Contents lists available at ScienceDirect

Nuclear Materials and Energy

journal homepage: www.elsevier.com/locate/nme

Physics Informed Neural Networks towards the real-time calculation of heat fluxes at W7-X

E. Aymerich [a], F. Pisano [a], B. Cannas [a], G. Sias [a], A. Fanni [a], Y. Gao [b], D. Böckenhoff [b], M. Jakubowski [b], the W7-X Team[c]

[a] *Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy*
[b] *Max-Planck-Institut für Plasmaphysik, Teilinstitut Greifswald, Greifswald, Germany*
[c] *See Sunn Pedersen et al 2022 (https://doi.org/10.1088/1741-4326/ac2cf5) for the W7-X Team*

A B S T R A C T

Wendelstein 7-X, the world largest superconducting advanced stellarator, aims to demonstrate high-performance steady-state experiments lasting up to 30 min. To this purpose, high heat flux (HHF) divertors capable of withstanding steady-state heat fluxes up to 10 MW/m^2 are being installed on the machine, in preparation for the next experimental campaign (OP2.1). The real-time heat flux estimation is pivotal for controlling the divertor heat loads during the experiments. Currently, the THEODOR (Thermal Energy Onto DivertOR) code computes the heat flux offline by numerically solving the heat equation, but the computation time does not allow the application of this approach to the real-time operation of the device. In this work, a new approach based on Physics Informed Neural Networks (PINNs) is proposed for solving the heat equation and estimating the heat fluxes on the divertor tiles in real time. PINN models are Neural Networks that learn Partial Differential Equations (PDEs) by minimizing the PDE loss. The inputs of a PINN are the independent variables of the PDE, e.g., spatiotemporal coordinates, while the loss of the model is designed to make the neural network satisfy the PDE and related initial and boundary conditions. Hence, the model can be trained without any experimental data. Only the initial and boundary conditions of the PDE are necessary for constructing the model. First intermediate results are discussed considering a normalized tile and starting from a constant thermal diffusivity.

1. Introduction

Monitoring and limiting the heat flux on the divertor tiles in real-time is a key objective for the high-performance fusion operation. For this reason, several diagnostics are needed to monitor the state of the device during the experiments. In this regard, one of the fundamental issues for magnetic fusion devices is to ensure the integrity of the PFCs during high-performance operation. At W7–X, infrared (IR) cameras monitor the plasma facing components (PFCs) by measuring their surface temperature [1,2]. Typically, the heat flux is localized on specific high load regions of the divertor called strike lines. Since localized high heat flux values can damage the PFCs, a lot of effort is devoted to the estimation and control of the heat flux on the divertor tiles. Regarding the estimation, starting from the measured temperature at the target surface, the internal temperature distribution can be computed by solving the transient heat conduction equation. Several heat flux reconstruction codes have been developed following this approach, such as TACO [3], NANTHELOT [4] and THEODOR [5,6], which is currently employed at W7–X. THEODOR is a 2D numerical Finite Difference Method code and at W7-X is currently only employed for offline data analysis. However, for heat load control purposes [7], fast computation schemes for the real-time heat flux estimation are required. In this work a Physics Informed Neural Network (PINN) model is proposed to speed up the heat-flux computation towards the real-time implementation. PINNs have several advantages with respect to the other numerical PDE solvers: they can be used to regress nonlinear PDE operators; they are mesh-free and can handle irregular domains; they are able to exploit the parallel computing capabilities of Graphical Processing Units (GPUs) [8]–[10]. The PINN is essentially a traditional Neural Network (NN), where a part of the loss function constrains the network to respect a physic law, in the form of an Ordinary or Partial Differential Equation. This fact makes the architecture of the PINN quite flexible, since many traditional NN architectures can be exploited such as Feed-Forward NNs, Convolutional NNs, Recurrent NNs, etc. Among them, the Feed-Forward NN is often used due to its simplicity, however, more complex architectures such as the Recurrent NNs (e.g., Long Short Term Memory NNs) and Convolutional NNs are also used in the literature. In nuclear fusion, a first example of application of a PINN is reported in [11], where a PINN is trained to reconstruct the 2D plasma equilibrium model. More recent applications of PINNs include the resolution of the Grad-Shafranov equation from magnetic signals [12] and learning reduced order turbulent models [13,14]. In the present paper, a Feed-Forward NN architecture is proposed to reconstruct the temperature distribution in the bulk and the heat-flux on the surface of a typical divertor profile. The model is developed using the DeepXDE library [15] and extending some of its functionalities. The paper is organized as follows: section 2 details the heat flux computation problem; section 3 reports the heat flux PINN architecture, whereas section 4 describes the

https://doi.org/10.1016/j.nme.2023.101401
Received 30 June 2022; Accepted 23 February 2023
Available online 26 February 2023

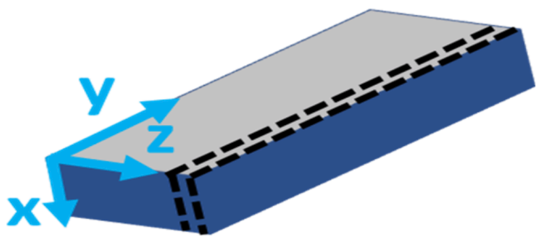

Fig. 1. Sketch of the divertor tile. × is the direction along the depth of the tile, *y* is the direction along the length of the tile (poloidal direction) and z is the toroidal direction. Dashed lines segment a profile, the computation domain of the PDE.

testing results assuming a constant or temperature-dependent diffusion coefficient. Finally, results and future developments are discussed in section 5 and some conclusions are provided in section 6.

2. Heat flux calculation

To calculate the heat flux entering the divertor profile, the temperature distribution inside the profile must be known. It is possible to reconstruct the temperature distribution by solving the heat partial differential equation

$$\frac{\partial u}{\partial t} = D(u)\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{1}$$

Here, $D(u)$ is the heat diffusion coefficient, x is the direction along the depth of the tile, y is the poloidal direction and z is the toroidal direction, as depicted in the sketch of the tile of Fig. 1. Finally, u is the heat–potential, defined as:

$$u(T) = \int_0^T k(T)dT \tag{2}$$

where $k(T)$ is the thermal conductivity coefficient and T is the temperature.

In this formulation, the toroidal heat diffusion is neglected due to the presence of a homogeneous distribution of the strike line in finite toroidal range, as also reported in [16]. To solve a PDE (i.e., to make the solution unique), initial and boundary conditions must be specified. In this paper, the surface temperature is sampled from the infrared cameras and acts as a boundary condition at the surface of the profile, while the lateral edges are considered adiabatic. The initial condition can be either assumed as a uniform constant temperature (at the beginning of the experiment) or reconstructed from the previous frames (during the experiment). The same conditions are assumed in the THEODOR code.

3. Heat flux PINN architecture

Physics informed models can be trained to numerically solve partial differential equations by including physics-based criteria in the NN loss function. The “physics informed” models exploit the possibility of calculating the gradient of the output with respect to the input. In this case, the model is trained using as input the spatial position and time instant at which the solution is to be calculated, while the function to be minimized is based on the PDE in equation (1).

Moreover, to make the solution unique, also the surface boundary condition, the lateral boundary condition and the initial conditions are included in the model loss. In this way it is possible to constrain the output function of the NN to solve the differential equation without using reference data. A sketch of the proposed model is reported in Fig. 2. The inputs of the network are the scalar values of the time and spatial position where the u should be computed, while the network is a Feed-Forward NN with 10 hidden layers, each of them with 97 neurons, and with a hyperbolic tangent (*tanh*) activation function. The number of layers, the number of neurons per layer and the learning rate were optimized with a Bayesian optimization scheme, an automatic optimization scheme where the network performance is modeled as a sample from a Gaussian Process [17]. As previously described, the loss function is the sum of several contributions:

$$L = L_{PDE} + L_{t0} + L_{bN} + L_{bD} \tag{3}$$

where L_{PDE} is the Mean Squared Error (MSE) on the approximation of the PDE, L_{t0} is the MSE with respect to the value of the initial condition, L_{bN}andL_{bD} are the MSEs with respect to the value of each boundary condition. In this case it is said that the network weakly satisfies the boundary and initial conditions. In other approaches in the literature, the boundary condition is directly enforced by cancelling the output of the network on the boundary of the domain. The boundary condition is then satisfied by a function that constrains the output to the values of the boundary conditions [18,19]. In this work, the contributions to the loss have been specified as:

$$\begin{aligned}
L_{PDE} &= \left\langle \left(\frac{\partial \widehat{u}}{\partial t}(x_i, y_i, t_i) - D_N\left(\frac{\partial^2 \widehat{u}}{\partial x^2} + \frac{\partial^2 \widehat{u}}{\partial y}\right)(x_i, y_i.t_i)\right)^2 \right\rangle \\
L_{t0} &= \left\langle (\widehat{u}(x_i, y_i, 0) - u_{t0}(x_i, y_i))^2 \right\rangle \\
L_{bN} &= \left\langle \left(\frac{\partial \widehat{u}}{\partial y}(x_i, y_{bN}, t_i) - u_{bN}(x_i, y_{bD}, t_i) \right)^2 \right\rangle \\
L_{bD} &= \left\langle (\widehat{u}(0, y_i, t_i) - u_{bD}(0, y_i, t_i))^2 \right\rangle
\end{aligned} \tag{4}$$

where u_{t0} is the initial condition of the PDE, u_{bD} the boundary condition on the surface of the tile, expressed as the value of the function at the edge of the tile (or Dirichlet condition), and u_{bN} is the adiabatic

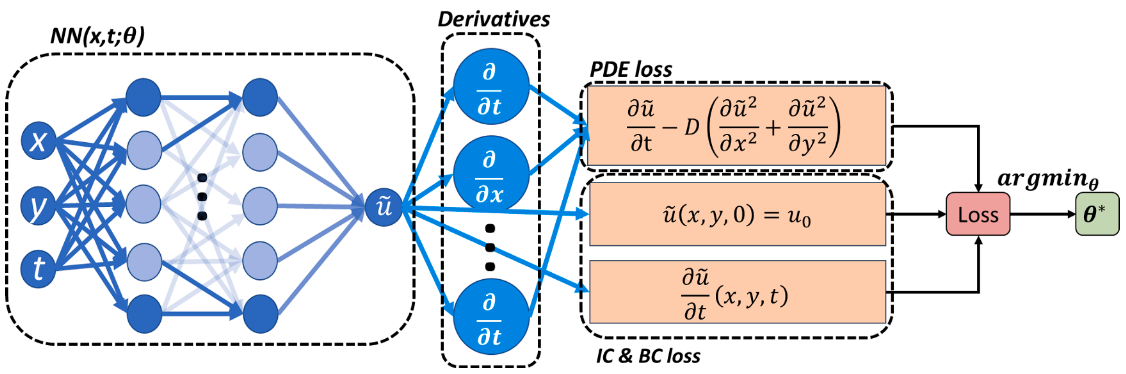

Fig. 2. Scheme of a Physics Informed NN: the inputs are the scalar values of the time and spatial position where solution of the PDE should be computed. The network output can be automatically derived with respect to the inputs using automatic differentiation, enabling the satisfaction of the PDE. The other components of the loss are the boundary and initial conditions of the PDE. Adapted from [15]

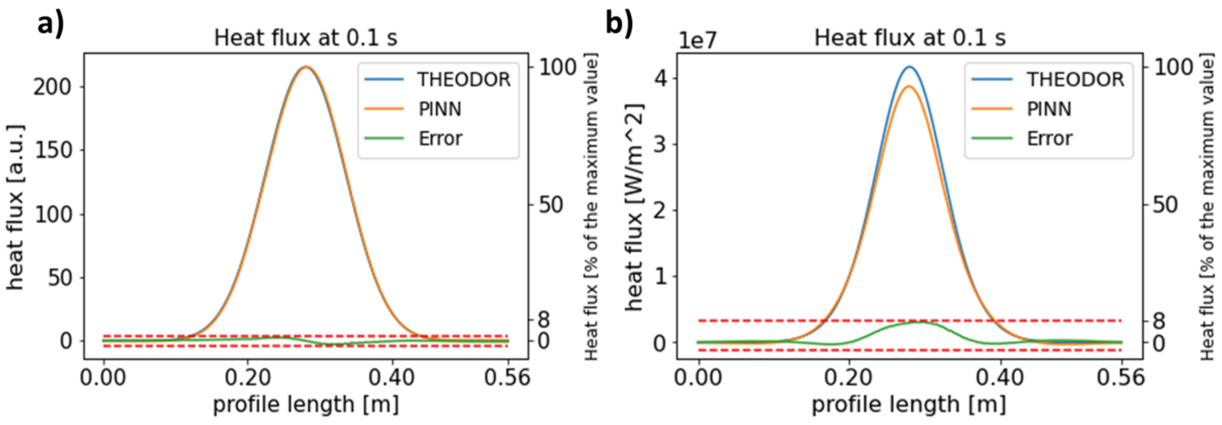

Fig. 3. A) - constant D case: Heat flux on the surface of a tile by THEODOR (blue line), PINN (orange line) and Error (green line) in the absolute and percentage scale, respectively at the left and right side of the plot. Red dashed lines delimit the error range in [-2,2]% (right y-axis). b) – *D(T)* case: Heat flux on the surface of a tile with THEODOR (blue line), PINN (orange line) and Error (green line) in the absolute and percentage scale, respectively at the left and right side of the plot. Red dashed lines delimit the error range in [-3,8]% (right y-axis). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

condition at the lateral edges. Each of the losses is a MSE, hence it is normalized by the number of points. The derivative at the lateral boundary (u_{bN}) should be zero (Neumann boundary condition), since the edges of the tile are adiabatic.

The network is trained by randomly sampling 10,000 points in the computational domain, and then the same linear density of points is maintained over all the edges at the boundary. The average sampling steps in $\times$ and y (dx, dy) must be equal, while the average sampling step in t is related to the average sampling step in space through D_N: $dt = dx^2/D_N$. The training points are resampled randomly every 10,000 iterations, while the same number of validation points are sampled uniformly on the computational domain. The network is trained using the Adam optimizer, and the training is stopped if the loss on the training set does not improve for 10,000 iterations. The model with the minimum validation loss is saved.

4. Results

In the following sub-section, the results of two simulations of the heat diffusion in the divertor tile are discussed. For both examples, the computational domain is consistent with the typical tile profile size, with $(x_{max}, y_{max}) = (28\text{ mm}, 560\text{ mm})$ sides, and PDE evolution time up to $t_{end} = 0.1$ s. The network spatial inputs are rescaled by dividing them with respect to y_{max}, and the time is divided by t_{end}. Finally, the diffusion coefficient D is then rescaled accordingly. The computational domain for the NN becomes:

$$\begin{cases} x_N \in [0, 0.05] \\ y_N \in [0, 1] \\ t_N \in [0, 1] \end{cases}, D_N = D\cdot\frac{t_{end}}{{y_{max}}^2}$$

where D_N is the dimensionless diffusion coefficient for the normalized equation and (x_N, y_N, t_N) the normalized inputs to the PINN. In the first example, the PINN learns the equation with a constant $D = 70\cdot10^{-6}\text{mm}^2/\text{s}$, while in the second one material properties are introduced and *D(T)* depends on the temperature (hence on the heat potential u). In both examples the initial and boundary conditions are fixed.

4.1. Diffusion with constant D

The initial condition of this example is a profile with a uniform heat potential of 0, while the top boundary condition (in the normalized domain) is a gaussian heat potential on the upper surface ($x = 0$), with amplitude 1, centered in 0.5 and with a standard deviation of 0.1. The amplitude of the gaussian is normalized between 0 and 1, but a simple rescaling of the output would allow to adapt the range to the real case. Fig. 3a compares the heat flux estimated with the PINN model to the ones computed with a 2D version of THEODOR, with a relative error below 2% on the heat flux values.

4.2. Diffusion with material properties

In this second example, the temperature dependency has been implemented by using the following nonlinear interpolation for D

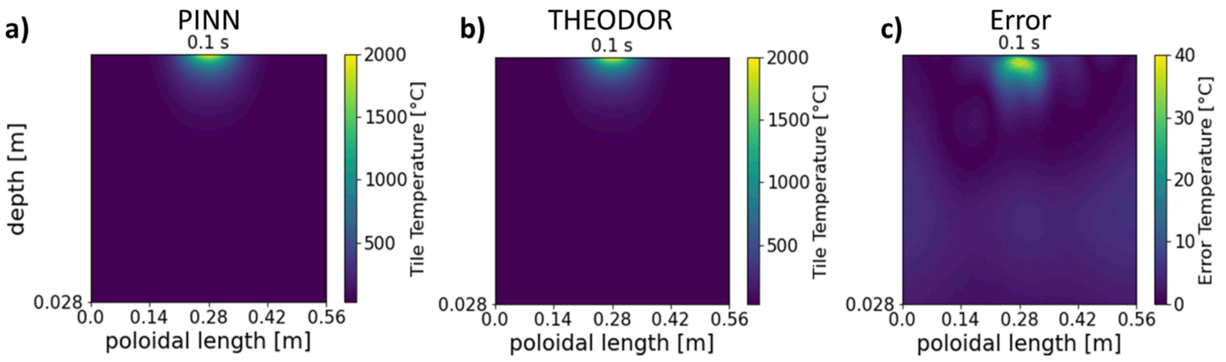

Fig. 4. A): pinn reconstruction of the temperature distribution in the profile at t = 0.1 s. b): THEODOR reconstruction of the temperature distribution in the profile at t = 0.1 s c): Error computed as the absolute difference value between the two reconstructions.

$$D(T) = a_{d0} + b_{d0} \Big/ \left(1 + \frac{T}{T_{d0}}\right)^2 \quad (5)$$

which is the same applied in THEODOR [6,16]. This formula has been implemented by modifying the PDE loss: the T is computed from the heat potential u by inverting equation (2) and then $D(u)$ is computed and used in the L_{PDE}. Since there is a nonlinear dependency between D and T, the boundary condition was not normalized as in the previous case, but it was considered as a gaussian of temperature between 25 °C and 2000 °C. In this case, at the end of the simulation it was possible to achieve an error smaller than 8% on the temperature and heat-flux reconstructions. The results of the heat flux reconstruction are reported in Fig. 3b, while the temperature reconstruction for t = 0.1 s is shown in Fig. 4.

5. Further developments

Unfortunately, the problem addressed with the PINN is still a simple one: the network only needs to solve one PDE, with given (fixed) boundary and initial conditions, while this is different in real applications. In a real-time framework, the initial conditions will change, and the model must be able to solve the PDE starting from any initial condition. Hence, an improvement of the model is under development, exploiting the possibility to enter the parameters of the PDE, such as the boundary and initial conditions, as input to the network. This is currently a problem under study in the physics informed machine learning community [8]–[10,20]. A first step towards the development of the model is the extraction or generation of a set of initial condition to train the physics informed model, which should be representative of the variety of the possible experimental conditions. Then a parametrized model, such as the ones in [9,20], will be trained by sampling the initial condition in a fixed set of points and a training procedure will be performed.

6. Conclusions

In this work, a Physics-Informed Neural Network approach is proposed to speed up the solution of the heat partial differential equation. The PINN method is based on recent advances in the automatic differentiation and machine learning and provides a very flexible model to solve a PDE in a mesh-free domain. Moreover, the PINN can be straightforwardly run on a GPU, hence allowing the real-time use of the method. In the simulations shown in this work, the PINN allows to compute the PDE solution and the heat flux computation error with respect to THEODOR is lower than 8%. Future work on the PINN model will focus on providing initial and boundary conditions as inputs, hence solving a parametrized PDE and enabling the real-time evaluation of the heat flux.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The github repository with the codes can be made available

Acknowledgments

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

References

[1] F. Pisano, et al., Rev. Sci. Instrum. 89 (12) (Dec. 2018), 123503, https://doi.org/10.1063/1.5045560.
[2] M.W. Jakubowski, et al., Rev. Sci. Instrum. (2018), https://doi.org/10.1063/1.5038634.
[3] K.F. Gan, et al., Rev. Sci. Instrum. 84 (2) (Feb. 2013), 023505, https://doi.org/10.1063/1.4792595.
[4] C.S. Kang, et al., Rev. Sci. Instrum. 87 (8) (Aug. 2016), 083508, https://doi.org/10.1063/1.4961030.
[5] A. Herrmann, et al., Plasma Phys. Control. Fusion 37 (1) (Jan. 1995) 17–29, https://doi.org/10.1088/0741-3335/37/1/002.
[6] B. Sieglin, et al., Rev Sci Instrum (2015) 7.
[7] F. Pisano, et al., Learning control coil currents from heat-flux images using convolutional neural networks at Wendelstein 7-X, Plasma Phys. Control. Fusion (2020), https://doi.org/10.1088/1361-6587/abce19.
[8] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Nat. Rev. Phys. vol. 3, no. 6, Art. no. 6 (Jun. 2021), https://doi.org/10.1038/s42254-021-00314-5.
[9] S. Wang, H. Wang, and P. Perdikaris, *Sci. Adv.*, vol. 7, no. 40, p. eabi8605, doi: 10.1126/sciadv.abi8605.
[10] L. Lu, P. Jin, G.E. Karniadakis, Nat. Mach. Intell. 3 (3) (Mar. 2021) 218–229, https://doi.org/10.1038/s42256-021-00302-5.
[11] B.P. van Milligen, V. Tribaldos, J.A. Jiménez, Phys. Rev. Lett. 75 (20) (Nov. 1995) 3594–3597, https://doi.org/10.1103/PhysRevLett.75.3594.
[12] S. Joung, et al., Nucl. Fusion 60 (1) (Jan. 2020), 016034, https://doi.org/10.1088/1741-4326/ab555f.
[13] A. Mathews, M. Francisquez, J.W. Hughes, D.R. Hatch, B. Zhu, B.N. Rogers, Phys. Rev. E 104 (2) (Aug. 2021), 025205, https://doi.org/10.1103/PhysRevE.104.025205.
[14] A. Mathews, N. Mandell, M. Francisquez, J.W. Hughes, A. Hakim, Phys. Plasmas 28 (11) (Nov. 2021), 112301, https://doi.org/10.1063/5.0066064.
[15] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, SIAM Rev. 63 (1) (Jan. 2021) 208–228, https://doi.org/10.1137/19M1274067.
[16] Y. Gao, et al., Nucl. Fusion 59 (6) (Jun. 2019), 066007, https://doi.org/10.1088/1741-4326/ab0f49.
[17] J. Snoek, H. Larochelle, and R. P. Adams, in *Advances in Neural Information Processing Systems*, 2012, vol. 25. Accessed: May 16, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html.
[18] K.S. McFall, J.R. Mahan, IEEE Trans. Neural Netw. 20 (8) (Aug. 2009) 1221–1233, https://doi.org/10.1109/TNN.2009.2020735.
[19] I.E. Lagaris, A. Likas, D.G. Papageorgiou, IEEE Trans. Neural Netw. (2000), https://doi.org/10.1109/72.870037.
[20] Y. Nakamura, S. Shiratori, H. Nagano, K., Shimano, presented at the 7th World Congress on Mechanical, Chemical, and Material, Engineering (Aug. 2021), https://doi.org/10.11159/htff21.113.