Preprints.org

Article

Not peer-reviewed version

Heat Conduction Control using Deep Q-Learning approach with Physics-Informed Neural Networks

N.D. Gonçalves and Jhonny de Sá Rodrigues *

Posted Date: 14 June 2024

doi: 10.20944/preprints202406.0978.v1

Keywords: Physics-Informed Neural Network; Deep Q-Learning; Model Predictive Control; Heat Transfer

Preprints.org is a free multidiscipline platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Heat Conduction Control Using Deep Q-Learning approach with Physics-Informed Neural Networks

N.D. Gonçalves [1,2,*,†] and Jhonny de Sá Rodrigues [1,2,†]

[1] Institute of Science and Innovation in Mechanical and Industrial Engineering (INEGI), Rua Dr. Roberto Frias, 400, 4200-465 Porto, Portugal; jsrodrigues@inegi.up.pt (J.R.); ngoncalves@inegi.up.pt (N.D.G)
[2] LAETA—Associated Laboratory of Energy, Transports and Aerospace, 4200-265 Porto, Portugal
* Correspondence: ngoncalves@inegi.up.pt
† These authors contributed equally to this work.

Abstract: As modern systems are becoming more complex their control strategy cannot longer relay only on measurement information that usually comes from probes but also from mathematical models. Those systems models can lead to unbearable computation times due to their own complexity, turning the control process non-viable, which leads to the implementation of surrogate models that enable to achieve estimates within acceptable time to take decisions. A control trained with Deep Reinforcement Learning algorithm, using a Physics-Informed Neural Network to obtain the temperature map on the following time step, replaces the need of running Direct Numerical Simulations. On this work we considered an 1D heat conduction problem, which temperature distribution feeds a control system to activate a heat source aiming to obtain a constant, previously defined, temperature value. With this approach, control training becomes much faster without the need of performing numerical simulations or laboratory measurements, as well the control is taken based on Neural Network enabling its implementation on simple processors to edge computing.

Keywords: physics-informed neural network; deep Q-learning; model predictive control; heat transfer

1. Introduction

Differential equations serve as the primary representation of complex interactions that are primarily non-linear in the study of real-world phenomena [1]. Although it is possible to model those systems, the analytical solutions of the resulting equations 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 remain unresolved for a significant period, making it challenging to be used in a real application to make timely decisions to apply certain control measures. Often, surrogate models and numerical solutions are utilised to address this issue, leading to alternative methods for finding the actual solution.

Some research studies employ data-driven models to facilitate the capture of knowledge from spatial and temporal discretizations. For instance, the research conducted by Iakovlev *et al.* [2] highlights the capacity of a trained model to effectively handle unstructured grids, arbitrary time intervals, and noisy observations.

One way of modelling data is through the employment of neural networks [3,4], which are computational models that approximate the behaviour of a system and are composed of multiple layers to represent data with multiple levels of abstraction [5]. This approach has achieved superior results across a wide range of applications [6], such as visual object recognition, video classification [7] and voice recognition [8].

The typical process for training a neural network involves having a set of data that contains true inputs and their corresponding true outputs. Then, the neural network parameters, weights, and biases are initialised with random values. After such initialization, the goal resides in minimising the error between true known data and the prediction generated by a neural network model. The error is minimised by modifying the neural network parameters using an optimisation strategy.

The objective of standard neural network training is to reduce the error between the model's predicted values and real values. However, there are cases where gathering the true resulting values is

© 2024 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

not possible, thereby preventing the application of this training process. To address such an issue, an alternative approach for training neural network models, as proposed by Lagaris *et al.* [9], states that the training can be performed using a set of diferential equations.

The objective is to approach to zero the error generated by these equations when evaluating a given set of data, collocation points, that are contained in the differential equations domain. This set of data also involves initial and boundary conditions for such differential equations. This methodology does not rely on training using expected results; instead, it employs evaluation data exclusively. This leads to quicker training procedures compared to methods that involve running simulations or conducting experiments before or during the training phase.

This training approach has been defined as Physics-Informed Neural Networks (PINNs). Some work has been successfully done in the fields of robotics [10], power transformers [11] and control strategies [12].

Despite the conceptual simplicity of PINNs, their training process can be challenging [3]. In addition to configuring Neural Network hyperparameters, such as number of layers, nodes per layer, activation functions, and loss functions, PINNs demand balancing multiple terms of the loss function, partitioning of results, and time marching to prevent convergence to undesirable solutions [13–18].

On the other hand, a system model defined using a PINN strategy can be used as a digital twin to provide information to a decision-making system that could also take information delivered by sensors [19], or coupled with a control system to define actuation actions in real-time situations [20]. In the same sense of digital twin, Liu *et al.* [21] proposed a multi-scale model utilising PINNs to predict the thermal conductivity of polyurethane-phase change materials foam composites.

In terms of control of nonlinear dynamical systems, the implementation of digital twins in the form of neural networks has been increasingly employed. Those digital twins are employed to estimate the state of the system at any given location and time, acting as virtual sensors. An example of such implementation to estimate flow around obstacles, stabilise vortex shedding, and reduce drag force has been presented in the work of Fan *et al.* [22] and Déda *et al.* [23]. One way to define control strategies is through the implementation of reinforcement learning algorithms, which is a machine learning area able to iteratively improve a policy within a model-free framework [24].

One of the reinforcement learning algorithms is the Deep Q-Learning algorithm. Let's suppose a discrete time controller that uses a Markov decision process to define an action according to a structure of events. Once an action is defined and executed, this action changes the state of the system. The system states are compared against a reference. The result of such a comparison generates a reward related to such an action. The system then 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 Deep Q-Learning algorithm is a reinforcement learning algorithm that uses the Bellman equation to evaluate future incomes (Q-value) [25], starting from the current state, and a neural network (Deep) [26] to simultaneously evaluate the Q-values to all possible actions at some stage.

This work presents 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 to 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

In this section, the Deep-Q Learning framework is presented by introducing the physics-informed neural network model and control model employed further in this work.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

2.1. System Control

A simple control system can be represented by a few components with information travelling between them (see Figure 1). One of the components represents a set of actuators which are initially configured with some possible values. The action coming out from those actuators result in an impact on the environment that is composed out of the system and its bounds interactions, and which states are measured by sensors. Another component is the control unit, that uses the information obtained by the sensors to update the system settings to obtain an objective.

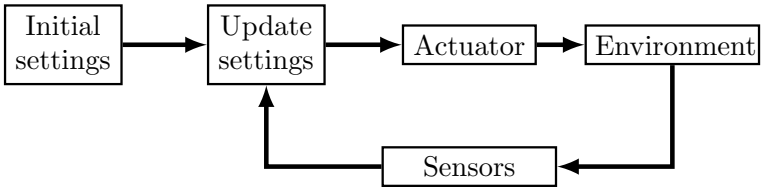


Figure 1. Control flowchart.

In this system, 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, enabling one to predict its behaviour for some system settings and therefore allow one to train the control beforehand. The control system needs to be defined, i.e., one needs to define a policy that decides the best system setting for an environmental state, represented by the sensor's measurements.

In this work the environment behaviour will be modelled with a PINN strategy, and the definition of the control unit will be made with a Reinforcement Learning strategy (Deep Q-Learning).

2.2. PINN

Artificial Neural Networks have been developed since the 1950's inspired by rats cortex functioning [27,28]. A Neural Network is a set of simple computational units, perceptrons (see Figure 2), where a set of inputs are multiplied by weights and its sum with a bias is sent to an activation function. This activation function can be used to polarize the result, i.e., the image is close to a minimum (corresponding to without a property) or close to a maximum (corresponding to having a property).

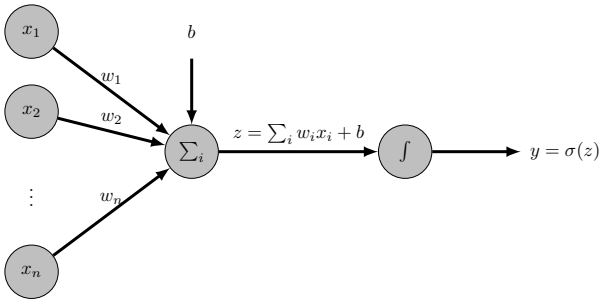


Figure 2. Perceptron.

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 Figure 3).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

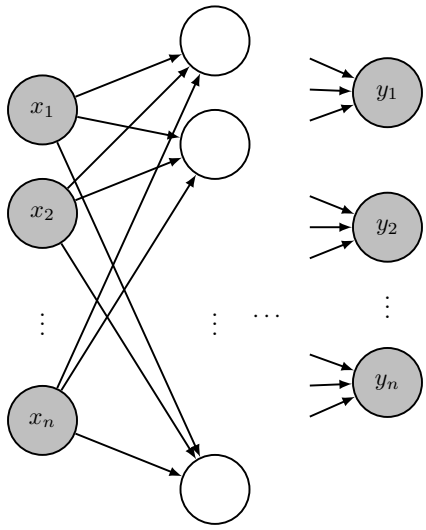


Figure 3. Neural Network.

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.

On "standard" Neural Networks, the results estimated by the Neural Network Y_{NN} (see Figure 4) are compared with known results Y (e.g. obtained by numerical simulations or laboratory measurements) resulting in a loss, $\mathcal{L}_{data}$. This loss can be calculated in several ways, e.g., in this work it was considered the mean square error.

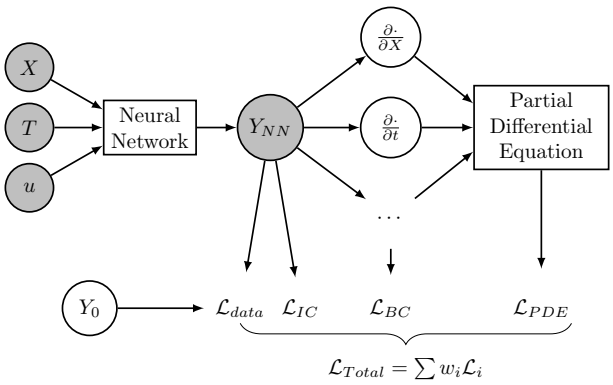


Figure 4. Physics-Informed Neural Network.

In Physics-Informed Neural Networks, the previously referred data can be used to estimate a total loss, or the train can be performed without any data, making this process much faster, especially when data comes from time-consuming processes.

During PINN training, in addition to estimative Y_{NN}, its derivatives are also estimated and replaced on the differential equations to assess the resulting error, $\mathcal{L}_{PDE}$. The boundary conditions and initial conditions are assessed, leading to the losses $\mathcal{L}_{BC}$ and $\mathcal{L}_{IC}$, respectively. A total loss, $\mathcal{L}_{TOTAL}$, is calculated with the several partial losses multiplied by weights. In this work, the weights were defined as the inverse of the cardinality of each set.

At Table 1 is presented the PINN training algorithm to determine the Neural Network Y_{NN} that receives the coordinates X, time t, control u as well the initial values $T(X, t = 0)$ and returns the $T(X, t = t)$. During this training, is required the definition of a finite number of possibilities of each of these parameters. In this work was not considered partitioning of results and time marching to avoid convergence.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

Table 1. PINN algorithm.

PINN algorithm
1: set a set of initial conditions $Y(t = 0)$
2: set training data intervals to consider
3: define the Neural Network parameters: layers, nodes, and activation function.
4: set learning parameters: learning rate, loss function
5: train the Neural Network aiming to minimize loss function

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 generalization, and Neural Networks can be used to perform this task. At Deep Q-learning (DQL), instead of a function that looks at a table of previous results to decide the better action for a given state, an NN is used to evaluate the expected future reward (Q-value) as the sum of future rewards weighted with a discount factor, $\sum_{t>0} \gamma^t r_t$, for a set of possible actions [28].

The use of such a supervised learning algorithm enables one to progressively improve the model (see Figure 5).

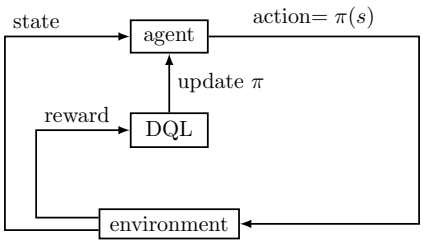


Figure 5. Deep Q-learning training flowchart.

Q-learning algorithm is based on the Bellman equation (Eq. 1)

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

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 $max_a Q(s_t, a_t)$ 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 [29]. Whereas Q_{main} is trained periodically every pre-determined number of iterations, Q_{target} is updated getting the values of Q_{main} less frequently.

At Table 2 is presented the DQL algorithm to determine the Neural Network Q that receives the environment state s and returns the reward Q-values (total discouted future rewards r) for each of available actions a.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

Table 2. Deep Q-learning algorithm .

DQL algorithm
Initialize policy parameters
$step_{global} \leftarrow 0$
for $episode = 0, episode_{max}$ do
$step_{global} \leftarrow step_{global} + 1$
for $step = 0, step_{max}$ do
Draw a random value $\omega \in [0, 1]$
if $\omega < \epsilon$ then
choose a random action
else:
choose the action (available from the current state) that maximizes Q
end if
Execute the action and get a new state and reward (save on batch)
if $step_{global} \equiv 0 \, mod \, steps_{train}$ then
train models
update policy parameters
end if
if $step_{global} \equiv 0 mod steps_{copy}$ then
$Q_{target} \leftarrow Q_{main}$
end if
end for
end for

3. Case Study

In this section, we present the details of the problem used to test the proposed methodology, that aims to utilise a dimensionless model, compatible with neural networks.

3.1. Geometry, Boundary Conditions, and Mesh

The scenario under examination involves a 1D rod of length L, where natural convection occurs at its left end and a controlled heat source is present at its right end (see Figure 6).

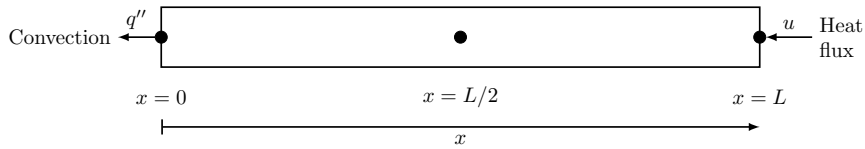


Figure 6. 1-dimensional bar as case of study.

The goal of this case is to reach a specific temperature in the centre of the geometry, at $x = L/2$. The governing equation is then the Energy Conservation equation (2)

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

where T is the temperature, t the time, α the thermal diffusivity, and x the spatial coordinate.

The natural convection equation (3) relies on the heat transfer coefficient h, which characterises the heat convection condition at the surface of the rod, the thermal conductivity property of the rod material k, and the external temperature T_{ext}.

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

At the opposite end, $x = L$, the heat source is controlled by a u function, leading to the condition shown in equation (4).

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

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

As initial conditions, we consider an initial temperature T_{ini}.

The above governing equation (Eq. 2) can be adimentionalized by the following replacement of variables as presented in (5), (6) and (7).

$$\theta = \frac{T - T_{\text{ext}}}{T_{\text{ini}} - T_{\text{ext}}}, \tag{5}$$

$$\xi = \frac{x}{L}, \tag{6}$$

$$\tau = \frac{\alpha}{L^2}t, \tag{7}$$

obtaining the equivalent equation (8).

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

Alternatively, it was considered the normalised values directly on Eq. 2, thermal diffusivity $\alpha = 1$, heat transfer coefficient $h = 1\ [\text{W} \cdot \text{m}^{-2}\text{K}^{-1}]$, $T \in [T_{\text{ext}}, T_{\text{ini}}]$ 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.

3.2. Solution

For validation purposes, the analytical solution was a approximate for a numerical method, the Finite Volume Method, i.e., by integration on space (δ_x) and time (δ_t) at each mesh cell, leading to $\delta_x\left(T_i - T_i^{old}\right) = \delta_t \sum_f \alpha \frac{\partial T}{\partial \mathbf{n}}$. The FVM discretization was performed considering a zero volume elements at both domain ends to tackle with boundary conditions, and the remaining elements with length δ_x.

3.3. PINN Parameters

In this case study, several PINN configurations were tested until acceptable results were achieved. A deeper analysis and optimization of such a model is not a goal of this work.

The $x[\text{m}]$ 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 the 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.

Regarding the Neural Network, the input parameters are 14: the corrdinate x, the time t, the control u and the eleven initial tamperatures ($T(x = 0, t = 0)$, $T(x = 0.1, t = 0)$, ..., $T(x = 1, t = 0)$). The output is just one value, $T(x, t)$. The Neural Network was defined with 5 inner layers with 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 the last layer, which was set with a linear activation function. The optimizer algorithm employed was Adam, and the train was performed with 200 iterations per each of the learning rates: $10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-6}$.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

3.4. DQL Parameters

As stated before, the input to DQL is a state s of the system at a given time t that should be enough to decide the action to take. However, it's possible to increase this information to feed the control with the data of previous time steps. Therefore, this study considered sending the temperature at each coordinate x not only at time t but the last $N_{s_t,h}$, and evaluating the difference control response.

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. The last layer with activation function was set with softmax to return a probability distribution of the decision to make. The optimizer algorithm was set 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 20 maximum steps per episode, saving the results in a 16 long batch before each training, with 4 iterations (epochs), copying Q_{main} to $Q_{trarget}$ every 64 steps. The random actions were taken with a initial probability 1.0, decreasing 1% every steps until achiving a minimum probability 0.1.

The learning rate β was set as 0.9, discount factor $\gamma = 0.1$ and the space of actions (set of possible values of u) in $\{0, 0.25, 0.5, 0.75, 1\}$.

As initial values were considered linear distribution between both ends with temperatures in $[0, 1]$ plus a random perturbation in $[0, 0.01]$.

Several possible Nt_{fw} future values were considered to evaluate the control system, to take into account the delay between the action and the system response.

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 prediction using the PINN is obtained almost instantaneously.

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 0.1 s.

Due to the integration interval, the FVM was employed considering previous time step (old) values to evaluate the following one, whereas PINN model is not limited by this issue.

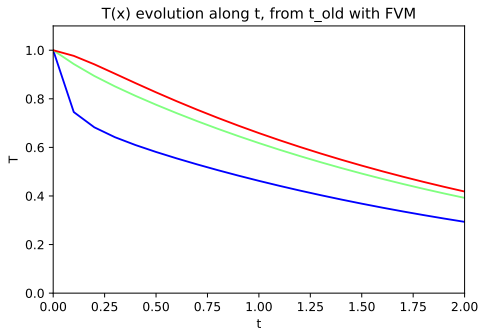
When the initial conditions are defined at $t = 0$, one can predict the evolution of the temperature at three points ($x = 0$ m, $x = 0.5$ m and $x = 1$ m) along the time (see Figure 7). The temperature decreases faster on the boundary with natural convection ($x = 0$ m) whereas the two other places cool slower. The comparison of results obtained with PINN model with those obtained with FVM, enables one to note a good agreement, and consequently validate the PINN model (see Figure 7).

Regarding the mesh influence, from Figure 7 it can be noticed that similar results were obtained, with just some perturbation on the right boundary (red lines) temperatures obtained with PINN.

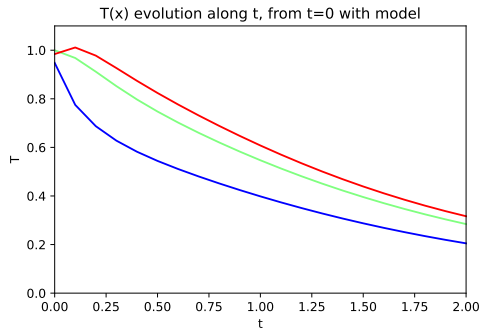
A similar analysis can be performed considering the evolution of the temperature distribution on the domain along the time (see Figure 8).

The comparison of the temperatures predicted by the PINN model and those obtained with FVM (see Figure 8) are similar with a slight difference on values to $t = 0$, where the model seems to be affected by the boundary condition with natural convection. Several approaches were tried, namely different weights to losses and more training iterations, leading to continuous improving of these results.

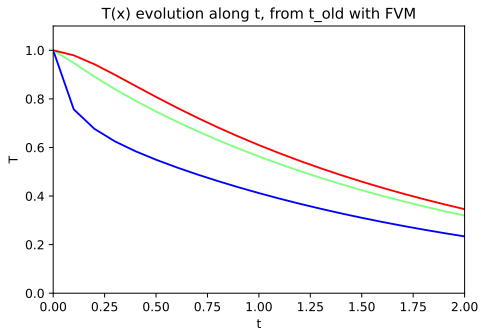
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1



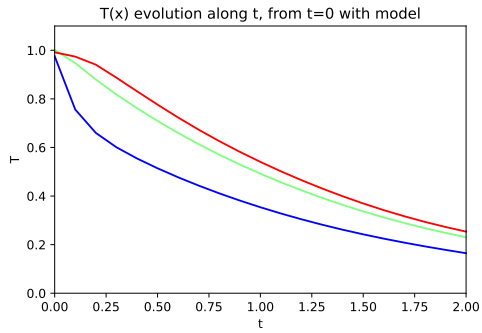
(a) n=5, FVM



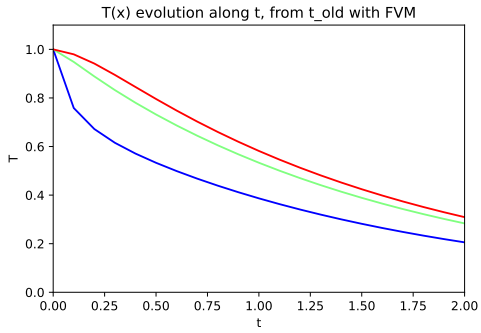
(b) n=5, PINN



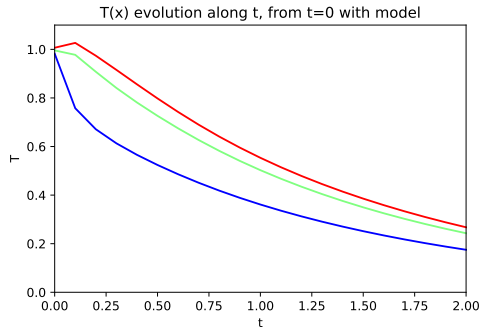
(c) n=11, FVM



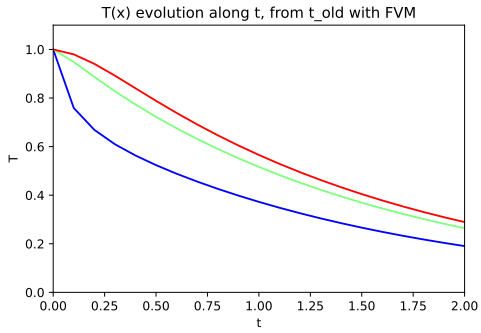
(d) n=11, PINN



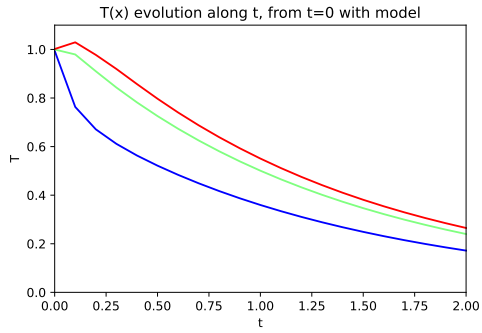
(e) n=21, FVM



(f) n=21, PINN



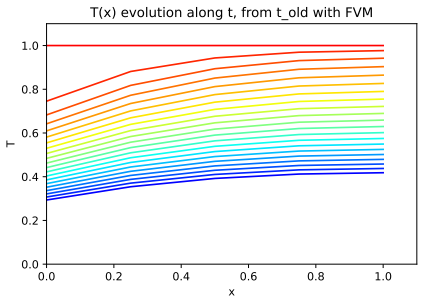
(g) n=41, FVM



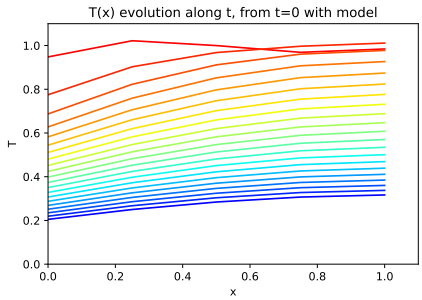
(h) n=41, PINN

Figure 7. Temperature distribution along time at three points: $x = 0$ (red line), $x = 0.5$ (green line) and $x = 1$ (blue line), with no heating source.

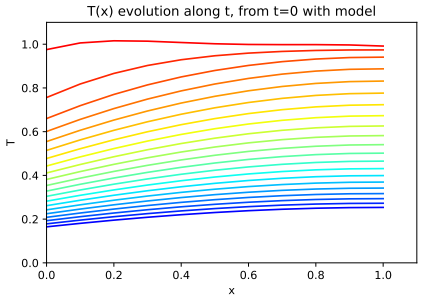
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1



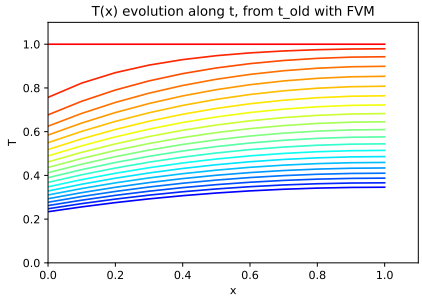
(a) n=5, FVM



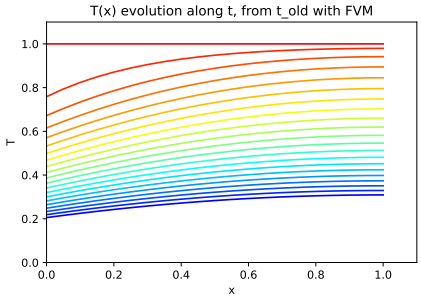
(b) n=5, PINN



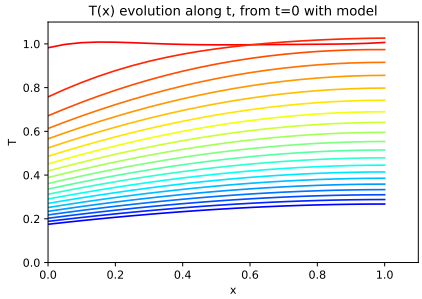
(c) n=11, FVM



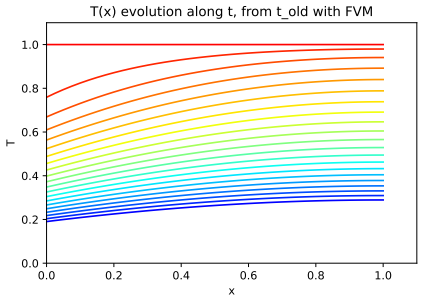
(d) n=11, PINN



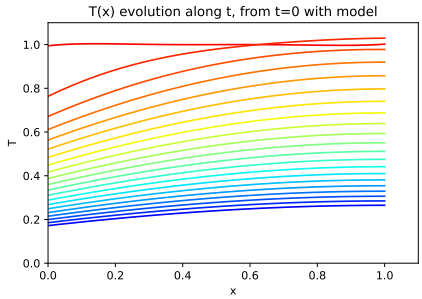
(e) n=21, FVM



(f) n=21, PINN



(g) n=41, FVM



(h) n=41, PINN

Figure 8. Temperature distribution along x for $t = 0$ s (red line) up to $t = 2$ s (blue line), with no heating source.

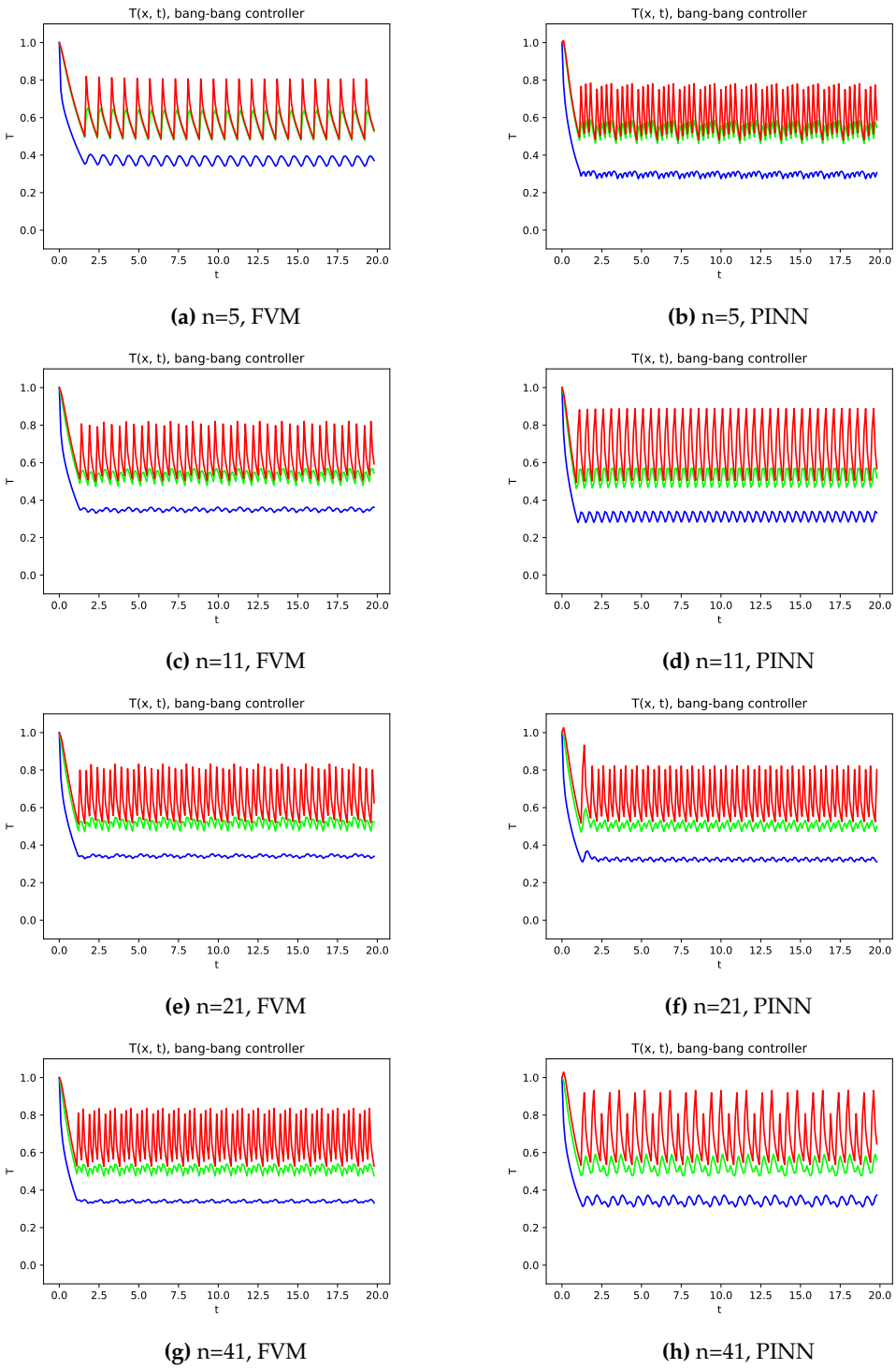
4.2. DQL Results

With the PINN defined, i.e., trained, it was possible to train the policy model, with the Deep Q-learning algorithm. The training was performed on the same processor already referred and 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″.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

To set a baseline, a bang-bang controller, i.e. the heating is turned on allways than 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 $x = 1$) were monitored, and from this evolution with FVM versus PINN model (see Figure 9) 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.



(a) n=5, FVM

(b) n=5, PINN

(c) n=11, FVM

(d) n=11, PINN

(e) n=21, FVM

(f) n=21, PINN

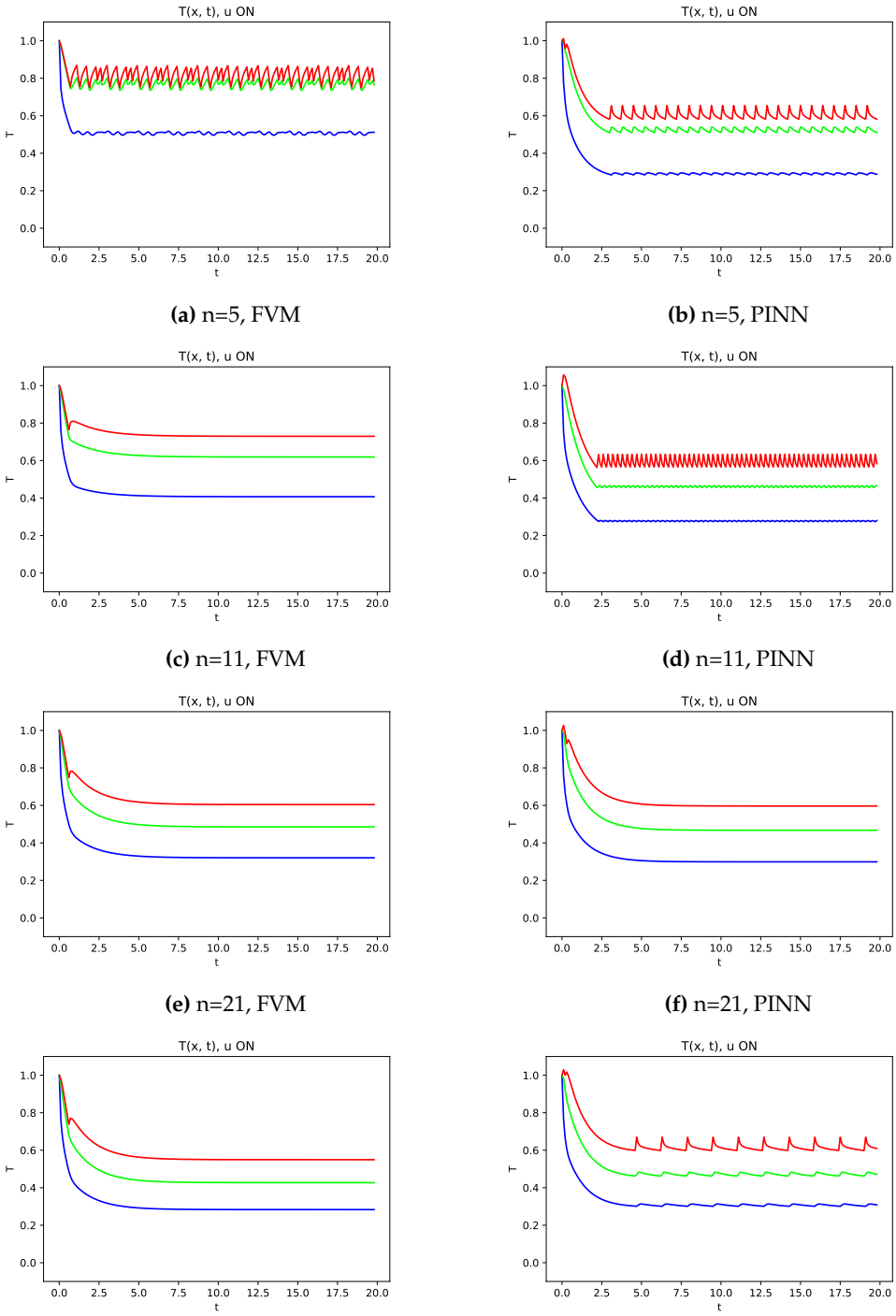
(g) n=41, FVM

(h) n=41, PINN

Figure 9. Temperature distribution along time at three points: $x = 0$ (red line), $x = 0.5$ (green line) and $x = 1$ (blue line), with bang-bang controller.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

Aiming to evaluate the performance of the controller, the temperature evolutions along time at points $x = 0$, $x = 0.5$ and $x = 1$ were represented in Figure 10, and it can be seen that the transition from the initial state to the oscillatory final state is smother when the controller defined with DQL was used. From the comparison of FVM and PINN several hyperparameters of PINN were tested with better results obtained when the control considered a forecast on time 11 time steps in the future than the current value. However, it should be noted that these results can be improved with a deeper study which is not the main goal of this work.



(a) n=5, FVM

(b) n=5, PINN

(c) n=11, FVM

(d) n=11, PINN

(e) n=21, FVM

(f) n=21, PINN

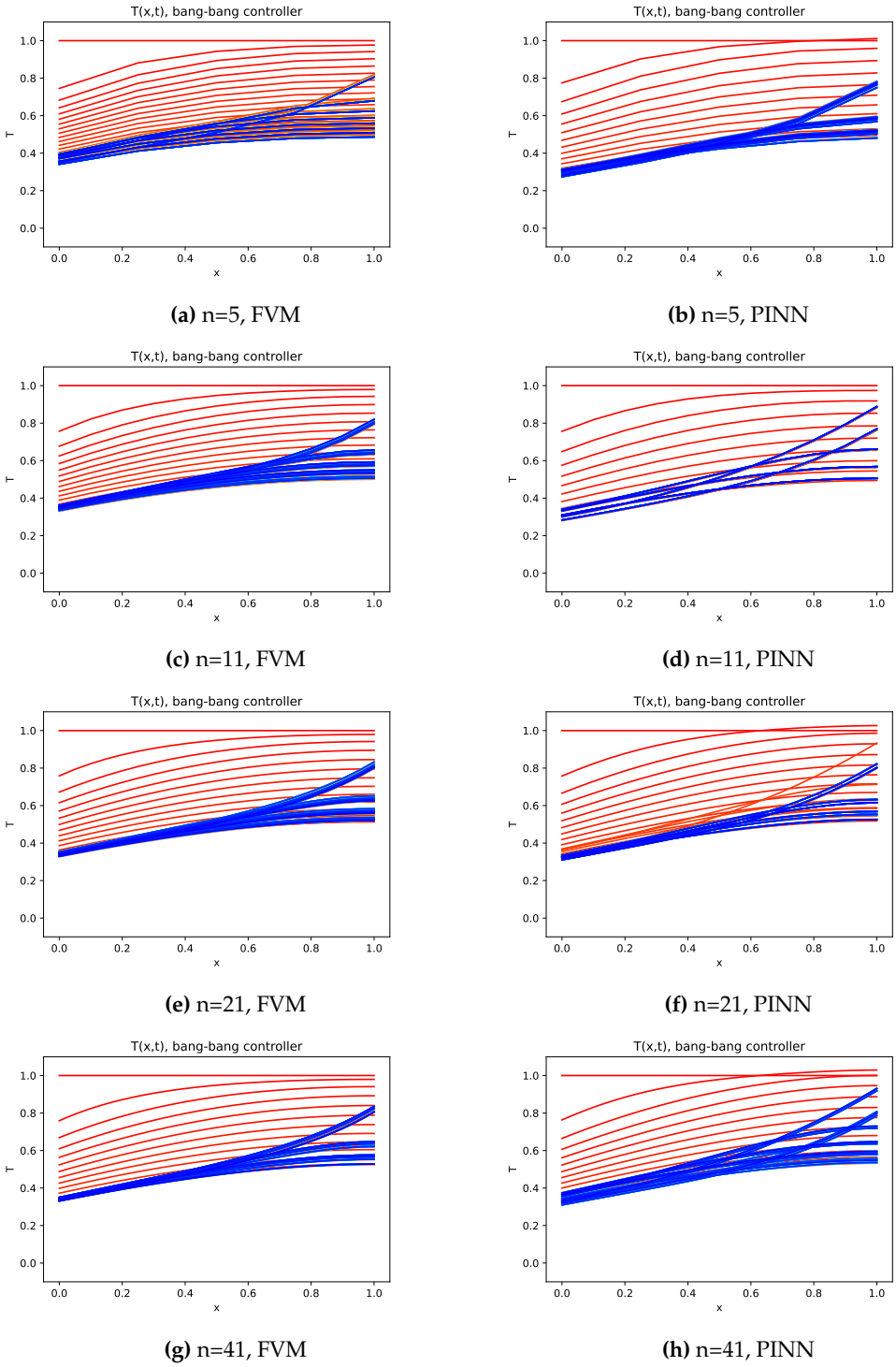
(g) n=41, FVM

(h) n=41, PINN

Figure 10. Temperature distribution along time at three points: $x = 0$ (red line), $x = 0.5$ (green line) and $x = 1$ (blue line), with DQL controller.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

The evolution of the temperature on the domain along the time are presented in Figure 11 and Figure 12, respectively to Bang-bang controller and DQL controller, where the natural convection in the left boundary is identified by the characteristic temperature gradient and a progressive evolution to the final almost stationary state.



(a) n=5, FVM **(b)** n=5, PINN

(c) n=11, FVM **(d)** n=11, PINN

(e) n=21, FVM **(f)** n=21, PINN

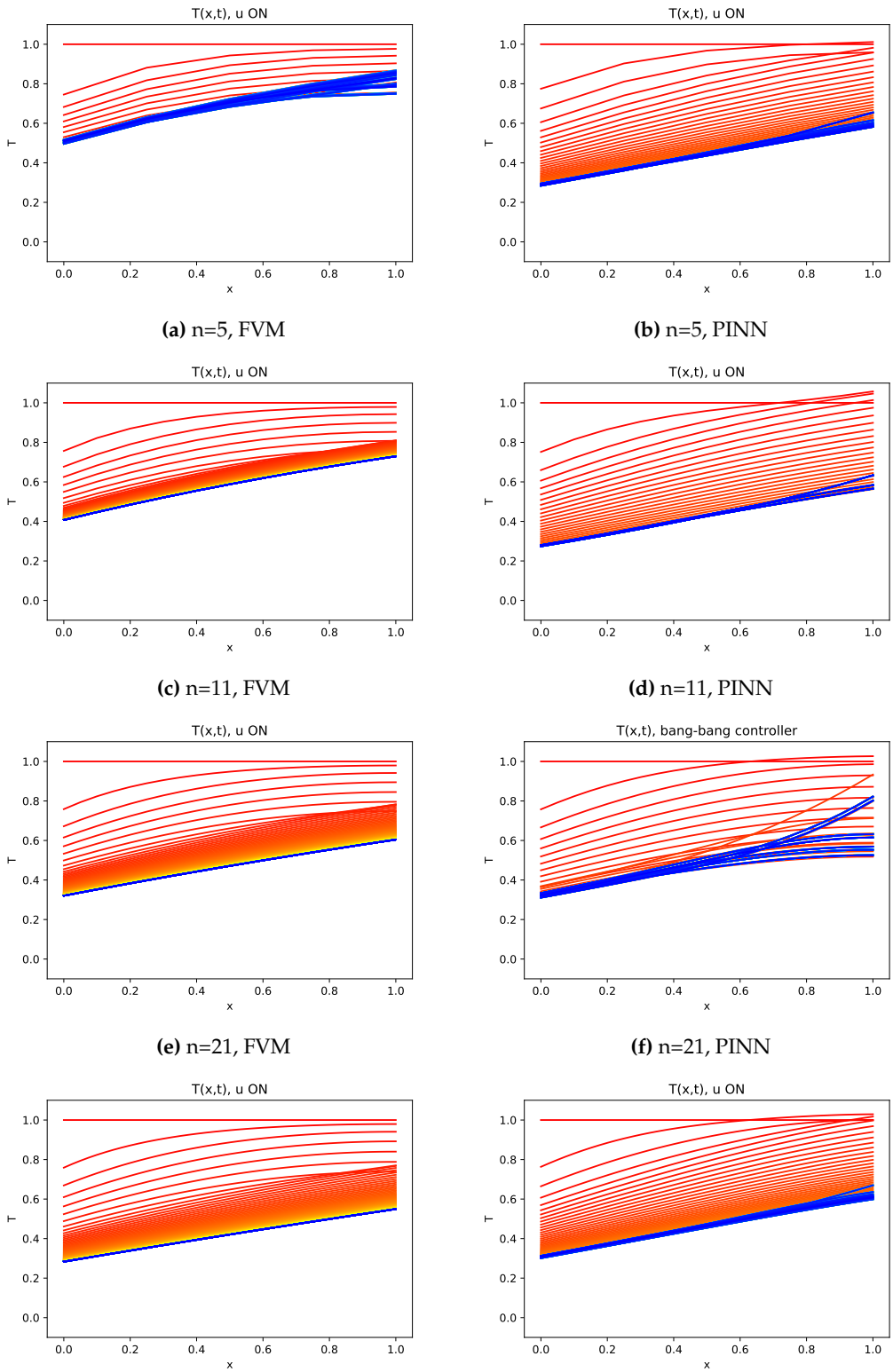
(g) n=41, FVM **(h)** n=41, PINN

Figure 11. Temperature evolution on the domain along the time, from initial stage (red line) up to final stage at $t = 2$ (blue line) with Bang-bang control.

Since Bang-bang controller isn't able to learn the time evolution of the temperature, cannot produce a stable solution with strong variations in temperature distributions (see Figure 11). On the

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

other hand, the control defined with DQL strategy enables us to obtain much more stable solutions (see Figure 12).



(a) n=5, FVM **(b)** n=5, PINN

(c) n=11, FVM **(d)** n=11, PINN

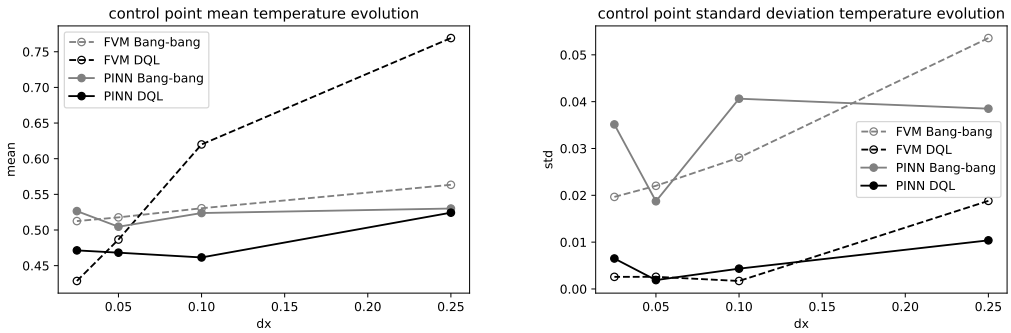
(e) n=21, FVM **(f)** n=21, PINN

(g) n=41, FVM **(h)** n=41, PINN

Figure 12. Temperature evolution on the domain along the time, from initial stage (red line) up to final stage at $t = 2$ (blue line) with model control.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

Finally, the temperature of the control point (at $x = 0.5$) 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 Figure 13, against the mesh element size dx, i.e. lower values (when compared with Bang-bang control strategy) to finner meshes, and it can be seen that the Deep Q-Learning applied to PINN model enables one to obtain mean values closer to the objective as the mesh is finner alongside the standard deviation diminishing. Regarding the PINN versus FVM model, it can be noted that the PINN enables one to obtain values closer to the goal and with lower variations.



(a) mean **(b)** standard deviation

Figure 13. Temperature at the control point $x = 0.5$ obtained with progressivly finner meshes.

5. Conclusions

In this work, was trained a Physics-Informed Neural Network to enable its use in edge computing hardware instead of more resource-demanding alternatives. This model was validated by comparison with a Finite Volume Method solving. Next, this implementation was used on a Deep Q-learning algorithm to find a control policy aiming to attain a predefined temperature at a specific point. Such a control strategy was compared with the Bang-bang control, analysing a 1D problem and the results obtained enable one to show that this strategy is an interesting alternative implementation to use in more complex problems where the need of fast evaluations are required to make decisions.

Author Contributions: Conceptualization, N.G.; methodology, N.G. and J.R.; software, N.G. and J.R.; validation, N.G. and J.R.; formal analysis, N.G. and J.R.; investigation, N.G. and J.R.; writing—review and editing, N.G. and J.R.; All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Evans, L.C. *Partial differential equations*; Vol. 19, American Mathematical Society, 2022.
2. Iakovlev, V.; Heinonen, M.; Lähdesmäki, H. Learning continuous-time pdes from sparse data with graph neural networks. *arXiv preprint arXiv:2006.08956* **2020**.
3. Haitsiukevich, K.; Ilin, A. Improved training of physics-informed neural networks with model ensembles. 2023 International Joint Conference on Neural Networks (IJCNN). IEEE, 2023, pp. 1–8.
4. Han, B.Z.; Huang, W.X.; Xu, C.X. Deep reinforcement learning for active control of flow over a circular cylinder with rotational oscillations. *International Journal of Heat and Fluid Flow* **2022**, *96*, 109008.
5. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *nature* **2015**, *521*, 436–444.
6. Ling, J.; Kurzawski, A.; Templeton, J. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics* **2016**, *807*, 155–166.
7. Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; Fei-Fei, L. Large-scale video classification with convolutional neural networks. Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2014, pp. 1725–1732.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 14 June 2024 doi:10.20944/preprints202406.0978.v1

8. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.r.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N.; others. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine* **2012**, *29*, 82–97.
9. Lagaris, I.E.; Likas, A.; Fotiadis, D.I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks* **1998**, *9*, 987–1000.
10. Nicodemus, J.; Kneifl, J.; Fehr, J.; Unger, B. Physics-informed neural networks-based model predictive control for multi-link manipulators. *IFAC-PapersOnLine* **2022**, *55*, 331–336.
11. Laneryd, T.; Bragone, F.; Morozovska, K.; Luvisotto, M. Physics informed neural networks for power transformer dynamic thermal modelling. *IFAC-PapersOnLine* **2022**, *55*, 49–54.
12. Bolderman, M.; Fan, D.; Lazar, M.; Butler, H. Generalized feedforward control using physics—informed neural networks. *IFAC-PapersOnLine* **2022**, *55*, 148–153.
13. Wang, S.; Teng, Y.; Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing* **2021**, *43*, A3055–A3081.
14. Wang, S.; Wang, H.; Perdikaris, P. 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* **2021**, *384*, 113938.
15. Wang, S.; Sankaran, S.; Perdikaris, P. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404* **2022**.
16. Krishnapriyan, A.; Gholami, A.; Zhe, S.; Kirby, R.; Mahoney, M.W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems* **2021**, *34*, 26548–26560.
17. Wight, C.L.; Zhao, J. Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks. *arXiv preprint arXiv:2007.04542* **2020**.
18. Mattey, R.; Ghosh, S. A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations. *Computer Methods in Applied Mechanics and Engineering* **2022**, *390*, 114474.
19. Prantikos, K.; Tsoukalas, L.H.; Heifetz, A. Physics-Informed Neural Network Solution of Point Kinetics Equations for a Nuclear Reactor Digital Twin. *Energies* **2022**, *15*, 7697.
20. Antonelo, E.A.; Camponogara, E.; Seman, L.O.; Jordanou, J.P.; de Souza, E.R.; Hübner, J.F. Physics-informed neural nets for control of dynamical systems. *Neurocomputing* **2024**, p. 127419.
21. Liu, B.; Wang, Y.; Rabczuk, T.; Olofsson, T.; Lu, W. Multi-scale modeling in thermal conductivity of Polyurethane incorporated with Phase Change Materials using Physics-Informed Neural Networks. *Renewable Energy* **2024**, *220*, 119565.
22. Fan, D.; Yang, L.; Triantafyllou, M.S.; Karniadakis, G.E. Reinforcement learning for active flow control in experiments. *arXiv preprint arXiv:2003.03419* **2020**.
23. Déda, T.; Wolf, W.R.; Dawson, S.T. Backpropagation of neural network dynamical models applied to flow control. *Theoretical and Computational Fluid Dynamics* **2023**, *37*, 35–59.
24. Buşoniu, L.; De Bruin, T.; Tolić, D.; Kober, J.; Palunko, I. Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control* **2018**, *46*, 8–28.
25. Watkins, C.J.; Dayan, P. Q-learning. *Machine learning* **1992**, *8*, 279–292.
26. Garnier, P.; Viquerat, J.; Rabault, J.; Larcher, A.; Kuhnle, A.; Hachem, E. A review on deep reinforcement learning for fluid mechanics. *Computers & Fluids* **2021**, *225*, 104973.
27. Rosenblatt, F. A perceiving and recognizing automation. Technical report, Cornell Aeronautical Laboratory, 1957.
28. Rabault, J.; Ren, F.; Zhang, W.; Tang, H.; Xu, H. Deep reinforcement learning in fluid mechanics: A promising method for both active flow control and shape optimization. *Journal of Hydrodynamics* **2020**, *32*, 234–246.
29. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. Proceedings of the AAAI conference on artificial intelligence, 2016, Vol. 30.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.