Model Order Reduction techniques for a parametric nonlinear elliptic problem

Athos Innocenti

***Abstract*—Given the objective of solving a parametric nonlinear elliptic problem, this work compares three Model Order Reduction techniques: Proper Orthogonal Decomposition (*POD*), Physics Informed Neural Networks (*PINNs*), and POD-NN. The methods are evaluated in terms of L^2 and H^1 errors, as well as computational Speed-Up. The results indicate that PINNs exhibit the highest errors among the methods, while POD-NN achieves the best Speed-Up, solving the problem significantly faster than POD with an accuracy almost comparable to the latter.**

I. Introduction

The numerical solution of parametric PDEs plays a crucial role in various scientific and engineering applications. These problems often involve high-dimensional parameter spaces and require multiple evaluations for different parameter values, leading to significant computational costs. Model Order Reduction techniques have emerged as a powerful approach to alleviate the computational burden associated with solving parametric PDEs. By reducing the dimensionality of the problem while preserving its essential features, MOR methods enable rapid and efficient approximations of the solution manifold. Among the most prominent MOR techniques are *Proper Orthogonal Decomposition*, data-driven methods such as *Physics Informed Neural Networks*, and hybrid approaches like *POD-NN*, which combines traditional projection-based methods with machine learning.

First, the parametric nonlinear elliptic problem is introduced. Then, a brief theoretical introduction to these methods is given. Finally, numerical results are presented and discussed.

II. Parametric nonlinear elliptic problem

Considering a two-dimensional spatial domain $\Omega = (0,\, 1)^2$, the aim is to solve the following parametrized problem: given $\boldsymbol{\mu} = (\mu_0,\, \mu_1) \in \mathcal{P} = [0.1,\, 1]^2$, find $u(\boldsymbol{\mu})$ such that

$$-\Delta u(\boldsymbol{\mu}) + \frac{\mu_0}{\mu_1}\big(e^{\mu_1 u(\boldsymbol{\mu})} - 1\big) = g(\mathbf{x};\boldsymbol{\mu})$$

with homogeneous Dirichlet boundary condition and the source term defined as

$$g(\mathbf{x};\boldsymbol{\mu}) = 100 \sin 2\pi x_0 \cos 2\pi x_1$$

$\forall\, \mathbf{x} = (x_0,\, x_1) \in \Omega$. Compactly, the problem reads

$$\begin{cases} -\Delta u(\boldsymbol{\mu}) + \frac{\mu_0}{\mu_1}\big(e^{\mu_1 u(\boldsymbol{\mu})} - 1\big) = g(\mathbf{x};\boldsymbol{\mu}) & \text{in } \Omega \\ u = 0 & \text{in } \partial\Omega \end{cases}$$

It is possible to derive the weak formulation of this problem by multiplication on both the left and right hand sides for a test function $v \in V$ and integration by parts on Ω while taking into account the homogeneous Dirichlet condition on the boundary. The latter formulation reads: find $u(\boldsymbol{\mu}) \in V := H_0^1(\Omega)$ such that

$$\int_\Omega \nabla u(\boldsymbol{\mu}) \cdot \nabla v + \frac{\mu_0}{\mu_1}\int_\Omega \big(e^{\mu_1 u(\boldsymbol{\mu})} - 1\big)v - \int_\Omega g(\mathbf{x};\boldsymbol{\mu})v = 0$$

$\forall\, v \in V$, $\forall\, \boldsymbol{\mu} \in \mathcal{P}$. In a more compact way, introducing

$$f_1(u, v; \boldsymbol{\mu}) = \int_\Omega \nabla u(\boldsymbol{\mu}) \cdot \nabla v$$

$$f_2(u, v; \boldsymbol{\mu}) = \frac{\mu_0}{\mu_1}\int_\Omega \big(e^{\mu_1 u(\boldsymbol{\mu})} - 1\big)v$$

$$f_3(u, v; \boldsymbol{\mu}) = -\int_\Omega g(\mathbf{x};\boldsymbol{\mu})v$$

it can be written as

$$f_1(u, v; \boldsymbol{\mu}) + f_2(u, v; \boldsymbol{\mu}) + f_3(u, v; \boldsymbol{\mu}) = 0$$

III. Newton scheme

However, the problem under analysis is nonlinear, which means that nonlinear terms will appear in its weak form representation, and these will need to be linearised in order to implement the model order reduction algorithms accordingly. In fact, using the last formulation mentioned, it can be seen that the term $f_2(u, v; \boldsymbol{\mu})$ is nonlinear. To overcome this issue the problem is linearized through the *Newton method*.

Starting from the functional

$$f(u, v; \boldsymbol{\mu}) := f_1(u, v; \boldsymbol{\mu}) + f_2(u, v; \boldsymbol{\mu}) + f_3(u, v; \boldsymbol{\mu})$$

and considering its Taylor expansion, it follows that at the *n*-th step of the Newton method the following equation holds

$$f(u_{n+1}) = f(u_n) + J_f[\delta u]\big|_{u_n}\delta u + O(||\delta u||^2)$$

where $u_{n+1} = u_n + \delta u$ is the estimated solution of the problem at the *n*-th iteration of the algorithm and $J_f[\delta u]\big|_{u_n}$ is the derivative J_f of f evaluated with respect to the function u_n along the unknown direction δu and it can be computed using the *Gateaux functional derivative* defined as

$$J_f[\delta u]\big|_{u_n} = \lim_{h\to 0}\frac{f(u_n + h\delta u) - f(u_n)}{h}$$

In the general scheme, since the Newton method looks for the zeros of f, the *n*-th iteration consists in finding the unknown δu such that

$$J_f[\delta u]\big|_{u_n}\delta u = -f(u_n, v; \boldsymbol{\mu}) \qquad \forall\, v \in V$$

It follows from the properties of the Gateaux derivative that if f_α is linear, then its Gateaux derivative is $f_\alpha(\delta u)$. Due to this, since $f_1(u, v; \boldsymbol{\mu})$ is linear, its Gateaux derivative is

$$J_{f_1}[\delta u]\big|_{u_n} = f_1(\delta u, v; \boldsymbol{\mu}) = \int_\Omega \nabla \delta u \cdot \nabla v$$

Moreover, since $f_3(u, v; \boldsymbol{\mu})$ depends only on g and v, its Gateaux derivative is

$$J_{f_3}[\delta u]\big|_{u_n} = 0$$

Finally, the Gateaux derivative of $f_2(u, v; \boldsymbol{\mu})$ is

$$\begin{aligned} J_{f_2}[\delta u]\big|_{u_n} &= \lim_{h\to 0} \frac{f_2(u_n + h\delta u) - f_2(u_n)}{h} \\ &= \frac{\mu_0}{\mu_1}\Big[\lim_{h\to 0}\frac{1}{h}\int_\Omega e^{\mu_1 u_n}\big(e^{\mu_1 h\delta u} - 1\big)v\Big] \\ &= \mu_0 \int_\Omega e^{\mu_1 u_n} v \end{aligned}$$

which is a reaction term. Putting everything together, knowing u_n, the n-th iteration consists in finding δu such that

$$\begin{aligned} \int_\Omega \nabla\delta u \cdot \nabla v + \mu_0 \int_\Omega e^{\mu_1 u_n} v = &- \int_\Omega \nabla u_n \cdot \nabla v + \\ &- \frac{\mu_0}{\mu_1}\int_\Omega \big(e^{\mu_1 u_n} - 1\big)v + \int_\Omega gv \end{aligned} \tag{1}$$

From a practical point of view, the Newton scheme has been implemented with a fixed tolerance of 10^{-5} and a maximum of 25 possible iterations. In addiction, since first and last terms in the equation do not depend on the solution u_n, they can be computed once.

IV. A priori error

From a validation perspective, it is convenient to perform an *a priori* error estimation on the problem, solving the nonlinear elliptic problem by choosing the possible exact high fidelity solution

$$u_{ex}(\mathbf{x}; \boldsymbol{\mu}) = 16x_0x_1(1 - x_0)(1 - x_1) \tag{2}$$

and computing the respective right hand side given by

$$\begin{aligned} g_{ex}(\mathbf{x}; \boldsymbol{\mu}) &= -\Delta u_{ex}(\mathbf{x}; \boldsymbol{\mu}) + \frac{\mu_0}{\mu_1}\Big(e^{\mu_1 u_{ex}(\mathbf{x};\boldsymbol{\mu})} - 1\Big) \\ &= 32x_0(1 - x_0) + 32x_1(1 - x_1) + \\ &\quad + \frac{\mu_0}{\mu_1}\Big(e^{\mu_1 16x_0x_1(1-x_0)(1-x_1)(\mathbf{x};\boldsymbol{\mu})} - 1\Big) \end{aligned} \tag{3}$$

Numerous theoretical results show that when the Galerkin method is used to discretize a variational problem of the form $a(u, v) = F(v)$ with $\mathbb{P}_k$ finite elements, and the solution u belongs to $H^r(\Omega)$ (where $r \leq k + 1$), the following error bound is satisfied

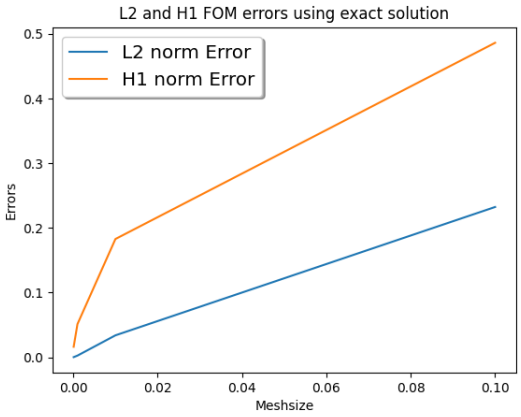
$$||u - u_h||_{1,\Omega} \leq Ch^{\frac{1}{2}(r-1)}|u|_{1,\Omega}$$

denoting by h the *mesh size* of an admissible and regular *triangulation* $\mathcal{T}$ of Ω. Thus, considering $\mathbb{P}_1$ finite elements and $u \in H^2(\Omega)$, two boundaries hold

$$\begin{aligned} \mathcal{E}_{L^2} &= ||u - u_h||_{L^2(\Omega)} \sim ch|u|_2 \\ \mathcal{E}_{H_0^1} &= ||u - u_h||_{H_0^1(\Omega)} \sim ch^{\frac{1}{2}}|u|_2 \end{aligned}$$

This implies that if these two errors are plotted against h on a logarithmic scale, a linear decay is expected, with the slopes $m_{L^2} = 1$ and $m_{H_0^1} = \frac{1}{2}$ corresponding to the convergence orders of the errors in the L^2 and H^1 norm, respectively.

In order to validate these theoretical results and to provide an estimate of the *a priori* error, the nonlinear elliptic problem has been solved using the Newton method applied to the Galerkin discretization of a finite element full order model. The exact solution (2) and the corresponding right hand side (3) were used. Since the two errors are defined as functions of the mesh size h, the solution was calculated for four different mesh sizes: $h \in \{0.1, 0.01, 0.001, 0.0001\}$. Under these conditions, the empirical orders of convergence in the L^2 and H^1 norms were found to be $m_{L^2} = 0.996$ and $m_{H_0^1} = 0.498$, respectively, which are in agreement with the expected theoretical predictions.



As shown in the figure above, the L^2 and H^1 errors decrease as the mesh size h decreases. The errors are plotted on a logarithmic scale.

V. Reduced order models

Given a generic parametric variational problem, the goal is to find a solution $u(\boldsymbol{\mu}) \in V$ such that

$$a(u(\boldsymbol{\mu}), v; \boldsymbol{\mu}) = F(v; \boldsymbol{\mu}) \quad \forall\, v \in V, \forall\, \boldsymbol{\mu} \in \mathcal{P}$$

where $\boldsymbol{\mu}$ are the parameters of the problem taking values in a properly defined parameter space $\mathcal{P}$. It is well known from theory that the Galerkin discretization method relies on a subspace $V_\delta = Span\{\phi_i\}_{i=1}^{N_\delta} \subset V$ of dimension $dim(V_\delta) = N_\delta \leq N = dim(V) < +\infty$ in order to discretize the initial problem into a new one which reads: find $u_\delta(\boldsymbol{\mu}) \in V_\delta$ such that

$$a(u_\delta(\boldsymbol{\mu}), v_\delta; \boldsymbol{\mu}) = F(v_\delta; \boldsymbol{\mu}) \quad \forall\, v_\delta \in V_\delta, \forall\, \boldsymbol{\mu} \in \mathcal{P}$$

where $u_\delta(\boldsymbol{\mu})$ is the so called *high fidelity solution* of the *full order problem*. A geometric interpretation of V_δ follows from the *Céas lemma*: $u_\delta(\boldsymbol{\mu}) \in V_\delta\ \forall\, \boldsymbol{\mu} \in \mathcal{P}$ is the best approximation of the exact solution $u(\boldsymbol{\mu}) \in V$ which can be obtained by fixing $\{V_\delta\}_{\delta\geq 0}$. In fact $||u(\boldsymbol{\mu}) - u_\delta(\boldsymbol{\mu})||_V \leq C_\alpha||u(\boldsymbol{\mu}) - v_\delta||_V$ holds $\forall\, v_\delta \in V_\delta$.

Defined the *discrete manifold*

$$\mathcal{M}_\delta = \{u_\delta(\boldsymbol{\mu}) \in V_\delta \mid \boldsymbol{\mu} \in \mathcal{P}\} \subset V_\delta$$

and the associated *discrete mapping*

$$S_\delta : \mathcal{P} \to V_\delta : \boldsymbol{\mu} \mapsto u_\delta(\boldsymbol{\mu})$$

the main focus of model order reduction is to build an approximation which is cheap to evaluate $\forall\, \boldsymbol{\mu} \in \mathcal{P}$. With the aim of approximating the map S_δ, the idea is to select a subset of parameter vectors $\{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_N\} \subset \mathcal{P}$ and to compute the corresponding high fidelity solutions

$$\{\omega_1 = u_\delta(\boldsymbol{\mu_1}), \dots, \omega_N = u_\delta(\boldsymbol{\mu_N})\} \subset V_\delta$$

called *snapshots*, which will be used to create a surrogate of S_δ called *low fidelity model* or *reduced order model*, indicated as S_N. Assuming that snapshots are linearly independent, a new subspace $V_N \subset V_\delta \subset V$ called *reduced space* is defined as

$$V_N = Span\{\omega_i\}_{i=1}^N$$

with $dim(V_N) = N << N_\delta \leq N$. Finally, the reduced order variational problem is formulated: find $u_N \in V_N$ such that

$$a(u_N(\boldsymbol{\mu}), v_N; \boldsymbol{\mu}) = F(v_N; \boldsymbol{\mu}) \quad \forall\, v_N \in V_N, \forall\, \boldsymbol{\mu} \in \mathcal{P}$$

As for V_δ, also for V_N it is possible to generate the *reduced manifold*

$$\mathcal{M}_N = \{u_N(\boldsymbol{\mu}) \in V_N \mid \boldsymbol{\mu} \in \mathcal{P}\} \subset V_N$$

and the *reduced mapping*

$$S_N : \mathcal{P} \to V_N : \boldsymbol{\mu} \mapsto u_N(\boldsymbol{\mu})$$

VI. Proper Orthogonal Decomposition

The main core of the Proper Orthogonal Decomposition method is to leverage *Galerkin orthogonality* and *Singular Value Decomposition* to construct a *basis* for V_N in order to construct a *low fidelity solution* $u_N(\boldsymbol{\mu})$.

A random set $\mathcal{P}_M = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_M\} \subset \mathcal{P}$ of parameters is supposed available and it is used to generate M high fidelity solutions $\{\omega_1 = u_\delta(\boldsymbol{\mu_1}), \dots, \omega_M = u_\delta(\boldsymbol{\mu_M})\}$ so that it is possible to build the *snapshot matrix*

$$\mathbf{W} = [\omega_1 \,|\, \omega_2 \,|\, \dots \,|\, \omega_M] \in \mathbb{R}^{N_\delta \times M}$$

with $rank(\mathbf{W}) \leq M$ so that they are not enough to build a basis. The idea is to remove all the unnecessary snapshots. Singular value decomposition is used to create a basis such that $\min_{W_{SVD}} ||\mathbf{W} - \mathbf{W_{SVD}}||^2 \leq \epsilon$ where ϵ is a fixed threshold. Since $\mathbf{W_{SVD}} = \mathbf{U\Sigma V}^\top$ and $\mathbf{U}$ columns are $\mathcal{R}(\mathbf{W_{SVD}}) = \mathbf{B} := \{\text{basis functions of } V_N\}$, $\mathbf{W_{SVD}}$ is the reduced basis approximation of $\mathbf{W}$.

A. POD algorithm

Practically, the POD can be divided into two phases: the *offline phase* and the *online phase*.

In the *offline phase*, a random set $\mathcal{P}_M = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_M\} \subset \mathcal{P}$ of parameters is generated. For each parameter $\boldsymbol{\mu}_i$, the corresponding high fidelity solution $u_\delta(\boldsymbol{\mu}_i)$ is computed by solving the problem (1) thanks to the Newton method. The M high fidelity solutions are then arranged as columns to form the snapshot matrix $\mathbf{W} = [\omega_1 \,|\, \dots \,|\, \omega_M]$. To introduce the inner product associated with the H_0^1-norm, the matrix $\mathbb{X}_\delta$ is defined as

$$[\mathbb{X}_\delta]_{ij} = (\nabla\phi_i, \nabla\phi_j)_{L^2} \tag{4}$$

where $\{\phi_i\}$ are the basis functions of the finite element space. Using this definition, the *covariance matrix* is computed as

$$\mathbf{C} = \mathbf{W}^\top \mathbb{X}_\delta \mathbf{W}$$

The eigenvalues and eigenvectors of $\mathbf{C}$, denoted by $\{(\lambda_i, \mathbf{v}_i)\}_{i=1}^M$, are then computed. The dimension N of the reduced space V_N is chosen as the largest integer satisfying

$$\frac{\sum_{i=1}^N \lambda_i}{\sum_{j=1}^M \lambda_j} \geq tol$$

where *tol* is a prescribed tolerance. Finally, the *basis matrix* $\mathbb{B}$ is constructed as

$$\mathbb{B} = [\psi_1 \,|\, \psi_2 \,|\, \dots \,|\, \psi_N] \quad \psi_i = \frac{\mathbf{W}\mathbf{v}_i}{\|\mathbf{W}\mathbf{v}_i\|_{H_0^1}} \quad 1 \leq i \leq N$$

This basis matrix $\mathbb{B}$ is then used to project and reduce the high fidelity system, enabling a significant reduction in computational complexity while preserving accuracy.

In the *online phase*, considering fixed parameters $\boldsymbol{\mu} \in \mathcal{P}$, during the k-th iteration of the Newton scheme, the low fidelity solution of the problem is obtained by following these steps:

1) Compute full matrices $\mathbf{A}_\delta(\boldsymbol{\mu})$, $\mathbf{R}_\delta(\boldsymbol{\mu})$ and forcing terms $\mathbf{g}_{i\delta}(\boldsymbol{\mu})$ associated to the high fidelity problem
2) Reduce the linear system with the basis matrix

$$\mathbf{A}_N(\boldsymbol{\mu}) = \mathbb{B}^\top \mathbf{A}_\delta(\boldsymbol{\mu})\mathbb{B}$$
$$\mathbf{R}_N(\boldsymbol{\mu}) = \mathbb{B}^\top \mathbf{R}_\delta(\boldsymbol{\mu})\mathbb{B}$$
$$\mathbf{g}_{iN}(\boldsymbol{\mu}) = \mathbb{B}^\top \mathbf{g}_{i\delta}(\boldsymbol{\mu})$$

3) Solve the reduced system in order to find the increment $\delta\mathbf{u}_N^k$ used to update the reduced solution

$$\mathbf{u}_N^{k+1} = \mathbf{u}_N^k + \delta\mathbf{u}_N^k$$

4) Project back the reduced solution into the discrete space V_δ using the basis matrix

$$\mathbf{u}_{\delta,\, proj}^{k+1} = \mathbb{B}\mathbf{u}_N^{k+1}$$

the projected solution will be used to build the matrices and the forcing terms associated to the high fidelity problem in the next step of the Newton method.

B. Numerical results

The POD method has been implemented on a triangulation $\mathcal{T}$ of the domain Ω with a mesh size of $h = 0.001$.

As a preliminary step, an *a priori* error estimation was performed and the full order model was compared with the reduced order one, using the exact solution (2). From this analysis, the empirical convergence orders (as functions of h) in the L^2 and H^1 norms were determined to be $m_{L^2} = 0.936$ and $m_{H_0^1} = 0.467$, respectively. Additionally, for a test set of 50 parameters, the average relative error was found to be 3.216×10^{-6}.

Following this initial evaluation, the analysis of the POD method without relying on the exact solution was carried out. During the offline phase, a random set $\mathcal{P}_M$ of $M = 300$ parameters was generated, and the corresponding high fidelity

solutions $\{u_\delta(\boldsymbol{\mu}_i)\}_{i=1}^{300}$ were computed to construct the snapshot matrix. Using the induced inner product and the resulting covariance matrix, the basis matrix $\mathbb{B}$ was then constructed, considering a maximum of $N_{max} = 10$ basis functions and a tolerance of $tol = 10^{-7}$.

Finally, during the online phase, the reduced solution $u_N(\boldsymbol{\mu})$ was computed for fixed parameters $(\mu_0, \mu_1) = (0.8, 0.4)$ by solving the reduced system using the Newton method.

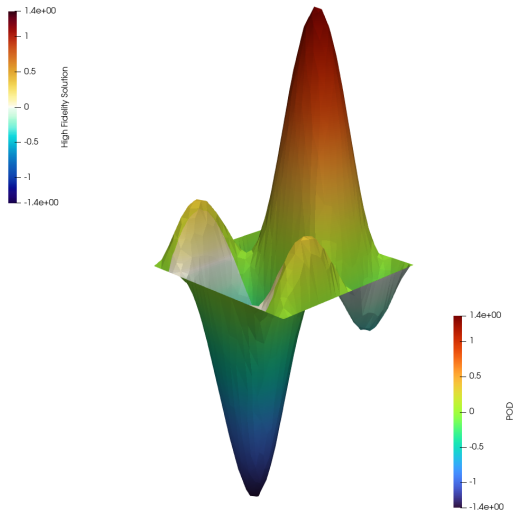


Fig. 1. High fidelity solution compared with the POD reduced one

The resultant reduced solution $u_N(\boldsymbol{\mu})$ is depicted in Figure 1 alongside the high fidelity solution. It can be inferred that the POD method offers a satisfactory approximation of the high fidelity solution when evaluated visually (*norm of the eye*). The POD solution effectively captures the main features of the high fidelity solution, such as peaks and valleys, preserving the global behaviour. However, localized discrepancies may occur in regions with steep gradients, where finer details are harder to approximate. Visually, the amplitude range is well preserved, but slight deviations in surface shapes can be observed in areas with pronounced features.

To provide a more rigorous assessment of the accuracy of the POD solution compared to the high fidelity one, an error analysis has been conducted. This analysis evaluates the discrepancies between the two solutions in terms of both the L^2-norm, which measures global differences, and the H^1-norm, which emphasizes variations in gradients. This ensures a comprehensive understanding of the approximation's quality across Ω. Given a test set $\mathcal{P}_{\text{test}}$ consisting of 20 randomly selected parameter pairs $(\mu_0, \mu_1) \in \mathcal{P}$, both the low fidelity and high fidelity solutions were computed. These solutions were then used to evaluate the absolute and relative errors in both the L^2 and H^1 norms. Relative errors are defined as

$$\mathcal{E}_{L^2}^{rel} = \frac{||u_\delta(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})||_{L^2(\Omega)}}{||u_\delta(\boldsymbol{\mu})||_{L^2(\Omega)}} \tag{5}$$

$$\mathcal{E}_{H_0^1}^{rel} = \frac{||u_\delta(\boldsymbol{\mu}) - u_N(\boldsymbol{\mu})||_{H_0^1(\Omega)}}{||u_\delta(\boldsymbol{\mu})||_{H_0^1(\Omega)}} \tag{6}$$

The resulting average relative errors obtained were

$$\mathcal{E}_{L^2}^{rel} = 3.66 \cdot 10^{-4} \qquad \mathcal{E}_{H_0^1}^{rel} = 2.22 \cdot 10^{-4}$$

Finally, as the primary objective of reduced order models is to decrease the computational time and costs associated with the classical Galerkin method, the *Speed-Up* ratio was analysed, defined as follows

$$SpeedUp = \frac{T_{FOM}}{T_{ROM}} \tag{7}$$

having T_{FOM} and T_{ROM} the times needed to compute the high fidelity solution in the full order model (generally slower due to its computational complexity) and the low fidelity solution in the reduced order model (a simplified version and therefore quicker to compute), respectively. The empirical ratio obtained was

$$SpeedUp_{POD} = 0.84$$

Thus the POD does not provide a significant reduction in computational time compared to the full order model, which turns out to be slightly faster due to the nonlinearity of the problem.

VII. Physics Informed Neural Networks

A Physics Informed Neural Network is a neural network designed to solve problems governed by differential equations by embedding physical laws directly into the learning process. Instead of relying solely on data, PINNs minimize a loss function that incorporates the residuals of the differential equation and boundary condition constraints. The network receives as inputs both domain points $\mathbf{x} \in \Omega$ and parameters $\boldsymbol{\mu} \in \mathcal{P}$, and generates an output: the approximated solution $u(\mathbf{x}, \boldsymbol{\mu})$. The network is trained by minimizing a loss function that includes the residuals of the governing PDE at the interior points of the domain and ensures the satisfaction of the boundary conditions at the domain's boundaries. Given a PINN $\tilde{\omega}(\mathbf{x}, \boldsymbol{\mu})$, the loss function is defined as the sum of two components

$$MSE \doteq MSE_{residual}(\boldsymbol{\mu}) + \lambda \cdot MSE_{boundary}(\boldsymbol{\mu})$$

where the boundary MSE is

$$MSE_{boundary}(\boldsymbol{\mu}) \doteq \frac{1}{N_b} \sum_{i=1}^{N_b} |\tilde{\omega}(\mathbf{x}_i^b, \boldsymbol{\mu}_i^b) - u(\mathbf{x}_i^b, \boldsymbol{\mu}_i^b)|^2$$

for $(\mathbf{x}_i^b, \boldsymbol{\mu}_i^b) \in \partial\Omega \times \mathcal{P}$. The residual MSE is the physical part

$$MSE_{residual}(\boldsymbol{\mu}) \doteq \frac{1}{N_r} \sum_{i=1}^{N_r} |\mathcal{R}(\tilde{\omega}(\mathbf{x}_i^r, \boldsymbol{\mu}_i^r))|^2$$

where $(\mathbf{x}_i^r, \boldsymbol{\mu}_i^r) \in \Omega \times \mathcal{P}$ and $\mathcal{R}$ is the *residual* of the PDE.

λ is an hyper parameter used to equilibrate the two losses and it was fixed to the value 2750 in order to ensure that they are of the same order of magnitude.

The architecture of the PINN consists of $L = 4$ hidden layers, with the *sigmoid* function serving as the *activation function*. This function is *continuous*, *non-affine*, and belongs to the class $\mathcal{C}^\infty$. According to the *Universal Approximation Theorem*, the input layer has a dimension of $N_1 = n = 4$, corresponding to the position $\mathbf{x} = (x_1, x_2)$ and the parameters $\boldsymbol{\mu} = (\mu_1, \mu_2)$, while the output layer has a dimension of $N_L = m = 1$, representing $u(\mathbf{x}, \boldsymbol{\mu})$. Consequently, each hidden layer $l \in \{2, \dots, L-1\}$ contains a fixed number of nodes

$$N_l = n + m + 2 = 7$$

A. Training of the network

To train the PINN, three distinct training sets were randomly generated, each with the same cardinality TRAINING SIZE = 4000. The first set consists of interior points, i.e. points located within the domain Ω. The second set is composed of boundary points, which lie on the frontier $\partial\Omega$ and are used to enforce the boundary conditions. Finally, the third set contains parameter values sampled from the parametric space $\mathcal{P}$. The network training process was carried out in such a way that both overfitting and underfitting were avoided.

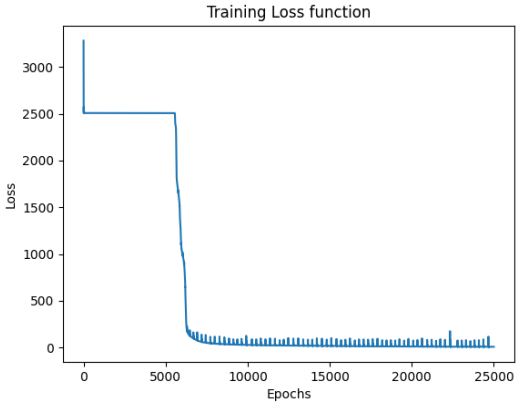


Fig. 2. Training loss function of the PINN

After a series of trials and simulations conducted in order to optimise the learning process, the learning rate was set at 0.01 (corresponding to the lowest averaged value of the loss) and the maximum number of possible iterations was set to 25000.

B. Numerical results

As done for the POD, a discretization of the domain Ω with a mesh size of $h = 0.001$ was considered. The reduced solution $u_N(\boldsymbol{\mu})$ was then predicted for fixed parameters $(\mu_0, \mu_1) = (0.8, 0.4)$ on the degrees freedom of the mesh, using the trained PINN.

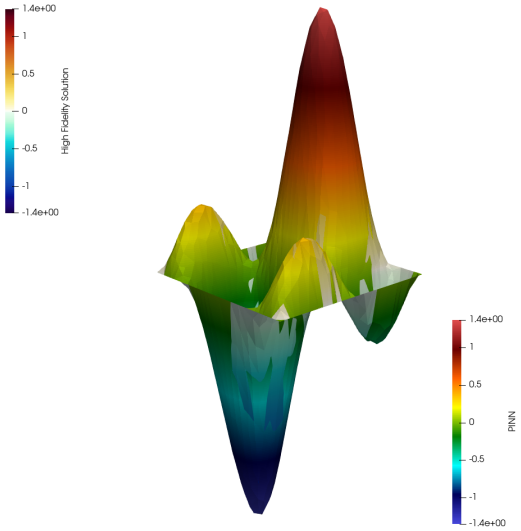


Fig. 3. High fidelity solution compared with the PINN reduced one

The predicted reduced solution $u_N(\boldsymbol{\mu})$ is illustrated in Figure 3, presented alongside the high fidelity solution computed for the same parameter values. The comparison between the high fidelity solution and the PINN predicted one shows that the PINN successfully captures the overall structure and main features of the target function. However, noticeable discrepancies may arise in regions with steep gradients or rapid variations, where the neural network struggles to fully replicate the high fidelity details. Boundary conditions appear to be accurately enforced, demonstrating the network's ability to integrate such constraints during training. Furthermore, it can be said that the solution predicted by PINN seems to be less accurate than the one predicted by POD.

To further investigate the accuracy of the PINN solution, an error analysis was performed. Using a test set of 100 randomly selected parameter pairs, both the low fidelity and high fidelity solutions were computed. These solutions were then used to evaluate the metrics defined in (5) and (6), resulting in the following average relative errors:

$$\mathcal{E}^{rel}_{L^2} = 1.44 \cdot 10^{-2} \qquad \mathcal{E}^{rel}_{H^1_0} = 5.05 \cdot 10^{-2}$$

confirming what had already been estimated using the norm of the eye: the solution predicted by PINN approximates the high fidelity solution rather crudely and is considerably less accurate than that estimated using the POD.

Lastly, the *Speed-Up* ratio (7) was computed, obtaining

$$SpeedUp_{PINN} = 158$$

proving that the PINN provides a significant computational efficiency, making it highly suitable for applications requiring rapid simulations. However, this performance gain must be evaluated alongside the accuracy of the solution.

VIII. POD-NN

The POD-NN combines the efficiency of Proper Orthogonal Decomposition with the flexibility of neural networks. POD is used to extract a low dimensional representation of the solution space by identifying a set of optimal basis functions. The neural network is used to approximate the reduced dynamics by mapping parameter inputs directly to the coefficients of the reduced basis. In summary, the POD-NN technique involves constructing the reduced order model space and predicting the reduced solution $u_N(\boldsymbol{\mu})$. The reduced space V_N is generated using POD. A neural network $\Pi^{NN}(\boldsymbol{\mu}) : \mathcal{P} \to V_N$ is then trained to take the parameters as input and predict the corresponding reduced solution. Finally, the full order discrete solution $u_\delta(\boldsymbol{\mu})$ is reconstructed from the reduced representation. This method enhances the efficiency of the POD algorithm in handling nonlinear problems by removing the projection steps usually carried out during each iteration of the Newton method, which are responsible for higher computational costs and extended processing time.

A. Feed forward neural network and its training

The *feed forward* network is composed of 4 hidden layers with $N_l = 30$ nodes. The input layer has a dimension of $N_1 = 2$ corresponding to the parameters $\boldsymbol{\mu} = (\mu_1, \mu_2)$. The

output layer has a dimension of $N_L = |\mathcal{R}(\mathbb{B})| = |V_N|$. The activation function used in the hidden layers is *tanh*, whereas the identity was used in the final layer.

To train the FFNN, a training set $\mathcal{P}_{train}$ consisting of 300 randomly selected parameter pairs was generated by sampling from the parameter space $\mathcal{P}$. The corresponding reduced solutions were computed and assembled into the snapshot matrix. Subsequently, the covariance matrix, the inner product matrix, and the basis matrix $\mathbb{B}$ were constructed. Following this, the reduced inner product was computed using the inner product matrix and the basis matrix. Finally, the training set was built by projecting the snapshot matrix onto the reduced space using the reduced inner product.

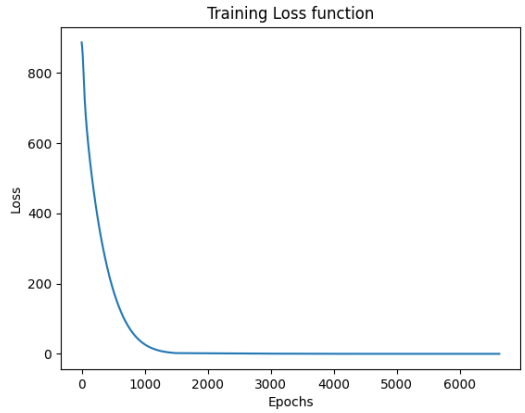


Fig. 4. Training loss function for the FFNN

As a final step, the neural network was trained with a learning rate initially set to 0.001 (and reduced to 0.0001 after 1500 iterations), a tolerance of 10^{-6}, and a maximum limit of 20000 possible iterations.

B. POD-NN algorithm

The loss function for the feed forward network is defined as the *mean squared error* of the predicted reduced solution and the projection of u_δ

$$\mathcal{L} = \sum_{\boldsymbol{\mu}\in\mathcal{P}_{train}} ||\Pi^{NN}(\boldsymbol{\mu}) - \mathbf{P_g} u_\delta(\boldsymbol{\mu})||^2$$

From theory, it can be proven that the optimal projection with respect to the *energy norm*, is the projection $\mathbf{P_g}$ given by

$$\mathbf{P_g} u_\delta(\boldsymbol{\mu}) = \mathbb{B}(\mathbb{B}^\top \mathbb{X}_\delta \mathbb{B})^{-1} \mathbb{B}^\top \mathbb{X}_\delta u_\delta(\boldsymbol{\mu})$$

where $\mathbb{B}$ is the basis matrix of V_N, $\mathbb{X}_\delta$ is the inner product matrix for V_δ (4), $\mathbb{B}^\top \mathbb{X}_\delta \mathbb{B} = \mathbb{X}_N$ is the reduced version of the inner product matrix, and $\mathbf{P_g} u_\delta(\boldsymbol{\mu}) = \mathbb{B} u_N(\boldsymbol{\mu})$ so that

$$u_N(\boldsymbol{\mu}) = (\mathbb{B}^\top \mathbb{X}_\delta \mathbb{B})^{-1} \mathbb{B}^\top \mathbb{X}_\delta u_\delta(\boldsymbol{\mu})$$

moving $(\mathbb{B}^\top \mathbb{X}_\delta \mathbb{B})^{-1}$ to the left

$$\mathbb{X}_N u_N(\boldsymbol{\mu}) = \mathbb{B}^\top \mathbb{X}_\delta u_\delta(\boldsymbol{\mu})$$

The algorithm can be organized into two phases as the classic POD: the *offline phase* and the *online phase*.

The *offline phase* consists of three steps

1) Build the reduced basis matrix $\mathbb{B}$ for the reduced space V_N executing the *offline phase* of the POD
2) Solve the linear system $\mathbb{X}_N u_{N_i} = \mathbb{B}^\top \mathbb{X}_\delta u_\delta(\boldsymbol{\mu})$ for all the snapshots collected before
3) Train the FFNN $\Pi^{NN}(\boldsymbol{\mu})$ with the reduced solutions $\{u_{N_i}\}_{i=1}^N$ of the snapshots

During the *online phase*, given a new parameters $\boldsymbol{\mu}^*$, the reduced solution is predicted by the trained FFNN as $u_N(\boldsymbol{\mu}^*) = \Pi^{NN}(\boldsymbol{\mu}^*)$, and projected back to the discretized space V_δ as $u_\delta^{prj} = \mathbb{B}\Pi^{NN}(\boldsymbol{\mu}^*)$.

C. Numerical results

As done for the previous methods, a discretization of the domain Ω with a mesh size of $h = 0.001$ was considered. The reduced solution $u_N(\boldsymbol{\mu})$ was then predicted for fixed parameters $(\mu_0, \mu_1) = (0.8, 0.4)$ using the trained FFNN.

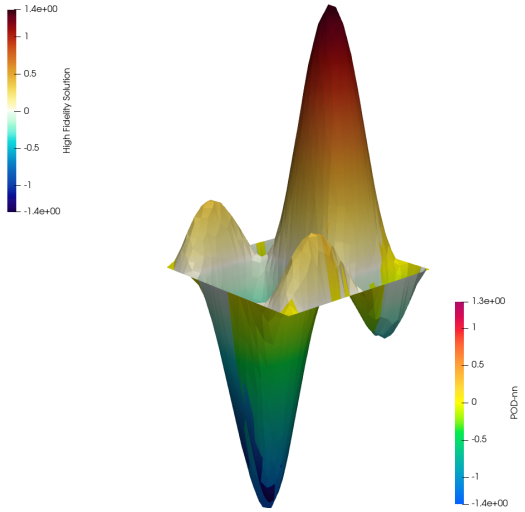


Fig. 5. High fidelity solution compared with the POD-NN reduced one

The reduced solution $u_N(\boldsymbol{\mu})$ is illustrated in Figure 5, alongside the high fidelity solution computed for the same parameter values. The visualization demonstrates that the POD-NN performs well in approximating the high fidelity solution, effectively capturing the global structure and main features of the surface. However, slight discrepancies are noticeable in regions with rapid variations or extreme values, indicating a slightly reduced accuracy in representing fine local details. Overall, the performance of the POD-NN appears promising.

To be more accurate, an error analysis was performed. Using a test set of 100 randomly selected parameter pairs, both the low fidelity and high fidelity solutions were computed and used to evaluate the relative errors (5) and (6), resulting in the following results:

$$\mathcal{E}^{rel}_{L^2} = 7.88 \cdot 10^{-3} \qquad \mathcal{E}^{rel}_{H^1_0} = 5.31 \cdot 10^{-3}$$

pointing out the strong predictive capability of the POD-NN approach, demonstrating its capacity to deliver accurate approximations of the high fidelity solution while effectively capturing its gradient behaviour, despite a slightly lower precision in the overall approximation (L^2 error).

Lastly, the *Speed-Up* ratio (7) was computed, obtaining

$$SpeedUp_{POD-NN} = 1235$$

emphasizing its efficiency in reducing computational time while maintaining a high level of accuracy.

IX. Conclusions

Taking into account the numerical results obtained, it is possible to compare the performance of the three methods mentioned.

	L^2 norm Error	H^1 norm Error	Speed-Up
POD	$3.66 \cdot 10^{-4}$	$2.22 \cdot 10^{-4}$	0.84
PINN	$1.44 \cdot 10^{-2}$	$5.05 \cdot 10^{-2}$	158
POD-NN	$7.88 \cdot 10^{-3}$	$5.31 \cdot 10^{-3}$	1235

It is clear that the POD algorithm offers the highest accuracy in terms of relative error in both the L^2 and H^1 norms, relative to the high fidelity solution, as shown in the table below. However, due to the fact that the problem in nonlinear, POD does not provide a significant reduction in computational time compared to the full order model, as it is more efficient under the affine hypothesis. This can be deduced from the fact that the Speed-up of POD is only 0.84. On the other hand, the PINN approach shows a faster inference time, but at the cost of reduced accuracy, with errors two orders of magnitude higher than those obtained with the POD. Finally, the POD-NN method strikes the best balance, delivering results that are almost as accurate as the POD, but with a significantly lower inference time, represented by the highest value of the Speed-Up. In conclusion, PINN is the least effective reduced order model with the worst relative errors, and the POD-NN is the most efficient choice for approximating the reduced solution.

X. Code

GitHub repository: code