mathematics

MDPI

Article

A Parsimonious Separated Representation Empowering PINN–PGD-Based Solutions for Parametrized Partial Differential Equations

Chady Ghnatios [1,*] and Francisco Chinesta [1,2]

[1] PIMM Research Laboratory, UMR 8006 CNRS-ENSAM-CNAM, Arts et Metiers Institute of Technology, 151 Boulevard de l'Hôpital, 75013 Paris, France; francisco.chinesta@ensam.eu
[2] CNRS@CREATE Ltd., 1 Create Way, #08-01 CREATE Tower, Singapore 138602, Singapore
* Correspondence: chady.ghnatios@ensam.eu

Abstract: The efficient solution (fast and accurate) of parametric partial differential equations (pPDE) is of major interest in many domains of science and engineering, enabling evaluations of the quantities of interest, optimization, control, and uncertainty propagation—all them under stringent real-time constraints. Different methodologies have been proposed in the past within the model order reduction (MOR) community, based on the use of reduced bases (RB) or the separated representation at the heart of the so-called proper generalized decompositions (PGD). In PGD, an alternate-direction strategy is employed to circumvent the integration issues of operating in multi-dimensional domains. Recently, physics informed neural networks (PINNs), a particular collocation schema where the unknown field is approximated by a neural network (NN), have emerged in the domain of scientific machine learning. PNNs combine the versatility of NN-based approximation with the ease of collocating pPDE. The present paper proposes a combination of both procedures to find an efficient solution for pPDE, that can either be viewed as an efficient collocation procedure for PINN, or as a monolithic PGD that bypasses the use of the fixed-point alternated directions.

Keywords: proper generalized decomposition; physics informed neural network; machine learning; parsimonious learning; separated representation

MSC: 90C08; 90C10

check for updates

Citation: Ghnatios, C.; Chinesta, F. A Parsimonious Separated Representation Empowering PINN–PGD Based Solutions for Parametrized Partial Differential Equations. *Mathematics* **2024**, *12*, 2365. https://doi.org/10.3390/math12152365

Academic Editor: Giovanni Stabile

Received: 3 July 2024
Revised: 23 July 2024
Accepted: 26 July 2024
Published: 29 July 2024

1. Introduction

Parametrized partial differential equations are typical in the domain of science and engineering. The solution of such equations allows accessing the problem solutions for any value of the parameters involved in those PDEs. However, pPDE induces the curse of dimensionality when many parameters are involved in the problem model and solution [1].

Addressing the efficient solution of these multi-parametric pPDEs is of great interest in engineering because, as soon as the parametric solution is known, its evaluation for any combination of parameters is almost instantaneous. Such a solution would pave the way towards possible simulation, optimization, control, or uncertainty propagation, all under the stringent real-time constraint [2].

For this purpose, different model order reduction techniques have been proposed and extensively applied, among them being the reduced basis [3–7] and the so-called proper generalized decomposition (PGD) [1,8,9]. PGD computes the parametric solution by operating directly on the PDE, without requiring the usual pre-calculations of a posteriori MOR techniques, for instance the reduced bases method.

If we consider field u, depending on space $\mathbf{x}$ and time t, and we can assume that the field satisfies the following equation

$$\mathcal{L}(u(\mathbf{x},t)) = f(\mathbf{x},t), \tag{1}$$

where $\mathcal{L}$ is the differential operator (here assumed linear), with parameter μ (here, it is a single parameter for the sake of simplicity, but without any loss of generality) in the model, field u will depend parametrically on it; that is, $u(\mathbf{x}, t; \mu)$. Even if the loading term f involves parameters, here, it is assumed for the sake of simplicity that it is not the case.

The PGD framework treats the parameter as an extra coordinate (or as many extra-coordinates as parameters) and then expresses the solution u as $u(\mathbf{x}, t, \mu)$, representing the solution at each point $\mathbf{x} \in \Omega_x \subset \mathbb{R}^d$ ($d = 1, 2, 3$), at each time $t \in \Omega_t \subset \mathbb{R}^+$, and for each value of the parameter $\mu \in \Omega_\mu \subset \mathbb{R}$.

With the choice of considering the model parameters as extra coordinates, the problem's dimensionality increases, leading to the curse of dimensionality in mesh-based approximations. In order to circumvent this issue, PGD assumes a separated representation of the unknown field $u(\mathbf{x}, t, \mu)$, according to

$$u(\mathbf{x}, t, \mu) \approx \sum_{i=1}^{N} \mathcal{X}_i(\mathbf{x}) \mathcal{T}_i(t) \mathcal{M}_i(\mu), \tag{2}$$

where functions $\mathcal{X}_i$, $\mathcal{T}_i$ and $\mathcal{M}_i$ must be computed.

For this purpose, we assume a greedy construction method. By assuming that at step $n-1$ (where $n > 1$ and $n - 1 < N$), the approximation of the solution at that step is expressed by

$$u^{n-1}(\mathbf{x}, t, \mu) \approx \sum_{i=1}^{n-1} \mathcal{X}_i(\mathbf{x}) \mathcal{T}_i(t) \mathcal{M}_i(\mu), \tag{3}$$

the solution update u^n is

$$u^n(\mathbf{x}, t, \mu) \approx \sum_{i=1}^{n} \mathcal{X}_i(\mathbf{x}) \mathcal{T}_i(t) \mathcal{M}_i(\mu) = u^{n-1}(\mathbf{x}, t, \mu) + \mathcal{X}_n(\mathbf{x}) \mathcal{T}_n(t) \mathcal{M}_n(\mu). \tag{4}$$

In order to compute the three unknown functions—$\mathcal{X}_n$, $\mathcal{T}_n$, and $\mathcal{M}_n$—the weak form of pPDE is considered

$$\int_{\Omega_x \times \Omega_t \times \Omega_\mu} u^* \, \mathcal{L}(u) \, d\mathbf{x} dt d\mu = \int_{\Omega_x \times \Omega_t \times \Omega_\mu} u^* \, f(\mathbf{x}, t) \, d\mathbf{x} dt d\mu \tag{5}$$

Now, within a Galerkin setting, the test function $u^*(\mathbf{x}, t, \mu)$ is expressed as follows

$$u^*(\mathbf{x}, t, \mu) = \mathcal{X}^*(\mathbf{x}) \mathcal{T}_n(t) \mathcal{M}_n(\mu) + \mathcal{X}_n(\mathbf{x}) \mathcal{T}^*(t) \mathcal{M}_n(\mu) + \mathcal{X}_n(\mathbf{x}) \mathcal{T}_n(t) \mathcal{M}^*(\mu), \tag{6}$$

when introduced into Equation (5), it results in the following three equations at step n:

$$\int_{\Omega_x \times \Omega_t \times \Omega_\mu} \mathcal{X}^* \mathcal{T}_n \mathcal{M}_n \, \mathcal{L}(u^{n-1} + \mathcal{X}_n \mathcal{T}_n \mathcal{M}_n) \, d\mathbf{x} dt d\mu = \int_{\Omega_x \times \Omega_t \times \Omega_\mu} \mathcal{X}^* \mathcal{T}_n \mathcal{M}_n \, f(\mathbf{x}, t) \, d\mathbf{x} dt d\mu, \tag{7}$$

$$\int_{\Omega_x \times \Omega_t \times \Omega_\mu} \mathcal{X}_n \mathcal{T}^* \mathcal{M}_n \, \mathcal{L}(u^{n-1} + \mathcal{X}_n \mathcal{T}_n \mathcal{M}_n) \, d\mathbf{x} dt d\mu = \int_{\Omega_x \times \Omega_t \times \Omega_\mu} \mathcal{X}_n \mathcal{T}^* \mathcal{M}_n \, f(\mathbf{x}, t) \, d\mathbf{x} dt d\mu \tag{8}$$

and

$$\int_{\Omega_x \times \Omega_t \times \Omega_\mu} \mathcal{X}_n \mathcal{T}_n \mathcal{M}^* \, \mathcal{L}(u^{n-1} + \mathcal{X}_n \mathcal{T}_n \mathcal{M}_n) \, d\mathbf{x} dt d\mu = \int_{\Omega_x \times \Omega_t \times \Omega_\mu} \mathcal{X}_n \mathcal{T}_n \mathcal{M}^* \, f(\mathbf{x}, t) \, d\mathbf{x} dt d\mu. \tag{9}$$

Being nonlinear, the updating problem is usually solved through an alternate direction fixed-point algorithm, which iterates by solving the three aforementioned equations. If the iteration is denoted by the superscript $\bullet^r$, the previous equations can be rewritten as:

$$\int_{\Omega_x \times \Omega_t \times \Omega_\mu} \mathcal{X}^* \mathcal{T}_n^{r-1} \mathcal{M}_n^{r-1} \, \mathcal{L}(u^{n-1} + \mathcal{X}_n^r \mathcal{T}_n^{r-1} \mathcal{M}_n^{r-1}) \, d\mathbf{x} dt d\mu =$$

$$\int_{\Omega_x \times \Omega_t \times \Omega_\mu} \mathcal{X}^* \mathcal{T}_n^{r-1} \mathcal{M}_n^{r-1} \, f(\mathbf{x}, t) \, d\mathbf{x} dt d\mu, \tag{10}$$

which updates $\mathcal{X}_n$ from $\mathcal{T}_n$ and $\mathcal{M}_n$ from the previous iteration,

$$\int_{\Omega_x \times \Omega_t \times \Omega_\mu} \mathcal{X}_n^r \mathcal{T}^* \mathcal{M}_n^{r-1} \, \mathcal{L}(u^{n-1} + \mathcal{X}_n^r \mathcal{T}_n^r \mathcal{M}_n^{r-1}) \, d\mathbf{x} dt d\mu =$$

$$\int_{\Omega_x \times \Omega_t \times \Omega_\mu} \mathcal{X}_n^r \mathcal{T}^* \mathcal{M}_n^{r-1} \, f(\mathbf{x}, t) \, d\mathbf{x} dt d\mu, \tag{11}$$

which updates $\mathcal{T}_n$ from the recently updated $\mathcal{X}_n$ and from $\mathcal{M}_n$ obtained in the previous iteration; and

$$\int_{\Omega_x \times \Omega_t \times \Omega_\mu} \mathcal{X}_n^r \mathcal{T}_n^r \mathcal{M}^* \, \mathcal{L}(u^{n-1} + \mathcal{X}_n^r \mathcal{T}_n^r \mathcal{M}_n^r) \, d\mathbf{x} dt d\mu =$$

$$\int_{\Omega_x \times \Omega_t \times \Omega_\mu} \mathcal{X}_n^r \mathcal{T}_n^r \mathcal{M}^* \, f(\mathbf{x}, t) \, d\mathbf{x} dt d\mu, \tag{12}$$

which finally updates $\mathcal{M}_n$ from the two updated functions $\mathcal{X}_n$ and $\mathcal{T}_n$.

Applying the differential operator and then integrating $\Omega_t \times \Omega_\mu$ into Equation (10), $\Omega_x \times \Omega_\mu$ into Equation (11), and $\Omega_x \times \Omega_t$ into Equation (12), the three equations are reduced to:

$$\begin{cases} \int_{\Omega_x} \mathcal{X}^* \, \mathcal{G}_x(\mathcal{X}_n^r) \, d\mathbf{x} = \int_{\Omega_x} \mathcal{X}^* \, h_x^r(\mathbf{x}) \\ \int_{\Omega_t} \mathcal{T}^* \, \mathcal{G}_t(\mathcal{T}_n^r) \, dt = \int_{\Omega_t} \mathcal{T}^* \, h_t^r(t) \\ \int_{\Omega_\mu} \mathcal{M}^* \, \mathcal{G}_\mu(\mathcal{M}_n^r) \, d\mu = \int_{\Omega_\mu} \mathcal{M}^* \, h_\mu^r(\mu) \end{cases}, \tag{13}$$

where $(\mathcal{G}_x, \mathcal{G}_t, \mathcal{G}_\mu)$ refer to the resulting single-coordinate differential operations, and similarly to functions (h_x, h_t, h_μ).

The fixed point of the these equations determines the updated search, i.e., $\mathcal{X}_n$, $\mathcal{T}_n$, and $\mathcal{M}_n$. Further updates are computed until obtaining a small enough residual for the differential equation, at iteration N, which determines the solution approximation $u(\mathbf{x}, t, \mu) \approx u^N(\mathbf{x}, t, \mu)$.

For additional details, the interested reader can refer to [9,10] and the references therein. The main drawbacks of such a procedure are as follows:

- The alternate direction iteration algorithm can become computationally expensive and exhibits low convergence.
- Implementation requires expressing the domain as the Cartesian product of the domains related to the different coordinates. This becomes a challenging issue in the multi-parametric case. To avoid the curse of dimensionality, the decomposition $\Omega_\mu = \Omega_{\mu_1} \times ... \times \Omega_{\mu_P}$ is needed, where P is the number of parameters considered as extra coordinates.
- Optimal implementation requires an affine decomposition for all of the terms involved in the weak form.
- The solution to nonlinear models becomes challenging because the nonlinearity can compromise the affine decompositions.

Recently, PINN was proposed as a versatile procedure for solving pPDE. The main salient points are as follows:

- A general approximation provided by a deep neural network: $u = \mathcal{NN}(\mathbf{x}, t, \mu)$.
- A collocation scheme that computes the derivatives involved in the differential operator $\mathcal{L}(\cdot)$ at different collocation points using automatic derivatives.

- A training procedure for calculating the neural network $\mathcal{NN}$ parameters based on the residual evaluated at the $\mathtt{C}$ collocation points $(\mathbf{x}_i, t_i, \mu_i)$, where $i = 1, ..., \mathtt{C}$. Using the p-norm, the residual is expressed as:

$$\mathcal{R} = \sum_{i=1}^{\mathtt{C}} |\mathcal{L}(u(\mathbf{x}_i, t_i, \mu_i)) - f(\mathbf{x}_i, t_i, \mu_i)|^p, \tag{14}$$

where $\mathcal{L}(u(\mathbf{x}_i, t_i, \mu_i)) \equiv \mathcal{L}(u(\mathbf{x}, t, \mu))|_{\mathbf{x}_i, t_i, \mu_i}$.

PINNs are used in multiple applications to solve complex problems, with multiple researchers working on the injection of physical laws [11]. It is used to solve physical problems, while overcoming the complexity of the computation time and stability conditions required by classical techniques, while having their own stability conditions in transient problems [12]. It is used for innovative approaches where more usual techniques cannot be applied, like when determining the optimal path or velocity of a drone [13,14].

Despite its conceptual simplicity, PINNs are a valuable procedure for addressing general linear and nonlinear pPDE; however, like traditional discretization techniques, they encounter difficulties when dealing with highly multi-parametric models due to the challenge of defining appropriate collocations that enable comprehensive coverage of the multi-dimensional domain.

The present paper proposes a parsimonious PINN–PGD that takes profit of the advantages of both techniques, the separated representations, at the heart of PGD-based discretization and the collocation and NN-based approximation that characterize PINN-based discretizations.

2. Parsimonious PINN–PGD

First, in order to circumvent the curse of dimensionality, a separated representation, at the heart of the PGD, is retained. In the case discussed above, this reads again as:

$$u(\mathbf{x}, t, \mu) \approx \sum_{i=1}^{N} \mathcal{X}_i(\mathbf{x}) \mathcal{T}_i(t) \mathcal{M}_i(\mu), \tag{15}$$

where we assume the $n-1$ first modes are already computed, looking at the present step for the update $\mathcal{X}_n(\mathbf{x}) \mathcal{T}_n(t) \mathcal{M}_n(\mu)$.

Now, inspired by PINN, three neural networks (as many as problem coordinates—including the so-called extra-coordinates) are defined in the present step:

$$\begin{cases} \mathcal{X}_n = \mathcal{NN}_n^x(\mathbf{x}) \\ \mathcal{T}_n = \mathcal{NN}_n^t(t) \\ \mathcal{M}_n = \mathcal{NN}_n^\mu(\mu) \end{cases}. \tag{16}$$

Then, three sets of collocation points are defined: (i) $\mathbf{x}_i$, $i = 1, \ldots, \mathtt{A}$; (ii) t_j, $j = 1, \ldots, \mathtt{B}$; and (iii) μ_k, $k = 1, \ldots, \mathtt{D}$. By using automatic differentiation, the different derivatives of the neural networks can be calculated using the defined collocation points, in order to evaluate all of the terms involved in the differential operator.

Now, the residual at step n, $\mathcal{R}^n$, can be defined in two ways:

1. Adding the collocation point residuals

$$\mathcal{R} = \sum_{m=1}^{\mathtt{C}} |\mathcal{L}(u^n(\mathbf{x}_m, t_m, \mu_m)) - f(\mathbf{x}_m, t_m, \mu_m)|_p, \tag{17}$$

where m refers to each of the collocation points resulting from the Cartesian product of $\mathbf{x}_i$, t_j, and μ_k, with $\mathtt{C} = \mathtt{A} \times \mathtt{B} \times \mathtt{D}$. It is important to note that in step n, the term involved by the differential operator reads

$$\mathcal{L}(u^n(\mathbf{x}_m, t_m, \mu_m)) \equiv \mathcal{L}(u^{n-1}(\mathbf{x}_m, t_m, \mu_m) + \mathcal{X}_n(\mathbf{x}_i)\mathcal{T}_n(t_j)\mathcal{M}_n(\mu_k))) \tag{18}$$

with the indexes (i, j, k) corresponding to m.

2. When considering the $L2$-norm and a fully affine decomposition, we can express

$$(\mathcal{L}(u(\mathbf{x}, t, \mu)) - f(\mathbf{x}, t, \mu))^2 = \sum_{s=1}^{\mathtt{S}} F_s(\mathbf{x})G_s(t)H_s(\mu), \tag{19}$$

which allows for writing the residual from

$$\mathcal{R} = \|\mathcal{L}(u(\mathbf{x}, t, \mu)) - f(\mathbf{x}, t, \mu)\|^2 =$$

$$\sum_{s=1}^{\mathtt{S}} \left(\int_{\Omega_x} F_s(\mathbf{x}) d\mathbf{x} \right) \times \left(\int_{\Omega_t} G_s(t) \right) \times \left(\int_{\Omega_\mu} H_s(\mu) \right). \tag{20}$$

The Parametric Transient Heat Equation as an Example

For the sake of illustrative purposes, the parsimonious PINN–PGD mentioned here is applied first to solve the one-dimensional transient and parametric heat equation involving the temperature field $u(x, t; \mu)$

$$\frac{\partial u}{\partial t} - \mu \frac{\partial^2 u}{\partial x^2} = Q \tag{21}$$

with $x \in \Omega_x = [0, L]$ and $t \in \Omega_t = [0, T)$, where T is large enough to ensure that the transient problem reaches the steady state, μ is the thermal diffusivity, and Q is the heat source uniformly distributed in $\Omega_x \times \Omega_t$. The initial and boundary conditions are $u(x, t = 0) = u^0$ and $u(x = 0, t) = u(x = L, t) = u_d$.

As discussed before, after introducing the diffusivity μ as a problem extra coordinate, the temperature field is searched in the separated form

$$u(x, t, \mu) \approx \sum_{i=1}^{N} \mathcal{X}_i(x)\mathcal{T}_i(t)\mathcal{M}_i(\mu). \tag{22}$$

The loss function consists of the PDE residual, that at step n reads

$$\mathcal{R}^n = \left\| \sum_{i=1}^{n} \mathcal{X}_i \frac{\partial \mathcal{T}_i}{\partial t} \mathcal{M}_i - \mu \sum_{i=1}^{n} \frac{\partial^2 \mathcal{X}_i}{\partial x^2} \mathcal{T}_i \mathcal{M}_i - Q \right\|_2. \tag{23}$$

The problem is solved with the architecture depicted in Figure 1. The network is built such that, at every iteration, the three networks are trained simultaneously

$$\begin{cases} \mathcal{X}_n = \mathcal{NN}_n^x(\mathbf{x}) \\ \mathcal{T}_n = \mathcal{NN}_n^t(t) \\ \mathcal{M}_n = \mathcal{NN}_n^\mu(\mu) \end{cases}. \tag{24}$$

When training the three networks, $\mathcal{X}_n$, $\mathcal{T}_n$, and $\mathcal{M}_n$, the previously trained networks $\mathcal{X}_j$, $\mathcal{T}_j$, and $\mathcal{M}_j$ for $j < n$ are frozen and set as non trainable.

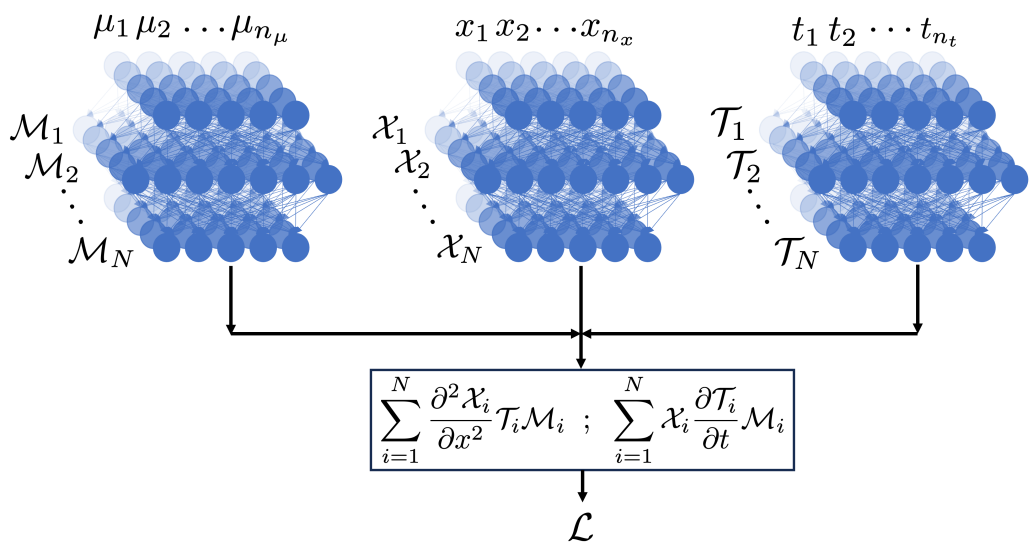


Figure 1. The architecture of the network used to build the parsimonious PINN–PGD.

The training in this example is easy to perform and exhibits fast convergence with a low number of required vectors N. However, the construction of the loss function (23) requires the reconstruction of the whole multidimensional domain $\Omega_x \times \Omega_t \times \Omega_\mu$, fact that compromise the strategy efficiency when operating in highly multi-parametric settings.

To circumvent this difficulty, instead of the $L2$-norm considered in the loss (23), one could consider the alternative loss given by Equation (17) or the one given by Equation (20).

3. Numerical Results

This section solved the heat transfer problem (21), using the proposed PINN–PGD approach just introduced.

The used neural networks are resumed in Table 1. We note the need to use a larger network in time, when compared to the other networks, to achieve the convergence of the PINN–PGD. All the biases in the PINN–PGD networks are set to zero, since the solution of a differential equation doesn't include biases. The layers were initialized using the Glorot uniform approach [15]. The gradient descent applies in two steps:

1. First, the ADAM algorithm with cosine decay is used, where the learning rate is reduced from 10^{-3} to 10^{-6} over 150 epochs.
2. A second step of fine tuning, where the ADAM algorithm is used again, with a constant learning rate of 10^{-6} over 150 epochs.

The optimizer is initialized after training every layer of the parsimonious network. Here, 100 uniform collocation points are selected in $x \in [0, 1]$ m, 200 in $t \in [0, 5]$ s, and 40 in $\mu \in [0.1, 1]$ m^{2}s^{-1}.

The boundary conditions are enforced by construction through the change in variables:

$$\begin{cases} \mathcal{X}_n = (x - x_0)(x - x_L) \times \mathcal{NN}_n^x(\mathbf{x}) \\ \mathcal{T}_n = t \times \mathcal{NN}_n^t(t) \\ \mathcal{M}_n = \mathcal{NN}_n^\mu(\mu) \end{cases}, \tag{25}$$

with x_0 and x_L being the minimum and maximum values of x, respectively.

The problem is solved for homogeneous boundary conditions in this case. However, the same transformation can be used for non homogeneous boundary conditions, by first considering a mode satisfying the required boundary conditions [16,17].

Table 1. For the neural networks used in PINN–PGD modeling, all of the layers had the same structure.

$\mathcal{X}$ Networks	Layer Input	Layer Type	Activation
a_x	x	Dense with 60 neurons	*tanh*
b_x	a_x	Dense with 60 neurons	*relu*
c_x	b_x	Dense with 60 neurons	*relu*
d_x	c_x	Dense with 1 neurons	*linear*
$\mathcal{T}$ **Networks**	**Layer Input**	**Layer Type**	**Activation**
a_t	t	Dense with 120 neurons	*tanh*
b_t	a_t	Dense with 120 neurons	*relu*
c_t	b_t	Dense with 120 neurons	*relu*
d_t	c_t	Dense with 1 neurons	*linear*
$\mathcal{M}$ **Networks**	**Layer Input**	**Layer Type**	**Activation**
a_μ	μ	Dense with 60 neurons	*tanh*
b_μ	a_μ	Dense with 60 neurons	*relu*
c_μ	b_μ	Dense with 60 neurons	*relu*
d_μ	c_μ	Dense with 1 neurons	*linear*

Ten modes were computed using the PINN–PGD approach and the solution was compared with the finite element one for the selected diffusivity μ values. Figures 2, 3, and 4 show the PINN–PGD computed modes in x, t, and μ, respectively. As expected, the first mode computed in x is almost a quadratic curve; in time, it increases to reach a steady state, and in the parametric domain μ, a decreasing trend is observed with an increasing μ. It is also noted, especially in the time domain, that the modes' amplitude decreases with increasing n. The decrease in the total amplitude of the modes demonstrates the convergence of the algorithm when increasing the number of modes n.

Figure 5 shows, for $\mu = 0.1$ m^{2}s^{-1}, both the solution obtained using 10 PINN–PGD modes and the one obtained using the finite element method (FEM) with an implicit time integration scheme, using the same mesh in space and time. The relative error $\mathcal{E}$ between both solutions is shown in Figure 6. The relative error is computed using:

$$\mathcal{E} = \frac{\|u_{PINN--PGD} - u_{FEM}\|_1}{\|u_{FEM}\|_\infty} \tag{26}$$

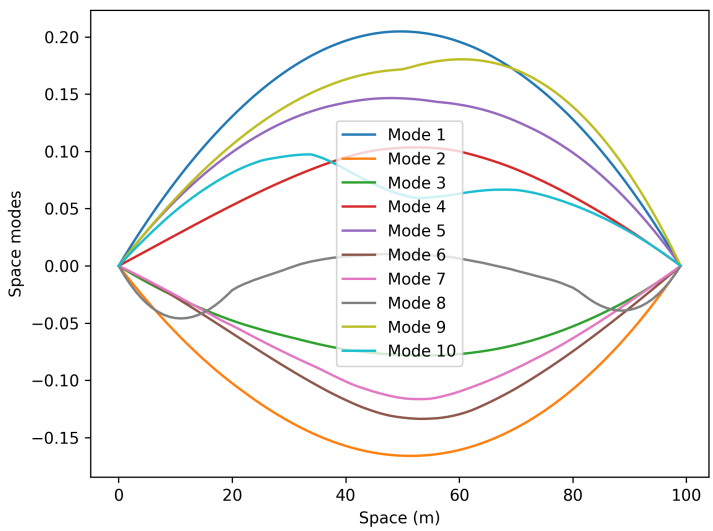


Figure 2. The space modes $\mathcal{X}_n$ found by the Parsimonious PINN–PGD in the space dimension x, for $n = 1, \cdots, 10$.

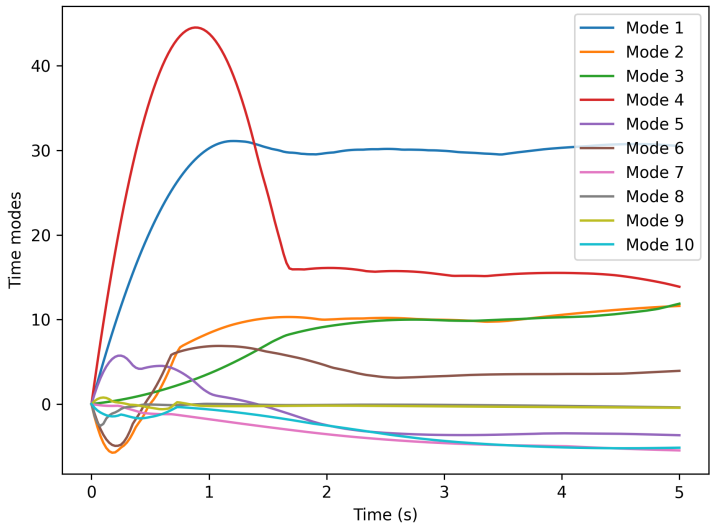


Figure 3. The time modes $\mathcal{T}_n$ found by the Parsimonious PINN–PGD in the time dimension t, for $n = 1, \cdots, 10$.

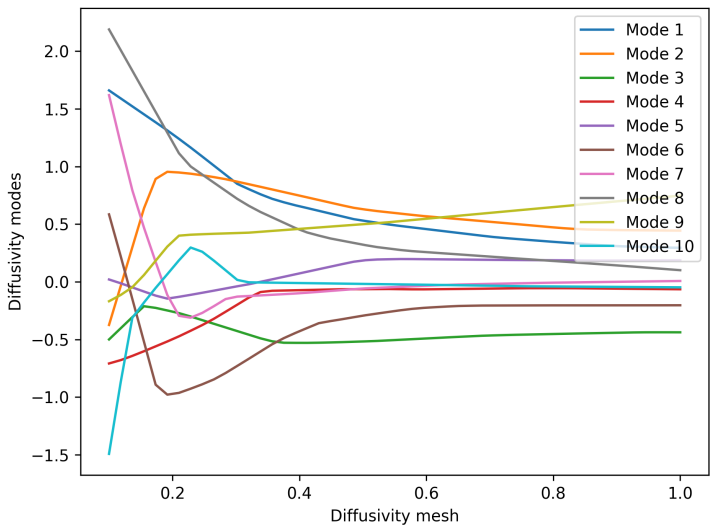
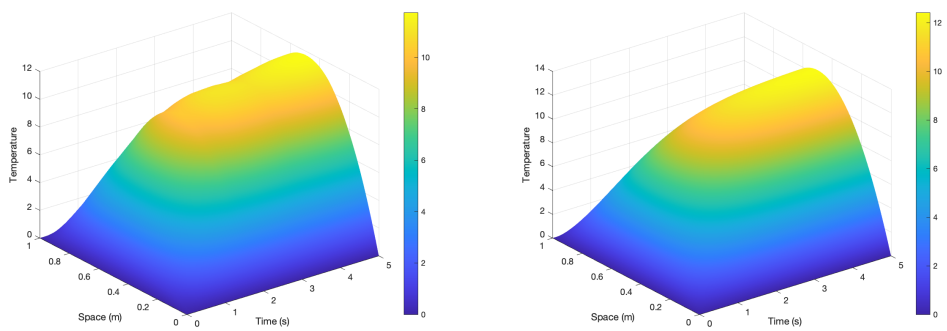


Figure 4. The diffusivity modes $\mathcal{M}_n$ found by the Parsimonious PINN–PGD in the diffusivity dimension μ, for $n = 1, \cdots, 10$.



(**a**) PINN–PGD solution (**b**) FEM solution

Figure 5. PINN–PGD solution and FEM solution for $\mu = 0.1$ m^{2}s^{-1}.

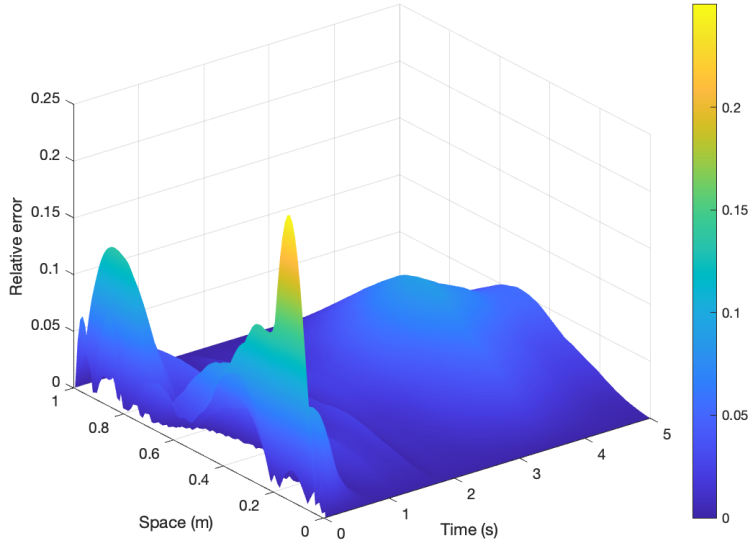
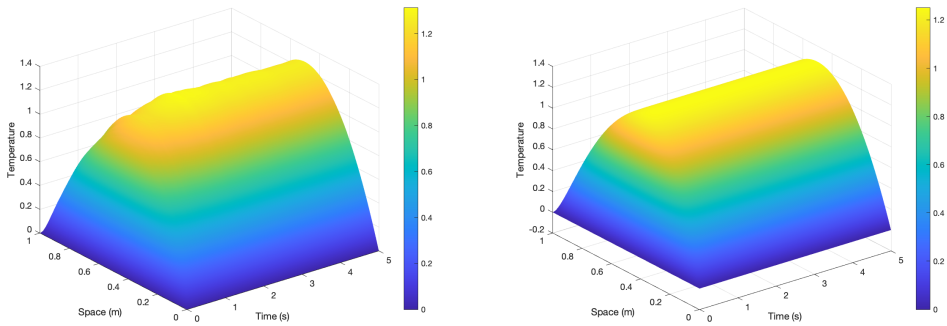


Figure 6. Relative error between the 10-modes for PINN–PGD and FEM solutions, for $\mu = 0.1$ m^{2}s^{-1}.

The relative error is large at the beginning of the time integration, where the solution still has a small amplitude. This is expected as the neural network converges using an average loss, first fitting the large amplitudes, and, later on, the lower amplitudes of the solution. Figure 7 shows the solutions for PINN–PGD and FEM for $\mu = 1$ m^{2}s^{-1}, and Figure 8 shows the relative errors map. The same observation applies for low amplitudes with a larger relative error. Both of the solutions illustrate some small oscillations in time for the PINN–PGD problem. The oscillations can be reduced using further fine tuning and/or more modes in the solution.



(**a**) PINN–PGD solution (**b**) FEM solution

Figure 7. PINN–PGD solution and FEM solution for $\mu = 1$ m^{2}s^{-1}.

The problem complexity scales in the proposed PINN–PGD linear approach as a function of the number of nodes per dimension. For instance, in the problem at hand, the number of unknowns to calculate is N_x for the x dimension, N_t for the time dimension, and N_μ for the diffusivity. There are a total of $(N_x + N_t + N_\mu) \times n$ variables to compute, with n stating the total number of PGD modes. However, for the finite element problem, a problem of dimension N_x needs to be solved for every time step and each selected value of the parameter μ, resulting in a total of $(N_x \times N_t \times N_\mu)$ problems. Thus, the problem complexity scales linearly with the problem dimensionality when using the proposed PINN–PGD approach, while it scales exponentially when using the usual finite element discretization.

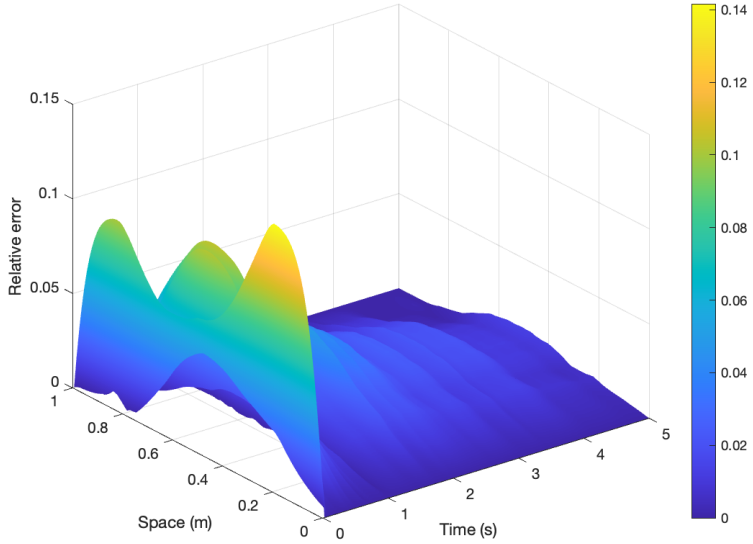


Figure 8. Relative errors between the 10 PINN–PGD modes and the FEM solutions, for $\mu = 1$ m^{2}s^{-1}.

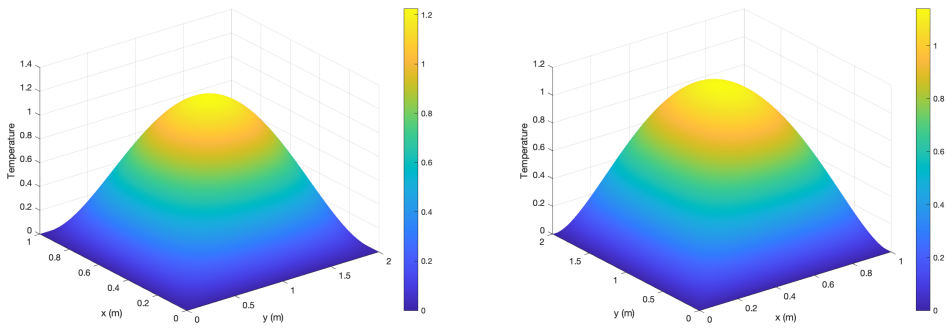
4. Additional Examples

4.1. 2D Steady State Example

In this section, we solve the steady state heat transfer equation in (x, y), illustrated below:

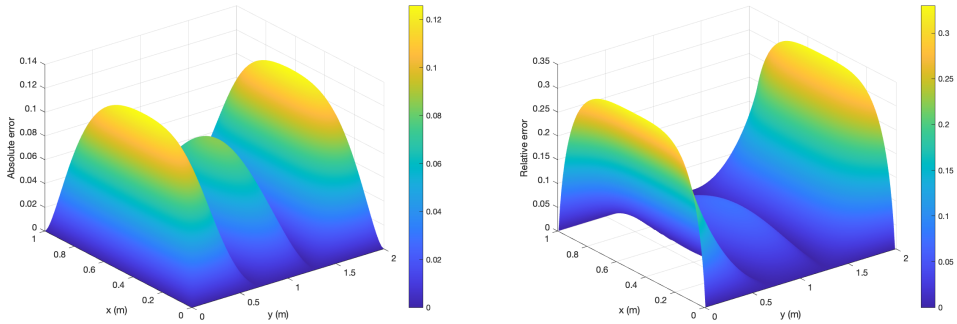
$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 10 \tag{27}$$

The problem is solved in the rectangular domain $x \in [0;1]$ and $y \in [0;2]$ along with 100 collocation points in x and 200 in y. The problem is also solved using 2D finite elements in a mesh involving $100 \times 200 = 20{,}000$ nodes in the 2D domain. The results are compared in Figure 9. The relative errors increase where the temperature is close to zero, but it is, in general, below 5% (as a consequence of the relative error definition), as illustrated in Figure 9.



(**a**) PINN–PGD solution for $n = 5$ (**b**) FEM solution

Figure 9. *Cont.*



(**c**) Absolute error (**d**) Relative error

Figure 9. Steady state 2D PINN–PGD solution and FEM solution for $\mu = 1\ \mathrm{m^2 s^{-1}}$ and $Q = 10$.

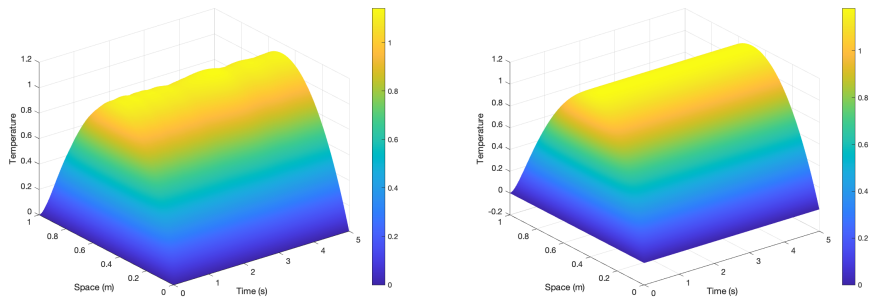
4.2. Nonlinear Example

We propose the following 1D nonlinear problem in (x, t):

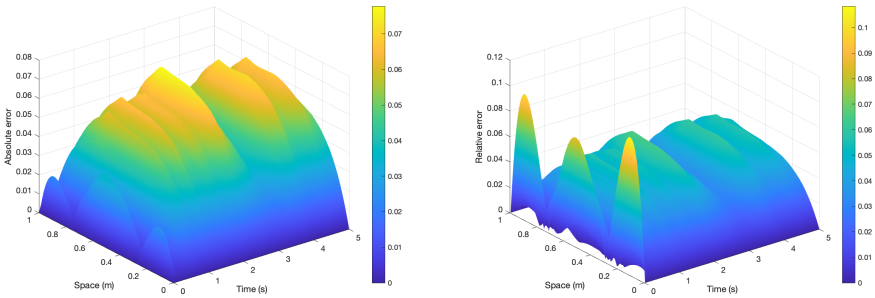
$$\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left(\mu(u)\frac{\partial u}{\partial x}\right) = 10 \tag{28}$$

The problem is solved for $x \in [0; 1]$ and $t \in [0; 5]$, along with 100 collocation points in x and 200 in t, using $\mu = 1 + 0.1u$. In this section, the loss given in Equation (18) is used, as the separation of loss in a sum of products in x and t dimensions becomes difficult.

The problem is also solved using incremental finite elements using 100 nodes in x and 200 time steps, while linearizing the problem using, for time step i, $\mu_i = \mu(u_{i-1})$. The results are compared in Figure 10. Again, we note the relative errors increase where the temperature is close to zero (as a consequence of the relative error definition), but it is, in general, below 5%, as illustrated in Figure 10.



(**a**) PINN–PGD solution for $n = 10$ (**b**) FEM solution



(**c**) Absolute error (**d**) Relative error

Figure 10. Nonlinear transient PINN–PGD solution and FEM solution for $\mu = 1 + 0.1u$ and $Q = 10$.

5. Conclusions

This work proposed a parsimonious PINN–PGD approach, able to construct the separated representation solution of a pPDE in a separated manner, without the need to reconstruct the full collocation points quadrature.

A novel monolithic method to learn the networks simultaneously computing the modes in different dimensions is proposed.The choice of the loss function being the square of the residuals allows for learning without the need to reconstruct the full dimensional space. The proposed approach allows for learning larger parametric problems without computer memory issues, alleviating the so-called curse of dimensionality.

The proposed formulation is parsimonious. The solution is built incrementally, on the fly, without prior need to know the involved number of modes in the solution. Moreover, only small networks can be used, as the solution can be continuously improved in a parsimonious manner by adding new networks after reaching the convergence of the previous ones.

Author Contributions: Conceptualization, C.G. and F.C.; Methodology, C.G. and F.C.; Software, C.G.; Validation, C.G. and F.C.; Formal analysis, C.G. and F.C.; Investigation, C.G.; Resources, F.C.; Writing—original draft, C.G.; Writing—review & editing, F.C.; Visualization, C.G.; Supervision, F.C.; Project administration, C.G. and F.C.; Funding acquisition, C.G. and F.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Acknowledgments: Authors acknowledge the support of the ESI and SKF research chair at ENSAM.

Conflicts of Interest: Author Francisco Chinesta was employed by the company CNRS@CREATE Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Chinesta, F.; Ammar, A.; Cueto, E. Recent Advances in the Use of the Proper Generalized Decomposition for Solving Multidimensional Models. *Arch. Comput. Methods Eng.* **2010**, *17*, 327–350. [CrossRef]
2. Ghnatios, C.; Rodriguez, S.; Tomezyk, J.; Dupuis, Y.; Mouterde, J.; Da Silva, J.; Chinesta, F. A hybrid twin based on machine learning enhanced reduced order model for real-time simulation of magnetic bearings. *Adv. Model. Simul. Eng. Sci.* **2024**, *11*, 3. [CrossRef]
3. Amsallem, D.; Zahr, M.; Choi, Y.; Farhat, C. Design optimization using hyper-reduced-order models. *Struct. Multidiscip. Optim.* **2015**, *51*, 919–940. [CrossRef]
4. Allery, C.; Beghein, C.; Hamdouni, A. Applying Proper Orthogonal Decomposition to the Computation of Particle Dispersion in a Two-Dimensional Ventilated Cavity. *Commun. Nonlinear Sci. Numer. Simul.* **2005**, *10*, 907–920. [CrossRef]
5. Ryckelynck, D.; Benziane, D.M. Multi-level A Priori Hyper-Reduction of Mechanical Models Involving Internal Variables. *Comput. Methods Appl. Mech. Eng.* **2010**, *199*, 1134–1142. [CrossRef]
6. Bernardi, C.; Maday, Y. Spectral Methods. *Handb. Numer. Anal.* **1997**, *5*, 209–485.
7. Atwell, J.A.; King, B.B. Proper Orthogonal Decomposition for Reduced Basis Feedback Controllers for Parabolic Equations. *Math. Comput. Model.* **2001**, *33*, 1–19. [CrossRef]
8. Ghnatios, C.; Fayazbakhsh, K. Warping estimation of continuous fiber-reinforced composites made by robotic 3D printing. *Addit. Manuf.* **2022**, *55*, 102796. [CrossRef]
9. Chinesta, F.; Keunings, R.; Leygue, A. *The Proper Generalized Decomposition for Advanced Numerical Simulations*; Springer: Cham, Switzerland, 2014. [CrossRef]
10. Chinesta, F.; Ammar, A.; Leygue, A.; Keunings, R. An Overview of the Proper Generalized Decomposition with Applications in Computational Rheology. *J. Non-Newton. Fluid Mech.* **2011**, *166*, 578–592. [CrossRef]
11. Cuomo, S.; Di Cola, V.S.; Giampaolo, F.; Rozza, G.; Raissi, M.; Piccialli, F. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *J. Sci. Comput.* **2022**, *92*, 88. [CrossRef]
12. Stiasny, J.; Chatzivasileiadis, S. Physics-informed neural networks for time-domain simulations: Accuracy, computational cost, and flexibility. *Electr. Power Syst. Res.* **2023**, *224*, 109748. [CrossRef]
13. Ghnatios, C.; Di Lorenzo, D.; Champaney, V.; Ammar, A.; Cueto, E.; Chinesta, F. Optimal trajectory planning combining model-based and data-driven hybrid approaches. *Adv. Model. Simul. Eng. Sci.* **2024**, *11*, 10. [CrossRef]

14. Ghnatios, C.; di Lorenzo, D.; Champaney, V.; Cueto, E.; Chinesta, F. Optimal velocity planning based on the solution of the Euler-Lagrange equations with a neural network based velocity regression. *Discret. Contin. Dyn. Syst.-S* **2024**, *17*, 2323–2333. [CrossRef]
15. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR, Sardinia, Italy, 13–15 May 2010; Volume 9, pp. 249–256.
16. Ghnatios, C.; Mathis, C.H.; Simic, R.; Spencer, N.D.; Chinesta, F. Modeling soft permeable matter with the proper generalized decomposition (PGD) approach, and verification by means of nanoindentation. *Soft Matter* **2017**, *13*, 4482–4493. [CrossRef]
17. Ghnatios, C. Modélisation Avancée des Procédés Thermiques Rencontrés Lors de la Mise en Forme des Composites. Ph.D. Thesis, Ecole Centrale Nantes, Nantes, France, 2012.