



## IMPLEMENTAÇÃO DE PINN EMBARCADA EM AMBIENTE SOFTWARE-IN-THE-LOOP COMO ANALISADOR VIRTUAL PARA UM SISTEMA DE TANQUES ESFÉRICOS

SILAS HENRIQUE ALVES ARAÚJO  
LEONARDO SILVA DE SOUZA, RAONY MAIA FONTES, MÁRCIO ANDRÉ  
FERNANDES MARTINS

# Seção 1

## Introdução

## Objetivos

- Explorar o uso de Redes Neurais Baseadas em Física (PINNs) como uma alternativa aos métodos tradicionais de simulação de sistemas dinâmicos;
- embarcar uma PINN em um microcontrolador de baixo custo.

## Relevância

- Velocidade é essencial em aplicações que exigem alta velocidade e respostas em tempo real, como no Controle Preditivo Baseado em Modelo (MPC).
- Hardware de baixo custo torna a tecnologia viável para mais aplicações, inclusive em contextos industriais e acadêmicos.

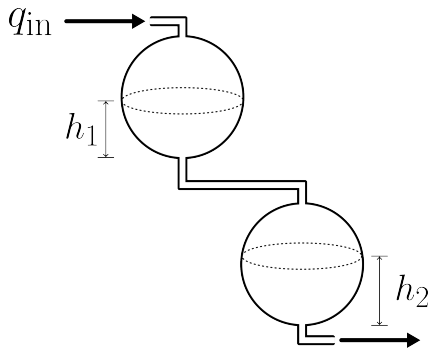


Figura 1: Representação esquemática do sistema de dois tanques esféricos.

Sistema de equações diferenciais dos tanques  
(Wildson 2024)

$$\begin{cases} \frac{dh_1(t)}{dt} = \frac{q_{in} - \alpha_1 s_1 \sqrt{2gh_1}}{\pi(2Rh_1 - h_1^2)} \\ \frac{dh_2(t)}{dt} = \frac{\alpha_1 s_1 \sqrt{2gh_1} - \alpha_2 s_2 \sqrt{2gh_2}}{\pi(2Rh_2 - h_2^2)} \end{cases} \quad (1)$$

Tabela 1: Constantes do sistema de equações 1.

Símbolo	Descrição	Valor
$\alpha_1$	Fator de correção 1	0.56
$\alpha_2$	Fator de correção 2	0.30
$s_1$	Área da seção de saída do tanque 1	$0.50 \text{ cm}^2$
$s_2$	Área da seção de saída do tanque 2	$0.50 \text{ cm}^2$
$g$	Aceleração da gravidade	$980.665 \text{ cm/s}^2$
$R$	Raio dos tanques	$14.85 \text{ cm}$

# Mas, o que são PINNs?

Introduzidas por Raissi, Perdikaris e Karniadakis em 2017, as PINNs são uma forma de integrar princípios físicos ao treinamento de redes neurais. Elas têm o potencial de:

- Acelerar simulações em relação aos métodos numéricos tradicionais;
- melhorar a qualidade das previsões em comparação com redes neurais convencionais.

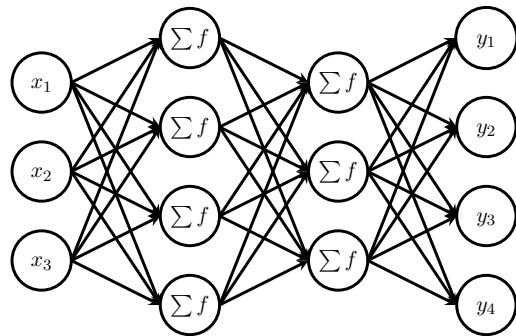


Figura 2: Representação de uma Rede Neural Artificial (ANN).

## Seção 2

### A PINN para Solução de EDOs

A rede neural utilizada consiste em 5 camadas com a função de ativação tangente hiperbólica ( $\tanh$ ) entre elas.

- Entrada:  $x = t$ ;
- camada linear 1: 32 neurônios;
- camada linear 2: 32 neurônios;
- camada linear 3: 32 neurônios;
- camada linear 4: 32 neurônios;
- saída:  $\mathbf{y} = [h_1, h_2]$ .

Totalizando  $32 + 32 + 32 + 32 + 2 = 130$  neurônios.



# Entendendo a função *loss* utilizada

A função *loss* utilizada combina o erro baseado nas equações diferenciais com o erro de dados simulados.

$$\mathcal{L} = w_1 \cdot \mathcal{L}_{\text{EDO}} + w_2 \cdot \mathcal{L}_{\text{IC}} + w_3 \cdot \mathcal{L}_{\text{data}} \quad (2)$$

onde:

- $w_1, w_2, w_3$ : pesos para o cálculo da *loss*;
- $\mathcal{L}_{\text{EDO}}$ : erro das equações diferenciais;
- $\mathcal{L}_{\text{IC}}$ : erro das condições iniciais;
- $\mathcal{L}_{\text{data}}$ : erro dos dados;
- $\mathcal{L}$ : erro total.

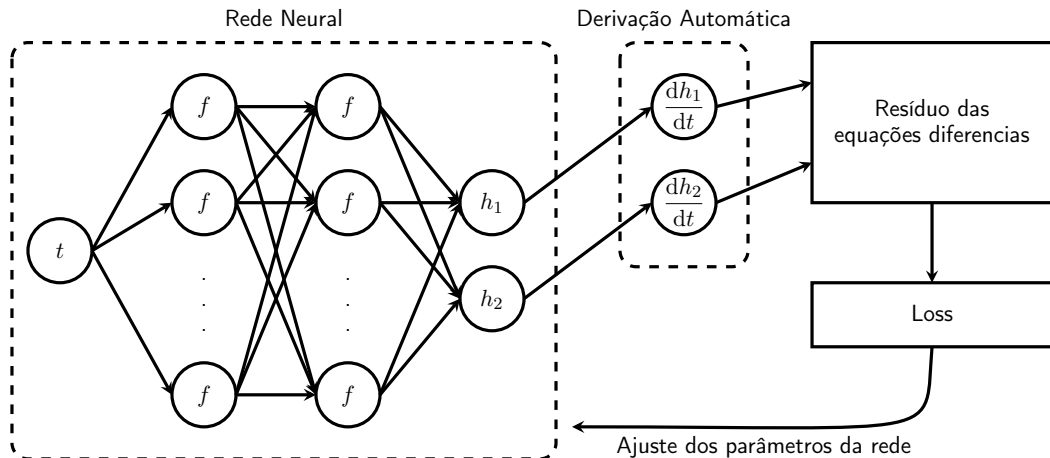


Figura 3: Diagrama da PINN utilizada

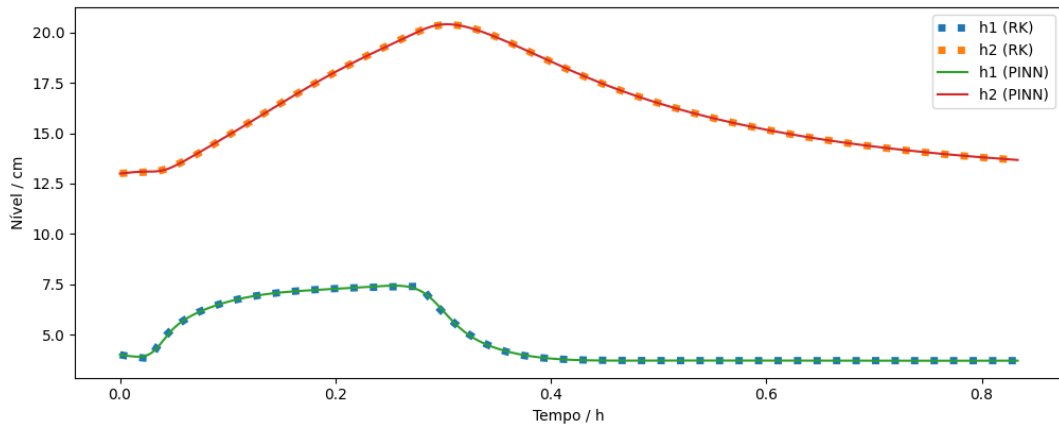
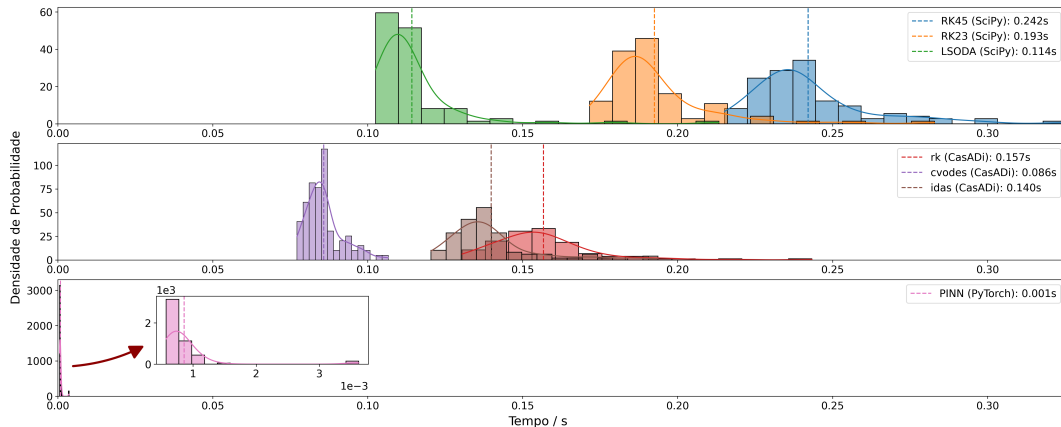


Figura 4: Comparação entre as previsões da PINN e o método numérico (RK45).

# Comparação de Tempos de Execução

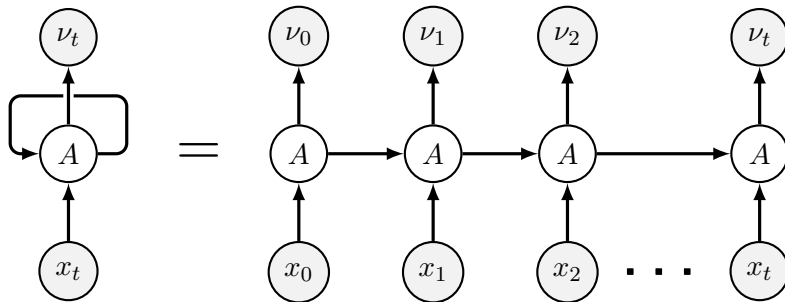


**Figura 5:** Densidade de probabilidade dos tempos de execução dos métodos avaliados. Cada método foi executado 100 vezes; as linhas tracejadas indicam os tempos médios.

## Seção 3

### A PIRNN (Physics-Informed Recurrent Neural Network)

**PIRNNs** incorporam memória recorrente e permitem uma modelagem mais precisa de sistemas dinâmicos (Zheng et al. 2023).



**Figura 6:** Representação de uma RNN nas formas *folded* e *unfolded*. Fonte: Adaptado de Stack Exchange (2019)

**Redes Neurais Recorrentes (RNNs)** usam o *hidden state* ( $\nu_t$ ) para alimentar o mesmo neurônio com as informações dos passos anteriores. Repetindo o processo até obter o estado oculto final (Ansel et al. 2024).

$$\nu_t = \sigma(x_t \cdot W_{i\nu}^T + b_{i\nu} + \nu_{t-1} \cdot W_{\nu\nu}^T + b_{\nu\nu})$$

A PIRNN utilizada consiste em 5 camadas, todas com a função de ativação tangente hiperbólica ( $\tanh$ ), com exceção da saída, que não possui função de ativação:

- Entrada:  $\mathbf{X} = [[h_1^{(t-1)}, h_1^{(t)}], [h_2^{(t-1)}, h_2^{(t)}], [q^{(t-1)}, q^{(t)}]]$ ;
- **camada RNN: 32 neurônios;**
- camada linear 1: 32 neurônios;
- camada linear 2: 32 neurônios;
- camada linear 3: 32 neurônios;
- saída:  $\mathbf{y} = [h_1^{(t+1)}, h_2^{(t+1)}]$ .

Totalizando  $32 + 32 + 32 + 32 + 2 = 130$  neurônios.



## A função *loss* utilizada

A implementação da função *loss* é um tanto diferente: agora não podemos usar a derivação automática, pois o tempo ( $t$ ) não é uma entrada.

Aqui, foi utilizada a derivada regressiva de 3 pontos (os 2 pontos fornecidos para a rede e o ponto previsto).

$$\mathcal{L} = w_1 \cdot \mathcal{L}_{\text{EDO}} + w_2 \cdot \mathcal{L}_{\text{data}} \quad (3)$$

onde:

- $w_1, w_2$ : pesos para o cálculo da Loss;
- $\mathcal{L}_{\text{EDO}}$ : erro das equações diferenciais;
- $\mathcal{L}_{\text{data}}$ : erro dos dados;
- $\mathcal{L}$ : erro total.

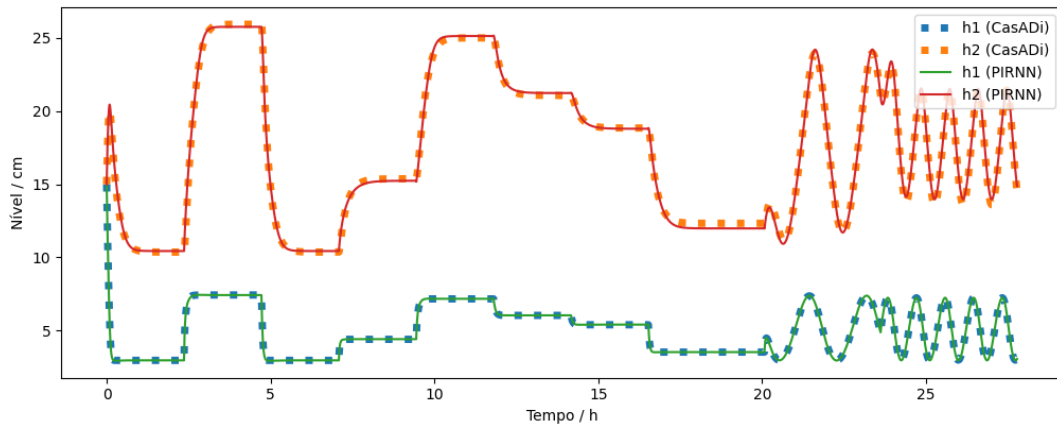
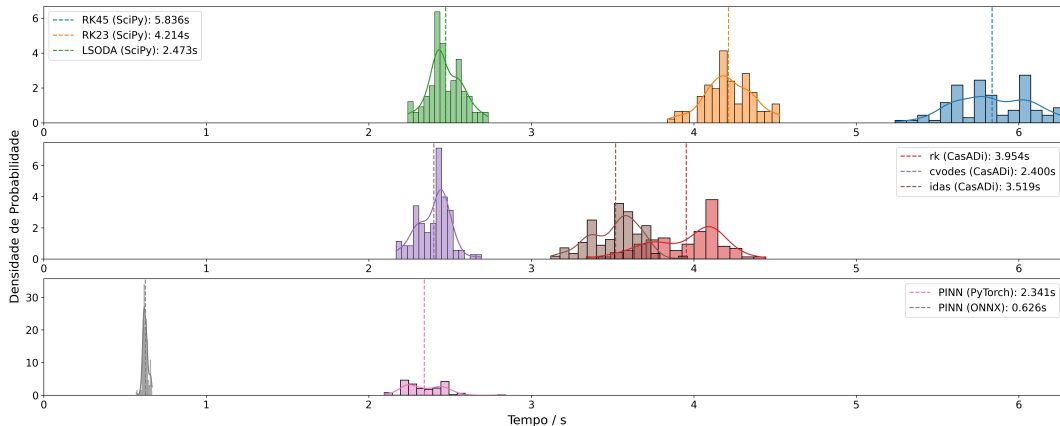


Figura 7: Comparação entre as previsões da PIRNN e o método numérico (RK)

# Comparação de Tempos de Execução



**Figura 8:** Densidade de probabilidade dos tempos de execução dos métodos avaliados. Cada método foi executado 100 vezes; as linhas tracejadas indicam os tempos médios.

# Por que o ONNX Runtime é tão rápido?

O ONNX Runtime é focado em desempenho de inferência e possui uma série de otimizações:

- “constant folding”;
- “redundant node eliminations”;
- “node fusions”.

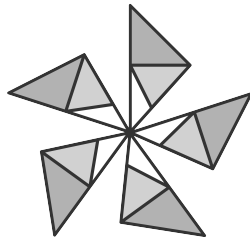


Figura 9: Logo do ONNX Runtime. Fonte: [onnxruntime.ai](https://onnxruntime.ai) (2017)

## Seção 4

### Implementação no Arduino

# O que é o Arduino?

O Arduino é uma plataforma prototipagem amplamente utilizada por programadores devido ao seu baixo custo e facilidade para desenvolver (Hughes 2016). Entre as placas mais populares, destaca-se o Arduino UNO, a escolhida para esse trabalho.

Memória Flash	Memória RAM
32 kilobytes	2 kilobytes

Tabela 2: Memória disponível no Arduino UNO.

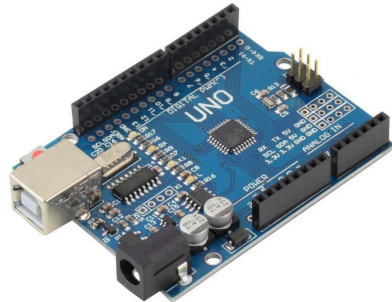


Figura 10: Placa Arduino UNO.  
Fonte: makerhero.com

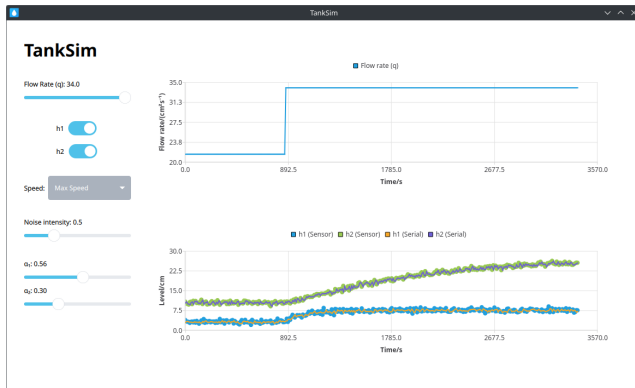


Figura 11: Captura de tela do *TankSim*.

O software *TankSim* foi desenvolvido para:

- simular e manipular o funcionamento de uma planta;
- realizar a comunicação com o Arduino;
- visualizar os valores previstos pela rede neural;

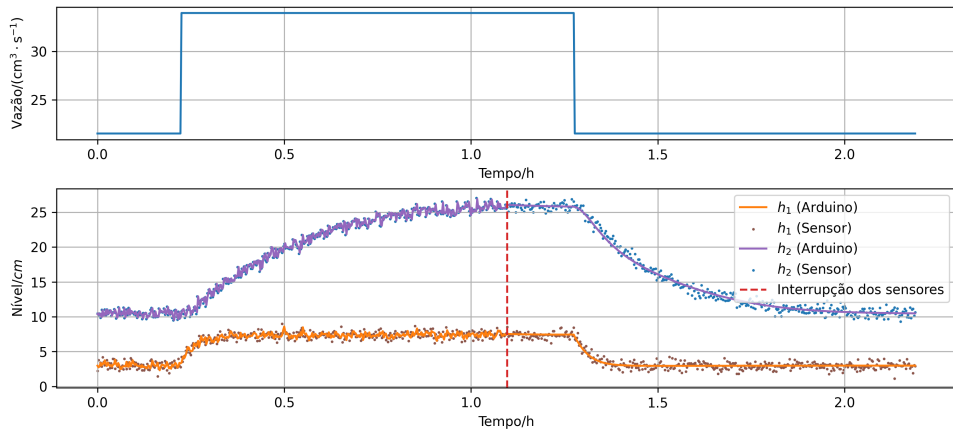


Figura 12: Vazão de entrada  $q_{in}$  e nível dos tanques  $h_1$  e  $h_2$ , previstos pelo simulador e pela PIRNN embarcada no Arduino.








## Seção 5

### Considerações finais

- O Arduino demonstrou capacidade para executar Redes Neurais Recorrentes fisicamente informadas, com ampla margem de recursos (apenas **68,4%** da memória flash e **13,2%** da memória RAM foram utilizados).
- Ao simular ruído nas medições, ou na ausência de medições, a PIRNN mantém a qualidade das previsões relativamente boas, mostrando a robustez do modelo.
- Usando o onnxruntime, a PIRNN foi mais de três vezes mais rápida que os métodos numéricos tradicionais no mesmo computador.

Agradeço ao PRH 35.1 pelo suporte acadêmico e à Agência Nacional do Petróleo, Gás Natural e Biocombustíveis (ANP) pelo suporte financeiro e apoio ao desenvolvimento deste trabalho.



-  Ansel, Jason et al. (2024). “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation”. Em: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. ASPLOS '24. La Jolla, CA, USA: Association for Computing Machinery, pp. 929–947. ISBN: 9798400703850.
-  Hughes, J M (mai. de 2016). *Arduino: A Technical Reference*. “O'Reilly Media, Inc.” ISBN: 9781491934500.
-  Raissi, Maziar, Paris Perdikaris e George Em Karniadakis (2017). “Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations”. Em: *arXiv preprint arXiv:1711.10561*.
-  Wildson (2024). “TCC - Trabalho de Conclusão de Curso”. Trabalho não publicado.
-  Zheng, Yingzhe et al. (2023). “Physics-informed recurrent neural network modeling for predictive control of nonlinear processes”. Em: *Journal of Process Control* 128, p. 103005. ISSN: 0959-1524.