A Fast Learning-Based Surrogate of Electrical Machines using a Reduced Basis

Anonymous Authors[1]

Abstract

A surrogate model approximates the outputs of a solver of Partial Differential Equations (PDEs) 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. This method is conceived to work in real-time, thus aimed for being used in the context of digital twins, where a user can perform an interactive analysis of results based on the proposed surrogate. We present promising results on two use cases concerning electrical machines. These use cases are not toy examples but are produced an industrial computational code, they use meshes representing non-trivial geometries and contain non-linearities.

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. In the last five years, the use of machine 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. Numerous methods for building surrogates have been proposed, in section 2 we discuss some of them, which we find representative of the variety of learning-based surrogates currently being developed. However, after analyzing the technical literature, we found two main limitations of the current state-of-the-art. First, the majority of use cases correspond to toy examples. For instance, the 1D Burgers equation is a popular choice. In general, the test use cases found in the literature are far in geometric complexity and size from what computation codes deal with in the current industrial practice. Second, the question of the eventual scaling of the method or its real-time capabilities is not often studied.

In this article, we propose a method to build surrogates in the context of parameterized Partial Differential Equations (PDEs), which depend on a set of parameters, that may represent boundary or initial conditions, materials laws, or other physical properties. In the general case, constructing surrogates for parameterized PDEs is a complex task. Our objective is to develop a learning-based surrogate that can be used to power a Digital Twin. This surrogate should be executed on a computer sufficiently fast to allow interactive exploration of the parameters of the underlying PDEs but also on space and time. This means that the surrogate should reconstruct a scalar or vectorial field at any time step and for any combination of the parameters. This is a state-of-the-art problem, especially when this task is performed in real time.

We have designed a method hybridizing the Proper Orthogonal Decomposition (POD, see chapter 11 of (Brunton & Kutz, 2019)) and several Support Vector Regression machines (SVR, see (Drucker et al., 1996)), where the Reduced Basis obtained by the POD is used to facilitate the training of our Machine-Learning system. We have designed a two-step simple method which is very fast at inference due to its simplicity. The method deals with geometric domains represented by non-regular meshes. Concerning the temporal discretization, it takes a *direct time* approach, which means it produces the state corresponding to the time step provided as input. This approach, which avoids iterating over time, is faster than autoregressive methods and thus well-adapted to real-time applications. The output of the proposed surrogate is an array containing a whole vectorial or scalar field, corresponding to the specified input set of parameters.

This article is structured as follows. Section 2 discusses relevant related work. Section 3 describes our proposed method for constructing learning-based surrogates. Section 4 introduces two use cases, concerning electric machines. These use cases are not toy examples but are produced by an industrial computational code. They use meshes representing

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

non-trivial geometries and contain non-linearities. Results are shown in Section 5. Section 6 discusses the real-time capabilities of the proposed method. Section 7 concludes the article. Finally, a bibliography section is given, followed by Appendix A which contains a mathematical proof for the inference error upper bound of the proposed method.

2. Related Work

Reducing the computational complexity of numerical simulations is not a new subject. The Reduced Order Model (ROM) community has traditionally tackled this problem. A ROM can be considered a type of surrogate model. Constructing surrogates for parameterized PDEs (the problem of this article) is, in the general case, a very complex task (Quarteroni et al.). One of the most popular techniques for building ROMs is the Proper Orthogonal Decomposition (POD, see chapter 11 of (Brunton & Kutz, 2019)), which we use in our work. On the other hand, building surrogates of simulations by using machine learning techniques is a recent research domain. From the seminal work of Raissi et al. (Raissi et al., 2019), which introduced the so-called physics-informed neural networks (PINNs), numerous techniques have been developed. Most of the learning-based surrogates found in the literature are trained in a supervised manner from simulation data. PINNs could stand as an exception, as they can be trained unsupervised. This is because PINNs are trained by minimizing the residual error of the PDE at random collocation points (Raissi et al., 2019; Sirignano & Spiliopoulos, 2018; Wandel et al., 2021). But PINNs also benefit from training with simulation data (Krishnapriyan et al., 2021; Lucor et al., 2022). Our method is also a supervised learning method that learns from the results of an ensemble of pre-executed numerical simulations.

Building learning-based surrogates still presents several important challenges, which are associated with different aspects of the numerical simulations. In the following, we discuss some of these aspects.

Spatial discretization. When the meshes supporting the numerical simulations are regular, they can be seen as images and convolutional networks can be employed successfully (Zhu et al., 2019; Ronneberger et al., 2015; Kasim et al., 2021). For instance, U-Net architectures are employed in (Thuerey et al., 2020; Wang et al., 2020), or auto-encoders in (Kim et al., 2019). For non-regular meshes Graph Neural Networks (GNNs) (Bronstein et al., 2017; Battaglia et al., 2018; Brandstetter et al., 2022) have been used. As an example, Deep-Mind introduced a GNN-based framework for learning mesh-based simulations (Pfaff et al., 2020; Sanchez-Gonzalez et al., 2020). Our method also supports the use of non-regular meshes.

Time discretization. Not only does the space discretization influence the design of the architectures, but the handling of the time dimension also distinguishes *autoregressive* from *direct* models. Autoregressive models mimic the iterative process of traditional solvers, where the current state is used as input to predict the next one. One of the challenges for autoregressive models is the error accumulation along a trajectory, leading to various mitigation strategies (Brandstetter et al., 2022; Pfaff et al., 2020; Takamoto et al., 2022). Some researchers have also integrated the time dimension using recurrent architectures (Tang et al., 2020) or attention mechanisms (Li et al., 2023). Direct models produce the state corresponding to the time step provided as input. PINNs are an example of direct models, our proposed method also uses this approach.

Respecting physical laws. From (Karniadakis et al., 2021), three main ways allow a learning system to represent correct physical laws: observational, inductive, and learning biases. PINNs (Raissi et al., 2019) focus on the learning biases by introducing physical constraints on the loss function. Techniques such as GNNs (Pfaff et al., 2020; Sanchez-Gonzalez et al., 2020) focus on designing specialized neural network architectures that implicitly embed prior knowledge about the problem (in this case information about the spatial domain) thus using inductive biases. Observational bias is the fact that the data used for training the system contains relevant and correct physical information. In this article, we use observational bias by preparing designs of experiences that execute an ensemble of meaningful numerical simulations (Santner et al., 2003).

Operators. PDEs are theoretically treated in the context of operators, which are maps between infinite-dimensional function spaces. However, neural networks learn mappings between finite-dimensional Euclidean spaces or finite sets (Kovachki et al., 2023). Efforts exist to create the so-called Neural Operators, which are neural networks aiming to map between functional spaces. Examples of this approach are techniques such as DeepONets (Lu et al., 2021), Graph Kernel Networks for PDEs (Li et al., 2020), or Fourier neural operators (Li et al., 2021). Neural operators aim at building surrogates of parametrized PDEs. Our method cannot be strictly considered a Neural Operator but it certainly maps from a parameter's space to a discretized functional space.

In summary, the method proposed in this article aims at building surrogates of parametrized PDEs. It is a direct time method that supports irregular meshes. The method learns from the results of an ensemble of pre-executed numerical simulations, thus using observational bias. A difference with the current literature is that its focus is not only on the quality of the reconstruction but also on the speed of the inference.

3. 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 before these two steps, an ensemble of N simulations should be run in order to create the dataset necessary for the learning. This step is not specific to our method, any learning-based surrogate needs a dataset to be trained on.

3.1. Finding a Reduced Basis

We obtain a Reduced Basis using the Proper Orthogonal Decomposition (POD), see chapter 11 of (Brunton & Kutz, 2019) or chapter 6 of (Quarteroni et al.). The fundamental step of the POD is the application of a Singular Value Decomposition (Gilbert & Strang, 1993) 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 precompute an ensemble of N spatio-temporal simulations. Each simulation contains t times steps and has a different vector of parameters λ. Afterward, we can build the snapshot matrix as follows:

$$\mathrm{X} = \begin{bmatrix} | & & | & & | & & | \\ X_{\lambda_1,t_1} & \dots & X_{\lambda_1,t_T} & \dots & X_{\lambda_N,t_1} & \dots & X_{\lambda_N,t_T} \\ | & & | & & | & & | \end{bmatrix} \tag{1}$$

where each column X_{λ_i,t_j} of X contains a vector $x_{i,j}$ indexed by (λ_i, t_j). Thus, discretized and linearized fields, representing a physical quantity provided by the solver, are contained in each vector $x_{i,j}$. We can build a *matrix of snapshots* X of size (n, m) by concatenating these vectors. Note that n is equal to the number of mesh cells (or nodes) used by the simulations multiplied by the number of components of the field (one for a scalar field, 2 or 3 for vectorial fields); a column of the matrix contains a spatially discretized field with its components stacked inside. The matrix has $m = NT$ columns, where N is the number of pre-computed simulations and T is the number of time steps.

SVD: A description of the singular value decomposition (SVD) can be found in any introductory linear algebra book, such as (Gilbert & Strang, 1993). Let $X \in \mathbb{R}^{n\times m}, U \in \mathbb{R}^{n\times n}, \Sigma \in \mathbb{R}^{n\times m}, V \in \mathbb{R}^{m\times m}$ the SVD of X is the decomposition $X = U \cdot \Sigma \cdot V^T$, where U and V are unitary matrices and Σ 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, which is defined by:

$$E = \frac{\sigma_1 + \sigma_2 + ... + \sigma_r}{\sigma_1 + \sigma_2 + ... + \sigma_m} \tag{2}$$

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

$$X^r = U^r \cdot \Sigma^r \cdot V^{Tr}. \tag{3}$$

3.2. 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, λ) as inputs, where t is a time step and λ is a vector of parameters. The SVR outputs a prediction $\hat{c}_i$, corresponding to the i-th coefficient on the reduced space, $i = 1...r$.

Figure 1. A SVR takes as input a time step t and a vector of parameters λ. It outputs a prediction $\hat{c}_i$, corresponding to the i-th coefficient on the reduced space, $i = 1...r$.

Preparing for the training phase. Our idea is to project the snapshot matrix X into the reduced space C. For this, we use U^r from equation 3 but we call this projection matrix U^r_{POD} to reinforce the fact that it is obtained by the POD. Consequently, the operation $C = U^r_{POD} \cdot X$ 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 is therefore subdivided into X_{train} and X_{val}, thus

$$X = \left[\ X_{train} \ \middle|\ X_{val}\ \right] \tag{4}$$

We can then construct the corresponding matrices of coefficients $C = [\, C_{train} \,|\, C_{val} \,]$ by matrix product:

$$C = \left[\ U^r_{POD} \cdot X_{train} \ \middle|\ U^r_{POD} \cdot X_{val}\ \right] \tag{5}$$

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, λ) as inputs. These inputs can be coded in the following matrix:

$$x = \begin{bmatrix} t_1 & t_2 & \dots & t_P & \dots & t_1 & t_2 & \dots & t_P \\ | & | & & | & & | & | & & | \\ \lambda_1 & \lambda_1 & \dots & \lambda_1 & \dots & \lambda_N & \lambda_N & \dots & \lambda_N \\ | & | & & | & & | & | & & | \end{bmatrix} \tag{6}$$

where each column X_{λ_i,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 $x = \left[\begin{array}{c|c} x_{train} & x_{val} \end{array}\right]$. 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 element of the vector C_{train}. For example, the first *SVR* is trained to predict the first element 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 *parametro-temporal* coefficients of the i-th mode in the reduced space generated by U^r_{pod}, 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 *SVR*s.

3.3. Tuning the hyperparameters of the SVRs

The SVR method contains several hyperparameters that should be tuned. In this work, we have used the SVR implementation of Scikit-Learn (Pedregosa et al., 2011) and performed tuning for three important parameters:

- Epsilon in the epsilon-SVR model. It specifies the epsilon-tube within which no penalty is associated with the training loss function with points predicted within a distance epsilon from the actual value.
- The regularization parameter associated with a squared penalty term.
- We decided to fix the kernel used in the algorithm, which is a Gaussian. Thus, the standard deviation σ of the Gaussian must be tuned.

We use the Optuna package (Akiba et al., 2019) for tuning the above-presented parameters. We specifically use Optuna's implementation of the Tree-structured Parzen estimator (TPE) algorithm (Bergstra et al., 2011), a Bayesian optimization method widely used in recent parameter tuning frameworks.

Table 1. Comparison of the two use cases concerning: the number of cells in the mesh, number of time steps, number of runs performed by the design of experiments, and size of the data stored on disk.

USE CASE	CELLS	STEPS	RUNS	STORAGE
TRANSFORMER	88,072	40	327	26G
INDUCTION	30,047	10	750	5.3G

We remark that SVRs can take several scalar inputs but they generate a unique scalar output. This implies that our method should train r SVRs, where r is the dimension of the reduced space. Thus a question arises concerning the $3r$ parameters to be optimized: is each SVR going to be treated independently, or can the $3r$ parameters be jointly tuned? We have found that, when jointly optimizing for the worst validation error, an upper bound on the error of the reconstructed fields exists.

We have proven that the reconstruction error of the physical field $\|\mathbf{X}_p - \hat{\mathbf{X}}_p\|_2$, computed at each cell $p \leq n$, has an upper bound that is linearly proportional to the highest validation error of all r SVR machines (the whole prove is given in Appendix A). Thus:

$$\|\mathbf{X}_p - \hat{\mathbf{X}}_p\|_2 \leq K_p.\mathbf{e}$$

where K_p is a positive real number, and $\mathbf{e}$ is the error. The constant K_p comprises terms of the projection matrix U^r_{POD} and standardization elements: S_i is the standard deviation and E_i is the mean of the coefficient vectors C_i. It is written as $K_p = \sqrt{A + 2B}$ where:

$$A = \sum_{k=1}^{k=r} (S_k.U^r_{POD_{pk}})^2$$

$$B = \sum_{1 \leq h < l \leq r} |S_h.U^r_{POD_{ph}}|.|S_l.U^r_{POD_{pl}}|$$

4. Description of the use cases

We present two electrical machines use cases generated by Code_Carmel (code-carmel.univ-lille.fr), which is an industrial computational code. Table 1 shows a comparison of these use cases concerning: the number of cells in the mesh, number of time steps, number of runs performed by the design of experiments, and size of the data when stored on disk (compressed).

4.1. An induction plate

In the first use case, we are interested in a magneto-quasistatic phenomenon by the modeling of a magnetic electric current inductor. This could represent several applications, such as an induction hotplate in a kitchen or the induction charger of a mobile phone. The inductor is a coil of 700 turns at which a current is injected with a sinusoidal frequency f. The intensity of the current in the coil is calculated from the electrical voltage U imposed and a characteristic resistance of the inductor R_{ind} via a standard circuit equation. The plate, made of copper, has an electrical conductivity of $\sigma = 5 \cdot 10^7 S \cdot m^{-1}$.

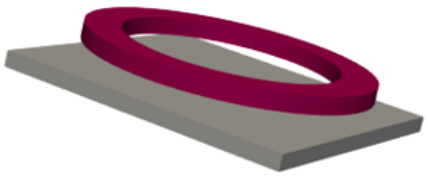

Figure 2. Mesh of the induction plate use case.

Design of experiments. Four parameters are considered in this use case: f, U, R_{ind}, and the time t. We run 750 simulations, each one corresponding to a parameter set. The values of f, U, R_{ind} are sampled from a non-informative uniform distribution. The sampling of t is given by the computational code.

Why this use case? The choice of this use case is motivated by two factors. First, the dynamics of the problem impose a time resolution of the finite elements model which can be computationally expensive. Then, we are interested in the accuracy of the proposed method for time-dependent problems, as it might provide large calculation speedups. Second, the calculation of the source current through a circuit equation impacts the dynamical evolution of the problem. Therefore, we aim to evaluate the capacity of the proposed method to adapt not only to the electromagnetic dynamics but also to changes in the circuit equation parameters (f, U, R_{ind}).

4.2. A three-phase transformer

A power network is a complex system, with a large variety of voltage levels at generation, transmission, and distribution points. In this context, transformers are vital components, as they allow transitioning from one voltage level to the other, ensuring the transmission of electric energy through the network. They are static devices, consisting of one or multiple windings and a magnetic core, used extensively by electric companies in all types of production sites. For these reasons, power transformers are the subject of numerous numerical simulations for diagnosis and design review purposes, and surrogate models have been proposed in recent years to reduce computational costs (Henneron & Clenet, 2014).

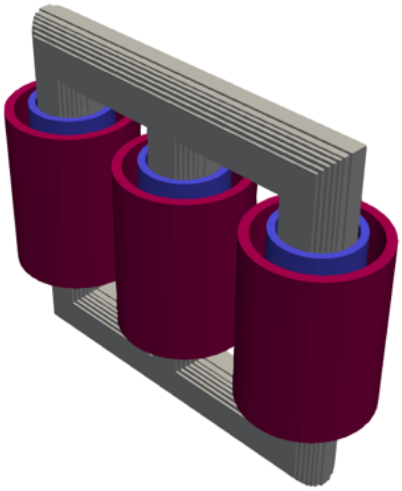

Figure 3. Mesh of the three-phase transformer use case.

Figure 3 shows the mesh used in the finite elements simulations: grey voxels belong to the core, red voxels to the primary windings, and blue voxels to the secondary windings. The space around the core and windings is also meshed but not represented. In this figure, we can see the three columns of a three-phase transformer, each of them associated with two windings. A high voltage is imposed in the primary circuit, associated with a high amplitude electric current, which generates a magnetic flux traveling through the core. The core consists of an arrangement of ferromagnetic laminated sheets, associated with a non-linear magnetic anhysteretic permeability as shown in Figure 4. It will be approximated using the so-called Frohlich model (Fröhlich, 1881):

$$\mu(H) = \mu_0 + \frac{\alpha}{\beta + H} \tag{7}$$

In this configuration, we are interested in the magnetic flux density distribution present in the transformer and around it, when the secondary circuit is short-circuited $I_2 = 0$. The problem is parameterized by the current amplitude I of the primary windings and by the coefficients α and β of the Frohlich approximation (equation 7).

Design of experiments. Four parameters are considered in this use case: the intensity I, the two coefficients α and β of the behavior law of the core, and the time t. We run 327 simulations, each one corresponding to a parameter set. The values of I are sampled from a beta distribution centered around $1.4A$, this is done to better learn a known non-linearity physically appearing at this current value. Parame-

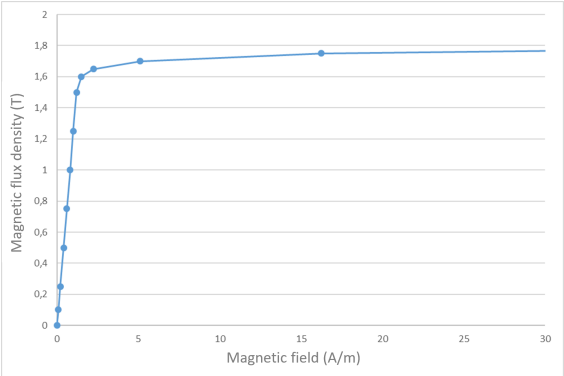

Figure 4. Non-linear behavior law of the transformer core.

ters α and β are sampled from a non-informative uniform distribution. The sampling of t is given by the computational code.

Why this use case? The choice of this use case is motivated by several main arguments. First, it is a 3D model initially created for the industrial purpose of design review. Then, applying the proposed method to this use case gives us useful insight, into its capacity to generate accurate results for real-life engineering problems. Second, the non-linear behavior law of the magnetic core is not often well known by engineers when performing numerical simulations for diagnosis or design review purposes. Then, we aim not merely to evaluate the accuracy of the proposed method on a nonlinear problem, but also to assess the possibility of using this method to solve inverse problems for the deduction of the parameters of the non-linear behavior law. Third, in the literature, we can find recent works on surrogate models (Henneron & Clenet, 2014) and deep learning (Gong & Tang, 2022) developments for similar devices, setting a favorable context for future comparison.

5. Results

In this section, we jointly discuss the results of the three use cases described in section 4. We start by presenting, in table 2, the number of modes necessary to represent the solutions while keeping 95% , 98%, and 99% of the accumulated energy, as specified in equation 2. By comparison with the number of cells given in table 1, we observe that the POD (which is the first step of our algorithm) introduces a strong compression on the vector fields associated to the simulation mesh. This can be considered a sort of spatial compression, which is induced by the coding we have chosen for the snapshot matrices.

We calculated, for each of our use cases, the relative RMSE (Root Mean Square Error) and relative AME (Absolute Mean Error). These errors are calculated as follows:

Table 2. Comparisson of the two use cases concerning the number of modes necessary to represent the problem, for three levels of cumulated energy (95%, 98%, and 99%).

USE CASE	# MODES 95%	# MODES 98%	# MODES 99%
INDUCTION	2	3	3
TRANSFORMER	4	7	10

$$\delta RMSE = \frac{\frac{1}{N}\sum_{k=1}^{N}\frac{1}{T}\sum_{j=1}^{T}\frac{1}{n}\sum_{i=1}^{n}\|\hat{v}-v\|_2^2}{\frac{1}{N}\sum_{k=1}^{N}\frac{1}{T}\sum_{j=1}^{T}\frac{1}{n}\sum_{i=1}^{n}\|v\|_2^2} \quad (8)$$

$$\delta AME = \frac{\frac{1}{N}\sum_{k=1}^{N}\frac{1}{T}\sum_{j=1}^{T}\frac{1}{n}\sum_{i=1}^{n}\mid \hat{v}-v \mid}{\frac{1}{N}\sum_{k=1}^{N}\frac{1}{T}\sum_{j=1}^{T}\frac{1}{n}\sum_{i=1}^{n}\mid v \mid} \quad (9)$$

where δ indicates "relative", v is a reference vector from the test set, $\hat{v}$ is the estimate provided by our method, N is the number of simulations in the design of experiences, T is the number of time steps, and n is the number of mesh cells. We choose to explicitly write three sum signs to remark on the parametric and spatio-temporal nature of the estimated vectorial fields. Table 3 shows these errors, which are multiplied by 100 to express percentages, for the test sets of each use case.

Table 3. Errors obtained by the surrogate models, on the simulations test sets, for two levels of cumulated energy (95% and 98%).

USE CASE	δ RMSE 95%	δ AME 95%	δ RMSE 98%	δ AME 98%
INDUCTION	4.5%	4.3%	0.98%	0.97%
TRANSFORMER	2.9%	3.2%	2.3%	2.5%

Figure 5 presents a visual comparison between the modulus of a reconstructed magnetic field (top image) and the reference magnetic field (bottom image), on the test set of the three-phase transformer introduced in Section 4.2. For the shown time step and set of parameters, the magnetic field is oscillating on the core. We observe positive values in red, negative in blue, and grey indicates near zero values. We observed that reconstructed and reference images are visually very similar, which was one of the objectives of the proposed surrogate. More importantly, the relative errors shown in Table 3 are very satisfactory.

6. Real-Time Inference

We recall that our final objective is to develop a surrogate that can be used to power a Digital Twin. This surrogate

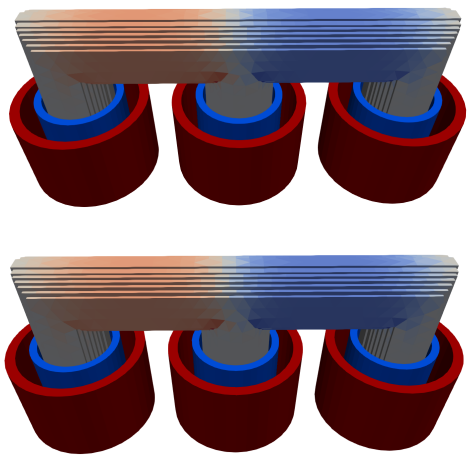

Figure 5. Visual comparison between the modulus of a reconstructed magnetic field (top image) and the reference magnetic field (bottom image), on the test set of the three-phase transformer use case introduced in Section 4.2.

should be executed on a computer sufficiently fast to allow the interactive exploration of its outputs. The introduced method fulfills this objective for several reasons. First, when coding the snapshot matrix as indicated in equation 3.2, the POD acts as a spatial compression system. This means that simulations based on large meshes can potentially be reduced to a vector of relatively few coefficients. Thus the SVRs can perform the estimation of these coefficients very fast. Once the coefficients are obtained, a simple matrix multiplication reconstructs the desired scalar or vectorial field. Second, the Direct Time approach avoids iterating overtime at the inference step.

When designing a visualization system for a Digital Twin, performing an interactive exploration of the results imposes constraints on the response time of the surrogate model. Response times in human-computer interactions have been studied for years, see for instance (Shneiderman, 1984). A classical reference (Nielsen, 1994) stated that 0.1 second is about the limit for having the user feel that the system is reacting instantaneously. Thus, we performed an analysis of the response times of the proposed surrogate. For this, we measured the time of inference of a single parameter set, which corresponds to computing a discretized vectorial field for a fixed time step and fixed parameters. We perform this testing in a single node of a cluster, this node contains an Intel-Xeon Platinum 8260 processor operating at 2,40 GHz, and it does not contain GPUs. No special optimization has been performed on the code, composed of Phyton scripts run sequentially. The results are shown in Table 4, times are given in milliseconds. Each shown time is the mean of 100 measurements, the standard deviations are always on the scale 10^5 thus we consider the measurement reliable.

We observe that, for all our use cases the inference time is approximately 2ms, thus around 50 times less (in our slower case) than the time necessary for a fluid real-time user interaction.

Table 4. Execution time of the surrogate.

USE CASE	# MODES 95%	# MODES 98%
INDUCTION	0.43 MS	0.56 MS
TRANSFORMER	1.23 MS	2.2 MS

7. Conclusion

We have conceived a novel 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. We found an error upper bound of the proposed method and a mathematical proof of this bound is included. We have performed tests on two use cases concerning electrical machines. These use cases are not toy examples. They are produced by an industrial computational code, they use 3D meshes representing non-trivial geometries and contain non-linearities. The obtained results show a good reconstruction accuracy. Furthermore, we have studied the response time of the proposed surrogate. The tests indicate that it is adapted to real-time tasks.

The current results indicate that the method can be applied to the interactive exploration of use cases based on much larger meshes or for higher intrinsic complexity. Indeed, the slower case we treated was 50 times faster than needed for interactive exploration. Moreover, no code optimization has been performed yet. Thus, the proposed method presents great potential for building parameterized surrogates of industrial-level numerical simulations.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. Algo-

rithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

Brandstetter, J., Worrall, D. E., and Welling, M. Message passing neural PDE solvers. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=vSix3HPYKSU.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

Brunton, S. L. and Kutz, J. N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019. doi: 10.1017/9781108380690.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A., and Vapnik, V. Support vector regression machines. *Advances in neural information processing systems*, 9, 1996.

Fröhlich, O. Investigations of dynamoelectric machines and electric power transmission and theoretical conclusions therefrom. *Elektrotech. Z*, 2:134–141, 1881.

Gilbert and Strang. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.

Gong, R. and Tang, Z. Further investigation of convolutional neural networks applied in computational electromagnetism under physics-informed consideration. *IET Electric Power Applications*, 16(6):653–674, 2022.

Henneron, T. and Clenet, S. Model order reduction of non-linear magnetostatic problems based on pod and dei methods. *IEEE Transactions on Magnetics*, 50(2):33–36, 2014.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Kasim, M. F., Watson-Parris, D., Deaconu, L., Oliver, S., Hatfield, P., Froula, D. H., Gregori, G., Jarvis, M., Khatiwala, S., Korenaga, J., et al. Building high accuracy emulators for scientific simulations with deep neural architecture search. *Machine Learning: Science and Technology*, 3(1):015013, 2021.

Kim, B., Azevedo, V. C., Thuerey, N., Kim, T., Gross, M., and Solenthaler, B. Deep fluids: A generative network for parameterized fluid simulations. In *Computer Graphics Forum*, volume 38, pp. 59–70. Wiley Online Library, 2019.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Krishnapriyan, A. S., Gholami, A., Zhe, S., Kirby, R. M., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 26548–26560, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/df438e5206f31600e6ae4af72f2725f1-Abstract.html.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A. M., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=c8P9NQVtmnO.

Li, Z., Meidani, K., and Farimani, A. B. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=EPPqt3uERT.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Lucor, D., Agrawal, A., and Sergent, A. Simple computational strategies for more effective physics-informed neural networks modeling of turbulent natural convection. *Journal of Computational Physics*, 456:111022, 2022.

Nielsen, J. *Usability engineering*. Morgan Kaufmann, 1994.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2020.

Quarteroni, A., Manzoni, A., and Negri, F. *Reduced Basis Methods for Partial Differential Equations: an introduction*. doi: 10.1007/978-3-319-15431-2.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020.

Santner, T. J., Williams, B. J., Notz, W. I., and Williams, B. J. *The design and analysis of computer experiments*, volume 1. Springer, 2003.

Shneiderman, B. Response time and display rate in human performance with computers. *ACM Computing Surveys (CSUR)*, 16(3):265–285, 1984.

Sirignano, J. and Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., and Niepert, M. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35: 1596–1611, 2022.

Tang, M., Liu, Y., and Durlofsky, L. J. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *Journal of Computational Physics*, 413:109456, 2020.

Thuerey, N., Weißenow, K., Prantl, L., and Hu, X. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020.

Wandel, N., Weinmann, M., and Klein, R. Learning incompressible fluid dynamics from scratch - towards fast, differentiable fluid models that generalize. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=KUDUoRsEphu.

Wang, R., Kashinath, K., Mustafa, M., Albert, A., and Yu, R. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1457–1466, 2020.

Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., and Perdikaris, P. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394: 56–81, 2019.

A. Appendix: inference error upper bound

A.1. Notation

- Let $X \in \mathbb{R}^{n \times m}$ be a snapshot matrix and $\hat{X}$ its inferred equivalent.
- Let $C, \hat{C} \in \mathbb{R}^{r \times m}$ be their coefficient matrices which are the projection of $X, \hat{X}$ by $U^r_{POD} \in \mathbb{R}^{r \times n}$. They verify :

$$\begin{aligned} X &= U^r_{POD}.C \\ \hat{X} &= U^r_{POD}.\hat{C} \end{aligned} \tag{10}$$

- We denote $\mathbf{X}_p = (X_{p1}, ..., X_{pm})^\top$, $\hat{\mathbf{X}}_p = (\hat{X}_{p1}, ..., \hat{X}_{pm})^\top$ the snapshot values of X and $\hat{X}$ at the cell $p \leq n$.
- Similarly, we define $\mathbf{C}_k = (C_{k1}, ..., C_{km})^\top$ and $\hat{\mathbf{C}}_k = (\hat{C}_{k1}, ..., \hat{C}_{km})^\top$ the k-th component of the original and the inferred matrices for any $k \leq r$.
- By centering and reducing $\mathbf{C}_k = (C_{k1}, ..., C_{km})^\top$, we derive $\mathbf{c}_k$ and $\hat{\mathbf{c}}_k$ which correspond to the training vector and the output of the k-th SVR respectively. They verify :

$$\begin{aligned} \mathbf{C}_k &= S_k.\mathbf{c}_k + E_k \\ \hat{\mathbf{C}}_k &= S_i.\hat{\mathbf{c}}_k + E_k \end{aligned} \tag{11}$$

  where E_k is the mean of $\mathbf{C}_k$ and S_k its standard deviation.

A.2. Introduction

SVR machines are trained with respect to the *Root Mean Square Error* of inference $\mathbf{e}_k$ evaluated between the output $\hat{\mathbf{c}}_k$ and the standardized k-th component of the original coefficient matrix $\mathbf{c}_k$. Thus:

$$\mathbf{e}_k = \|\mathbf{c}_k - \hat{\mathbf{c}}_k\|_2$$

To optimize the learning hyper-parameters $\lambda_1, \lambda_2, ...$ of the metamodel, we aim to minimize an objective function $\mathbf{e}$, equal to the highest of validation errors $(\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_r)$ recorded among all $\mathbf{r}$ SVR machines such as :

$$\mathbf{e} = \max_{1 \leq k \leq r} \mathbf{e}_k \tag{12}$$

In fact, we can prove that the root mean square error of inference of X computed at any cell $p \leq n$ has an upper bound that is linearly proportional to $\mathbf{e}$ (*Theorem A.1*).

Theorem A.1. $\forall p \leq n \; \exists K_p > 0$ *that verifies*

$$\|\mathbf{X}_p - \hat{\mathbf{X}}_p\|_2 \leq K_p.\mathbf{e}$$

Thus, minimizing the objective function contributes to shrinking the validation error evaluated at any component $p \leq n$ of the snapshot matrix.

A.3. Proof

At first, we remind the analytical formulation of $\|X_p - \hat{X}_p\|_2$:

$$\|X_p - \hat{X}_p\|_2^2 = \frac{1}{m}\sum_{j=1}^{j=m}(X_{pj} - \hat{X}_{pj})^2$$

To find an upper bound, we start by expressing the relative error of inference $\Delta X_{pj} = X_{pj} - \hat{X}_{pj}$ of the $j-th$ snapshot at the cell p in terms of inference errors of all SVR machines.

$\forall p \in [1, N] \;\; \forall j \in [1, m]$:

$$\begin{aligned}
\Delta X_{pj} &= X_{pj} - \hat{X}_{pj} \\
(1) \implies &= \sum_{k=1}^{k=r} U^r_{POD_{pk}}(C_{kj} - \hat{C}_{kj}) \\
(2) \implies &= \sum_{k=1}^{k=r} U^r_{POD_{pk}}(S_k c_{kj} + E_k - S_k \hat{c}_{kj} - E_k) \\
&= \sum_{k=1}^{k=r} S_k.U^r_{POD_{pk}} \underbrace{(c_{kj} - \hat{c}_{kj})}_{\Delta c_{kj}}
\end{aligned}$$

$$\implies \Delta X_{pj} = \sum_{k=1}^{k=r} S_k.U^r_{POD_{pk}} \Delta c_{kj} \tag{13}$$

By squaring ΔX_{pj}, we obtain :

$$\begin{aligned}
\Delta X_{pj}^2 &= (\sum_{k=1}^{k=r} S_k.U^r_{POD_{pk}} \Delta c_{kj})^2 \\
&= \sum_{k=1}^{k=r}(S_k.U^r_{POD_{pk}})^2 \Delta c_{kj}^2 + 2\sum_{1\le h<l\le r}(S_h.U^r_{POD_{ph}}).(S_l.U^r_{POD_{pl}}).\Delta c_{hj}\Delta c_{lj} \\
\Delta X_{pj}^2 &\le \sum_{k=1}^{k=r}(S_k.U^r_{POD_{pk}})^2 \Delta c_{kj}^2 + 2\sum_{1\le h<l\le r}|S_h.U^r_{POD_{ph}}|.|S_l.U^r_{POD_{pl}}|.|\Delta c_{hj}\Delta c_{lj}|
\end{aligned}$$

Then, we sum over j:

$$\sum_{j=1}^{j=m} \Delta X_{pj}^2 \leq \sum_{j=1}^{j=m}\sum_{k=1}^{k=r}(S_k.U^r_{POD_{pk}})^2\Delta c_{kj}^2 + 2\sum_{j=1}^{j=m}\sum_{1\leq h<l\leq r}|S_h.U^r_{POD_{ph}}|.|S_l.U^r_{POD_{pl}}|.|\Delta c_{hj}\Delta c_{lj}|$$
$$\leq \sum_{k=1}^{k=r}(S_k.U^r_{POD_{pk}})^2 \underbrace{\sum_{j=1}^{j=m}\Delta c_{kj}^2}_{1} + 2\sum_{1\leq h<l\leq r}|S_h.U^r_{POD_{ph}}|.|S_l.U^r_{POD_{pl}}| \underbrace{\sum_{j=1}^{j=m}|\Delta c_{hj}\Delta c_{lj}|}_{2}$$

We recall that $\forall k \in [1, r]$:

$$\mathbf{e}_k^2 = \|\mathbf{c}_k - \hat{\mathbf{c}}_k\|_2^2 = \frac{1}{m}\sum_{j=1}^{j=m}\Delta c_{kj}^2 \tag{14}$$

Thus, term 1 immediately becomes $\sum_{j=1}^{j=m}\Delta c_{kj}^2 = m\mathbf{e}_k^2 \leq m\mathbf{e}^2$.

By virtue of Hölder's inequality and (5), we establish the following about term 2:

$$\sum_{j=1}^{j=m}|\Delta c_{hj}\Delta c_{lj}| \leq \left(\sum_{j=1}^{j=m}\Delta c_{hj}^2\right)^{1/2}\cdot\left(\sum_{j=1}^{j=m}\Delta c_{lj}^2\right)^{1/2}$$
$$\leq (m\mathbf{e}_h^2)^{1/2}\cdot(m\mathbf{e}_l^2)^{1/2}$$
$$\leq m\mathbf{e}_h\cdot\mathbf{e}_l \leq m\mathbf{e}^2$$

By introducing the new bounds on term 1 and 2, the latter inequality becomes :

$$\sum_{j=1}^{j=m}\Delta X_{pj}^2 \leq \sum_{k=1}^{k=r}(S_k.U^r_{POD_{pk}})^2\cdot(m\mathbf{e}^2) + 2\sum_{1\leq h<l\leq r}|S_h.U^r_{POD_{ph}}|.|S_l.U^r_{POD_{pl}}|\cdot(m\mathbf{e}^2)$$
$$\leq \left(\sum_{k=1}^{k=r}(S_k.U^r_{POD_{pk}})^2 + 2\sum_{1\leq h<l\leq r}|S_h.U^r_{POD_{ph}}|.|S_l.U^r_{POD_{pl}}|\right)\cdot m\mathbf{e}^2$$

$$\frac{1}{m}\sum_{j=1}^{j=m}\Delta X_{pj}^2 \leq \left(\sum_{k=1}^{k=r}(S_k.U^r_{POD_{pk}})^2 + 2\sum_{1\leq h<l\leq r}|S_h.U^r_{POD_{ph}}|.|S_l.U^r_{POD_{pl}}|\right)\cdot\mathbf{e}^2$$

$$\|\mathbf{X}_p - \mathbf{X}_p\|_2 \leq \left(\sum_{k=1}^{k=r}(S_k.U^r_{POD_{pk}})^2 + 2\sum_{1\leq h<l\leq r}|S_h.U^r_{POD_{ph}}|.|S_l.U^r_{POD_{pl}}|\right)^{1/2}\cdot\mathbf{e}$$

Hence :

$$\|\mathbf{X}_p - \mathbf{X}_p\|_2 \leq K_p\cdot\mathbf{e}$$

where $K_p = \left(\sum_{k=1}^{k=r}(S_k.U^r_{POD_{pk}})^2 + 2\sum_{1\leq h<l\leq r}|S_h.U^r_{POD_{ph}}|.|S_l.U^r_{POD_{pl}}|\right)^{1/2}$