The 9th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2024
3–7 June 2024, Lisboa, Portugal

HEAT CONDUCTION CONTROL USING DEEP Q-LEARNING APPROACH WITH PHYSICS-INFORMED NEURAL NETWORKS

NELSON D. GONÇALVES[1] AND JHONNY DE SÁ RODRIGUES[2]

[1] Institute of Science and Innovation in Mechanical and Industrial Engineering (INEGI)
Rua Dr. Roberto Frias, 400, 4200-465 Porto, Portugal
LAETA—Associated Laboratory of Energy, Transports and Aerospace, 4200-265 Porto, Portugal
e-mail: ngoncalves@inegi.up.pt

[2] Institute of Science and Innovation in Mechanical and Industrial Engineering (INEGI)
Rua Dr. Roberto Frias, 400, 4200-465 Porto, Portugal
LAETA—Associated Laboratory of Energy, Transports and Aerospace, 4200-265 Porto, Portugal
e-mail: jsrodrigues@inegi.up.pt

Key words: Physics-Informed Neural Network; Deep Q-Learning; Model Predictive Control; Heat Transfer.

Summary. As modern systems become more complex, their control strategy can no longer solely rely on measurement data gathered by instrumentation. Instead, it must also incorporate information derived from mathematical models. The complexity of system models can result in excessively long computation times, making the control process impractical. As a solution, surrogate models are implemented to provide estimates within an acceptable timeframe for decision-making purposes. The surrogate model can be a Physics-Informed Neural Network that is used to obtain the system state on the next time step; such information can be used with a Deep Reinforcement Learning algorithm to train a control strategy based on simulations, replacing the need for running direct numerical simulations. On this work, we explore a Deep Q-Learning strategy on 1D heat conduction problem in which temperature distribution feeds a control system to activate a heat source, aiming to obtain a constant, previously defined temperature value. The main goal is to stabilize the bar temperature at the middle point of it without recurring to numerical simulations.

1 INTRODUCTION

In the study of real-world events, differential equations are the main way to describe the complex interactions between a system, which are not mainly linear [1]. Although it is possible to model those systems, their analytical solutions are typically only available for simple cases that are primarily of interest in academic settings. Complex cases that do not have an analytical solution often take a significant period of time to be solved, making it challenging to use in a real-time application to make timely decisions in order to apply control measures. Often, surrogate models and numerical solutions are utilised to address this issue, leading to alternative methods for finding the actual solution.

Data-driven models are used in certain research investigations to make it easier to extract knowledge from temporal and spatial discretizations. As an example, the research by [2] focuses on how well a trained model can work with unstructured grids, random time intervals, and noisy observations to find a solution to a partial derivative equation.

An alternative method for data modelling is by using neural networks [3, 4]. Neural networks are computational models that approximate the behaviour of a system and are composed of multiple layers that combine multiple linear functions to represent data, which may have multiple levels of abstraction [5].

The typical procedure to train a neural network involves defining a dataset containing true inputs and their corresponding true outputs. Then, the goal is minimize the error between true known data and the computed prediction from the neural network model using arbitrary defined parameters. The error is minimised by modifying the neural network parameters using an optimisation strategy.

In not all cases, it is possible to compile a dataset due to impractical reasons such as a lack of sensors or lower amounts of recorded data. So, an alternative method for training a neural network proposed by Lagaris et al. [6] is to use the differential equations that describe the system. The objective is to reduce to zero the residual of evaluating a neural network as the answer to the differential equation. This method does not require true input data with it's corresponding true data; instead, collocation points are defined and evaluated. This training approach has been defined as Physics-Informed Neural Networks (PINN's), and has been successfully implemented in the fields of robotics [7], electrical transformers [8] and control strategies [9].

Despite the concept of PINN's being simple, their training process is not straightforward [3]. This challenge arises when configuring the Neural Network hyperparameters for Physics-Informed Neural Networks (PINNs), when it is necessary to consider the number of layers, nodes per layer, activation functions, and type of loss function. Additionally, PINNs require balancing multiple terms in the loss function, the partitioning of results, and the implementation of time marching techniques to avoid convergence towards undesirable solutions [10, 11, 12, 13, 14, 15].

Once a PINN has been successfully defined, it can be used as a digital twin of a physical system that would provide relevant information to a decision-making process. It could also take information gathered from physical sensors [16] to compute an action or set of actions to change the dynamics of such physical system in a real-time situation [17].

The application of neural networks in the form of digital twins is becoming more common in the context of controlling nonlinear dynamical systems. Those digital twins are employed to estimate the state of the system at any given location and time, acting as virtual sensors. As an implementation, it has been used to estimate flow around obstacles, stabilise vortex shedding, and reduce drag forces, as presented by Fan et al. [18] and Déda et al. [19]. A strategy to define the control system is by using reinforcement learning algorithms, which are a machine learning area that seeks to improve a policy within a model-free framework [20].

The Deep Q-Learning algorithm is one of the algorithms employed in reinforcement learning. Let's consider a scenario where a discrete time controller determines an action. Upon defining and executing such action, it is observed that the state of the system changes. The states of the system are compared to a reference. The outcome of this comparison yields a reward that is associated with the corresponding action. Then, the system defines another possible action, executes it, and generates another related reward. This process is repeated until an objective is achieved. Those rewards are successively appended and used to train the process model. The

Bellman equation is used to evaluate future incomes (Q-value) [21].

This work proposes a strategy for controlling the central point temperature of a rod in which a heat source is applied at one end and, at the other end, it has free convection into ambient air as a heat transfer phenomenon. The control strategy is trained with the Deep Q-learning approach using a Physics-Informed Neural Network to evaluate the state of the system and forecast near-future states. This approach is also compared with a standard control strategy. The problem is presented as a uni-dimensional and continuous medium problem, which implies the control strategy has to deal with the delay effects caused by the control action and the heat loss to the ambient.

2 DEEP-Q LEARNING FRAMEWORK

2.1 System Control

A few components with information flowing between them can represent a straightforward control system, as shown in Fig. 1. In such a representation, the set of actuators is initially configured with some possible action values. The action coming out of the set of actuators results in an impact on the environment, which is composed of the system and its bound's interactions. The system's state is then measured by sensors. The control unit is another component that modifies the system settings to achieve an objective using the data collected by the sensors.

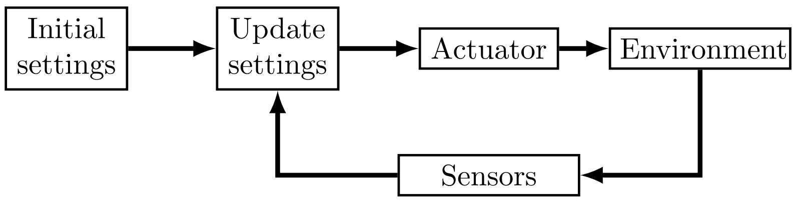
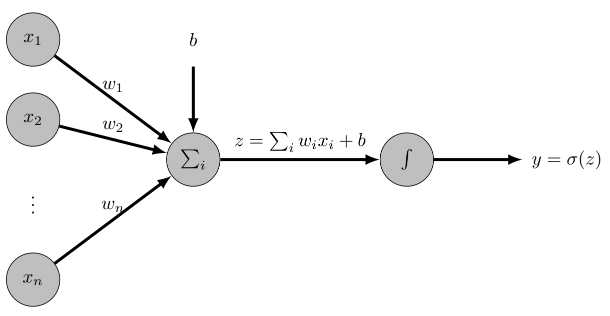


Figure 1: Basic flowchart of a control system

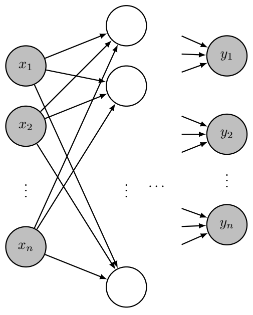
In the system presented in Fig. 1, the two components that are more relevant to this work are the environment and the control system. Regarding the environment, it can be replaced by a simulator to perform predictions. And the control system is to be trained using the environment to define a policy that decides the best system setting for an environmental state, represented by the sensor's measurements.

2.2 PINN

Artificial Neural Networks have been developed since the 1950's inspired by rats cortex functioning [22, 23]. A neural network consists of a number of straightforward computational units called perceptrons (see Fig. 2a). These perceptrons multiply a set of inputs by weights and compute their sum, along with a bias to an activation function.



(a)



(b)

Figure 2: Neural Network basic structure. (a) perceptron. (b) Group of perceptrons forming a Neural Network

In Neural Network a set of inputs can feed a set of perceptrons, forming a layer. A feedforward Neural Network is composed of a series of layers, with the outputs of those layers being used as inputs for the following layer (see Fig. 2b).

The number of inputs and outputs of a Neural Network is defined by the data that is being modelled; however, the number of layers, number of nodes per layer, and activation functions are not so easily defined. Data scientists usually use previous experience to set an initial estimate that is improved in the training stage. As stated before, to train a neural network, it is required to gather input values and their respective true output values to minimize a loss function.

On the other hand, for a PINN training process, the neural network is considered the solution of a differential equation, so a set of collocation points is defined to evaluate not only the neural network but also its respective derivatives. Then, a residual $\mathcal{L}_{\text{Total}}$ is evaluated based on the differential equation that describes the system's dynamics, including its boundary conditions and initial conditions, to be minimized towards zero by an optimization algorithm; a schematic representation of a PINN is shown in Fig. 3. This approach eliminates the required true results for such evaluation points.

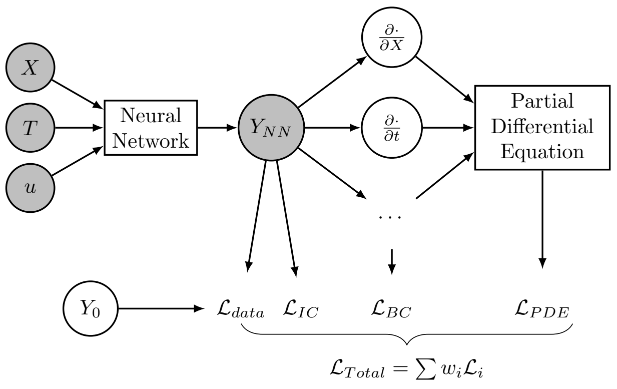

X
T
u
Neural Network
Y_{NN}
$\frac{\partial \cdot}{\partial X}$
$\frac{\partial \cdot}{\partial t}$
$\cdots$
Partial Differential Equation
Y_0
$\mathcal{L}_{data}$
$\mathcal{L}_{IC}$
$\mathcal{L}_{BC}$
$\mathcal{L}_{PDE}$
$\mathcal{L}_{Total} = \sum w_i \mathcal{L}_i$

Figure 3: Basic PINN architecture

2.3 DQL

The objective of the control unit is to decide the best action for a given environment state, i.e., the action that maximizes the reward: at some stage, taking an action from a given state, or during an entire episode, considering the expected future rewards available from the current state up to the episode end when the goal is achieved.

With a Reinforcement Learning strategy, each state, action taken and reward obtained are recorded in a table to support future decisions. When the set of possible states is too big or even infinite, one needs to make some kind of generalisation, and Neural Networks can be used to perform this task.

To find the best action for a given state in Deep Q-learning (DQL), a NN is used instead of a function that looks at a table of past results. The NN figures out the expected future reward (Q-value) by adding up all the future rewards that are weighted with a discount factor, $\sum_{t>0} \gamma^t r_t$, for each of the possible actions [23]. The general information flow for a DQL implementation is shown in Fig. 4.

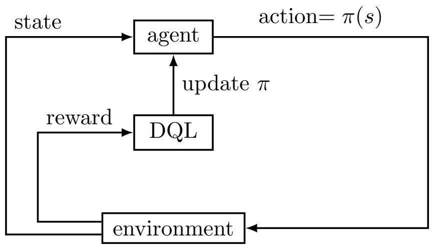


Figure 4: Deep Q-Learning training flowchart

The Q-learning algorithm is based on the Bellman equation, presented in Eq. 1.

$$Q_{new}\left(s_t, a_t\right) = Q_{main}\left(s_t, a_t\right) + \beta\left(r_t + \gamma \max_a Q_{target}\left(s_{t+\delta_t}, a\right) - Q_{main}\left(s_t, a_t\right)\right), \tag{1}$$

where β is the learning rate, r_t denotes the current reward obtained taking the action a_t from state s_t, γ is the discounting rate, a value in $]0, 1[$ used to set the importance of immediate rewards compared with future ones.

During the DQL training algorithm, the estimate of $max_a Q\left(s_t, a_t\right)$ leads to systematic overestimation, introducing a bias in the learning process. A solution to avoid this overestimation is to use two different estimators, Q_{main} and Q_{target}, trained at different stages [24]. Whereas Q_{main} is trained periodically every pre-determined number of iterations, Q_{target} is updated, getting the values of Q_{main} less frequently.

3 CASE STUDY

To make an example of the DQL implementation, a heat transfer process is proposed for a one-dimensional problem.

3.1 Case description

For demonstration purposes, let's suppose a one-dimensional rod of length L, with a natural convection phenomenon occurring at it's left end ($x = 0$) and a heat source at it's right end

($x = L$), see Fig. 5.

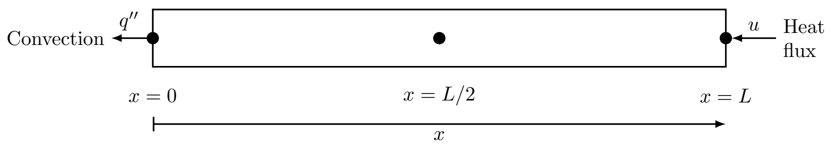


Figure 5: One-dimensional bar as case of study

The goal of the implementation is to keep the temperature of the rod at a defined value, located at the middle point of its length ($x = L/2$). For this problem, the governing equation is the energy conservation relation, as described in Eq. 2.

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}, \tag{2}$$

where T the temperature, t the time, α stands for the thermal diffusivity, and x is the spatial coordinate. For the convection heat transfer phenomena and the heat source, Eq. 3 and Eq. 4 are used, respectively.

$$-k \left[\frac{\partial T}{\partial x}\right]_{x=0} = -h \left(T_{\text{ext}} - [T]_{x=0}\right) \tag{3}$$

$$-k \left[\frac{\partial T}{\partial x}\right]_{x=L} = u, \tag{4}$$

where k stands for the thermal conductivity of the rod, the coefficient h is the convection coefficient, T_{ext} is the external temperature and u stands for the heat intensity that flows towards the rod. As initial conditions, we consider that the entore rod is at T_{ini}.

The governing equation, Eq. 2, can be adimentionalized by the change of variables presented in Tab. 1, by using Eq. 5, Eq. 6 and Eq. 7, obtaining an equivalent equation, Eq. 8.

Table 1: Replacement relations for adimentionalization of the governing equation

Temperature	**Coordinate**	**Time**
$\theta = \dfrac{T - T_{\text{ext}}}{T_{\text{ini}} - T_{\text{ext}}}$ (5)	$\xi = \dfrac{x}{L}$ (6)	$\tau = \dfrac{\alpha}{L^2} t$ (7)

$$\frac{\partial \theta}{\partial \tau} = \frac{\partial^2 \theta}{\partial \xi^2}, \tag{8}$$

Alternatively, it was considered the normalised values of thermal diffusivity $\alpha = 1$, heat transfer coefficient $h = 1$, $T \in [T_{\text{ext}}, T_{\text{ini}}]$ with $T_{\text{ext}} = 0$ and $T_{\text{ini}} = 1$, and $x \in [0, L]$ with $L = 1$. The time step was defined as $\delta_t = 0.1$ s. For the space discretization several meshes were considered, with 5, 11, 21 and 41 nodes.

The x[m] values considered in this work were in $[0, 1]$ with increment 0.1, leading to 11 possible points to the reference case. The mesh studies were performed with 5, 11, 21 and 41 points with the corresponding increments. The initial values were defined considering a degree 3 polynomial defined with the conditions: $[T]_{x=0} = T_0$, $[T]_{x=L} = T_L$, $\left[\frac{\partial T}{\partial x}\right]_{x=0} = T_0 - T_{ext}$ and $\left[\frac{\partial T}{\partial x}\right]_{x=L} = u$, where T_0 and T_L are the temperatures at $x = 0$ and $x = L$, respectively, with values in $[0, 1]$ with increment 0.1, and a random perturbation rnd in $[-0.005, 0.005]$. The time values, t [s], were set in $[0, 2]$ with increment 0.05 and $u \in \{0, 0.25, 0.5, 0.75, 1\}$. Therefore, a training set with $11 \times 21 \times 5 \times 11 \times 11 = 139755$ rows of eleven T_x values was set, and 10% of its values randomly chosen were used to train the PINN.

3.2 Solution

To validation purposes, the finite volumes method is used, keeping the same domain discretization as the PINN method. For the boundary conditions, a zero-volume element is considered to tackle them.

3.3 PINN parameters

The inputs of the proposed neural network typically contain the x coordinate, the evaluation time t, the input u value, and the number of initial temperatures, which corresponds to the discretization number of the domain. The output of the neural network is only one value, which makes reference to the temperature T at the x coordinate. The inner layers are 5, whose composition is: 2×14, 3×14, 2×14, 14 and 7 nodes, respectively. As activation functions, the hyperbolic tangent was chosen to activate the hidden layers, except for the last layer, in which a linear activation function was chosen. The optimizer algorithm employed was Adam, and the train was performed with 200 iterations per each of the learning rates: $10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-6}$.

3.4 DQL parameters

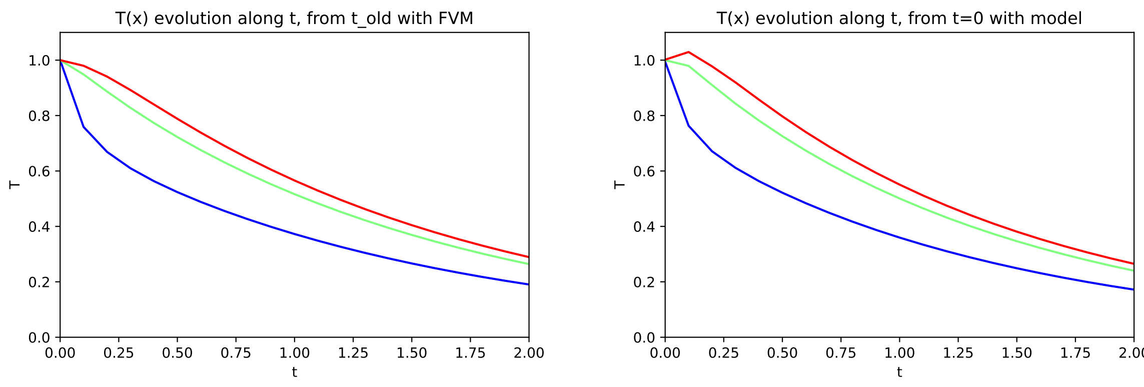
The DQL neural network was set with $11 \times N_{st,h}$ inputs for the defined number of previous time steps considered, multiplied by the number of temperatures of each time step (eleven). The 4 hidden layers with $2 \times 11N_{st,h}$, $4 \times 11N_{st,h}$, $2 \times 11N_{st,h}$ and $1 \times 11N_{st,h}$ nodes each, and sigmoid activation function. For the last layer, the activation function was set with Softmax. The optimizer algorithm was set to Adam with a learning rate of 10^{-3} and categorical cross-entropy as a loss function. The DQL was trained with 400 episodes, with a maximum of 20 steps per episode.

4 RESULTS

4.1 PINN results

The training of the PINN was performed in an Intel® Core™ i7-7700K CPU at 4.20 GHz (4 cores, 8 threads) and 64 GB of RAM, and took approximately $17'$ (with 5 elements), $2h6'$ (with 11 elements), $11h57'$ (with 21 elements), and $89h21'$ (with 41 elements).

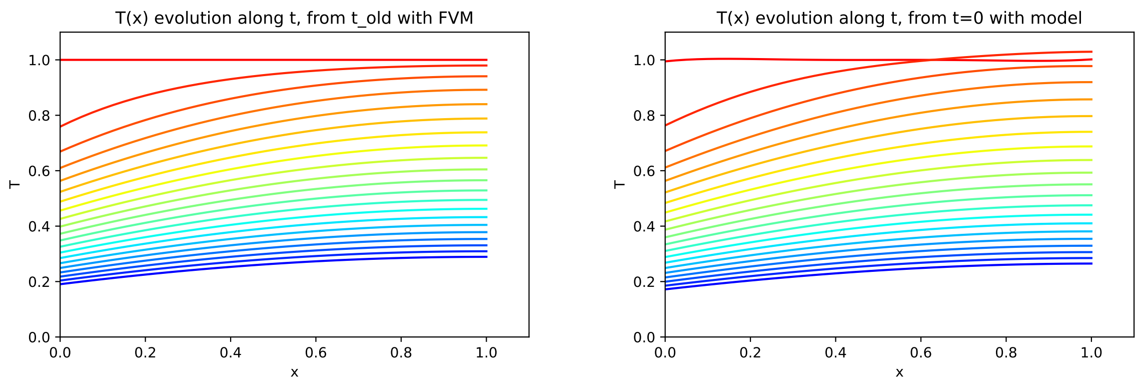
The PINN quality was evaluated through the analysis of its prediction to the temperature distribution for all (5, 11, 21 or 41) points homogeneously distributed along the range $[0, 1]$, from the initial condition (at $t = 0$ s) up to $t = 2$ s with a time step of 0.1 s. A comparison is made between the PINN results and the same problem using the Finite Volume Method, enabling us to note a good agreement and consequently validate the PINN model. The results for 41 distribution points are presented in Fig. 6.



(a) (b)

Figure 6: Temperature comparison between: (a) the Finite Volume Method approach and (b) the PINN approach to predict the temperature as function of time along the rod point's $x = 0$ (red line), $x = 0.5$ (green line) and $x = 1$ (blue line).

A similar analysis can be performed for the temperature profile prediction as a function of time along the entire rod domain, as shown in Fig. 7, for the case of 41 distribution points.



(a) (b)

Figure 7: Temperature profile comparison between: (a) the Finite Volume Method approach and (b) the PINN approach to predict the temperature as function of time along the rod profile. $t = 0$ s (red line), and $t = 2$ s (blue line).

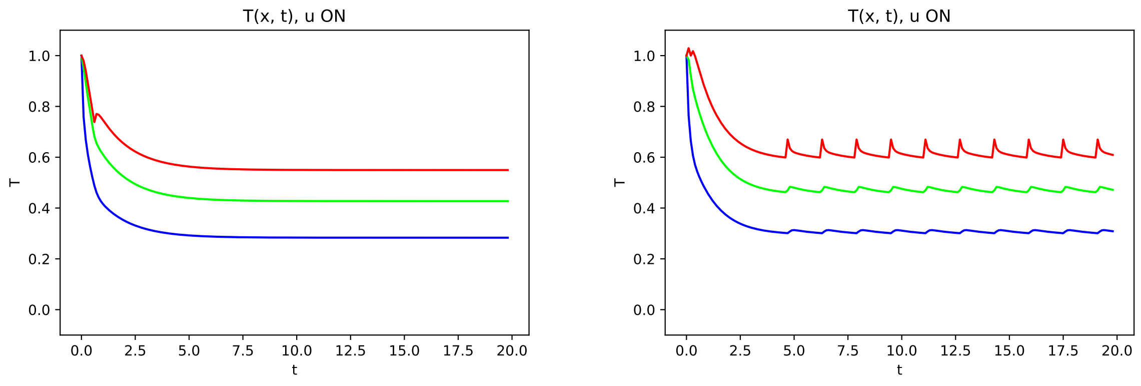
4.2 DQL Results

Once the PINN is trained, it is possible to train the policy model with the Deep Q-learning algorithm. The training was performed on the same processor already referred, it took about 18", 14", 11" and 11", respectively, for 5, 11, 21 and 41 nodes using the FVM, whereas the PINN model took 51", 36", 27" and 27", respectively. The simulation to test the DQL control took around 0.67" with FVM whereas with the PINN took around 1.11".

The control strategy was set to be a bang-bang approach. The reference temperature was set to $T = 0.5$ at $x = 0.5$.

To set a baseline, a bang-bang controller, i.e., the heating, is turned on when the temperature at $x = 0.5$ is equal to or lower than the goal temperature $T = 0.5$. The evolution of the temperatures at three points ($x = 0$, $x = 0.5$ and $x = 1$) was monitored, and from this evolution with FVM versus PINN model (see Fig. **??**) it seems that PINN enables a more stable control when coarser meshes are used, with FVM improving its performance with mesh refinement. However, it should be noted that the PINN hyperparameters can be changed with a deeper study to improve this behaviour.

To evaluate the performance of the control strategy, the temperature profiles at three different locations of the rod, $x = 0$, $x = 0.5$ and $x = 1$ are recorded. Fig.8, presents a comparison between the use of the Finite Volume Method and the PINN method in conjunction with the DQL method to control the $x = 0.5$ temperature. The rod was set to have 41 distribution points.

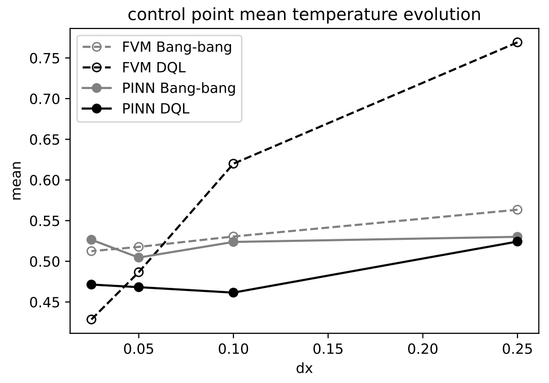


(a) (b)

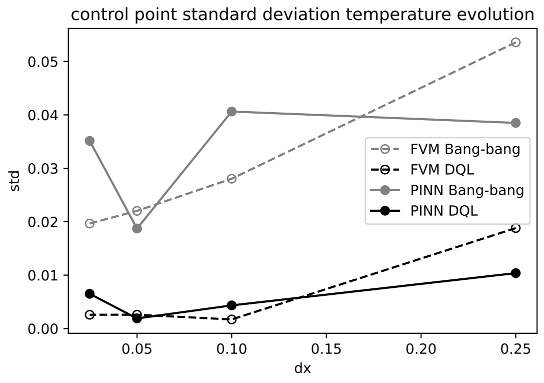
Figure 8: Temperature comparison between: (a) the Finite Volume Method approach and (b) the PINN approach with the control strategy and the DQL implementation. $x = 0$ (red line), $x = 0.5$ (green line) and $x = 1$ (blue line).

When FVM and PINN were compared, several PINN hyperparameters were tested. The control that looked at a forecast of 11 time steps in the future gave better results.

Finally, the temperature of the control point ($x = 0.5$) obtained with the different meshes was analysed, considering its mean value and its standard deviation on the last 150 (of 200) time steps. These values (mean and standard deviation) are represented in Fig. 9 against the mesh element size dx.



(a)



(b)

Figure 9: Temperature mean (a) and standard deviation (b), at the control point ($x = 0.5$) as function of the element size.

5 CONCLUSIONS

In this work, a Physics-Informed Neural Network was successfully trained, enabling its use outside of edge computing hardware and minimising the requirements for resource-demanding equipment.

This model was validated by comparison with a Finite Volume Method solver, proving its worthiness to predict dynamic behaviour.

The implementation of a PINN model with a Deep Q-learning algorithm, allowed for the computation of a control policy that satisfied the reference requirements, and was compared with a Bang-Bang control strategy.

In spite of presenting a 1D problem as the case of study, the results obtained enable one to show that this strategy is an interesting alternative implementation to use for more complex control problems.

References

[1] Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

[2] Valerii Iakovlev, Markus Heinonen, and Harri Lähdesmäki. Learning continuous-time pdes from sparse data with graph neural networks. *arXiv preprint arXiv:2006.08956*, 2020.

[3] Katsiaryna Haitsiukevich and Alexander Ilin. Improved training of physics-informed neural networks with model ensembles. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2023.

[4] Bing-Zheng Han, Wei-Xi Huang, and Chun-Xiao Xu. Deep reinforcement learning for active control of flow over a circular cylinder with rotational oscillations. *International Journal of Heat and Fluid Flow*, 96:109008, 2022.

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

[6] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[7] Jonas Nicodemus, Jonas Kneifl, Jörg Fehr, and Benjamin Unger. Physics-informed neural networks-based model predictive control for multi-link manipulators. *IFAC-PapersOnLine*, 55(20):331–336, 2022.

[8] Tor Laneryd, Federica Bragone, Kateryna Morozovska, and Michele Luvisotto. Physics informed neural networks for power transformer dynamic thermal modelling. *IFAC-PapersOnLine*, 55(20):49–54, 2022.

[9] M Bolderman, D Fan, M Lazar, and H Butler. Generalized feedforward control using physics—informed neural networks. *IFAC-PapersOnLine*, 55(16):148–153, 2022.

[10] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[11] Sifan Wang, Hanwen Wang, and Paris Perdikaris. On the eigenvector bias of fourier feature networks: From regression to solving multi-scale pdes with physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 384:113938, 2021.

[12] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*, 2022.

[13] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

[14] Colby L Wight and Jia Zhao. Solving allen-cahn and cahn-hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542*, 2020.

[15] Revanth Mattey and Susanta Ghosh. A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390:114474, 2022.

[16] Konstantinos Prantikos, Lefteri H Tsoukalas, and Alexander Heifetz. Physics-informed neural network solution of point kinetics equations for a nuclear reactor digital twin. *Energies*, 15(20):7697, 2022.

[17] Eric Aislan Antonelo, Eduardo Camponogara, Laio Oriel Seman, Jean Panaioti Jordanou, Eduardo Rehbein de Souza, and Jomi Fred Hübner. Physics-informed neural nets for control of dynamical systems. *Neurocomputing*, page 127419, 2024.

[18] Dixia Fan, Liu Yang, Michael S Triantafyllou, and George Em Karniadakis. Reinforcement learning for active flow control in experiments. *arXiv preprint arXiv:2003.03419*, 2020.

[19] Tarcísio Déda, William R Wolf, and Scott TM Dawson. Backpropagation of neural network dynamical models applied to flow control. *Theoretical and Computational Fluid Dynamics*, 37(1):35–59, 2023.

[20] Lucian Buşoniu, Tim De Bruin, Domagoj Tolić, Jens Kober, and Ivana Palunko. Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*, 46:8–28, 2018.

[21] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[22] Frank Rosenblatt. A perceiving and recognizing automation. Technical report, Cornell Aeronautical Laboratory, 01 1957. URL `https://bpb-us-e2.wpmucdn.com/websites.umass.edu/dist/a/27637/files/2016/03/rosenblatt-1957.pdf`.

[23] Jean Rabault, Feng Ren, Wei Zhang, Hui Tang, and Hui Xu. Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization. *Journal of Hydrodynamics*, 32:234–246, 2020.

[24] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.