A hybrid Reduced Basis and Machine-Learning algorithm for building Surrogate Models: a first application to electromagnetism

Alejandro Ribés, Ruben Persicot, Lucas Meyer, Jean-Pierre Ducreux

To cite this version:

Alejandro Ribés, Ruben Persicot, Lucas Meyer, Jean-Pierre Ducreux. A hybrid Reduced Basis and Machine-Learning algorithm for building Surrogate Models: a first application to electromagnetism. Workshop on Machine Learning and the Physical Sciences workshop at NeurIPS 2022, Dec 2022, New Orleans, United States. pp.1-6. hal-03850571

HAL Id: hal-03850571
https://hal.science/hal-03850571
Submitted on 14 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A hybrid Reduced Basis and Machine-Learning algorithm for building Surrogate Models: a first application to electromagnetism

Alejandro Ribés
Industrial AI Laboratory SINCLAIR
EDF Lab Paris-Saclay
91120 Palaiseau, France
alejandro.ribes@edf.fr

Ruben Persicot
Ecole des Ponts ParisTech
Marne-la-Vallée, France
ruben.persicot@eleves.enpc.fr

Lucas Meyer
Industrial AI Laboratory SINCLAIR
EDF Lab Paris-Saclay, Univ. Grenoble Alpes,
Inria, CNRS, Grenoble INP, LIG
lucas.meyer@inria.fr

Jean-Pierre Ducreux
EDF Lab Paris-Saclay
91120 Palaiseau, France
jean-pierre.ducreux@edf.fr

Abstract

A surrogate model approximates the outputs of a Partial Differential Equations (PDEs) solver with a low computational cost. In this article, we propose a method to build learning-based surrogates in the context of parameterized PDEs, which are PDEs that depend on a set of parameters but are also temporal and spatial processes. Our contribution is a method hybridizing the Proper Orthogonal Decomposition and several Support Vector Regression machines. We present promising results on a first electromagnetic use case (a primitive single-phase transformer).

1 Introduction

In the context of numerical simulation, a surrogate model approximates the outputs of a solver with a low computational cost. Solvers of differential equations, for instance, those based on finite element methods, often require long runs. Thus, they are not well-suited for real-time applications, prediction, or the resolution of inverse problems requiring multiple executions. In the last five years, the use of deep learning for constructing surrogate models has gained a lot of attention from industry and academics. These surrogates learn from simulation results and/or experimental data. One important example is the seminal work of Raissi et al. [9], which introduces physics-informed neural networks (PINNs). In the same context, other techniques have been proposed. When the meshes supporting the numerical simulations are regular the use of traditional deep learning algorithms for image analysis tasks is possible. For instance, U-Net architectures are employed in [11,12], or auto-encoders in [6]. For non-regular meshes Graph Neural Networks (GNNs) [2,1] have been used. As an example, Deep-Mind recently introduced a GNN-based framework for learning mesh-based simulations [7,10]. Numerous methods for building surrogates have been proposed, we just cited some examples to show their variety.

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.
In this article, we propose a method to build surrogates in the context of parametrized Partial Differential Equations (PDEs), which are PDEs that depend on a set of parameters (or indexed by a parameter vector that we call $\lambda$). The Reduced Order Model (ROM) community has traditionally tackled this problem. In fact, a ROM can be considered a type of surrogate model. Quarteroni et al. [8] give a complete introduction to ROMs in which it is evident that constructing surrogates for parameterized PDEs is, in the general case, a very complex task. Technical literature on the use of machine learning techniques for parameterized PDEs is, to our knowledge, much less numerous than for building surrogates of single spatio-temporal simulations.

Our contribution is a method hybridizing the Proper Orthogonal Decomposition (POD, see chapter 11 of [3]) and several Support Vector Regression machines (SVR, see [4]). This method is non-invasive because we do not perform Galerkin projection, the Reduced Basis obtained by the POD is only used to facilitate the training of our Machine-Learning system. Some approaches to building data-driven surrogates mimic the solver iterative process: they infer the next state of the physical system given its previous one. We take a less popular approach by directly inferring the state from a time input. Our system accepts as inputs the time and the vector of parameters to output the current spatio-temporal and parametric state of the system. We present promising results on a first electromagnetic use case (a primitive single-phase transformer).

2 Proposed Method

This section presents the technical details of the conceived algorithm that combines model reduction via the POD and support vector regression (SVR). It consists of two main steps: first, a reduced basis is found using the POD, and second several SVRs are trained. We remark that prior to these two steps an ensemble of $N_h$ high-fidelity simulations should be run. It is important to note that we allow using a different number of simulations in our two steps: $N_{POD}$ simulations for finding the reduced basis and $N_{SVR}$ for training the SVRs. This can have a great impact on computing time when a large number of simulations are treated.

2.1 Finding a Reduced Basis

We obtain a Reduced Basis using the Proper Orthogonal Decomposition (POD), see chapter 11 of [3] or chapter 6 of [8]. The fundamental step of the POD is the application of a Singular Value Decomposition [5] to a so-called snapshot matrix. Thus we define here how we construct this matrix for parameterized problems.

**Snapshot matrix for a parametric problem:** When dealing with parametric and spatio-temporal problems, we have an ensemble of $N$ spatio-temporal simulations. In this case, we can build the snapshot matrix by defining fields indexed by $(\lambda, t)$, which represents the field of interest provided by the solver in a linearized vector. We thus build a matrix of snapshots $X$ of size $(n, m)$ by concatenating each of these vectors as presented below. Note that $n$ is equal to the number of mesh nodes used to perform the simulations and that $m = N_h T$ (number of high fidelity simulations $N_h$ by number of time steps $T$).

$$X = \begin{bmatrix} X_{\lambda_1, t_1} & \cdots & X_{\lambda_1, t_T} & \cdots & X_{\lambda_N, t_1} & \cdots & X_{\lambda_N, t_T} \end{bmatrix}$$ (1)

Each column $X_{\lambda, t}$ of $X$ is associated with a vector $x_{i,j} = (\lambda_i, t_j)$. We can form a matrix $x$ of parameters from these vectors. In this article, we use $N_{POD} < N_h$ thus we will denote $X$ by $X_{POD}$ to indicate the snapshot matrix used for finding the reduced basis.

**SVD:** A description of the singular value decomposition (SVD) can be found in any introductory linear algebra book, such as [5]. Let $X \in \mathbb{R}^{n \times m}$, $U \in \mathbb{R}^{n \times n}$, $V \in \mathbb{R}^{m \times m}$, $\Sigma \in \mathbb{R}^{n \times m}$ the SVD of $X$ is the decomposition $X = U \cdot \Sigma \cdot V^T$, where $U$ and $V$ are unitary matrices and $\Sigma$ is a diagonal matrix containing the singular values of $X$, which are ordered by decreasing value.

Applying a Singular Value Decomposition to the snapshot matrix allows for finding an orthogonal basis. However, this basis is of the same size as the original non-transformed problem. The key to finding a Reduced Basis (RB) comes from the fact that not all the principal components need to be
kept. Keeping only the first $r$ principal components, produced by using only the first $r$ eigenvectors, gives the truncated transformation. The value $r$ is typically found by looking at the accumulated energy (also called accumulated variance), which is defined by:

$$E = \frac{\sigma_1 + \sigma_2 + \ldots + \sigma_r}{\sigma_1 + \sigma_2 + \ldots + \sigma_m}$$

(2)

where the $\sigma_i$ ($i = 1 \ldots m$) values are the diagonal elements of the matrix $\Sigma$. Once the value $r$ is chosen we obtain the following approximation of the matrix $X$:

$$X^r = U^r \cdot \Sigma^r \cdot V^{Tr}.$$  

(3)

2.2 Training the SVRs

The SVR being a supervised learning algorithm, it is necessary to constitute (input, output) pairs for its training. In figure 1, we depict what a single SVR takes in and out. The SVR accepts $(t, \lambda)$ as inputs, where $t$ is a time step and $\lambda$ a vector of parameters. The SVR outputs a prediction $\hat{c}_i$, corresponding to the i-th coefficient on the reduced space, $i = 1 \ldots r$.

![Figure 1: A SVR takes as input a time step $t$ and one or several parameters $\lambda$. It outputs a prediction $\hat{c}_i$, corresponding to the i-th coefficient on the reduced space, $i = 1 \ldots r$.](image)

Preparation for the training phase. Our idea is to project a snapshot matrix $X_{SVR}$ into the reduced space $C$, both these matrices contain $N_{SVR} < N_h$ simulations. $X_{SVR}$ is a sub-sampled version of the $X$ presented in equation 2.2 and thus present the same encoding. In order to project this snapshot matrix in the reduced space we use $U^r$ from equation 3. Strictly speaking we should call this projection matrix $U_{POD}^r$ because $X_{POD}^r = U_{POD}^r \cdot \Sigma_{POD}^r \cdot V_{POD}^{Tr}$. The operation $C = U_{POD}^r \cdot X_{SD}$ projects the snapshot matrix on the reduced space. However, we need not only a matrix for training but also a matrix for validation. The matrix $X_{SVR}$ is therefore subdivided into $X_{train}$ and $X_{val}$, thus $X = [X_{train} | X_{val}]$. We can then construct the corresponding matrices of coefficients $C_{train}$ and $C_{val}$ by matrix product: $C = [ C_{train} = U_{POD}^r \cdot X_{train} | C_{val} = U_{POD}^r \cdot X_{val} ]$.

At this point, we have defined how to form the training and validation sets (matrices in this case) for the outputs of the SVR. In figure 1, we observe that the SVR accepts $(t, \lambda)$ as inputs. These inputs can be coded in the following matrix:

$$x = \begin{bmatrix} t_1 & t_2 & \ldots & t_p & \ldots & t_1 & t_2 & \ldots & t_p \\ \lambda_1 & \lambda_1 & \ldots & \lambda_1 & \ldots & \lambda_N & \lambda_N & \ldots & \lambda_N \end{bmatrix}$$

(4)

where each column $X_{\lambda,t_j}$ of $X$ (in equation ) is associated with a vector $x_{i,j} = (\lambda_i, t_j)$. Thus the matrix $x$ encodes the time and parameters in exactly the same way as $X$. Similarly, $x$ is therefore subdivided into $x_{train}$ and $x_{val}$, thus $x = [ x_{train} | x_{val} ]$. Now it is possible to constitute the training and validation datasets which are respectively $(x_{train}, C_{train})$ and $(x_{val}, C_{val})$.

Training of $r$ SVRs The second stage of the training phase is to train $r$ SVRs, one for each line of $C_{train}$. For example, the first SVR is trained to predict the first line of $C_{train}$ from $x_{train}$. In other words, we can say that the i-th SVR is led to predict the value of the parametoro-temporal coefficients of the i-th mode in the reduced space $U_{pod}^r$, from the parameter values contained in $x_{train}$. We note that it is in general necessary to center and reduce the training and validation data sets before training the SVRs.
3 Use case: a single-phase transformer

Our use case is composed of a cylinder and two windings. A winding is one or more turns of wire that form a continuous coil through which an electric current can pass. As can be seen in Figure 2, only one 2D section of this set is studied, thus the cylinder becomes a torus and the coils are represented by rectangles. The cylinder is made of metallic material with non-linear permeability. It is characterized by two Frolich coefficients: \( \alpha = 0.00025 \) and \( \beta = 0.00018 \). The part of the domain not containing the cylinder contains air. The sections of the coils are represented by rectangles. The electromagnetic problem (a primitive single-phase transformer) is magnetostatic. The wires of each coil are oriented as shown in Figure 2. A dot indicates that the current is circulating towards the reader. Each coil has a section equal to \( 50e^{-3} \cdot 100e^{-3} \text{m}^2 = 0.005 \text{m}^2 \) and is composed of 1000 turns.

The boundary condition \( B \cdot n = 0 \) corresponding to a magnetic wall is imposed on all the boundaries of the domain. All the simulations were carried out using Code_Carmel (code-carmel.univ-lille.fr). A magnetic vector potential type formulation A was used, and the numerical problems were solved by conjugate gradient and using a Jacobi pre-conditioner. The mesh of the use case shown in figure 3 is irregular and composed of extruded triangles. It has 10,840 cells. The amplitude \( A_1 \) of the current \( I_1 \) is varied between 1 A and 20 A. The amplitude \( A_2 \) of the current \( I_2 \) is fixed at 0 A. Current \( I_1 \) is imposed and is sinusoidal with frequency \( f = 50 \text{Hz} \) (\( I_1(t) = A_1 \cdot \sin(2\pi ft) \), \( I_2(t) = 0 \)). Each simulation is composed of 41 time steps. Among all the information provided by Code_Carmel once each simulation has been completed, we are only interested here in the magnetic field \( B \) (this is one of the most interesting quantities) and more particularly in \( B_x \), its component along the x-axis, for the sake of simplicity.

4 Results

We apply the POD to a snapshot matrix built from ten simulations \( (N_{POD} = 10) \), which are run with different values of \( A_1 \in [1, 20] \text{A} \). We observe a clear decrease in the singular values associated to the POD decomposition, by choosing the first three we keep more than 99.9% of the variance. Thus, we train three SVRs taking the same two inputs (the electric current \( \lambda = I_1 \) and the time step \( t \)), and each SVR outputs one of the singular values. One hundred simulations are used to train the SVRs \( (N_{SVR} = 100) \) using the method described in section 2. For this, \( A_1 \) is drawn from a uniform distribution in the interval \([1, 20] \text{A}\). The choice of the interval \([1, 20] \text{A}\) is justified by the current-voltage characteristic of the constituent material of the cylinder. The operating range of the material is between 0 A and approximately 7 A, beyond which there is saturation. Thus, this interval contains material non-linearities. We run the algorithm in a single GPU node of CRONOS (a supercomputer included in the list www.top500.org) and the training took less than 2 seconds. However, we have not yet performed testing on large simulations. Figure 4 shows that the algorithm succeeds in learning the temporal evolution of the coefficients of the first three modes in the reduced
Figure 4: Temporal evolution of the coefficients of the first three modes ($A_1 = 11.36A$, $r=3$, $s=0.5$)

base. We observe that the fitting of the evolution of the coefficients is excellent. Furthermore, the mean root square error on the validation set is $8.95 \cdot 10^{-5}$, which is a remarkably small error for this use case. However, these results are preliminary and, even promising, more extensive testing and probably an evolution of the presented algorithm will be necessary.

5 Conclusion

We have conceived a method that combines model reduction via the Proper Orthogonal Decomposition (POD) and Support Vector Regression (SVR). The aim is the construction of a learning-based surrogate model for parameterized PDEs, which are also temporal and spatial processes. This method is non-invasive and uses a direct-time estimation strategy. We have performed tests on a first parametric electromagnetic use case, which presents dependence on a single parameter (an electric current) and contains material non-linearities. Obtained results are promising. However, we have presented ongoing research and these results are still preliminary.

6 Acknowledgments

This work was partially funded by REGALE (https://regale-project.eu/), part of the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 956560.

References


