Embarque de Rede Neural Recorrente Fenomenologicamente Informada para Controle de Nível em Tanques Esféricos

Silas Henrique Alves Araújo * **Wildson Oliveira dos Santos** *
Leonardo Silva de Souza * **Raony Maia Fontes** *
Márcio André Fernandes Martins *

* *Escola Politécnica, Universidade Federal da Bahia, BA, (e-mail: silasaraujo@ufba.br, wildson.oliveira@ufba.br, lssouza@ufba.br, raony@ufba.br, marciomartins@ufba.br)*

Abstract: The use of Physics-Informed Recurrent Neural Networks (PIRNNs) offers a promising framework for modeling dynamic systems. They combine traditional machine learning with the system's phenomenological model by incorporating differential equations directly into the training process. This work presents the deployment of a PIRNN on a low-cost microcontroller to operate as a virtual analyzer, tested in a hardware-in-the-loop environment for PI level control of a cascaded spherical tanks system. The performance and robustness tests demonstrate reductions of over three times in computation time and the ability to provide information in the absence of measurements. These results suggest that PIRNNs can be an efficient solution for systems requiring real-time control and process analysis applications.

Resumo: O uso de Redes Neurais Recorrentes Fenomenologicamente Informadas (PIRNNs) tem se mostrado promissor na modelagem de sistemas dinâmicos. Elas combinam o aprendizado de máquina tradicional com o modelo fenomenológico do sistema incorporando equações diferenciais diretamente no processo de treinamento. Este trabalho apresenta o embarque de uma PIRNN em um microcontrolador de baixo custo para operar como analisador virtual, testado em ambiente *hardware-in-the-loop* para o controle PI do nível de um sistema de tanques esféricos em cascata. Os testes de desempenho e robustez demonstram reduções superiores a três vezes no tempo de cômputo e a capacidade de fornecer informações na ausência de medições. Esses resultados sugerem que a PIRNN pode ser uma solução eficiente para sistemas que exigem controle em tempo real e aplicações de análise de processos.

Keywords: Physics-Informed Recurrent Neural Networks; Simulation of Nonlinear Systems; Low-Cost Microcontrollers; Virtual Analyzer; Hardware-in-the-loop.

Palavras-chave: Redes Neurais Recorrentes Fenomenologicamente Informadas; Simulação de Sistemas Não Lineares; Microcontroladores de Baixo Custo; Analisador Virtual; Hardware-in-the-loop.

1. INTRODUÇÃO

Um elemento crítico para o funcionamento de um sistema de controle com realimentação é a qualidade das medições realizadas. Sensores precisos e confiáveis são essenciais para fornecer dados em tempo real ao controlador, permitindo ajustes corretos no processo. Caso um sensor apresente falhas, como leituras imprecisas ou atrasos, o controle pode ser prejudicado, levando o sistema a operar inadequadamente. Em situações extremas, se os sensores deixarem de fornecer dados ao controlador, o sistema pode entrar em malha aberta, o que pode resultar em instabilidade, baixa eficiência e até riscos operacionais.

O uso de analisadores virtuais tem se tornado uma solução viável para esse problema. Também conhecidos como sensores de software (*soft sensors*) e sensores virtuais, eles são ferramentas capazes de estimar valores de variáveis não medidas diretamente, utilizando dados de sensores físicos e relações matemáticas entre variáveis conhecidas (Martin et al., 2021). Alguns autores ainda incluem observadores de estado nesse conjunto de ferramentas (Kadlec et al., 2009; Alvarado-Santos et al., 2022), embora essa associação não seja consensual. Apesar das variações conceituais, o objetivo comum dessas abordagens é fornecer estimativas confiáveis de variáveis mesmo quando determinadas medições diretas não são possíveis devido a limitações técnicas ou falhas de sensores, garantindo a continuidade e a precisão do controle de processos (Silva, 2023).

Entretanto, dependendo do modelo utilizado, a execução de um analisador virtual pode demandar recursos computacionais significativos. Isso pode ser um problema para sistemas embarcados de baixo custo, como o Arduino UNO, onde soluções eficientes são essenciais devido às limitações de memória e poder de processamento.

Uma abordagem alternativa de implementar um analisador virtual é o uso de redes neurais artificiais (*Artificial*

Neural Networks — ANNs). Essas são consideradas aproximadores universais de funções (Augustine, 2024) e têm o potencial de serem mais rápidas que os métodos numéricos convencionais por terem uma arquitetura baseada em operações simples e paralelizáveis. Em adendo, as Redes Neurais Fenomenologicamente Informadas (*Physics-Informed Neural Networks* — PINNs) foram introduzidas por Raissi et al. em 2017. Elas são uma metodologia de treinamento que integra o modelo fenomenológico do problema diretamente na função custo penalizando previsões que violam as relações físicas conhecidas. Isso é especialmente vantajoso para simular sistemas complexos e com poucos dados experimentais, por combinar a flexibilidade das ANNs com o rigor das leis de conservação do sistema em análise (Raissi et al., 2019).

Em 2022, Nicodemus et al. integraram uma PINN a um esquema de controle de um braço robótico com múltiplas articulações. No ano seguinte, Zheng et al. introduziram as Redes Neurais Recorrentes Fenomenologicamente Informadas (*Physics-Informed Recurrent Neural Networks* — PIRNNs). Uma abordagem que aprimora a capacidade das PINNs de capturar dependências temporais, sendo particularmente úteis para modelagem de sistemas dinâmicos. Como exemplo de processo químico, eles utilizaram a PIRNN para o controle de um reator perfeitamente agitado (Continuous Stirred-Tank Reactor — CSTR) (Zheng et al., 2023). No entanto, essa implementação foi realizada em um ambiente computacional convencional, sem as restrições de hardware impostas por sistemas embarcados.

Seguindo as ideias apresentadas por Zheng et al. (2023), este trabalho explora o uso de PIRNNs para o controle de sistemas dinâmicos, com foco na implementação em sistemas embarcados de baixo custo. Para demonstrar a aplicabilidade e o desempenho das PIRNNs, um sistema de dois tanques esféricos em cascata é utilizado como exemplo. Especificamente, visa-se utilizar uma PIRNN para manter o controle PI desse sistema e avaliar a redução no tempo de cômputo e a qualidade das previsões em comparação com os métodos numéricos tradicionais. Além disso, este trabalho apresenta uma abordagem para o embarque de analisadores virtuais baseados em PIRNNs em microcontroladores de baixo custo, permitindo a manutenção do controle mesmo em falhas de sensores.

2. METODOLOGIA

2.1 Descrição do Sistema

O sistema analisado neste trabalho é composto por dois tanques esféricos interligados sujeitos à pressão atmosférica. A configuração do sistema é ilustrada na Figura 1, onde h_1 e h_2 representam os níveis dos tanques em centímetros, q_{in} a vazão volumétrica de entrada no primeiro tanque (em $\text{cm}^2 \cdot \text{s}^{-1}$) e SP o *setpoint* para o nível do tanque 2. Os símbolos LT e LIC referem-se a instrumentação do sistema, sendo respectivamente, o transmissor de nível e o controlador de nível.

A modelagem matemática dos tanques baseia-se no princípio de conservação de massa, do qual se obtêm (1a) e (1b), que descrevem a variação temporal do nível de líquido em cada tanque.

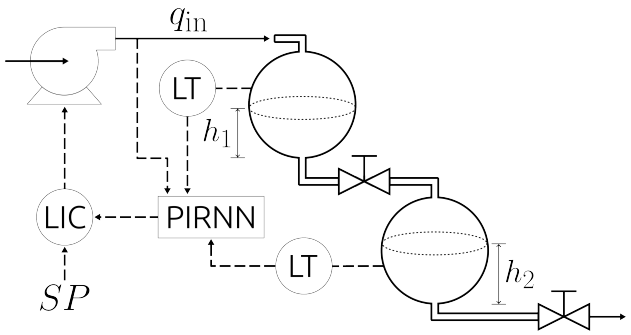

Figura 1. Representação esquemática do sistema de dois tanques esféricos.

$$\frac{\mathrm{d}h_1(t)}{\mathrm{d}t} = \frac{q_{\text{in}} - \alpha_1 s_1 \sqrt{2gh_1}}{\pi(2Rh_1 - h_1^2)} \tag{1a}$$

$$\frac{\mathrm{d}h_2(t)}{\mathrm{d}t} = \frac{\alpha_1 s_1 \sqrt{2gh_1} - \alpha_2 s_2 \sqrt{2gh_2}}{\pi(2Rh_2 - h_2^2)} \tag{1b}$$

O valor e a definição de cada constante utilizada na modelagem estão descritos na Tabela 1.

Tabela 1. Constantes do sistema de equações diferenciais formado por (1a) e (1b).

Símbolo	Descrição	Valor
α_1	Coeficiente de fluxo da válvula 1	0,56
α_2	Coeficiente de fluxo da válvula 2	0,30
s_1	Área da seção de saída do tanque 1	0,50 cm^2
s_2	Área da seção de saída do tanque 2	0,50 cm^2
g	Aceleração da gravidade	980,665 cm/s^2
R	Raio dos tanques	14,85 cm

2.2 Estrutura da Rede Neural Artificial

As Redes Neurais Recorrentes (*Recurrent Neural Networks* — RNNs) são modelos computacionais projetados para processar dados sequenciais, sendo amplamente utilizadas em aplicações que envolvem séries temporais, reconhecimento de padrões e modelagem dinâmica de sistemas. Diferentemente das ANNs tradicionais, que tratam cada entrada de forma independente, as RNNs possuem conexões recorrentes que permitem reter informações de estados anteriores, possibilitando a captura de dependências temporais nos dados (Mienye et al., 2024).

No entanto, as RNNs compartilham algumas das limitações das redes neurais tradicionais. Seu desempenho depende fortemente da disponibilidade de um grande volume de dados representativos, o que nem sempre é viável. Outro ponto crítico é que esses modelos podem gerar previsões inconsistentes com o conhecimento científico disponível, pois não consideram explicitamente as leis físicas que regem o sistema (Karniadakis et al., 2021).

Para superar essas limitações surgiram as PIRNNs. Uma variação das PINNs que combina a capacidade das PINNs de incorporar conhecimento físico com a habilidade das RNNs de processar dados sequenciais e capturar dependências temporais eficientemente (Zheng et al., 2023). Elas funcionam de maneira semelhante às RNNs tradicionais, mas com a diferença de que, além do custo associado aos dados, a função de treinamento também considera

o erro relativo ao problema físico. Dessa forma, a rede neural é penalizada sempre que suas previsões divergem das relações físicas conhecidas.

A PIRNN utilizada neste trabalho atua como um modelo dos dois tanques apresentados na Figura 1. A rede recebe como entrada os dois valores anteriores de três variáveis: os estados do sistema (h_1, h_2) e a vazão de entrada q_{in}. Sua arquitetura, ilustrada na Figura 2, é composta por uma camada recorrente do tipo *Elman* (Elman, 1990) seguida por duas camadas totalmente conectadas, todas utilizando a função de ativação tangente hiperbólica (σ) e com 32 neurônios cada. Por fim, uma camada linear projeta o resultado para duas saídas, correspondentes aos próximos valores dos níveis dos tanques.

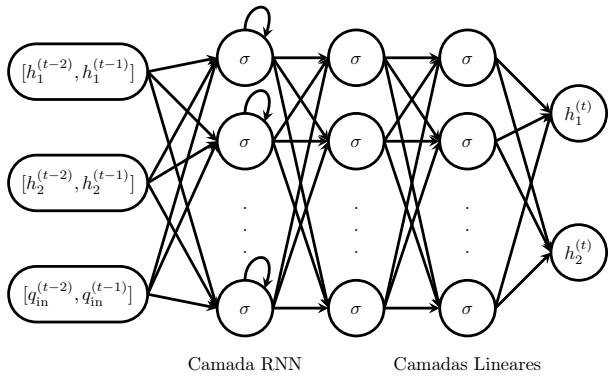

Figura 2. Diagrama da PIRNN utilizada.

Para treiná-la, a função custo utilizada é composta por dois termos: o primeiro, L_{EDOs}, corresponde ao erro relacionado às equações diferenciais do problema físico, enquanto o segundo, L_{data}, refere-se ao erro entre as previsões da rede e os dados observacionais. A função custo total L está expressa em (2) em que ω_1 e ω_2 são coeficientes ajustáveis que determinam a importância relativa de cada termo na função custo. Neste trabalho, ambos os coeficientes foram fixados em $\omega_1 = \omega_2 = 1$, atribuindo pesos iguais ao erro da física e ao erro dos dados.

$$L = \omega_1 \cdot L_{\text{EDOs}} + \omega_2 \cdot L_{\text{data}} \tag{2}$$

O primeiro termo, L_{EDOs}, é dado por (3), na qual $h_{1,\text{prev}}$ e $h_{2,\text{prev}}$ são os valores dos níveis dos tanques previstos pela rede neural artificial. Enquanto f_1 e f_2 são os resíduos relacionados ao sistema de equações diferenciais formado por (1a) e (1b).

$$\begin{aligned} L_{\text{EDOs}} = \frac{1}{N}\sum_{i=1}^{N} f_1(t^i, h^i_{1,\text{prev}})^2 + \\ + \frac{1}{N}\sum_{i=1}^{N} f_2(t^i, h^i_{1,\text{prev}}, h^i_{2,\text{prev}})^2 \end{aligned} \tag{3}$$

O resíduo f_1, relacionado ao primeiro tanque, é definido em (4). E o resíduo f_2, relacionado ao segundo tanque, é definido por (5).

$$f_1(t, h_1) = \frac{\mathrm{d}h_1}{\mathrm{d}t} - \frac{q_{\text{in}} - \alpha_1 s_1 \sqrt{2gh_1}}{\pi(2Rh_1 - h_1^2)} \tag{4}$$

$$\begin{aligned} f_2(t, h_1, h_2) = \\ \frac{\mathrm{d}h_2}{\mathrm{d}t} - \frac{\alpha_1 s_1 \sqrt{2gh_1} - \alpha_2 s_2 \sqrt{2gh_2}}{\pi(2Rh_2 - h_2^2)} \end{aligned} \tag{5}$$

As derivadas temporais de ambos os resíduos são calculadas por meio do método das diferenças finitas com três pontos, considerando os dois pontos fornecidos para a rede e o ponto previsto pela rede.

O segundo termo, L_{data}, é expresso por (6), em que $h_{1,\text{data}}$ e $h_{2,\text{data}}$ representam os dados utilizados para o treinamento, os quais foram obtidos por meio de uma simulação realizada com o método numérico de Runge-Kutta de quarta ordem (RK4), utilizando a biblioteca *CasADi*. A simulação abrangeu pouco mais de 33 horas de operação do sistema, com a introdução de perturbações na vazão de entrada, a fim de capturar a dinâmica completa do processo sob diferentes condições operacionais.

$$\begin{aligned} L_{\text{data}} = \frac{1}{N}\sum_{i=1}^{N} \left(h^i_{1,\text{data}} - h^i_{1,\text{prev}}\right)^2 + \\ + \frac{1}{N}\sum_{i=1}^{N} \left(h^i_{2,\text{data}} - h^i_{2,\text{prev}}\right)^2 \end{aligned} \tag{6}$$

O treinamento da rede foi realizado utilizando o otimizador *Adam* no framework PyTorch (Kingma and Ba, 2017; Ansel et al., 2024) em um processo de ajuste progressivo da taxa de aprendizado. Foram realizadas até 10.500 épocas de treinamento no total, iniciando com uma taxa de aprendizado de 0,01 por 500 épocas, seguida de 5.000 épocas com taxa de 0,001 e, por fim, 5.000 épocas com taxa de 0,0001. Além disso, foi empregado um mecanismo de *early stopping*, interrompendo o treinamento caso não houvesse melhora na função custo por 1.000 épocas consecutivas. Para otimizar a inferência em tempo de execução, o modelo treinado foi convertido para o formato ONNX (*Open Neural Network Exchange*), e os testes de velocidade foram realizados com o framework *ONNX Runtime* (ONNX Runtime developers, 2021).

2.3 Implementação no Arduino

O Arduino é uma plataforma de prototipagem amplamente utilizada por programadores devido ao seu baixo custo e facilidade para desenvolver. A plataforma conta com uma série de placas diferentes, cada uma com características específicas, para atender a diversas necessidades e aplicações. Entre as placas mais populares, destaca-se o Arduino UNO, uma das primeiras e mais acessíveis opções disponíveis e que foi a escolhida para esse trabalho (Hughes, 2016).

No entanto, o Arduino é bastante limitado em termos de capacidade de memória e processamento. O modelo UNO possui 32 kilobytes de memória flash e apenas 2 kilobytes de memória RAM, o que dificulta bastante a tarefa de implementar algoritmos mais complexos, como ANNs que possuem muitas camadas. Diante desse cenário, foi necessário realizar um *deploy* manual da rede neural artificial. As equações que descrevem a PIRNN foram programadas diretamente na linguagem C++, sem o uso de bibliotecas

especializadas. Para garantir que o código fosse compatível com as restrições do hardware, foram feitas otimizações para reduzir o uso de memória e armazenamento. Dentre elas, vale a pena destacar o uso da memória flash para armazenamento dos pesos e bias da rede neural por meio da diretiva *PROGMEM* (Margolis et al., 2020). Isso é possível, pois esses valores são constantes, uma vez que foram definidos na etapa de treinamento da rede.

Além disso, também foi implementado um controlador PI no Arduino para realizar o controle básico do sistema, demonstrando a integração entre a PIRNN e o controle. Os valores das constantes do controlador foram definidos como $K_p = 1,5$ cm·s^{-1} e $K_i = 0,0015$ cm·s^{-2}. Esses parâmetros foram ajustados empiricamente, com base na observação do comportamento dinâmico do sistema durante os testes.

Para realizar a comunicação com o Arduíno, foi desenvolvido o *software* TankSim (Figura 3), responsável por simular o funcionamento do sistema, enviar os dados simulados para o Arduino e receber os valores previstos pela rede neural, além de plotar os gráficos comparando os dados simulados com os previstos. A interface conta ainda com um controle deslizante que permite o usuário alterar o valor de SP para o tanque h_2, dois interruptores para suspender o envio dos dados simulados do nível dos tanques para o Arduino e outro controle deslizante para controlar o nível de ruído. A interface foi desenvolvida utilizando o framework `Qt` (`https://www.qt.io/`), que oferece uma ampla gama de componentes e ferramentas para a criação de interfaces.

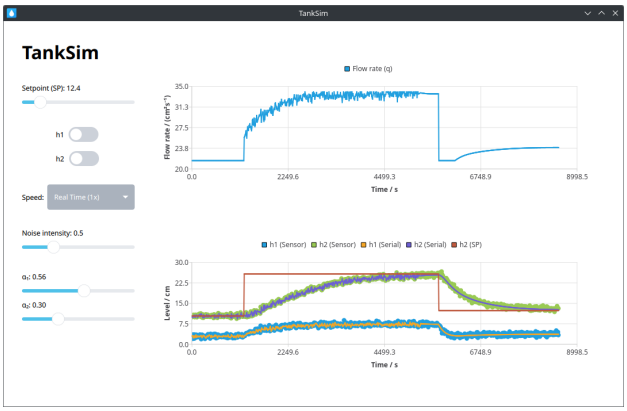

Figura 3. Captura de tela do TankSim.

Este sistema configura-se como um esquema *hardware-in-the-loop* (HIL) (Bacic, 2005), no qual o TankSim simula uma planta virtual e envia as leituras dos sensores para o Arduino. Com base nessas informações, o controlador PI implementado no Arduino calcula a ação de controle, ou seja, o novo valor da vazão q_{in}, visando ajustar o nível do tanque h_2 até o *setpoint* definido pelo usuário. Em seguida, o Arduino utiliza a PIRNN para prever os valores dos níveis de líquido em cada tanque para o próximo intervalo de tempo. Por fim, o TankSim aplica a ação de controle na planta simulada e exibe os resultados na interface.

Quando os dados dos sensores não estão disponíveis, simulando uma falha ou interrupção manual por meio dos interruptores da interface, o Arduino entra em um modo de retroalimentação. Nesse modo, em vez de receber a entrada do nível anterior dos tanques do simulador, a PIRNN passa a utilizar os valores previstos pela própria rede neural como entrada. Esse mecanismo permite que o sistema continue operando mesmo na ausência de dados dos sensores, garantindo a continuidade da previsão e do controle.

3. RESULTADOS E DISCUSSÃO

A PIRNN apresentou previsões altamente consistentes com a simulação de referência realizada pelo *CasADi* com o método RK4. A Figura 4 ilustra os resultados da validação, evidenciando que a PIRNN conseguiu reproduzir com elevada fidelidade a dinâmica do sistema mesmo ao aplicar intensas perturbações na vazão de entrada durante aproximadamente 33 horas de operação.

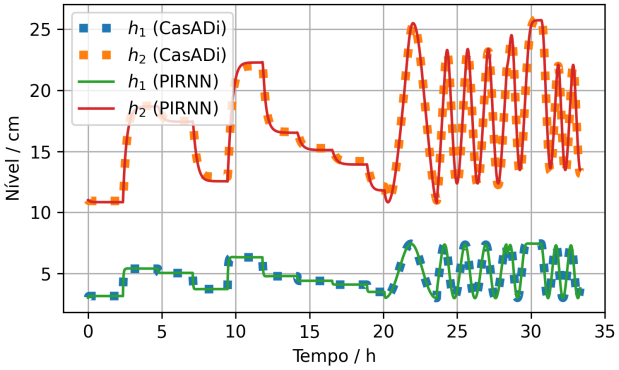

Figura 4. Comparação entre as previsões da PIRNN e o modelo de referência simulado no *CasADi*.

A implementação da PIRNN no Arduino demonstrou um uso eficiente dos recursos computacionais: verificou-se que o programa completo, que inclui a PIRNN, o controlador PI e o código responsável pela comunicação serial com o computador, utilizou 69,1% da memória flash e apenas 15,9% da memória RAM do Arduino. Esses números indicam que existem recursos disponíveis para implementação de modelos mais complexos ou execução de tarefas adicionais, caso necessário.

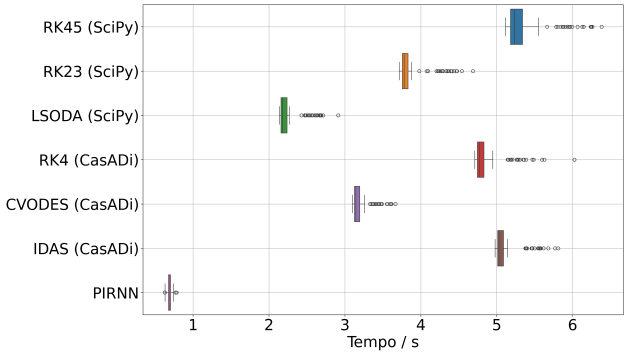

Figura 5. Boxplot dos tempos de execução dos métodos avaliados.

Nos testes de desempenho, a PIRNN demonstrou uma significativa vantagem em termos de velocidade quando comparada com métodos numéricos tradicionais. A comparação foi feita com a implementação dos métodos RK4, CVODES e IDAS da biblioteca *CasADi*, e dos métodos

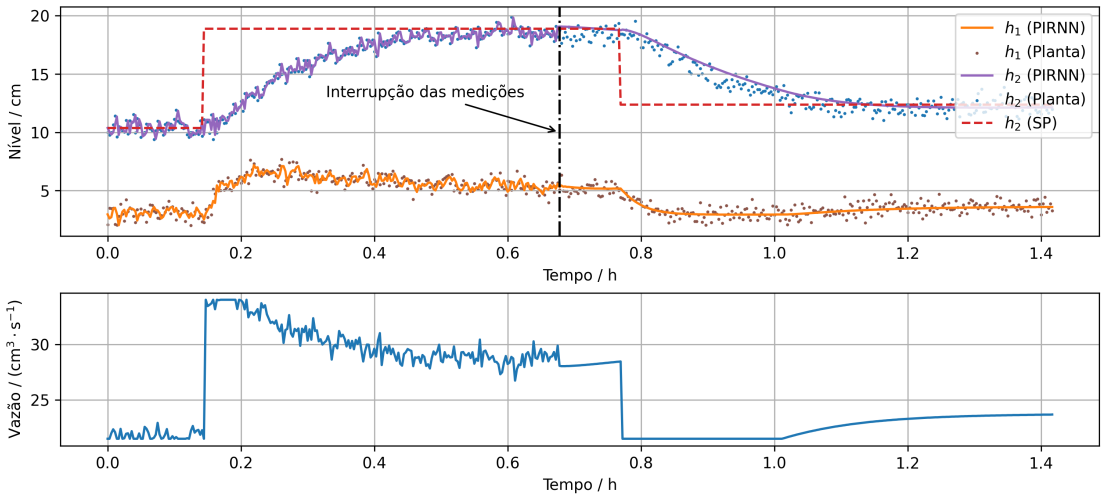

Figura 6. Comparação entre as leituras dos sensores, obtidas a partir dos níveis simulados com adição de ruído, e os níveis previstos pela PIRNN embarcada após uma perturbação do tipo degrau na vazão de entrada. A linha preta ponto-traço vertical indica o instante em que os valores dos sensores deixam de ser enviados ao Arduino, fazendo com que a PIRNN passe a se retroalimentar.

Runge-Kutta de quinta ordem com estimativa de erro de quarta ordem (RK45), Runge-Kutta de terceira ordem com estimativa de erro de segunda ordem (RK23) e LSODA da biblioteca *SciPy*, todos executados no mesmo computador. Para avaliar o desempenho de cada método, foram realizadas 100 execuções de cada algoritmo sob as mesmas condições de entrada utilizadas na validação. A Figura 5 apresenta o boxplot dos tempos de cômputo dos métodos avaliados, evidenciando a superioridade da PIRNN em termos de tempo de processamento.

Nos testes de robustez, foram simulados cenários com ruído nas medições e com ausência completa de medições. A Figura 6 apresenta os resultados obtidos nesses testes. Observa-se que no caso de medições ruidosas, a PIRNN capta as variações geradas pelo ruído, mas com um leve amortecimento, sugerindo uma capacidade de filtragem parcial das perturbações. Quando as medições são interrompidas completamente, a PIRNN passa a se retroalimentar. Nesse cenário, ela continua seguindo com precisão o comportamento esperado dos tanques, mesmo sem a entrada de dados dos sensores. Vale destacar que, mesmo nessas condições adversas, o controlador PI implementado no Arduino continua atuando corretamente no sistema, alcançando o *setpoint* com base apenas nas informações fornecidas pela PIRNN.

4. CONSIDERAÇÕES FINAIS

Neste trabalho, foi demonstrada a utilização de uma PIRNN embarcada em um Arduino UNO atuando como um analisador virtual para medição de nível de um conjunto de tanques.

Os resultados obtidos demonstram que a PIRNN é uma alternativa viável aos métodos numéricos tradicionais, apresentando desempenho superior no tempo de simulação, sem grandes prejuízos à fidelidade das previsões. Outro aspecto relevante é sua robustez, que se mantém mesmo na presença de falhas ou ruídos nas medições. Além disso, sua compatibilidade com plataformas de baixo custo, como o Arduino, torna-a uma solução acessível e eficiente para controle em tempo real, com aplicações potenciais em automação industrial, monitoramento de processos e como analisador virtual.

Contudo, em sistemas dinâmicos reais, é necessário considerar a degradação dos parâmetros do modelo ao longo do tempo, causada pelo desgaste ou envelhecimento dos componentes físicos. Essa degradação pode comprometer o desempenho da PIRNN, tornando necessário o retreinamento periódico do modelo para garantir sua acurácia. Nesse contexto, estudos futuros podem explorar técnicas de aprendizado por reforço, visando a adaptação em tempo real da PIRNN às mudanças dos parâmetros do sistema sem a necessidade de intervenções manuais.

Por fim, pesquisas adicionais podem investigar a implementação de modelos mais complexos, visando expandir a aplicabilidade das PIRNNs em sistemas dinâmicos descritos por equações diferenciais parciais.

AGRADECIMENTOS

Os autores gostariam de agradecer o suporte financeiro da ANP no âmbito do PRH 35.1.

REFERÊNCIAS

Alvarado-Santos, E., Mata-Machuca, J. L., López-Pérez, P. A., Garrido-Moctezuma, R. A., Pérez-Guevara, F., Aguilar-López, R., 2022. Comparative analysis of a family of sliding mode observers under real-time conditions for the monitoring in the bioethanol production. Fermentation 8 (9).

Ansel, J., Yang, E., He, H., Gimelshein, N., Jain, A., Voznesensky, M., Bao, B., Bell, P., Berard, D., Burovski, E., Chauhan, G., Chourdia, A., Constable, W., Desmaison, A., DeVito, Z., Ellison, E., Feng, W., Gong, J., Gschwind, M., Hirsh, B., Huang, S., Kalambarkar, K., Kirsch, L., Lazos, M., Lezcano, M., Liang, Y., Liang, J., Lu, Y., Luk, C. K., Maher, B., Pan, Y., Puhrsch, C., Reso, M., Saroufim, M., Siraichi, M. Y., Suk, H., Zhang, S., Suo, M., Tillet, P., Zhao, X., Wang, E., Zhou, K.,

Zou, R., Wang, X., Mathews, A., Wen, W., Chanan, G., Wu, P., Chintala, S., 2024. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. ASPLOS '24. Association for Computing Machinery, New York, NY, USA, p. 929–947.

Augustine, M. T., 2024. A survey on universal approximation theorems.

Bacic, M., 2005. On hardware-in-the-loop simulation. In: Proceedings of the 44th IEEE Conference on Decision and Control. pp. 3194–3198.

Elman, J. L., 1990. Finding structure in time. Cognitive Science 14 (2), 179–211.

Hughes, J. M., May 2016. Arduino: A Technical Reference. "O'Reilly Media, Inc.".

Kadlec, P., Gabrys, B., Strandt, S., 2009. Data-driven soft sensors in the process industry. Computers & Chemical Engineering 33 (4), 795–814.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., Yang, L., Jun. 2021. Physics-informed machine learning. Nature Reviews Physics 3 (6).

Kingma, D. P., Ba, J., 2017. Adam: A method for stochastic optimization.

Margolis, M., Jepson, B., Weldin, N. R., Apr. 2020. Arduino Cookbook. "O'Reilly Media, Inc.".

Martin, D., Kühl, N., Satzger, G., Mar 2021. Virtual sensors. Business & Information Systems Engineering 63 (3), 315–323.

Mienye, I. D., Swart, T. G., Obaido, G., 2024. Recurrent neural networks: A comprehensive review of architectures, variants, and applications. Information 15 (9).

Nicodemus, J., Kneifl, J., Fehr, J., Unger, B., 2022. Physics-informed neural networks-based model predictive control for multi-link manipulators. IFAC-PapersOnLine 55 (20), 331–336, 10th Vienna International Conference on Mathematical Modelling MATHMOD 2022.

ONNX Runtime developers, 2021. Onnx runtime. `https://onnxruntime.ai/`, version: 1.20.0.

Raissi, M., Perdikaris, P., Karniadakis, G. E., 2017. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. arXiv preprint arXiv:1711.10561.

Raissi, M., Perdikaris, P., Karniadakis, G. E., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics 378, 686–707.

Silva, A., 2023. Analisadores virtuais e controle avançado para uma planta industrial de polietileno linear e avaliação de seus benefícios econômicos. Master's thesis, Universidade Federal da Bahia.

Zheng, Y., Hu, C., Wang, X., Wu, Z., 2023. Physics-informed recurrent neural network modeling for predictive control of nonlinear processes. Journal of Process Control 128, 103005.