Utilização de Redes Neurais Artificiais no Controle de Motores CC

1st Douglas Ivan Schauren
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Câmpus Farroupilha
Farroupilha, Brasil
douglaschauren@yahoo.com.br

2nd Rodrigo Martini Riboldi
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Câmpus Farroupilha
Farroupilha, Brasil
rodrigo.riboldi@farroupilha.ifrs.edu.br

***Resumo* — A demanda atual por processos mais flexíveis também abrange o escopo dos controladores utilizados na indústria, exigindo uma nova geração deles, visto que os atuais controladores PI (Proporcional Integral) clássicos têm seu desempenho comprometido quando os parâmetros da planta, erros de medida, ruídos ou não-linearidades destoam dos originalmente planejado. Dessa forma, o presente trabalho propõe a implementação de uma Rede Neural Artificial (RNA) para o controle de processos, dentro da disciplina de Trabalho de Conclusão de Curso do curso de Engenharia de Controle e Automação do IFRS - Campus Farroupilha. Buscando incorporar um maior número de variáveis externas aos testes, foi desenvolvida uma planta física, a fim de avaliar as facilidades de implementação e capacidades de uma RNA no controle de motores CC (corrente contínua). Durante este trabalho, foram criadas duas RNAs, a primeira para simular o comportamento da planta física, e a segunda, foi construída para atuar como um controlador, sendo seu treinamento baseado em um PI clássico. Com isso, foi possível realizar comparações de desempenho, e assim, avaliar o papel das RNAs na nova geração de controladores.**

Palavras-chave - Processo não-linear, Rede Neural Artificial, Controle, Motores CC

I. INTRODUÇÃO

A evolução na área de controle é motivada pela necessidade de tratar sistemas cada vez mais complexos e interconectados, com alto grau de precisão e o mínimo de informações da planta [1]. No último século, o controlador PID tem atendido aos requisitos da indústria, sendo que representa cerca de 90 % das malhas existentes [2]. Esse sucesso é devido ao seu bom desempenho e relativa facilidade de implementação [3]. Porém, os controladores PI utilizados nas indústrias, para simplificação de sua configuração e projeto, linearizam os modelos físicos, implicando em perda de informação da planta física a ser controlada [4]. Segundo [1] e [5], definir os parâmetros para os controladores PI quando existem não-linearidades, variações de carga, atritos e outras incertezas, se torna trabalhoso, acarretando na degradação da performance do controlador.

Dessa forma, estudos com controladores não-lineares têm recebido maior atenção, tanto no ramo industrial quanto no acadêmico, justamente por conseguirem capturar a dinâmica de sistemas com um maior número de variáveis de entradas e saídas interconectadas, com poucas informações e com a capacidade de adaptação [6]. Conforme o trabalho de [7], as RNAs ganharam destaque e aceitabilidade por meio das indústrias, pois esses novos controladores são menos sensíveis a variações do sistema quando comparados com controladores PI de ganho fixo.

Ainda conforme [8], as RNAs tem sido aplicadas com sucesso na identificação e controle de sistemas lineares e não-lineares. Seu sucesso se deve pois apresenta a capacidade de reter conhecimento baseado em experiências e generalização de situações, extraindo o modelo desses sistemas. As RNAs podem até mesmo responder a estímulos a qual não haviam sido treinadas, e dessa maneira, podem ser implementadas como controladores em situações que a planta apresenta variações de comportamento, como ruídos e variações paramétricas [9].

A Indústria 4.0 também tem papel fundamental nessa transformação, pois está demandando controladores que permitam maior conexão, flexibilidade e inteligência. Sendo assim, este trabalho pretende explorar a aplicabilidade das RNAs em simular e controlar a planta física constituída por um motor CC. Esse dispositivo foi selecionado para constituir a planta de controle, pois é um bom meio para implementação de novos controladores, visto que apresenta um comportamento estrutural constante [8].

O objetivo do trabalho é implementar uma RNA que seja capaz de predizer o comportamento da planta física construída, ajudando na parametrização do controlador PI clássico. Tendo o controlador PI sido ajustado, ele será utilizado para obter os dados de treinamento da RNA que irá simular seu comportamento e à qual será inserida no sistema para controlar a planta física. Posteriormente, com o objetivo de validar a performance das RNAs frente a alteração de parâmetros, a planta física sofrerá alteração em sua dinâmica, ao adicionar-se carga. A utilização de uma planta física fornecerá um ambiente de controle mais parecido aos industriais, permitindo com que se possa avaliar melhor o comportamento das RNAs frente a interferências externas.

O trabalho está organizado da seguinte maneira: na seção II serão apresentadas diferentes abordagens no controle de processos com RNAs. A seção III conterá a fundamentação teórica, abordando a estrutura do motor CC, as redes neurais artificiais e as formas de treiná-la. Na seção IV será apresentada a proposta de RNA para a simulação e controle do sistema, bem como as etapas do projeto. Na seção V, serão demonstrados os resultados obtidos com o trabalho, avaliando o desempenho das RNAs implementadas. Por fim, na seção VI, serão apresentadas as observações dos resultados obtido.

II. REVISÃO BIBLIOGRÁFICA

Os controladores PID clássicos, com retroalimentação unitária, apresentam uma configuração muito semelhante entre si, normalmente tendo o projetista a necessidade de ajustar os ganhos do mesmo. Já as RNAs, podem apresentar

inúmeras variações, quer seja no esquema de ligação de controle, quer seja em sua estrutura e forma de treinamento.

Como elencado no trabalho de [5], é justamente devido à falta de generalização dos controladores PID clássicos, que seu desempenho em controles não-lineares acaba sendo, muitas vezes, inferior ao necessário. Nos motores CC, a não linearidade é ocasionada pelas zonas mortas presentes nesse tipo de sistema. Através de seu estudo, é demonstrada a eficiência das RNAs no controle dos sistemas nessas regiões, sendo os resultados obtidos superiores quando comparados com controladores PID clássicos.

Porém, a RNA só apresenta um bom resultado se for configurada corretamente conforme cada problema. No trabalho de [10] são demonstradas as diferentes formas de inserção da RNA em um processo, quer seja atuando diretamente sobre a planta, fazendo as vezes do controlador, ou mesmo sendo usada para predizer o comportamento da planta. O autor de [10] demonstrou que sistemas preditivos utilizando otimizadores matemáticos apesentam melhores resultados quando comparados a RNA atuando como controladores, porém, a complexidade dessa abordagem, comumente não a justifica para sistemas mais simples.

As RNAs têm seu cerne de funcionamento baseado em definir uma função que generalize o processo em questão. Para isso, ela utiliza neurônios em múltiplas camadas, as quais podem variar conforme a complexidade do problema. Conforme o estudo de [6], foi demonstrado que são suficientes três camadas para prever qualquer função, tanto linear como não-linear. Em seu trabalho, ao utilizar esse número de camadas, a RNA apresentou uma curva de controle mais rápida e assertiva em comparação ao PID clássico.

O número de neurônios em cada camada costuma ser realizado de forma empírica, porém o peso atribuído a cada um deles é realizado de forma dinâmica por um algoritmo de treinamento. Nos trabalhos de [9] e [11], a rede é treinada *offline* utilizando o *backpropagation*, nos quais os dados para treinamento são coletados aplicando um controlador PID e PI, respectivamente, na planta.

Já no trabalho de [1], onde são controladas as vazões de tanques de água, o treinamento também é realizado *offline* e com *backpropagation*, porém, os dados para o mesmo são obtidos em malha aberta, variando a abertura da válvula e verificando a vazão resultante. Com isso, o autor de [1] criou um método de auto-treinamento, mantendo a rastreabilidade por meio do controlador sem a necessidade de configurar o PID para obter os dados de treinamento.

Outros métodos de aprendizado de máquina também podem ser utilizados em sistemas de controle, como os algoritmos genéticos, que visam otimizar os resultados por meio da seleção do melhor algoritmo. Utiliza-se no trabalho de [2] a análise de uma planta multivariáveis, onde busca-se determinar os melhores coeficientes Kp, Ki e Kd dos controladores PID que a constituem.

Buscando aumentar a velocidade de desenvolvimento de um controlador neural, no trabalho de [12] a RNA foi utilizada para determinar o número ideal de neurônios na camada oculta da RNA de controle, processo realizado por treinamento online. Foi demonstrado que esse sistema se torna mais eficiente e rápido quando comparado com a busca empírica, porém, devido a utilização de duas RNAs, a complexidade de desenvolvimento aumenta, não sendo justificada em sistemas com poucas variáveis.

III. FUNDAMENTAÇÃ TEÓRICA

As bases teóricas que regem o projeto e que contribuem para compreender melhor os resultados obtidos, bem como a estrutura delimitada, serão apresentados como se segue: primeiramente será explorada as características do motor CC, planta a ser controlada, seguido da modelagem do controlador PI discretizado pelo método de Tustin. Por último, será explanado sobre as características de estrutura e treinamento das RNAs para controle de processos não-lineares.

A. Modelo não-linear do Motor CC

Os motores CC são amplamente utilizados na indústria para controle de inúmeros processos, motivado por sua estabilidade e controle eficiente de velocidade, torque e posição [11]. Eles podem ser modelados como sistemas SISO (*Single-Input Single-output)*, devido a sua característica de controle da velocidade por meio da tensão aplicada na armadura do motor [8].

Os modelos adotados normalmente para os motores CC são sistemas lineares de segunda ordem, porém, como salientado por [5], essa consideração implica em ignorar o atrito de *Coulomb* no motor. Ele é gerado por atrito e desgaste mecânico, formando as zonas mortas (situação em que é necessária uma certa magnitude de tensão para o motor iniciar o movimento), dificultando o controle, principalmente em baixas velocidades angulares. Segundo [13], atualmente para contornar as zonas mortas são utilizadas técnicas de controladores adaptativos ou robustos, as quais apresentam alto grau de dificuldade e modelagem matemática.

Neste trabalho, a planta a ser controlada consiste de um motor CC, assim como no trabalho [5] e conforme pode ser visto na Fig. 1.

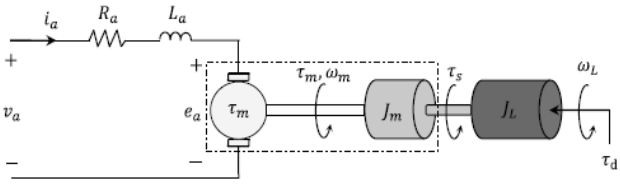

Fig. 1. Representação eletromecânica do sistema utilizado [5].

Para compreender melhor o sistema a ser controlado, serão demonstradas as equações que modelam o motor CC, importante principalmente para o desenvolvimento dos controladores PID. O equacionamento elétrico e mecânico (inércia e torque), conforme o trabalho de [5], é apresentado nas Eq. 1 e 2 respectivamente.

$$v_a = R_a i_a + L_a \frac{di_a}{dt} + e_a \quad (1)$$

$$J_m \frac{d\omega_m}{dt} = \tau_m - B_m \omega_m - \tau_s \quad (2)$$

Na Eq. 1, v_a é a tensão de armadura, R_a é a resistência da bobina da armadura, i_a é a corrente da armadura, L_a é a indutância da bobina da armadura e e_a é a força contraeletromotriz. Já na Eq. 2, J_m é o momento de inércia, ω_m é a velocidade angular do motor, τ_m é o torque do motor, B_m é o coeficiente de atrito viscoso e τ_s é o torque transmitido pelo eixo.

A carga adicionada sobre o motor será posicionada diretamente sobre eixo, devendo também ser considerada como parte integrante do sistema, e, portanto, seu torque e inércia são avaliados pela Eq. 3.

$$J_L \frac{d\omega_L}{dt} = \tau_s - B_L\omega_L - \tau_d \quad (3)$$

Na Eq. 1, J_L é a inércia da carga, ω_L é a velocidade angular, B_L é o atrito viscoso da carga e τ_d é o torque da carga. As Eq. 1, 2 e 3 são compostas apenas por elementos lineares, desconsiderando o atrito de *Coulomb*. Nessas situações, a Eq. 4 pode ser utilizada para inserir o atrito em modelos de sistemas mecânicos, simulando assim o comportamento não-linear do motor CC.

$$\tau_f(\omega) = \alpha_0 \mathrm{sgn}(\omega_m) + \alpha_1 e^{-\alpha_2|\omega_m|}\mathrm{sgn}(\omega_m) \quad (4)$$

Onde α_0, α_1 e α_2 são constantes, enquanto $\tau_{f(\omega)}$ é o torque gerado pelo atrito de *Coulomb*. Para que o sistema do motor CC e sua carga considerem na modelagem os efeitos não-lineares, a Eq. 4 é adicionada, como uma perda de torque, na Eq. 2 e 3, conforme apresentado nas Eq. 5 e 6

$$J_m \frac{d\omega_m}{dt} = \tau_m - B_m\omega_m - \tau_s - \tau_f(\omega_m) \quad (5)$$

$$J_L \frac{d\omega_L}{dt} = \tau_s - B_L\omega_L - \tau_d - \tau_f(\omega_L) \quad (6)$$

Dessa maneira é possível observar que o rotor só irá partir quando conseguir vencer o torque gerado pelo atrito de *Coulomb*, criando-se assim uma zona morta não-linear [5].

Para desenvolver um controlador PI com controle robusto ou adaptativo, seria necessário conhecer todos os parâmetros e constantes das equações apresentadas, tarefa difícil principalmente para as cargas, devido à complexidade que podem assumir [8]. Por isso, os controladores neurais artificiais têm sido tópicos de muitas pesquisas, visto que necessitam apenas dos dados de entrada e saída da planta para capturarem sua dinâmica [4].

B. Controlador PI discretizado

Os controladores PI, como o demonstrado na Eq. 7, estão no domínio da frequência, abordagem frequentemente utilizada pois facilita os cálculos matemáticos, além de permitir que os sistemas sejam modelados nesse domínio [3].

$$G_{PI}(s) = \frac{M(s)}{E(s)} = K_p + \frac{K_i}{s} \quad (7)$$

Na Eq. 7, $G_{PI}(s)$ é a ação de controle, K_p é o ganho proporcional e K_i é o ganho integral. Esses sistemas são utilizados com controladores analógicos, porém, com o crescimento dos sistemas digitais, os controladores PI também foram digitalizados, passando a atuar no domínio discreto (domínio da transformada z), visto que os dados devem ser amostrados a uma taxa fixa, chamada de período de amostragem [3].

Para utilizar o controle PI digitalmente, a Eq. 7 é discretizada utilizando o método de *Tustin* para o termo integrativo, assim como realizado no trabalho de [2]. Para isso, substitui-se a Eq. 7 em 8 gerando a Eq. 9, 10 e 11.

$$s = \frac{z-1}{z+1} * \frac{2}{T_s} \quad (8)$$

$$U(k) = U(k-1) + Ae(k) + Be(k-1) \quad (9)$$

$$A = \frac{2K_p + K_iT}{2} \quad (10)$$

$$B = \frac{-2K_p + K_iT}{2} \quad (11)$$

Nas Eq. 9, 10 e 11 $U(k)$ é a ação de controle atual, $e(k)$ é erro atual, K_p é o ganho proporcional, K_i é o ganho integral e T é o período de amostragem. Os ganhos proporcionais e integrais são facilmente definidos quando todas as características do processo a ser controlado são conhecidas, porém, quando não é possível gerar-se o modelo matemático no domínio da frequência, muitas vezes utiliza-se aproximações e testes para definição desses parâmetros, o que nem sempre gera a melhor performance para o controlador PI [9].

C. Rede Neural Artificial

Muitos processos atualmente apresentam grande complexidade em suas dinâmicas, motivados principalmente pela alta precisão exigida e pelo grande número de variáveis. Os desenvolvedores encontram dificuldade em estruturar todas as condições e restrições que estes processos podem conter, e por isso, técnicas que identifiquem e aprendam as características e singularidades dos problemas foram criadas, nas quais se destaca o aprendizado de máquina (AM) [14].

Uma das técnicas de AM mais utilizada são as redes neurais artificiais, sendo inspiradas nas redes neurais biológicas. As RNAs são modelos computacionais que apresentam a capacidade de se adaptar e aprender por meio de treinamento [4]. Conforme [14], as RNAs estão presentes em inúmeros projetos práticos nos últimos anos, especialmente por possuírem a capacidade de generalização e tolerância a falhas e ruídos. Isso é possível pois estão estruturadas na computação paralela, ou seja, a informação se distribui por toda a rede [1].

As redes neurais artificiais são usadas para encontrar uma função ou hipótese do problema a partir de treinamento. Se os dados de entrada forem um conjunto de valores nominais, tem-se um classificador, como observado na Fig. 2.a, já se os dados são infinitos e ordenados, tem-se um problema de regressão, como visto na Fig. 2.b. Para problemas envolvendo o controle de processos, deseja-se encontrar a função que melhor modela o sistema, e, portanto, é utilizado um modelo regressor [14].

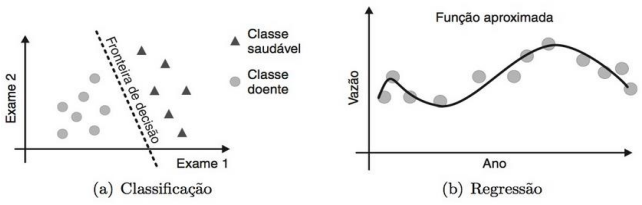

Fig. 2. Classificação das redes conforme o domínio do problema [14].

1) Arquitetura das redes neurais artificiais: As RNAs são compostas de neurônios, elementos de cálculos que interagem entre si por conexões que podem variar conforme a característica da rede [15]. A Fig. 3 representa a estrutura de

um neurônio, na qual a entrada escalar p é transmitida por uma conexão que será multiplicada por um peso escalar w. Além dessa informação, é adicionado um componente extra à soma, o *bias* b, valor utilizado para permitir que a rede se adapte melhor aos dados do treinamento. Essas informações são transmitidas para o somador, e sua saída n, é referida como a entrada da rede [4].

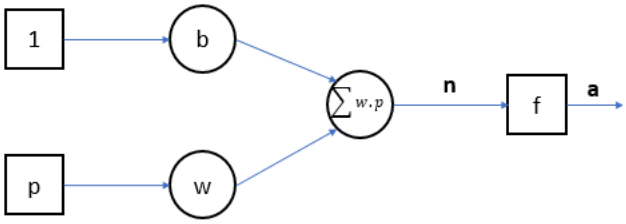

Fig. 3. Estrutura simplificada de um neurônio com entrada única [4].

O resultado do somador é aplicado na função de ativação, responsável por indicar o comportamento do neurônio para a respectiva entrada. São diversas as funções de ativação utilizadas, sendo que seu emprego depende do problema. Conforme elencado por [14], redes multicamadas utilizam funções de ativação não-lineares nas camadas intermediárias.

As funções mais utilizadas para projetos de controle são a tangente hiperbólica e a *sigmoid*, conforme ilustrado na Fig. 4 [1]. Para utilização dessas funções, os dados devem ser normalizados para que não ocorram singularidades matemáticas. Outra característica importante dessas funções, é o fato de serem diferenciáveis, contínuas e não decrescentes, características necessárias para algumas técnicas de aprendizado, como o *backpropagation* [4].

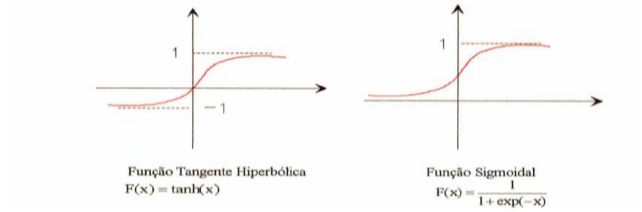

Fig. 4. Funções de ativação não-lineares [1].

Como comentado, a função de ativação é determinada pelo projetista conforme as características do problema, já os pesos *b* e *w*, são ajustados pela regra de aprendizado durante o treinamento [4]. A Eq. 12 ilustra matematicamente o funcionamento do neurônio, onde sua saída y é igual ao somatório de cada entrada p multiplicada pelo seu respectivo peso w mais o *bias* b aplicados na função de ativação φ [9].

$$y = \varphi(b + \sum_{i=1}^{n} p_i w_i) \qquad (12)$$

A RNA presente na Fig. 3 é composta apenas por uma camada, no entanto, as RNAs costumam apresentar várias delas. Conforme [2], as RNAs são compostas por 3 camadas conectadas, sendo elas: camada de entrada, camadas intermediárias (ocultas) e a camada de saída, cada qual podendo apresentar um número variado de neurônios. Na Fig. 5 é apresentado uma rede tipicamente utilizada no controle de sistemas, sendo a camada de entrada responsável por receber os valores dos atributos de entrada do sistema, a camada oculta, onde o processamento ocorre, e uma camada de saída, contendo a ação de controle.

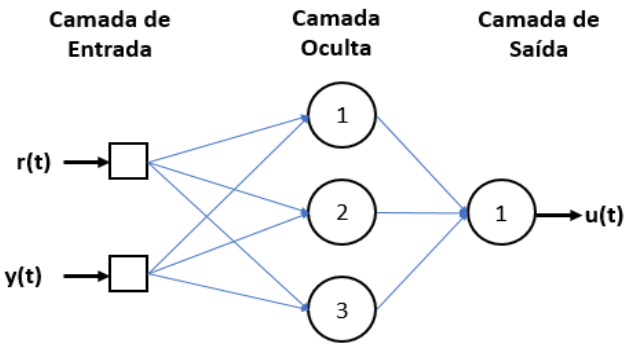

Fig. 5. Estrutura de uma rede neural com 3 camadas [9].

Quanto ao número de neurônio nas camadas de entrada e saída, é dependente das variáveis que interferem na dinâmica da planta e quais as variáveis de atuação existem para o controle da mesma. No trabalho de [7], o qual controla uma planta térmica simulada, dois neurônios são utilizados na camada de entrada. Já o autor de [11], que implementa um sistema físico para controle de motor CC, são utilizadas 4 entradas, visto que mais variáveis podem interferir na velocidade da planta. Porém, a camada de saída em ambos conta com apenas um neurônio, o qual será a ação de controle.

Já a definição da quantidade de neurônios na camada oculta, ocorre normalmente por tentativa e erro, não existindo um método de cálculo. Porém, sabe-se que quanto maior for o grau de dificuldade do processo controlado, maior o número de neurônios necessários [14] . A busca pela quantidade ideal pode ser obtida de duas formas:

- Empírica: o projetista testa diferentes arquitetura de redes até que se encontre a RNA que apresente a performance desejada. Esse método traz elevado custo e esforço, tanto computacional como do projetista.
- Construtiva: Incrementa-se novos neurônios na camada oculta por meio de um algoritmo que irá avaliar a RNA que apresenta o melhor resultado.

2) Algoritmo de Aprendizado: Sistemas PI clássicos necessitam o modelo da planta para que seus parâmetros sejam ajustados, já nas RNAs, são utilizados dados de conjuntos de amostras (pares entrada-saída) para seu treinamento no aprendizado supervisionado. Nessa técnica, um algoritmo avalia a capacidade da RNA em descrever o comportamento do sistema [4].

O conjunto de amostras deve ser fiel à planta, e dificuldades como ruídos, precisam ser tratados em um pré-processamento de dados. Além disso, outros dois fatores devem ser levados em consideração. O primeiro é quanto ao tamanho do conjunto de amostras, que necessita ser grande o suficiente para que a rede apresente boa acurácia, mas deve-se ter o cuidado para não perder a sua capacidade de generalização [14]. O segundo diz respeito a frequência de treinamento, onde em sistemas invariantes no tempo, é realizado o treinamento *offline*, visto que a RNA será treinada apenas uma vez. Já quando o sistema é variante no tempo, como variações de carga, é necessário utilizar uma estrutura *online*, e, portanto, a rede é treinada constantemente, criando um problema de otimização dinâmica [4].

Existem diversos algoritmos para treinamento das redes neurais, ou seja, para encontrar o valor ideal dos pesos. Para as redes multicamadas, um dos algoritmos mais empregado é o *backpropagation*, devido a sua tolerância às falhas e

possibilidade de ser utilizado em arquiteturas paralelas como as RNAs [14]. Conforme [1], com esse algoritmo, em uma RNA de três camadas, é possível estimar qualquer função contínua com qualquer grau de precisão, desde que haja na camada oculta um número suficiente de neurônios.

Esse algoritmo é um processo de otimização baseado na regra do delta e gradiente descendente que objetiva minimizar o erro quadrático médio entre a saída desejada e a calculada pela RNA. Ele consiste de duas etapas [4]:

- *Forward*: as entradas são processadas e multiplicadas pelos pesos atuais da rede, gerando como resultado as saídas da RNA, com as quais torna-se possível calcular o erro quadrático médio.
- *Backward*: de posse do erro, é utilizado o gradiente descendente para calcular a direção que leva ao menor erro de cada camada, e consequentemente, o novo peso. Esse processo ocorre da camada de saída para a de entrada.

O processo de *forward* e *backward* ocorre para cada par de entrada e saída até que todos os dados do conjunto de amostras tenham sido processados, formando assim uma época. Porém, algumas vezes apenas uma época não é suficiente, e dessa maneira, época após época o processo de *backpropagation* ocorre, até que um critério de parada tenha sido alcançado. Existem diferentes abordagens para a implementação do algoritmo de *backpropagation*, alterando-se conforme a implementação e linguagem de programação utilizada.

O treinamento normalmente é encerrado quando a taxa de erro for menor que o erro admitido, porém, deve-se ter cuidado para que a generalização da RNA não seja degenerada, ocasionando o *overtraining*, momento no qual a rede apresenta um erro baixo, porém a capacidade de generalização é perdida [4].

Para o treinamento, utiliza-se em muitos casos bibliotecas já desenvolvidas, nas quais se destaca a *scikit-learn*, projetada em *Python.* É otimizada para trabalhar com diferentes formas de aprendizado de máquina, tendo para a regressão o modelo *MLPRegressor.* Esse modelo otimiza a perda quadrática usando a descida do gradiente estocástico, treinando de forma iterativa, visto que a cada época, as derivadas parciais são calculadas para atualizar os parâmetros [16].

Para avaliar o desempenho da rede neural após o treino, existem diferentes cálculos matemáticos, nas quais pode-se citar a raiz quadrada da média dos erros do inglês *Root mean Squared error* (RMSE) que mede o grau de dispersão em torno da linha de regressão. Já o parâmetro R^2, conhecido como coeficiente de determinação, mede com qual acurácia a rede neural explica o modelo de regressão, variando de 0 até 1, no qual, quanto mais próximo de 1, mais assertiva a rede é.

3) Estrutura da RNA no controle de processos: As tarefas de aprendizado com RNAs podem ser divididas em duas, a descritiva e a preditiva. Os controladores neurais preditivos são amplamente utilizados para controlar processos [4], nesse tipo de abordagem, a rede neural constrói um estimador do processo, tentando prever qual a melhor ação de controle [14].

Existem inúmeras estratégias de controle envolvendo redes neurais artificiais preditivas, cada qual variando seu nível de dificuldade e robustez.

Ainda segundo [10], a utilização das redes neurais em controle pode ser classificada em três categorias:

- Controle direto por rede (*Direct network control):* nesse modelo, a RNA tenta imitar o comportamento de um controlador com retroalimentação, sendo que seu treinamento pode ser realizado utilizando os dados de um controlador PI, com a vantagem de conseguir-se melhores resultados devido a sua capacidade de generalização. Essa abordagem pode ser visualizada na Fig. 6.

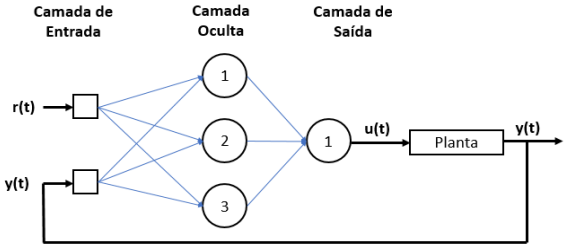

Fig. 6. Controle direto com RNA. Adaptado de [9].

- Controle Inverso por rede (*Inverse network controle*): envolve o treinamento da RNA para que compreenda o processo de forma inversa, ou seja, prediz a entrada necessária (ação de controle) de um processo para obter a saída desejada (*setpoint).* Nessa estrutura, a RNA é utilizada como visto na Fig. 7. Esse tipo de abordagem deve ser aplicado em processos simples, pois não apresenta alta robustez.

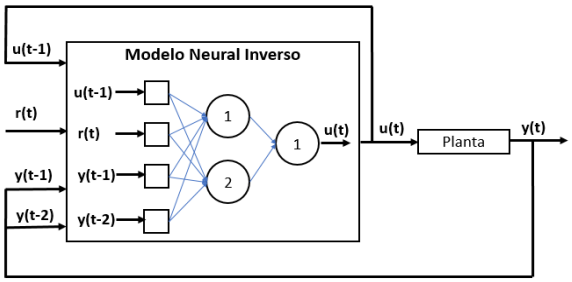

Fig. 7. Controle inverso com RNA. Adaptado de [10].

- Controle indireto por rede (*Indirect network controle):* nesta abordagem, a rede neural irá servir como modelo do processo, prevendo seu comportamento dado uma determinada entrada. O controlador por sua vez, pode ser implementado por diversas maneiras, como otimizadores matemáticos ou uma rede neural inversa, processo denominado Controle com Modelo Interno (CMI) o qual pode ser visto na Fig. 8. Nessa metodologia, é necessário o treinamento de duas redes, uma sendo utilizada como modelo do processo (gm) e outra rede inversa para ser utilizada como controlador (gc).

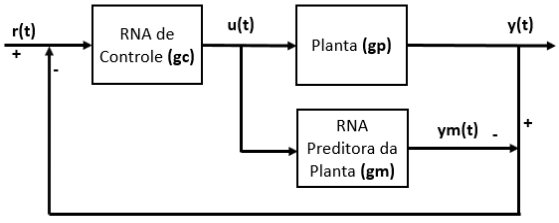

Fig. 8. Controle com IMC. Adaptado de [10].

IV. PROPOSTA

Visando a introdução nos processos controlados por redes neurais artificiais, o trabalho irá explorar sua aplicação em um motor CC, buscando assim verificar como variáveis externas podem interferir na acurácia do controle pela RNA.

O desenvolvimento para tanto, será focado em dois aspectos essenciais: a obtenção com baixo ruído da rotação do motor, bem como seu controle de velocidade por meio de um sistema de potência PWM (*Pulse Width Modulation*) e o treinamento da RNA para compreender e controlar a dinâmica do sistema proposto.

A. *Planta controlada*

Para simular a planta física, será utilizado um motor de 12V e 3 A de corrente, da marca *Bosh*, visto na Fig. 9, comumente utilizado em limpadores de para-brisas de veículos. Esse motor terá a caixa de redução de seu eixo retirada e foi escolhido por apresentar facilidade de controle e ter uma potência e inércia maior, pois sabe-se que a inércia, seja do próprio motor ou da carga, interferem no controle ao alterar a dinâmica de aceleração e desaceleração.

Fig. 9. Motor CC utilizado para construir a planta de controle.

Para controlar a tensão aplicada à armadura do motor e consequentemente a velocidade, será utilizado um controlador de potência com chaveamento PWM. Para atender aos requisitos de tensão e corrente CC, será utilizada a ponte H BTS7960, vista na Fig. 10, constituída por dois drivers modelo BTS7960, com capacidade de operar motores de até 43A. Ela permite controle por PWM, além de conter proteção térmica, de sobretensão e sobrecorrente.

Para o controle do PWM, será utilizado um *Raspberry PI* 3 B+, o qual tem saídas PWM implementadas via *software*, juntamente com optoacopladores modelo 4N25, realizando a interface de comunicação com a ponte H, bem como o retorno do *feedback*, mantendo a alimentação do *Raspberry PI* 3 B+ totalmente independente da planta construída, e assim, protegendo o microprocessador de sobrecargas.

Fig. 10. Ponte H BTS7960 para controle de potência do motor CC.

Já para o *feedback* do controle, ou seja, para medir a rotação do motor, será utilizado um sensor de interrupção ótica como o visto na Fig. 11. Este modelo de sensor funciona perante a interrupção da luz que flui entre o diodo emissor de luz infravermelho e o fototransistor receptor. Para realizar o trem de pulsos (interrupção da passagem de luz), um disco com ranhuras será acoplado ao motor, e este irá percorrer o espaço entre o emissor e receptor do sensor, não necessitando contato físico e, evitando assim, interferências mecânicas.

Fig. 11. Sensor de rotação com fototransistor.

A quantidade de ranhuras no disco irá definir a frequência de amostragem do RPM. Inicialmente, será testado um disco com 16 delas, na qual esse número baixo, embora possa trazer menor precisão de leitura da velocidade, irá evitar a sobrecarga do controlador utilizado, podendo-se facilmente aumentar ou diminuir o número de ranhuras conforme for necessário.

Na Fig. 12 é possível observar uma representação em 3D da planta de controle, incluindo os sensores e a parte de potência e processamento.

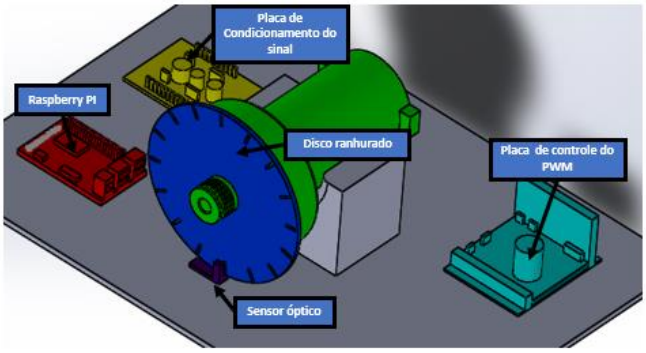

Fig. 12. Estrutura da planta a ser controlada.

B. *Estruturas das RNAs utilizadas*

Conforme visto anteriormente, existem diversas configurações de RNAs passíveis de serem utilizadas nos processos de controle, dependendo principalmente do nível de precisão e complexidade da planta. Embora o controle do motor CC seja um processo rápido, ele apresenta poucos parâmetros que interferem no controle, e, portanto, será utilizada a configuração de controle direto por rede, onde a RNA irá reproduzir o funcionamento de um controlador PI.

Essa abordagem irá facilitar o desenvolvimento do controlador neural, proporcionando uma maior imunidade a ruídos provenientes do sistema de sensoriamento e controle. Além disso, o uso da RNA nessa configuração irá permitir a comparação direta entre esse sistema de controle e os sistemas com PI clássico, principalmente no que diz respeito a variações e controle em regiões não-lineares da planta.

Primeiramente, para identificar o correto funcionamento do sensoriamento e controle PWM do motor CC, será desenvolvida uma RNA que seja capaz de simular o comportamento da planta física. Os dados para seu treinamento e validação serão obtidos aplicando *steps* de tensão na planta a ser controlada, adquirindo assim o par de registros com as ações de controle u(t), ou seja, a tensão aplicada e as respectivas velocidades do motor y(t). Para fins de treinamento, a ação de controle será considerada como um valor numérico que representa o PWM aplicado na planta, já

a velocidade, será um valor em RPM, obtido por meio de uma função que converte os pulsos do sensor.

Para tanto, serão aplicados 10 *steps* de tensão, divididos igualmente, até se atingir a tensão de 12 V, tensão máxima do motor, nos quais em cada aplicação, inicialmente, 200 amostras serão extraídas. Com estes 10 subconjuntos de dados, se totalizarão 2000 pares de entrada e saída para o treinamento e validação da rede neural.

Essa RNA será constituída por 5 parâmetros de entrada conforme visto na Fig. 13, nas quais duas são as ações de controle u(t-1) e u(t-2) defasadas no tempo, duas são as respectivas saídas de velocidade da planta y(t-1) e y(t-2) também defasadas no tempo e por fim, tem-se a ação de controle atual u(t). Já a saída é composta de apenas um neurônio, sendo ele a velocidade atual da planta y(t) quando aplicada a entrada u(t). Ao utilizar o mecanismo de entradas e saídas em tempos t-n, possibilita-se que a RNA consiga compreender melhor a forma da curva de resposta da planta.

O número de neurônios na camada oculta costuma ser encontrado por testes empíricos durante o treinamento, mas para auxiliar a escolha, será utilizada a Eq. 13, onde N_{CO} é o número de neurônios na camada oculta, N_{CE} é o número de neurônios na camada de entrada e N_{CS} é o número de neurônios na camada de saída. Utilizando a Eq. 13, chega-se que o número necessário de neurônios são de 3 na camada oculta.

$$N_{CO} = \frac{N_{CE}+N_{CS}}{2} \quad (13)$$

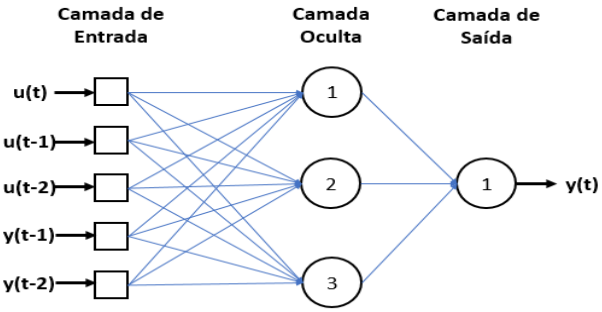

Fig. 13. Estrutura da RNA preditora da planta.

De posse da RNA que prediz o comportamento da planta, será possível utilizá-la para uma pré-configuração do PI clássico, isso implica em um menor desgaste da planta física durante os testes para a obtenção dos ganhos K_p e K_i. Esse tipo de simulação, utilizando dados físicos para a modelagem da planta, traz maior precisão quando comparado as simulações meramente teóricas, principalmente por conseguirem capturar as regiões não-lineares.

Os ganhos do controlador PI clássico serão obtidos empiricamente, por meio de testes do controlador na planta simulada, podendo realizar variações abruptas nos parâmetros, visto que estes estarão sendo executados em um ambiente de simulação. Após obter-se resultados satisfatórios de controle utilizando a RNA preditora e o controlador PI, se irá realizar o ajuste fino dos ganhos na planta real, buscando assim, ter-se um controlador PI com uma maior precisão quando aplicado à planta física.

De posse do controlador PI ajustado, pode-se extrair os dados para treinar e validar a RNA que irá reproduzir seu comportamento. Para isso, serão aplicadas 10 velocidades de referência no controlador PI, divididas igualmente até atingir a velocidade máxima de 3000 RPM. O controlador irá acionar a planta até que atinja a velocidade especificada, nas quais, 200 amostras da ação de controle u(t), a velocidade do motor y(t), o erro e(t) e a velocidade de referência y_r serão registradas. Novamente se terão 10 subconjuntos de dados, totalizando 2000 amostras, que serão divididas entre o treinamento e validação da RNA.

Essa RNA por sua vez, demandará 6 entradas, conforme visto na Fig. 14, nas quais duas são as ações de controle u(t-1) e u(t-2) defasadas no tempo, duas são as respectivas saídas de velocidade da planta y(t-1) e y(t-2) também defasadas no tempo, uma será a referência de velocidade y_r e por fim, tem-se o erro entre a velocidade atual e a referência e(t). Na camada de saída, tem-se apenas um neurônio, sendo ele a ação de controle atual u(t). Utilizando-se da Eq. 13, descobre-se que o número de neurônios necessários na camada oculta são 4.

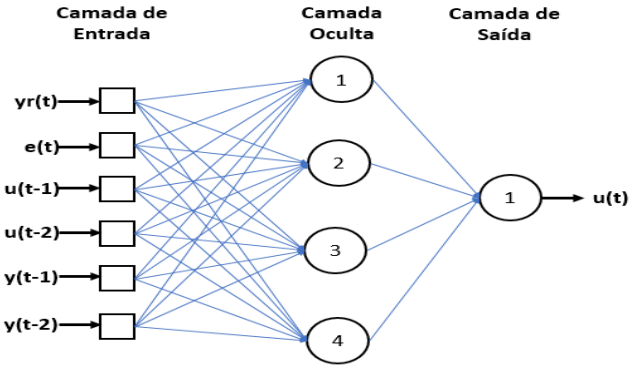

Fig. 14. Estrutura do controlador PI-neural.

Como observado, ambas RNAs terão uma estrutura *Feedforward*, ou seja, todos os neurônios de uma camada estarão ligados a todos os neurônios da camada seguinte. Essa estrutura se faz necessária devido ao algoritmo de *backpropagation* utilizado para o treinamento, escolha feita pois permite aproximar qualquer função com apenas 3 camadas e apresenta tolerância a ruídos.

C. Treinamento da RNA

Com a estrutura definida e sabendo-se as entradas e saídas necessárias para o treinamento e funcionamento da RNA, é necessário a preparação dessas informações no banco de dados. Esse pré-processamento objetiva eliminar inconsistência, ocorridas principalmente por interferências e ruídos nos sensores.

Para este trabalho, o algoritmo de controle do PWM e implementação da RNA será realizado na linguagem *Python*, visto que esta contém uma vasta quantidade de bibliotecas voltadas ao treinamento por *backpropagation*, além de ser capaz de interagir com as entradas e saídas digitais do *Raspberry PI*.

O conjunto de dados para treinamento será de 40 % de cada subconjunto obtido ao aplicar os *Steps* na planta física e no controlador PI clássico. Enquanto que os outros 60 % de cada subconjunto obtido ao aplicar os *Steps*, serão utilizados posteriormente na fase de validação da rede, na qual aplica-se a entrada na RNA e compara-se a saída gerada com a esperada, verificando assim sua assertividade.

O treinamento com o *backpropagation* inicia-se pelo processo de *forward*, nos quais os pesos, inicialmente gerados aleatoriamente, são multiplicados e somados pelos respectivos valores de entrada da RNA aos quais estão conectados. Essa soma, por sua vez, irá passar pela função de ativação *sigmoid*. Após definido o valor de cada neurônio da camada oculta, o processo descrito anteriormente ocorre também para a camada de saída, gerando assim, o resultado da RNA. De sua posse, é possível calcular o erro quadrático médio, que pode ser obtido por meio da Eq. 16, onde y_n é a saída obtida e y_e é a saída esperada.

$$erro = \frac{1}{n} * \sum_{n=1}^{k} (y_e - y_n)^2 \quad (16)$$

Após se encontrar o erro, ocorre o processo de *backward*, no qual os novos pesos são calculados, da camada de saída para a camada de entrada. Nesse processo, utiliza-se a descida do gradiente, que irá direcionar para o menor erro possível. Para isso, aplica-se a derivada parcial D_s no resultado da função *sigmoid* V_s, como visto na Eq. 17. Esse resultado irá indicar a direção da alteração dos pesos, para que se convirja ao menor erro, como pode ser visto na Fig. 15.

$$D_s = V_s * (1 - V_s) \quad (17)$$

Fig. 15. Gradiente descendente do erro.

De posse da derivada, é necessário calcular o valor do delta, parâmetro utilizado para ajustar efetivamente o peso. Para a camada de saída, o delta é encontrado multiplicando o erro pela derivada da função *sigmoid*, como visto na Eq. 18. Já para os neurônios das demais camadas, o delta é encontrado multiplicando a derivada da função *sigmoid* daquele neurônio, pelo peso que o liga até a camada de saída, pelo delta da camada de saída, como pode ser visto na Eq. 19.

$$Delta_{CSaída} = erro * D_{sSaída} \quad (18)$$

$$Delta_{COculta} = D_{SOculta} * Peso * Delta_{Csaída} \quad (19)$$

Por fim, calcula-se o peso utilizando a Eq. 20, na qual, o novo peso é igual ao antigo peso, somado as entradas x de todos registros daquele neurônio multiplicada pelos deltas *D* de todos os registros daquele neurônio e pela taxa de aprendizado ta , sendo essa um parâmetro utilizado para determinar a intensidade da mudança do peso a cada iteração.

$$w_{ij}(t+1) = w_{ij}(t) + ta * \sum_{i=1}^{n} x_i * D_i \quad (20)$$

Para exemplificação dos passos descritos acima, é possível observar o fluxograma da Fig. 16, que descreve o funcionamento do *backpropagation*.

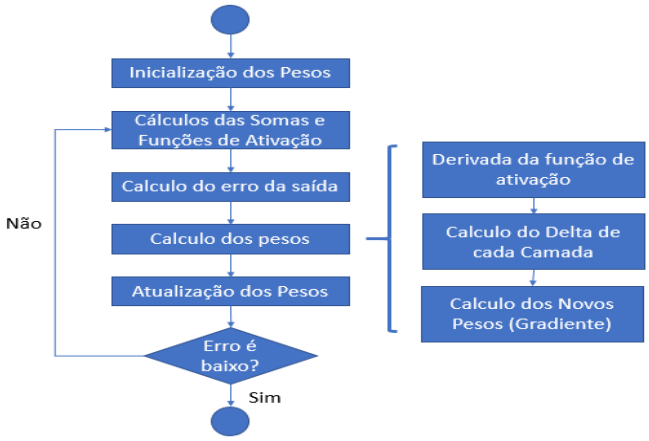

Fig. 16. Fluxograma do algoritmo de treinamento Backpropagation.

V. RESULTADOS

Nessa sessão serão demonstrados os resultados obtidos durante o trabalho, baseados nas hipóteses de desenvolvimento abordadas na proposta. Primeiramente será exibida a planta construída para realizar os testes, posteriormente será demonstrada a rede neural que prediz o comportamento da planta real, seguido da obtenção dos parâmetros do PI clássico, e por último, a rede neural que simula um controlador PI-neural.

A. *Planta construída para os ensaios*

um dos objetivos do trabalho foi utilizar as RNAs em um ambiente físico, buscando verificar o comportamento da mesma em uma situação onde existem ruídos, imperfeições de leitura e controle, tempos de processamento e aquisição dos sinais, além de atritos mecânicos. Essas variações estão presentes em praticamente todos sistemas físicos, sendo difíceis de serem modeladas matematicamente e incluídas em simulações, o que afetaria as análises sobre o comportamento das RNAs as quais esse trabalho fará.

A estrutura da planta foi confeccionada com o auxílio do laser, no qual foram cortadas as peças de MDF (placa de fibra de média densidade do inglês *Medium-density fiberboard)* de 6 mm com encaixes, as quais foram unidas com cola, tornando a estrutura rígida. Após, o motor DC de 12 V foi fixado sobre a plataforma, com o disco de 16 ranhuras, alinhando-o com o sensor de interrupção óptica responsável por gerar o feedback. Foram testadas diferentes configurações de ranhuras, na qual a utilização de 16 delas apresentou resolução que permitiu manter constante a leitura de velocidade com um PWM fixo. Na Fig. 17 é possível observar uma visão frontal da planta construída, com destaque para o disco de leitura.

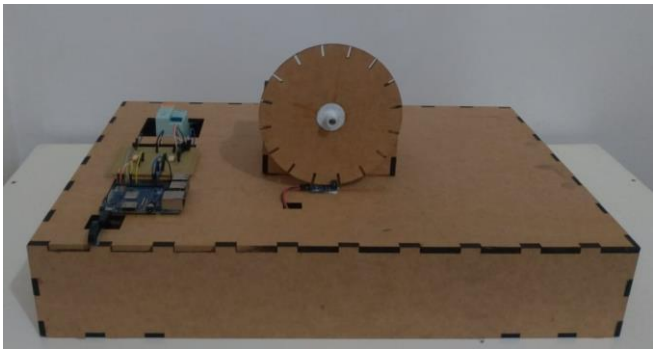

Fig. 17. Visão frontal da planta construída

O controle de potência foi realizado por meio da placa de potência BTS7960, demonstrando tempo de resposta adequado, sendo utilizada frequência de 300 Hz no PWM, limitação imposta pelo processamento do *Raspberry PI* 3 B+, a qual não demonstrou interferir no controle de potência. O interfaceamento entre o sensor e controlador de potência, com o *Raspberry PI* 3 B+, foi realizado por meio do optoacoplador 4N25, trazendo maior segurança ao circuito. Na Fig. 18 tem-se uma visão superior da planta, com destaque para o *Raspberry PI* 3 B+ e também para a placa de condicionamento de sinal.

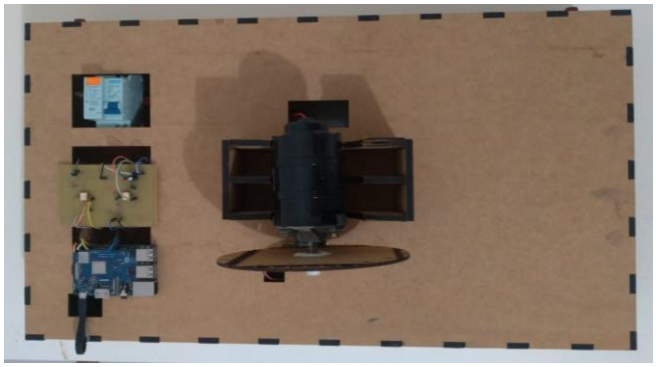

Fig. 18. Visão superior da planta construída

Devido a estrutura realizada em MDF e o motor utilizado, o qual não foi projetado para trabalhar constantemente em alta rotação, foi limitado via *software* a porcentagem máxima do PWM em 80%, visto que acima desta, a estrutura apresentava alta vibração mecânica.

Outro objetivo do trabalho é facilitar academicamente o acesso ao estudo de redes neurais no controle e simulação de plantas físicas, e por isto, ela foi desenvolvida para facilitar o manuseio. Na parte de proteção, conta com um IDR (Interruptor diferencial residual) de 10 A na entrada da alimentação, como visto na Fig. 18. Ainda conta com dois fusíveis para proteção na entrada e saída da fonte de alimentação do motor, como visto na Fig. 19. Todo o sistema é móvel, contando com encaixe rápido para alimentação em 220V. A conexão ao *Raspberry* PI 3 B+ ocorre por meio de um acesso VNC (*Virtual Network Computing*).

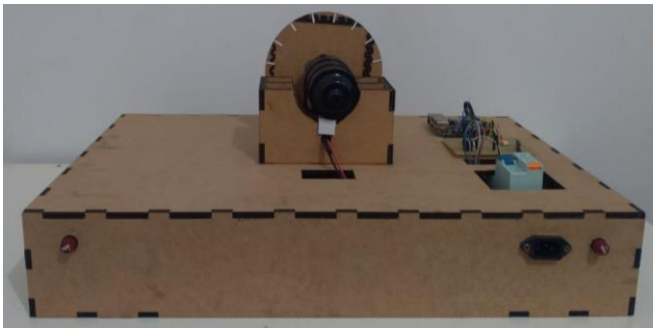

Fig. 19. Visão posterior da planta construída

B. Simulação da planta real

O primeiro passo proposto no trabalho foi a obtenção de uma rede neural artificial que predissesse o comportamento da planta física, objetivando com isso compreender como a escolha de variáveis de entrada e saída da rede neural, bem como os dados de treinamento, influenciavam na performance da mesma. Também validou-se com estes testes o funcionamento do controle de potência e aquisição do feedback vindos da planta.

A obtenção dos dados de treinamento foi realizada aplicando *steps* crescentes de PWM, e consequentemente de tensão, na planta física, e após foram realizados treinamentos com a RNA preditora da planta, conforme havia sido proposto. Essa abordagem não demonstrou bons resultados, visto que não conseguia-se capturar a dinâmica da planta, principalmente na região de transição. Para contornar esse problema, foram realizados dois ajustes em relação a proposta, o primeiro sendo a mudança da estrutura da rede e o segundo sendo a alteração dos dados para treinamento.

Para capturar melhor a dinâmica da planta na região transitória, foi utilizada uma quantidade maior de entradas defasadas de velocidades, o que permitiu a rede entender qual o comportamento da planta quanto era submetida a variações de PWM. Para tanto, foram adicionadas 6 entradas de velocidades defasadas, indo do tempo t-1, até o tempo t-6, frentes as 2 que haviam sido propostas. As entradas da rede que se referiam ao PWM defasado, foram mantidas em 2 defasagens, pois não mostraram alterações significativas na predição a adição de mais defasagens. Na Fig. 20 é possível verificar a nova estrutura utilizada.

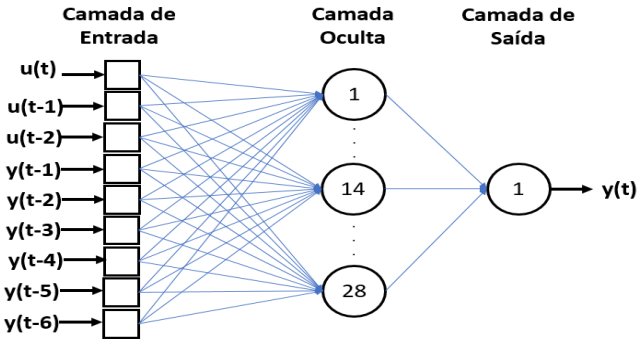

Fig. 20. Estrutura da rede neural artificial simulada final.

Ainda na Fig. 20, é possível reparar que o número de neurônios na camada oculta passou de 3, que haviam sido propostos, para 28, parâmetro determinado empiricamente. Outro ponto que sofreu alteração em relação ao proposto, foi a forma de treinamento, para que a rede pudesse compreender como a intensidade da variação do PWM afeta o comportamento da planta física. Sendo assim, foi utilizada a forma de onda PWM da Fig. 21 na planta física, na qual além das rampas de subida e descida, também aplicou-se *steps* partindo do zero e chegando a valores delimitados. Com essa estrutura de dados, foram obtidos 6429 pares de entrada e saída para treinamento.

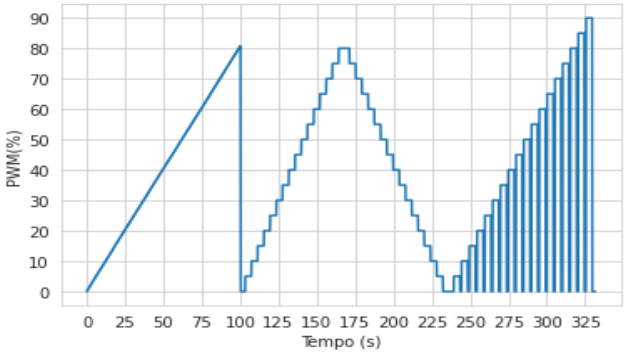

Fig. 21. Forma do PWM usado no treinamento da rede preditora

Foi utilizado o modelo *MLPRegressor*, como já comentado, o qual de forma simplificada permite parametrizar a rede, inserindo quantidade de neurônios na camada oculta, épocas de treinamento, função de ativação, entre outros. Durante todos os testes desse artigo, a classe *MLPRegressor* foi utilizada em conjunto com a biblioteca Pandas para aquisição dos dados de treinamento, os quais estavam armazenados em arquivos .csv gerados no *Raspberry* PI 3 B+ por meio da planta física.

No treinamento da rede neural preditora da planta, o *MLPRegressor* foi parametrizado para funcionar por 140000 épocas, 28 neurônios na camada oculta e utilizando a função de ativação *relu*, a qual obteve melhor desempenho quando comparada a função tangente ou *sigmoid*, principalmente no que diz respeito a velocidade de treinamento. A função *relu* diferentes das demais, não ativa todos os neurônios ao mesmo tempo, tornando a rede mais eficiente e fácil de treinar.

Com esses parâmetros de treinamento e estrutura de rede neural, foi obtido um R^2 de 0,99 e RMSE de 7,58, equivalente a 0,23% da amplitude total dos dados. Este resultado fornecerá uma predição próxima a real, porém devido ao R^2 muito elevado, ocorre o *overfitting*, o qual se caracteriza pela perda da generalização quando os dados de entradas da RNA forem diferentes daqueles utilizados para treinamento.

Para validação da RNA, foram aplicados tanto na RNA preditora quanto na planta física 4 *steps*, sendo eles de 20, 40, 60 e 80% do PWM máximo. Obteve-se os resultados visualizados na Fig. 22, na qual observa-se que a RNA conseguiu assimilar a dinâmica da planta física, seguindo a mesma forma de comportamento transitório. Interessante observar também que a rede neural apresenta um pequeno *overshoot*, indicando que em plantas que apresentem esse comportamento, ela será capaz de o replicar. A velocidade final, porém, mostrou um erro, principalmente em PWM's mais elevados, sendo ocasionado pela forma de treinamento, a qual poderia ser modificada para melhorar essa característica.

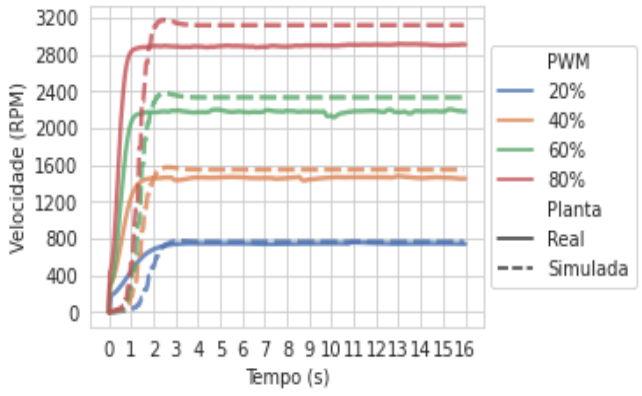

Fig. 22. Comparação entre planta física e simulada

C. *Parametrização do controlador PI*

Visto que obteve-se bons resultados com a planta simulada, ela foi utilizada para encontrar os parâmetros do K_p e K_i para o controlador PI. Esses testes foram feitos empiricamente, não se preocupando com o *overshoot*, mas sim focando-se em encontrar um menor erro de regime permanente com um baixo tempo de subida.

Após alguns testes com o PI clássico, discretizado utilizando o método de *Tustin*, e a RNA preditora da planta, determinou-se que os parâmetros K_p=0,00004, K_i =0,1 e T_s=0,025 permitiam a estabilização da planta, com tempo de subida inferior a 3 segundos. De posse desses parâmetros, aplicou-se o PI clássico na planta física, a fim de verificar seu comportamento. Como esperado, visto que a planta simulada se comportava como a planta física, o controlador projetado obteve controle satisfatório na planta física, não apresentando erro significativo, ou tornando-a instável.

Para validação final, foi aplicado o PI clássico em ambas as plantas, simulada e física, com 3 distintas referências de velocidade. A primeira iniciou-se em 0 segundos até 20 segundos, com uma velocidade de 2000 rpm, a segunda, partiu de 20 segundos até 40 segundos, com uma redução de velocidade para 800 rpm, e por último, de 40 segundos até 60 segundos, aumentou-se a velocidade para 1800 rpm. Esse perfil de velocidades é implementado em todos os testes no trabalho, facilitando as comparações.

Na Fig. 23 é possível verificar a comparação de resposta entre a planta física e a simulada. Como observado, o comportamento é muito semelhante, principalmente em regime permanente, onde com os parâmetros do PI clássico definidos por meio da planta simulada, a planta física também conseguiu ser controlada, apresentando baixo erro. Percebe-se, que na região transitória, existe uma tendência ao mesmo comportamento, exceto na desaceleração de 2000 rpm para 800 rpm, onde fica claro que a rede neural não conseguiu compreender corretamente a dinâmica da planta.

Ainda na Fig. 23, é possível observar que a planta física apresenta um maior *overshoot*, principalmente na mudança de velocidade de 800 rpm para 1800 rpm, no qual a planta simulada não apresenta o mesmo comportamento. Isso ocorre devido a velocidades de processamento, já que a planta simulada está rodando no mesmo processador, enquanto que na planta física existe o *delay* de envio do comando e obtenção do feedback.

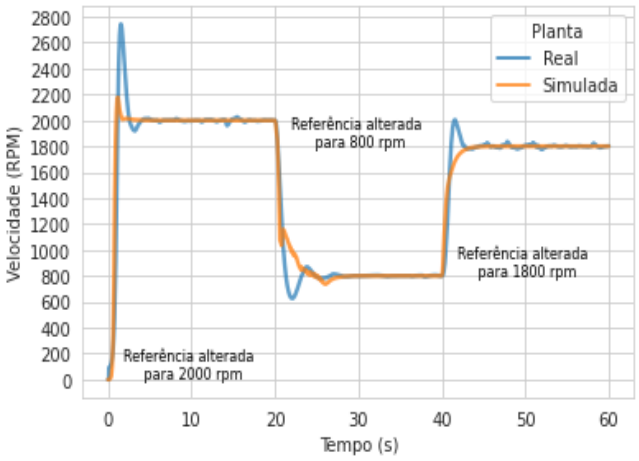

Fig. 23. Comportamento da velocidade do controlador PI clássico aplicado a planta física e simulada. Autor.

Para validação final da planta simulada, pode ser observado na Fig. 24 o comportamento do PWM gerado pelo controle PI clássico quanto aplicado a planta simulada e a física. Percebe-se que o comportamento é muito semelhante entre ambos, reforçando a correta acurácia da RNA preditora. Essa validação se torna importante pois em situações industriais reais, permite compreender com antecedência não

somente o comportamento da planta, mas também as ações de controle necessárias para se alcançar a referência desejada.

Ainda na Fig. 24, é possível verificar o mesmo comportamento obtido na Fig. 22, na qual a planta simulada apresenta uma maior rotação para a mesma porcentagem do PWM. Para compensar isto, o controlador PI na planta simulada, aplicou um PWM menor, demonstrando um comportamento consistente em seu funcionamento.

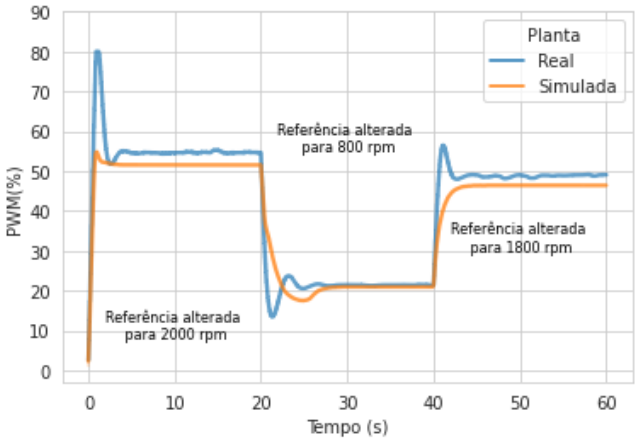

Fig. 24. Comportamento do PWM do controlador PI clássico aplicado a planta física e simulada

D. Desenvolvimento do PI-Neural

Com a validação do funcionamento dos sensores e atuadores da planta construída e estando o PI clássico configurado para atuar com baixo erro na planta física, foi possível obter os dados de treinamento para o desenvolvimento do PI-neural.

Primeiramente se obteve os dados de treinamento com base no PI clássico, no qual foram realizadas 2 rampas de referência de velocidade para o PI clássico seguir, sendo a primeira de 0 até 208 segundos, na qual iniciou-se em 300 rpm, valor mínimo para a planta iniciar o movimento constante, e incrementou-se 50 rpm a cada 4 segundos. Já a segunda rampa, iniciou em 208 e foi até 501 segundos, na qual iniciou-se em 400 rpm, acrescentando 100 rpm a cada 14 segundos, com a característica de sempre voltar a 0 rpm por 1 segundos antes de realizar o incremento.

Na Fig. 25 é possível verificar as duas rampas de referência aplicadas. A última rampa tem o objetivo de demonstrar à rede neural artificial a maior intensidade do PWM que deve ser aplicada quando a diferença entre o valor atual da velocidade e o de referência é grande.

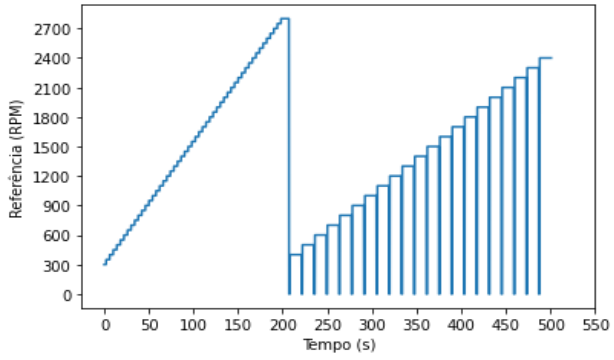

Fig. 25. Forma da referência de velocidade usada no treinamento do PI-Neural

De posse dos dados de treinamento, que totalizaram 45390, a rede foi treinada utilizando a estrutura proposta. Porém, com as velocidades defasadas em t-1, t-2 e a referência de velocidade como entradas, a rede não pode compreender o funcionamento do PI clássico, apresentando oscilações. Por isso, foi reformulada e utilizada a RNA vista na Fig. 26, na qual apenas os valores de PWM defasados e o erro são usados como entrada para a rede.

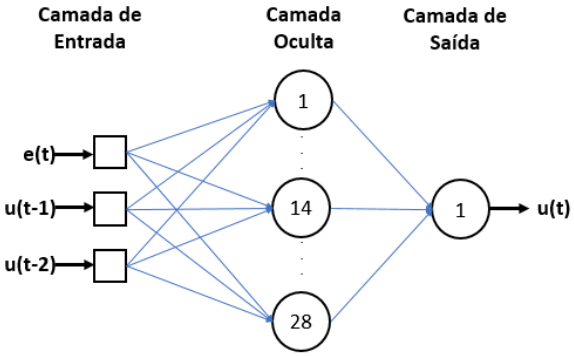

Fig. 26. Estrutura da rede PI-neural final

Essa RNA foi treinada utilizando a função de ativação *sigmoid*, com 50000 épocas e 28 neurônios na camada oculta. Com esses resultados, obteve-se um R^2 de 0,98 e RMSE 2,68, equivalente a 3,35% da amplitude total dos dados, o que trará boa acurácia para a rede PI-Neural. A estrutura vista na Fig. 26 obteve maior sucesso em relação a proposta anteriormente, pois a entrada de velocidade já está implícita no próprio erro, e ao adicionar as entradas de velocidade defasadas, acabava-se inserindo esta informação duplicada, de formas diferentes, interferindo na correta aquisição dos pesos de treinamento da RNA.

Para validação do PI-neural, este controlador e o PI clássico foram inseridos na malha de controle da planta física, na qual utilizou-se os mesmos valores de referência de controle vistos na Fig. 23. O controle por ambos os métodos pode ser visto na Fig. 27.

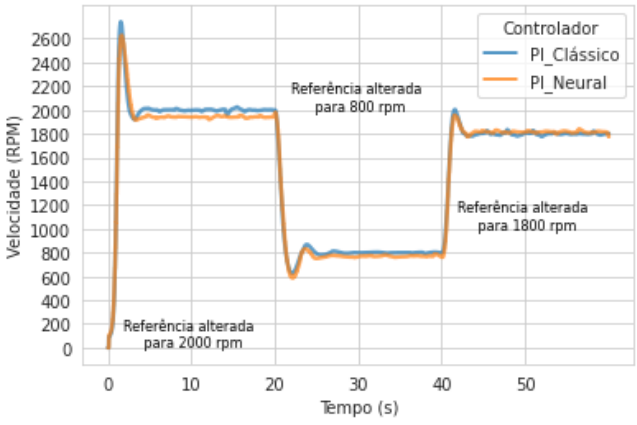

Fig. 27. Comparação de velocidade entre o controlador PI-neural e PI clássico na planta física

Como observado na Fig. 27, o controle PI-neural apresentou controle bastante parecido ao PI-clássico, porém com um erro maior na velocidade de 2000 rpm, causado pela baixa generalização que se obteve em seu treinamento, devido ao R^2 muito alto. Outro ponto a se observar é o *overshoot*, no qual a rede conseguiu o reduzir em relação ao PI clássico,

porém, na mesma proporção que o erro em 2000 rpm, o que não representa uma melhora real.

Observa-se que o comportamento a baixas velocidades, no qual a rede neural além de apresentar um comportamento muito semelhante ao PI clássico, também mostra um tempo de assentamento e erro baixo. Ademais, é interessante notar que a rede neural não foi treinada com o perfil de referência vista na Fig. 27, porém conseguiu capturar a dinâmica do controlador PI clássico, e por isso é observado tamanha semelhança entre os dois controladores.

Na Fig. 28 observa-se a ação PWM no controlador PI-neural e do PI clássico, onde novamente tem-se resultados muito parecidos, demonstrando um correto funcionamento da rede neural. Nos resultados obtidos com estes testes, foi possível verificar que os controladores PI-neurais funcionam, controlando de forma semelhante ao controlador PI clássico. No entanto, as abordagens de estrutura da rede neural e treinamento utilizadas, não conseguem melhora no *overshoot*, e, em alguns casos, tem-se um erro maior.

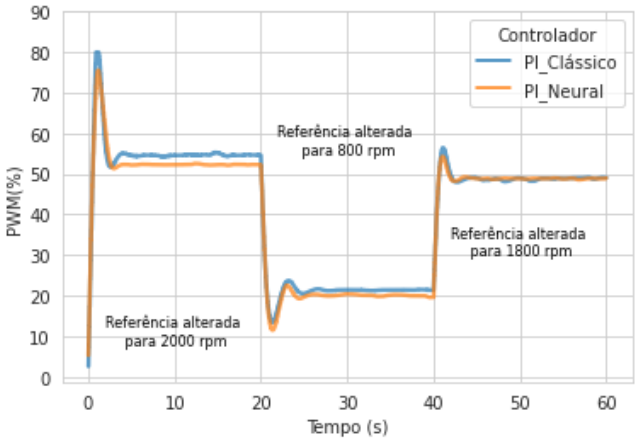

Fig. 28. Comparação do PWM entre o controlador PI-neural e PI clássico na planta física

E. Variação de carga na planta física

Um dos pontos levantados como hipótese, era a capacidade das RNAs em adaptar-se a situações as quais não haviam sido treinadas anteriormente. Para validação, foi adiciona uma carga ao eixo do motor DC, sendo utilizado para isso outro motor, o qual permanecerá desenergizado. Essa situação é muito comum nas industrias, nas quais bombas ou gerados são acoplados mecanicamente a motores.

A nova configuração da planta física, vista na Fig. 29, agora com a adição da carga, foi construída inserindo um motor AC de 220V e 4A, modelo YDM – 45B. Este motor foi acoplado utilizando uma mangueira flexível, o que corrige imperfeições de alinhamentos entre os dois motores, a qual foi presa com braçadeiras aos eixos. O novo motor, por estar desenergizado e não possuir imãs permanentes, não criará campo magnético que possa interferir na dinâmica da planta, porém, seu atrito mecânico e principalmente sua inércia irão interferir no comportamento da mesma.

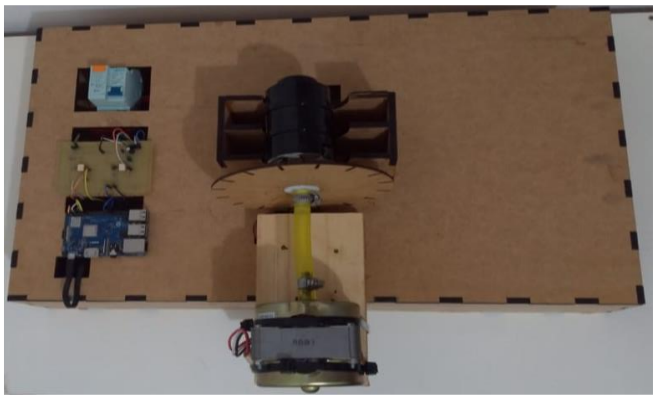

Fig. 29. Planta física com adição da carga

A nova configuração irá alterar o comportamento em relação as condições que o PI clássico e o PI-neural haviam sido projetados e treinados respectivamente, permitindo comparar sua performance frente a alteração da dinâmica da planta. Para obtenção dos dados, o mesmo perfil de velocidade utilizado na Fig. 27 será mantido.

Na Fig. 30 é possível observar o comportamento do controle PI clássico e PI-Neural, na qual percebe-se que o primeiro conseguiu um melhor resultado, muito semelhante quando aplicado a planta sem carga, porém com menor *overshoot*, visto que a carga torna o sistema mais lento. O PI-neural, no entanto, apesar de apresentar um *overshoot* menor que o da planta sem carga, não conseguiu anular o erro, principalmente em 1800 RPM, na qual manteve a planta estável, mas com erro considerável, quando comparado ao PI clássico.

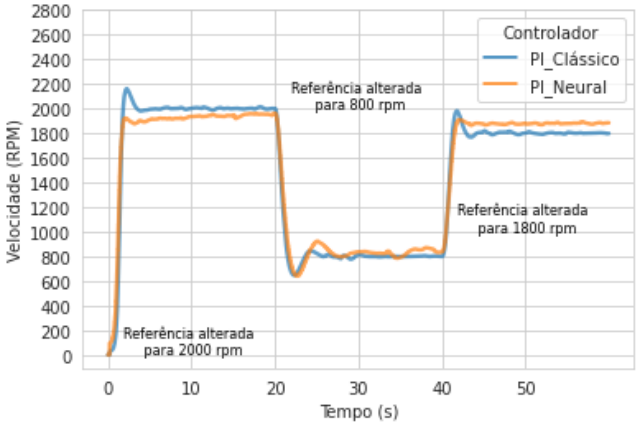

Fig. 30. Comparação de velocidade entre o controlador PI-neural e PI clássico na planta física com adição de carga

Os resultados do teste com carga na planta física demonstram a necessidade de alteração da forma de treinamento, para que se consiga capturar de forma mais precisa as variações. Porém, como observado na Fig. 31 o controlador PI-neural alterou o valor do PWM aplicado quando com carga e sem carga, demonstrando que ele foi capaz de compreender o funcionamento do controle baseado no PI clássico, necessitando apenas alguns ajustes para que consiga se comportar como ele também com variações de carga.

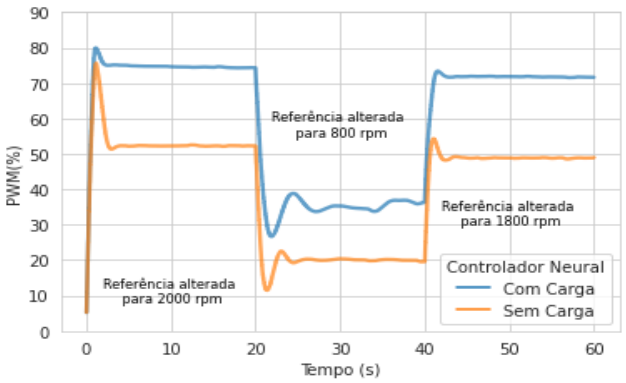

Fig. 31. Comparação de PWM no PI-neural aplicado com carga e sem carga.

F. Trabalhos futuros

o desenvolvimento das RNAs trabalhado nesse artigo abriu caminho para outras pesquisas que podem ser construídas utilizando as mesmas tecnologias, com duas vertentes principais. A primeira diz respeito a exploração das redes neurais artificiais para simulação de plantas industriais, capturando sua dinâmica transitória e em regime permanente. O segundo caminho a explorar-se, é o desenvolvimento de controladores neurais baseados em treinamento otimizado, sem a necessidade de um PI clássico para obter os dados.

A rede neural preditora da planta física elaborada nesse artigo, pode compreender de forma satisfatória o comportamento em regime permanente da planta física, porém a região transitória apresentou divergência no tempo de subida e forma de resposta. Em futuros trabalhos, pode-se analisar outras entradas que melhor representem o comportamento da planta, bem como a forma de treinamento, na qual alterando-se os *steps* e tempos de treinamento, pode-se conseguir melhores resultados.

Essa continuação se torna particularmente interessante pois permitirá ao projetista realizar diversos testes de controle em um ambiente simulado, podendo atingir limites para testes que a estrutura física não permitiria. Além disso, permite ao engenheiro projetar controladores para plantas industriais em funcionamento, sem necessidade de parada de produção, já que a parametrização do controlador pode ocorrer de forma simulada.

Além das aplicações industriais das simulações, o ambiente acadêmico também pode utilizar dessa abordagem, tendo o aluno a oportunidade de projetar controladores para diferentes tipos de plantas industriais, sem custosas plantas físicas. Permite também ao aluno um maior contato com o ambiente industrial que irá enfrentar em sua vida profissional, pois interferências presentes em plantas fabris já estarão inseridas na dinâmica da simulação.

A segunda vertente para trabalhos futuros, é o treinamento da RNA baseado em metodologias inteligentes de controle, e não em um PI clássico, que além de suas limitações, como já comentadas, necessita ser parametrizado, atingindo grande complexidade dependendo da planta.

A abordagem de controladores neurais sem um controlador clássico como base de treinamento, requer maiores estudos quanto ao algoritmo empregado para obtenção dos dados de treinamento, bem como a melhor estrutura de RNA, visando deixá-la com flexibilidade de controle. Embora desafiadora, essa nova abordagem de treinamento alternativa, permitirá uma maior integração entre o controlador neural, que estará atuando diretamente na planta, e as inteligências artificiais que fazem cada vez mais parte da área gerencial das indústrias.

VI. Conclusão

O presente trabalho iniciou a análise da viabilidade da utilização de redes neurais artificiais na simulação de plantas industriais e principalmente no controle de processo em substituição aos controladores PI clássico, que devido a demanda de maior conectividade aos sistemas de produção, não são capazes de atender os novos requisitos, principalmente as variações da produção.

Para essa análise, o processo escolhido foi o controle da velocidade de um motor CC, devido ao fato de conter poucas variáveis no controle e por apresentar não-linearidades, uma das grandes vantagens das RNAs frente aos controladores PI clássicos. Visando um ambiente real para treinamento e ensaios com as RNAs desenvolvidas, uma planta física foi construída, tendo atendido aos requisitos necessários, visto que apresentou estabilidade durante o seu controle e inseriu dinâmicas não-lineares aos ensaios.

A RNA preditora da planta apresentou bom resultado, conseguindo compreender a dinâmica transitória da planta, porém, apresentou erro em algumas velocidades. No entanto, com algumas alterações de treinamento e estrutura, ela poderá ser capaz de apresentar maior precisão.

O comportamento do PI-Neural também se mostrou satisfatório, embora não tenha apresentado melhora de desempenho quando comparado ao PI clássico. Ele conseguiu controlar a planta física com erro baixo, porém, não foi eficaz no controle da mesma quando esta sofreu alteração de carga, o que pode ser explicado principalmente pela estrutura da rede, que leva em consideração apenas o erro e as ações de controle anteriores.

No quesito facilidade de implementação, a construção das RNAs com as bibliotecas atuais, torna dinâmico e rápido o processo, podendo-se alterar os parâmetros da estrutura e treinamento de forma rápida e simples. O próprio conceito da rede neural artificial facilita o desenvolvimento, pois elas conseguem compreender a dinâmica de processos de uma forma natural, principalmente quando comparado as técnicas matemáticas

O trabalho, portanto, demonstrou que as RNAs têm um grande potencial, não apenas no controle de plantas físicas, mas também em simulações das mesmas, o que permitirá futuramente, ambientes totalmente virtualizados para desenvolvimento de novos controladores.

Referências

[1] D. J. Rita, “Controle de processos usando redes neurais artificiais: uma aplicação experimental,” Univaersidade Federal de Santa Catarina, p. 100, 1995.

[2] L. V. R. Arruda, F. Neves-J, M. C. S. Swiech, and M. R. Delgado, “Um método evolucionário para sintonia de controladores pi/pid em processos multivariáveis,” Revista Controle & Automação, p. 17, 2008. [Online]. Available: https://www.sba.org.br/revista/

[3] R. H. Bishop and R. C. Dorf, Sistemas de Controle Modernos, 11th ed. Rio de Janeiro, 2011.

[4] A. B. A. Mamani, “Utilização de redes neurais no controle da velocidade de um veículo experimental,” Universidade Estadual de Campinas, Campinas, p. 122, 2004.

[5] J. Peng and R. Dubay, “Identification and adaptive neural network control of a DC motor system with dead-zone characteristics,” ISA TRANSACTIONS, Fredericton, p. 588, 2011.

[6] L. H. G. POPOFF and M. MAITELLI, “Controle preditivo neural aplicado à processos petroquímicos,” Universidade Federal do Rio Grande do Norte, p. 6, 2009.

[7] Z. YU, Y.-B. Xie, Y.-Y. Jing, and X.-A. LU, “Applying neural networks to pid controllers for time-delay systems,” Fifth International Conference on Machine Learning and Cybernetics, Dalian, 2006.

[8] M. A. Alhanjouri, “Speed Control of DC Motor using Artificial Neural Network,” International Journal of Science and Research, 2015.

[9] V. A. Freire Junior, A. do N. Vargas, and A. Goedtel, “Uma abordagem pi-neural aplicado ao controle de um servo-mecanismo,” Universidade Tecnológica Federal do Paraná.

[10] A. J. Assis, “Identificação e controle de processos não lineares utilizando redes neurais artificiais,” Universidade Estadual de Campinas, Campinas/SP, p. 205, 2001.

[11] Rai; Neerparaj and Rai; Bijay, “Neural Network based Closed loop Speed Control of DC Motor using Arduino Uno.,” International Journal of Engineering Trends and Technology, 2013.

[12] M. Farahani, Bidaki Amir Reza Zare, and Enshaeieh; Mohammad, “Intelligent control of a DC motor using a self-constructing wavelet neural network,” Systems Science & Control Engineering: An Open Access Journal, 2014.

[13] Nise; Norman S., Engenharia de Sistemas de Controle, 6th ed. Rio de Janeiro, 2016.

[14] Carvalho; André and Faceli; Katti, Inteligência Artificial - Uma Abordagem de Aprendizado de Máquina. 2011.

[15] Bezerra ; Felipi Luiz de Assunção, “Desenvolvimento de um controle preditivo baseado em modelo de rede neural artificial em um processo de fermentação contínua de segunda geração,” Universidade Estadual de Campinas, Campinas-SP, p. 63, 2017.

[16] Scikit Learn, sklearn.neural_network.MLPRegressor, https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html (Acesso em: 18 jul. 2021).