Automata* (CARLA), introduzido por [Howell et al. 1997] e discutido em [Howell and Best 2000], para o desenvolvimento do agente responsável por convergir para um controlador que tenha um menor valor de custo. O código desenvolvido para esse agente está disponível na integra no GitHub[1]. Esse algoritmo 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.

O esquema do CARLA, representado na Figura 3, 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. Neste trabalho, foi criado um autômato para cada parâmetro do controlador PID.

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

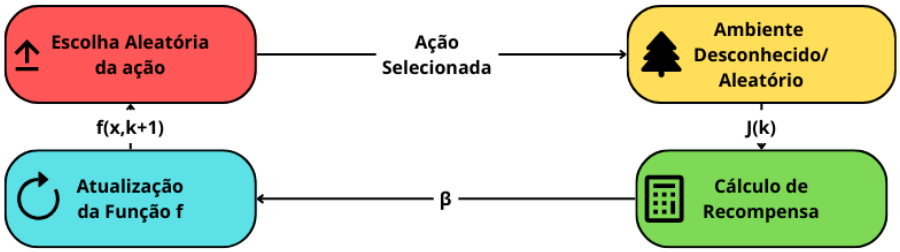


Figura 3. Esquema CARLA

À medida que o sistema continua aprendendo, a função de densidade probabilística converge gradualmente para uma distribuição em torno do valor desejado do 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 útil para lidar com problemas complexos em ambientes dinâmicos e incertos.

4. Resultados

Para facilitar o entendimento, a discussão dos resultados é feita em três partes. Primeiro, é discutida a precisão do modelo simulado. Depois, é avaliado o desempenho dos controladores com parâmetros aprendidos. Por fim, são discutidas as limitações dos achados científicos da pesquisa.

4.1. Precisão das simulações

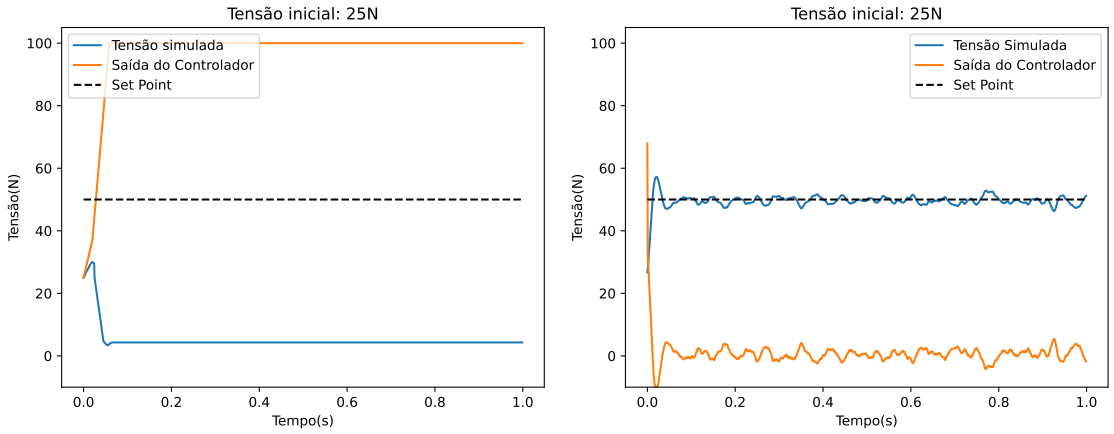
No ambiente de simulação, *offline*, o modelo de regressão é alimentado com as variáveis de entrada (saída do controlador, o penúltimo valor de tensão e o antepenúltimo valor de tensão) e deve estimar a tensão atual resultante. Neste cenário, define-se uma tensão inicial no sistema e o controlador deve encontrar um *set point* (objetivo). Assim, com base na estimativa de tensão o erro é calculado e a saída do controlador é calculada de acordo com a Equação 2, os novos valores são então reinseridos no modelo de forma iterativa: o antepenúltimo valor de tensão passa a ser o penúltimo valor de tensão e este passa ser a tensão estimada. Os resultados dessa simulação são apresentados na Figura 4.

O modelo baseado no AutoGluon obteve um desempenho satisfatório na predição dos dados presentes no banco de dados de teste, com um 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 simulação, observou-se que as predições geradas pelo modelo exigiam um tempo considerável para serem calculadas, cerca de 0,945 s por predição. Essa demora na realização das predições tornaria inviável o uso desse modelo no ambiente de produção, onde é necessário processar um grande volume de previsões em tempo hábil para que o controlador se ajuste. Portanto, não foi realizada a simulação do controlador com esse modelo.

O modelo treinado com RNA apresentou um desempenho também satisfatório quanto as estimativas, com $R^2 = 0{,}9994$, e um tempo de resposta adequado durante a

fase de validação, cerca de 0,0002 segundos por estimativa. No entanto, ao ser testado no ambiente de simulação o comportamento do controlador se tornou instável como mostra a Figura 4(a). No início da simulação o controlador (linha azul) aplica uma força sobre a tensão (amarela), o que deveria implicar em um crescimento da tensão no sistema. No início da simulação (até 0.05 s) isto até acontece, mas os valores gerados fazem com que o sistema se torne incoerente, levando a saídas do controlador para forças na ordem de 100 N e o preditor estima tensões baixas, distantes do *set point* de 50 N. Tornando o modelo ineficaz para representação da máquina real.

O modelo de regressão linear alcançou valores de coeficiente de determinação R^2 próximos aos obtidos pelos modelos anteriores, cerca de 0,9992. Além disso, apresentou a vantagem de ser mais previsível e estável. Durante a simulação, o modelo de regressão linear obteve resultados mais adequados para representar o equipamento real, como mostra a Figura 4(b), fazendo com que as tensões simuladas (preditas) estivessem de acordo com o comportamento da saída do controlador, com um ajuste próximo ao *set point* de 50 N.



(a) RNA (b) Regressão linear

Figura 4. Simulações utilizando modelos de regressão

Tendo em vista a precisão obtida, a estabilidade da simulação e o tempo de resposta, o preditor baseado em regressão linear se mostrou mais adequado para o ambiente de treinamento utilizado pelo agente de RL.

4.2. Ajuste dos parâmetros do controlador

O ajuste dos parâmetros do controlador através de RL é feito em dois ciclos, exploração (*offline*) e aprendizado (*offline* e *online*). No primeiro ciclo o agente escolhe as ações com uma probabilidade uniforme. Foi estabelecido que este ciclo teria 300 épocas, suficiente para que o agente conheça o ambiente no qual vai convergir. No final desse ciclo, após aproximadamente 1,67 horas de processamento, a função de distribuição de probabilidade de cada autômato é mostrada na Figura 5.

Em que, 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 5, 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

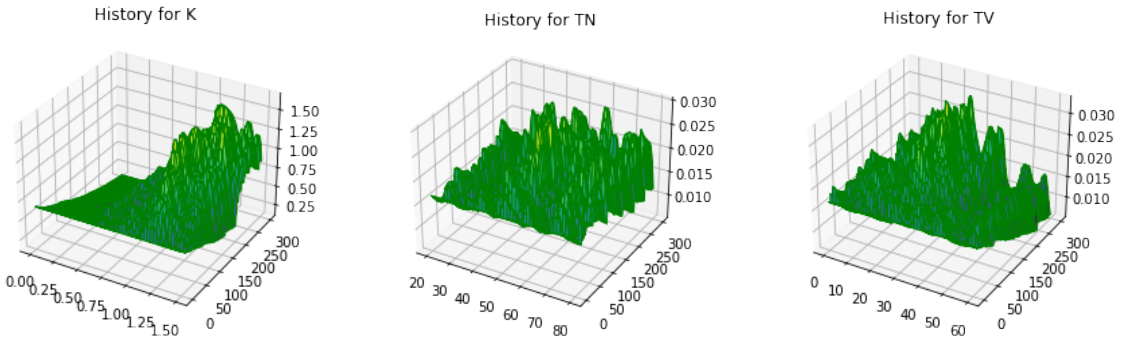


Figura 5. Desenvolvimento da função de distribuição de probabilidade na fase exploratória

nos locais onde o controlador teve um desempenho melhor que a média de desempenhos até o momento.

No segundo ciclo, de aprendizagem, o agente usa experiências passadas para escolher a ação, para esse ciclo foram definidas inicialmente 1000 épocas. A Figura 6 mostra a função de distribuição de probabilidade de cada autômato ao final das 1000 épocas, após aproximadamente 5.56 horas de processamento.

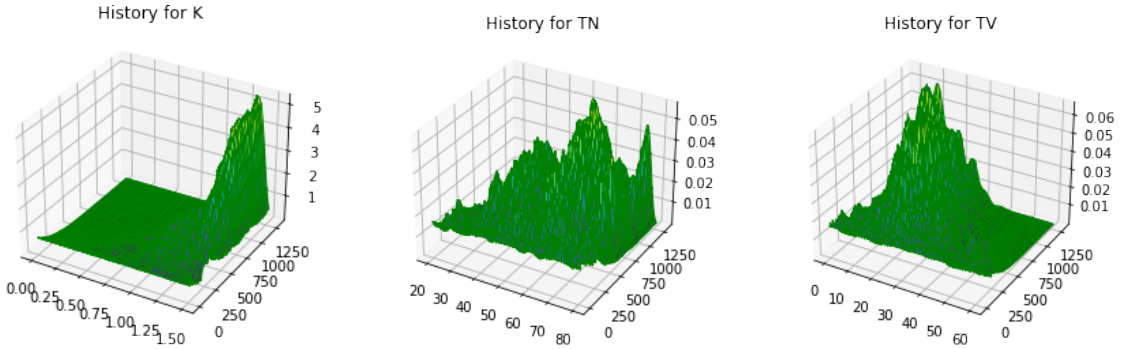


Figura 6. Desenvolvimento da função de distribuição de probabilidade na fase de aprendizado

Os eixos x, z e y tem o mesmo significado dos da Figura 5. Os resultados mostram que na época 1 as funções de probabilidade tem o mesmo formato da última época da fase exploratória de cada autômato (Figura 5), ao decorrer das épocas os valores que tem um bom desempenho tem sua probabilidade aumentada e, portanto, tem maior chance de serem selecionados novamente. Isso faz com que seja formado um pico cada vez maior sobre esse bons valores e os autômatos tendem a convergir para valores específicos e muito próximos um do outros. A Figura 7 mostra a função de distribuição de probabilidade na última época de cada autômato. Em que, 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.

Uma vez estabelecidos os melhores conjuntos de valores do controlador pelo agente de RL, a próxima etapa avalia o comportamento desses valores em uma máquina

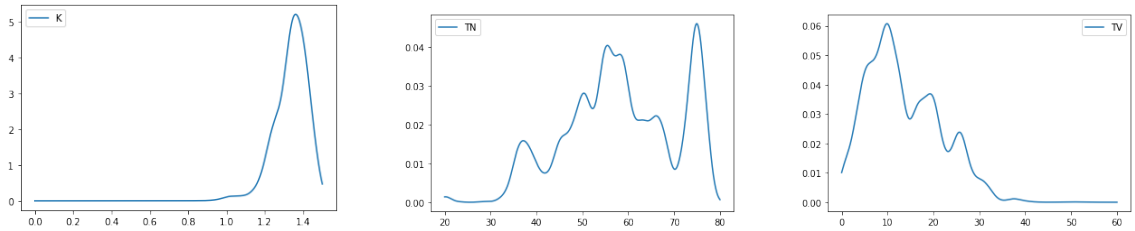


Figura 7. Função de distribuição de probabilidade ao final da fase de aprendizado

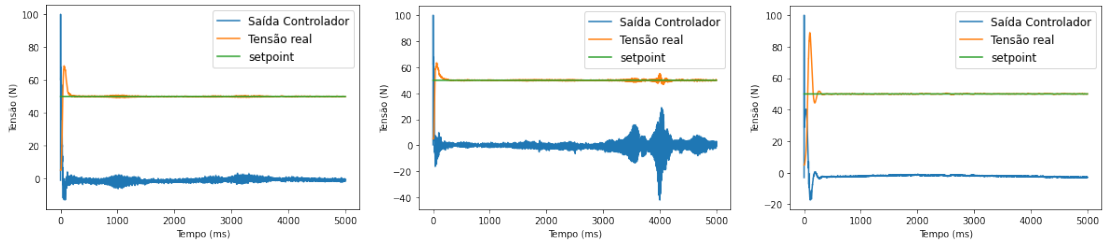
real (*online*). Cada controlador foi testado utilizando o mesmos critérios: a tensão foi inicialmente configurada para 5 N, que representa 10% do *set point*; cada controlador foi carregado no CLP, e; o *set point* foi configurado para 50 N. Após isto, foi iniciada a coleta dos dados por 5 segundos, sendo o processo reiniciado após a coleta para que o próximo controlador fosse testado. Foram selecionados 4 controladores resultantes 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. Em adição, foram testados outros 4 controladores previamente selecionados de forma empírica pelo operador da máquina. A Tabela 1 apresenta os controladores testados e o valor da função custo em cada um.

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%

Tabela 1. Desempenho dos controladores testados no ambiente real

Os resultados expostos na Tabela 1 mostram que os dois melhores controladores quanto a função custo foram definidos pelo agente CARLA. Em seguida, com um diferença percentual de 65,1% em relação ao melhor controlador, é um controlador definido pelo operador. 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 representativa no desempenho. A Figura 8 mostra em detalhes como foi o desempenho dos três melhores controladores no ambiente real.

A Figura 8 mostra que o melhor controlador manteve a tensão muito próxima do *set point* com um pico de sobressinal perto de 40%. Ainda, 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. Tais variações são consequência dos altos valores dos parâmetros PID, principalmente do parâmetro derivativo. Ademais, é possível observar 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



(a) 1º (b) 2º (c) 3º

Figura 8. Desempenho dos três melhores controladores no ambiente real

maior amplitude fez com que esse controlador obtivesse uma maior função custo.

O melhor controlador empírico tem um comportamento bem distinto daqueles definidos pelo agente RL, 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.

A principal limitação da proposta é a escolha da função de custo, que foca apenas na minimização do erro entre a tensão real e o *set point*, sem considerar fatores cruciais para o ambiente industrial, como pico de sobressinal, tempo de acomodação e estabilidade em torno do *set point*. Embora o controlador gerado pelo agente CARLA tenha sido o melhor segundo a função de custo, o 3° melhor controlador é mais adequado para uso industrial, devido à sua maior estabilidade e proximidade do *set point* após a acomodação, o que melhora a qualidade das superfícies produzidas no processo R2R.

5. Conclusão

A coleta de dados realizada diretamente em uma máquina R2R permitiu 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, 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.

Após o treinamento do agente de RL 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. O melhor controlador definido pelo agente obteve um valor de custo 65,1% menor em relação ao melhor controlador definido pelo operador no processo de tentativa e erro. Estes resultados mostram a viabilidade de utilizar algoritmos de RL para o ajuste de controladores em sistemas dinâmicos. Assim, pode-se concluir que o algoritmo CARLA, em conjunto com o ambiente de simulação baseado em um modelo de regressão linear, conseguiu ajustar os parâmetros do controlador PID para diminuir o valor da função custo definida, com um melhor desempenho que os controladores PID escolhidos de forma empírica.

Apesar dos resultados promissores, 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*, que afetam a qualidade do produto 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 para que o controlador se torne mais eficiente no processo de manufatura R2R e adequado para uso industrial.

Referências

(2022). Predicting columns in a table - in depth.

Dehui, W., Chen, C., Xiumiao, Y., Xuesong, L., and Yimin, H. (2014). Optimization of taper winding tension in roll-to-roll web systems. *Textile research journal*, 84(20):2175–2183.

Dogru, O., Velswamy, K., Ibrahim, F., Wu, Y., Sundaramoorthy, A. S., Huang, B., Xu, S., Nixon, M., and Bell, N. (2022). Reinforcement learning approach to autonomous pid tuning. *Computers Chemical Engineering*, 161:107760.

Gouda, M., Danaher, S., and Underwood, C. (2000). Fuzzy logic control versus conventional pid control for controlling indoor temperature of a building space. *IFAC Proceedings Volumes*, 33(24):249–254. 8th IFAC Symposium on Computer Aided Control Systems Design (CACSD 2000), Salford, UK, 11-13 September 2000.

Greener, J. (2018). *Roll-to-Roll Manufacturing*, chapter 1, pages 1–17. John Wiley Sons, Ltd.

Howell, M. N. and Best, M. C. (2000). On-line pid tuning for engine idle-speed control using continuous action reinforcement learning automata. *Control Engineering Practice*, 8(2):147–154.

Howell, M. N., Frost, G. P., Gordon, T. J., and Wu, Q. H. (1997). Continuous action reinforcement learning applied to vehicle suspension control. *Mechatronics*, 7(3):263–276.

Knospe, C. (2006). Pid control. *IEEE Control Systems Magazine*, 26(1):30–31.

Lawrence, N. P., Forbes, M. G., Loewen, P. D., McClement, D. G., Backström, J. U., and Gopaluni, R. B. (2022). Deep reinforcement learning with shallow controllers: An experimental application to pid tuning. *Control Engineering Practice*, 121:105046.

Lee, J., Byeon, J., and Lee, C. (2020). Theories and control technologies for web handling in the roll-to-roll manufacturing process. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 7(2):525–544.

Lee, Y.-S. and Jang, D.-W. (2021). Optimization of neural network-based self-tuning pid controllers for second order mechanical systems. *Applied Sciences*, 11(17).

Ogata, K. (1999). Modern control engineering. *Book Reviews*, 35(1181):1184.

Ribeiro, J. M. S., Santos, M. F., Carmo, M. J., and Silva, M. F. (2017). Comparison of pid controller tuning methods: analytical/classical techniques versus optimization algorithms. In *2017 18th International Carpathian Control Conference (ICCC)*, pages 533–538.

Tao, Y., Yin, Y., and Ge, L. (1998). New pid control and application.