Sancarlos *et al. Advanced Modeling and Simulation in Engineering Sciences(2023)10:4*
https://doi.org/10.1186/s40323-023-00240-4

Advanced Modeling and Simulation in Engineering Sciences

RESEARCH ARTICLE Open Access

Regularized regressions for parametric models based on separated representations

Abel Sancarlos[1], Victor Champaney[1], Elias Cueto[2] and Francisco Chinesta[1*]

*Correspondence: Francisco.CHINESTA@ensam.eu
[1]PIMM, ENSAM Institute of Technology ESI GROUP Chair on Advanced Modeling and Simulation of Manufacturing Processes, 151 Boulevard de l'Hopital, 75013 Paris, France
[2]Aragon Institute of Engineering Research, Universidad de Zaragoza, Calle de Mariano Esquillor, s/n, 50018 Zaragoza, Spain

Abstract

Regressions created from experimental or simulated data enable the construction of metamodels, widely used in a variety of engineering applications. Many engineering problems involve multi-parametric physics whose corresponding multi-parametric solutions can be viewed as a sort of computational vademecum that, once computed offline, can be then used in a variety of real-time engineering applications including optimization, inverse analysis, uncertainty propagation or simulation based control. Sometimes, these multi-parametric problems can be solved by using advanced model order reduction—MOR-techniques. However, solving these multi-parametric problems can be very costly. In that case, one possibility consists in solving the problem for a sample of the parametric values and creating a regression from all the computed solutions. The solution for any choice of the parameters is then inferred from the prediction of the regression model. However, addressing high-dimensionality at the low data limit, ensuring accuracy and avoiding overfitting constitutes a difficult challenge. The present paper aims at proposing and discussing different advanced regressions based on the proper generalized decomposition (PGD) enabling the just referred features. In particular, new PGD strategies are developed adding different regularizations to the *s*-PGD method. In addition, the ANOVA-based PGD is proposed to ally them.

Keywords: Model order reduction, Proper generalized decomposition, Sparse PGD, Data-driven models, LASSO, Ridge regression, ANOVA, Elastic net

Introduction

Model order reduction—MOR-techniques express the solution of a given problem (expressed as a partial differential equation—PDE, for instance) into a reduced basis with strong physical or mathematical content. By "strong physical content" we mean that they are extracted and motivated by the physical laws governing the system under study. In addition, the dynamic evolution of this type of basis is also computed based on the corresponding physics-based model. Very often, these bases are extracted from solutions of the problem at hand and are obtained offline. This can be done, for instance, by invoking the proper orthogonal decomposition—POD—or the reduced basis method—RB—[8]. When

computing with a reduced basis, the solution complexity scales with the size of this basis, which is in general much smaller than the size of the multi-purpose approximation basis associated with the finite element method—FEM, whose size scales with the number of nodes in the mesh.

Even if the use of a reduced basis implies a certain loss of generality, it enables impressive computing time savings and, as soon as the problem solution continues living in the space spanned by the reduced basis, the computed solution remains accurate enough. Obviously, as soon as one is interested in a solution that can not be accurately approximated within the space spanned by that reduced basis, the solution will be computed fast, but its accuracy is expected to be poor. To improve generality while ensuring accuracy, an appealing route consists of constructing the reduced basis and solving the problem simultaneously, as the Proper Generalized Decomposition—PGD—does [8]. However, this option becomes in general very intrusive, even more than the ones based on the employ of reduced bases. In this work, by intrusiveness we mean the degree of changes required by the MOR framework, with respect to standard simulation techniques, in the mathematical procedure to solve an industrial problem. These changes should be programmed in softwares that are already implemented in the market and therefore they already have the confidence of the client as well as several years of improvement and development. Companies' reluctance to make major changes to their long-established software promotes and favors the creation of methodologies with a low level of intrusiveness..

To alleviate intrusiveness, non-intrusive procedures were proposed. They proceed by constructing the parametric solution of the parametric problem from a number of high-fidelity solutions performed offline. In general, these are very expensive from the computing time viewpoint, for different choices of the model parameters that constitutes the design of experiments—DoE.

Among these techniques we can mention standard polynomial approximations on sparsely sampled parametric domains. Despite its simplicity, its use is not to be taken lightly. The use of orthogonal polynomial bases, with their associated Gauss–Lobatto points as DoE, allows us to obtain very accurate approximations. However, the sampling (DoE) increases exponentially with either the number of dimensions of the considered polynomial degree. Using randomly sampled DoE, or considering an approximation too rich with respect to the available amount of data (underdetermined approximation problem), results in noticeable overfitting effects. A way of attenuating these unfavorable effects, consists in using an approximation basis avoiding over-oscillating phenomena, as kriging approximations, for instance perform successfully [31], being a major protagonist of the so-called surrogate models (or metamodels) [12,30]. Another possibility consists in restricting polynomial approximations to a low degree, e.g., linear or moderately nonlinear regressions.

Other approaches concern the proper orthogonal decomposition with interpolation—PODI—[25], where usual regressions for expressing the dependence of the modal coefficients on the parameters are employed. Within the PGD rationale, Sparse Subspace Learning—SSL—[4] interpolates the pre-computed solutions related to the DoE associated to an structured grid (Gauss-Lobatto points) over the whole parametric space, by considering a hierarchical approximation basis for interpolating the precomputed solutions. This ensures the separated representation of the interpolated parametric solution. A sparsely sampled counterpart, the so-called sparse PGD, s-PGD, was proposed in [20].

The main limitations of SSL-based regression procedures is the volume of data, which increases exponentially with the number of parameters involved in the model. Thus, when considering P parameters, the lowest approximation level, the so-called *0-level*, which consists in a multi-linear approximation (the product of a linear approximation along each parametric dimension), needs 2^P data (each datum coming in fact from a high fidelity solution). On the other hand, s-PGD reduces the amount of required data, by considering a sparse sampling. However, the fact of combining higher degree approximations (induced by the separated representations) with very reduced amount of data, exacerbates the risk of overfitting. To avoid overfitting, in [20] the authors proposed the use of adaptive approximation bases, the so-called Modal adaptive Strategy—MAS—, whose degree is kept to a minimum in the first PGD modes (first terms of the finite sum decomposition expressing the variables separation which is at the heart of the PGD). This degree is then increased progressively for the calculation of higher level modes. Other choices of the approximation bases were also considered for limiting these spurious over-oscillating behaviors, as for example the employ of kriging. The s-PGD can thus be viewed as a nonlinear regression that makes use of the separation of variables. This enables its use in multi-parametric settings.

Regressions are widely employed in artificial intelligence in general, and more particularly in supervised scientific machine learning [7,16,37], in the development of cognitive or hybrid digital twins [9,28,32] or even in the field of neuroscience [35]. Regression can thus be seen as the main ingredient in the automatic construction of models of the surrounding physical reality. This is of utmost importance in the construction of an artificial intelligence able to maneuver in the physical world [27,29].

The main issues related to the implementation of regression in the low-data limit concern nonlinear behaviors in multi-parametric settings. This last factor leads to the so-called curse of dimensionality, i.e., the exponential growth in the number of degrees of freedom (equivalently, the number of necessary sampling points in the phase space) that is necessary to obtain accurate results [23].

When constructing models, it is always important to keep them as simple as possible. In other words, parsimonious models are always preferable to more complex ones. This principle, known as Occam's razor [7,37], implies that simpler explanations should be preferred among all the available ones to explain any physical phenomenon. In the literature this is achieved by imposing sparsity in the regression [7,15,17,19]. To obtain parsimonious models able to address sparsity, it is thus convenient to perform regression by combining L2 and L1 norms.

This paper aims at proposing robust, general, frugal and accurate regression methodologies able to operate in separated representation settings. For that purpose, three techniques will be proposed and analyzed. The first is based on an Elastic Net regularized formulation [14], called *rs*-PGD, and combines Ridge and Lasso regressions [5,13,14], that make use, respectively, of the L2 and L1 norms. Both use a rich approximation basis and, to avoid overfitting, the former favors specific solutions with smaller coefficients, while the last enforces the sparsest possible solution by retaining those contributing the most to the solution approximation.

Then, the doubly sparse regression, the so-called s^2-PGD technique will be introduced. The last makes use of the Lasso regularization (the one introduced above that looks for the

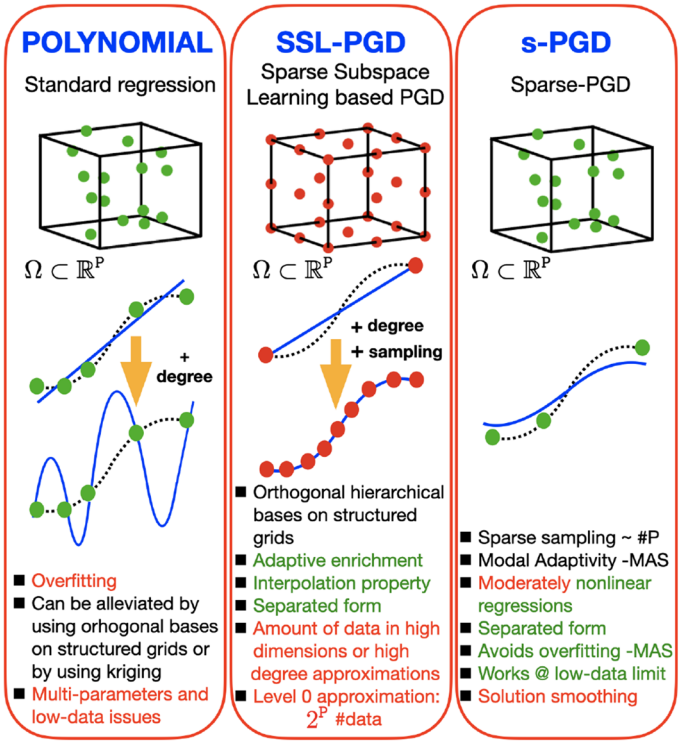

Fig. 1 Part A. Non-intrusive MOR techniques with the main sampling and approximation features, their pros (emphasized in the green text) and the cons (in red)

sparsest approximation through the use of the L1-norm) while searching for the sparsest dimensions.

The third and last technique, the ANOVA-PGD [22,36], aims at allying orthogonal hierarchical bases with a more favorable scaling (with respect to the SSL [4]) of the amount of data with the approximation richness. For that purpose, separated representations and sparse approximations (eventually regularized) will be combined for addressing multiple correlation terms.

Figures 1 and 2 sketches the just referred regression strategies, with the main sampling and approximation features, their pros (emphasized in the green text) and the cons (in red). A comparison on the different exposed techniques, the general workflow for allying them for the solution of a given problem, while addressing their scalability to address industrial problems involving extremely large solutions, constitutes a work in progress that will constitute the part two of the present work.

Regularized regressions: the regularized sparse PGD (*rs*-PGD) and the doubly sparse PGD (s^2-PGD)

In the present paper, the term "scarce data limit" does not refer to the fact that in some scenarios the number of samples is smaller than the number of features or basis elements. In our case, it refers to dealing with the exponential growth of a base when working with high-dimensional models, since the growth of base elements is accompanied by the same exponential growth of data to build the model. The idea is to stop the exponential growth of needed data by assuming a separated representation of the solution inspired by

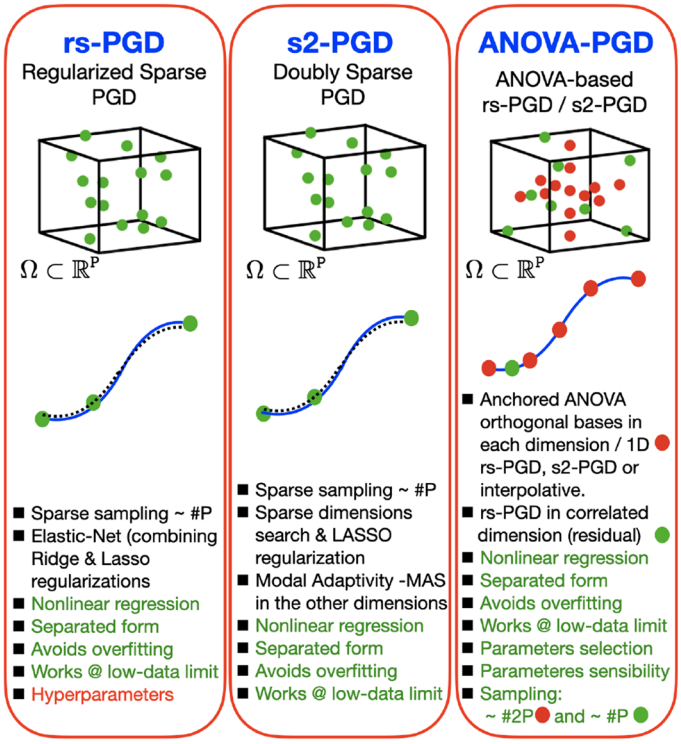

Fig. 2 Part B. Non-intrusive MOR techniques with the main sampling and approximation features, their pros (emphasized in the green text) and the cons (in red)

the so-called proper generalized decomposition [10,11]. The *s*-PGD is proved to be able to achieve good accuracy in this context as seen in [18,24] where a six-input parametric solution of a vehicle crash test is constructed by using less than 22 high-fidelity simulations. Therefore, seeing that the novel proposed strategies (*rs*-PGD, s^2-PGD) can outperform the *s*-PGD (see "Results" section), it seems that they can provide a new tool when addressing this type of challenging problems.

We strongly recommend the following references [18,20] to see how the *s*-PGD is compared with current state-of-the-art techniques in the frameworks where the *s*-PGD is appealing. The same comparison is still valid for the novel *s*-PGD-based strategies but taking into consideration that these last ones can improve results in the scenarios depicted and as shown during the present work.

The PGD-related methods are employed to construct fast multi-parametric solutions of high-fidelity physics-based models. This way, applications including optimization, inverse analysis, uncertainty propagation or simulation based control are enabled with a higher accuracy. An example where the *s*-PGD is widely employed is the aforementioned crash simulation (virtual recreation of a destructive crash test of a car). The proposed PGD-based solutions grow in interest both as the number of the dimensions increases and as the cost of obtaining the snapshots get bigger. The reason is that they deal with the exponential growth of the training data when increasing problem dimensionality. For instance, in [20], it can be observed how a 10-dimensional model is constructed using the *s*-PGD employing a reasonable amount of data. In the same way, a 11-dimensional model

is obtained in [32] to infer the cell battery behaviour of a high-fidelity battery model using a reasonable amount of snapshots and time.

The *rs*-PGD technique is designed to improve the *s*-PGD performance when variables are highly correlated, matrix is nearly singular or it is desired to decrease overfitting. On the other hand, the s^2-PGD is designed to strongly improve sparse identification when just a few terms of the interpolation basis are present thus significantly enhancing the final result.

Two different types of computational expense can be discussed: the one from obtaining the snapshots and the one needed to construct the PGD model.

On the one hand, considering the snapshot cost, the PGD-related techniques (*s*-PGD, s^2-PGD, *rs*-PGD) can greatly reduce the number of needed snapshots in high-dimensional problems. This is possible thanks to the PGD separated representation of the solution. In addition, companies can take advantage of the long-time simulations they have already carried out as well as the previous design of experiments (DoEs) due to the fact that these PGD-related techniques are not linked to a particular sampling strategy. In many applications, this is a great advantage. By contrast, imposing a specific sampling strategy can make them waste months or years of data simulation. However, [18,20] discuss the types of sampling that would be most suitable for this type of techniques when it is possible to create a new DoE. In that references, the LHS is recommended to maximize the rank information in each problem direction thus tending to increase the rank of the PGD operator. It is a convenient method when there is no prior information. Moreover, in [33], the LHS is combined with a mesh constrained to Chebyshev nodes to take advantage of their properties minimizing the Runge's phenomenon. In addition, other sampling strategies can be designed to address a particular problem, thus improving performace. However, to do that, additional insight and priori information about the problem is needed beforehand.

On the other hand, considering the computational effort to construct the model, the *s*-PGD computational expense was discussed in [18,20]. Here, it can be deduced that a light effort to obtain the solutions is achieved because of the choise of quick-computation basis such as polynomial basis. The computation time depends on the problem but it is often in the order of some seconds or minutes. Considering the s^2-PGD and the *rs*-PGD, the computation time can greatly change depending on the number of hyperparameters, the rate of convergence and the chosen tuning strategies. Anyway, they often can be maintained in a suitable time range even though they are more expensive than the *s*-PGD.

In this section, the novel numerical techniques, the regularized sparse PGD (*rs*-PGD) and the doubly-sparse PGD (s^2-PGD), are presented and discussed. The content is divided according to the following subsections:

- In subsection Theoretical background: the *s*-PGD, the theoretical background, from which the proposed methodologies are developed, is presented.
- In subsection *rs*-PGD, the regularized PGD is presented starting from the concepts discussed in Theoretical background: the *s*-PGD.
- In subsection s^2-PGD, the s^2-PGD is presented starting from the concepts presented in *rs*-PGD and Theoretical background: the *s*-PGD.

Theoretical background: the *s*-PGD

The *rs*-PGD and the s^2-PGD are constructed from the theoretical background of the *s*-PGD in the context of regression problems.[1] In this section, this theoretical basis is reviewed and discussed.

Let us consider an unknown function whose approximation is precisely the objective of this work:

$$f(s^1, \ldots, s^d) : \Omega \subset \mathbb{R}^d \rightarrow \mathbb{R},$$

which depends on d different variables s^k, $k = 1, \ldots, d$, considered as dimensions of the state space.

The sparse PGD (*s*-PGD) approach tries to approximate the function f using a low-rank separated (tensor) representation. As in standard PGD procedures, it approximates the function f using a sum of products of one-dimensional functions each one involving one dimension. Each sum is usually called a mode.

This separated form can be expressed as:

$$f(s^1, \ldots, s^d) \approx \tilde{f}^M(s^1, \ldots, s^d) = \sum_{m=1}^{M} \prod_{k=1}^{d} \psi_m^k(s^k), \tag{2.1}$$

where $\tilde{f}^M$ is the approximate, M is the number of modes and ψ_m^k are the one-dimensional function of the mode m and dimension k.

In the *s*-PGD context, functions ψ_m^k, $m = 1, \ldots, M$ and $k = 1, \ldots, d$ are expressed from standard approximation functions:

$$\psi_m^k(s^k) = \sum_{j=1}^{D} N_{j,m}^k(s^k) a_{j,m}^k = (\boldsymbol{N}_m^k)^\top \boldsymbol{a}_m^k, \tag{2.2}$$

where D represents the number of degrees of freedom (nodes) of the chosen approximation. In addition, $\boldsymbol{N}_m^k$ is a column vector with the set of basis functions for the k-th dimension and the m-th mode and $\boldsymbol{a}_m^k$ is a column vector with the coefficients for the k-th dimension and the m-th mode. The important issue here is to know which set of basis functions are best suited for the problem at hand. For example, a Fourier basis or a polynomial basis can be selected.

In the context of regression problems, the goal is to find an approximation $\tilde{f}^M$, which minimizes the distance (usually related to the L2-norm) to the sought function

$$\tilde{f}^M = \arg\min_{f^*} \sum_{i=1}^{n_t} (f(\boldsymbol{s}_i) - f^*(\boldsymbol{s}_i))^2, \tag{2.3}$$

where $\tilde{f}^M$ takes the separated form of Eq. (2.1), n_t is the number of sampling points to train the model and $\boldsymbol{s}_i$ are the different vectors which contain the data points of the training set.

The determination of the coefficients of each one-dimensional function for each mode $m = 1, \ldots, M$ is done by employing a greedy algorithm (described in the next sections) such that, once the approximation up to order $M - 1$ is known, the new M-th order term

[1] We would like to stress the fact that the *s*-PGD is based on some of the ideas of the standard Proper Generalized Decomposition (PGD) method for solving PDEs. For this reason, suggest the reader not familiar with the PGD to review previous works in the field such as [10,11,34], to name but a few.

is found using a non-linear solver (Picard or Newton, for instance):

$$\tilde{f}^M = \sum_{m=1}^{M-1} \prod_{k=1}^{n_d} \psi_m^k(s^k) + \prod_{k=1}^{n_d} \psi_M^k(s^k). \tag{2.4}$$

The final goal of the method is that the function $\tilde{f}$ has to approximate f not only when evaluated in the training set but, notably, in other previously unseen sampling points. This objective is essentially a particular form of *machine learning*. This second goal is more difficult to achieve, yet is more important because this evaluates the predictive ability of the model $\tilde{f}$, that is, the capacity to provide good predictions when the model is fed with previously unseen data. Achieving this is particularly difficult when confronted with a high-dimensional problem, for which data is nearly always sparse and/or scarce.

Indeed, the regression problem described by Eq. (2.3) only guarantees that the minimization is satisfied by the training set, without saying anything at different sampling points. Hence, if there is not an abundance of sampling points in the training set, in the low-data limit, high oscillations may appear out of these measured points because of the increased risk of overfitting. Usually, this is an undesirable effect because it affects the predictive ability of the constructed regression model.

In order to tackle this problem, the *s*-PGD uses the Modal Adaptivity Strategy (MAS) to take advantage of the greedy PGD algorithm. The idea is to minimize spurious oscillations out of the training set by starting the PGD algorithm looking for modes with low degree. When it is observed that the residual decreases slowly or stagnates, higher order approximation functions are introduced. By doing this, oscillations are reduced, since a higher-order basis will try to capture only what remains in the residual.[2]

The MAS has proved to be a good strategy to improve significantly the *s*-PGD performance in many problems, see for instance [2,18,32,33]. However, it has some limitations. For example, it has been observed that the desired accuracy is not achieved before reaching overfitting or the algorithm stops too early when using MAS in some cases. This last issue implies a PGD solution composed of low order approximation functions, thus not getting an as rich as desired function.

In addition, in problems where just a few terms of the interpolation basis are present (that is, there are just some sparse non-zero elements in the interpolation basis to be determined), the strategy fails in recognizing the true model and therefore converging to other one whose predictive performances are bad.

To solve these difficulties, the *rs*-PGD and the s^2-PGD are proposed in what follows. Specifically, the first one is used to increase the predictive capacity beyond the *s*-PGD capabilities and the second one is used to sparse identification and variable selection to construct parsimonious models improving the *s*-PGD explanatory and predictive capabilities.

rs-PGD

For the ease of the exposition and representation but without loss of generality, let us continue by assuming that the unknown objective function $f(x, y)$ lives in $\mathbb{R}^2$,

$$f(x, y) : \Omega \subset \mathbb{R}^2 \to \mathbb{R},$$

and that it is to be recovered from scarce data.

[2] We recommend the reading of [20] and [32] for more information about the MAS.

The goal is therefore to find a function $\tilde{f}^M$ which minimizes the distance to the sought function:

$$\tilde{f}^M = \arg\min_{f^*} \sum_{i=1}^{n_t} \left\| f(x_i, y_i) - f^*(x_i, y_i) \right\|_2^2,$$

and that takes the separated form

$$\tilde{f}^M(x, y) = \sum_{m=1}^{M} X_m(x) Y_m(y) = \sum_{m=1}^{M} \left((\boldsymbol{N}_m^x)^\top \boldsymbol{a}_m^x (\boldsymbol{N}_m^y)^\top \boldsymbol{a}_m^y \right),$$

where n_t is the number of sampling points employed to train the model (training set). Here, the superscript M is employed to highlight the rank of the sought function. How to determine the precise value of M will be detailed hereafter.

In the PGD framework, an iterative scheme based on an alternating direction strategy is usually used to solve the resulting non-linear problem—note that we look for products of one-dimensional functions—and compute $\boldsymbol{a}_M^x$ and $\boldsymbol{a}_M^y$. This strategy computes $\boldsymbol{a}_M^{x,k}$ from $\boldsymbol{a}_M^{y,k-1}$ and $\boldsymbol{a}_M^{y,k}$ from $\boldsymbol{a}_M^{x,k}$ where $\boldsymbol{a}_M^{y,k}$ indicates the values of $\boldsymbol{a}_M^y$ at iteration k of the nonlinear iteration algorithm. The iterations proceed until reaching a fixed point according to a user-specified tolerance.

Defining $\boldsymbol{N}_m^x(x_i)$ and $\boldsymbol{N}_m^y(y_i)$ as the vectors containing the evaluation of the interpolation basis of the m^{th} mode at x_i and y_i, respectively, we can write the following matrix equations defining the systems to solve:

$$\mathbf{M}_x \boldsymbol{a}_M^x = \boldsymbol{r}, \tag{2.5}$$

$$\mathbf{M}_y \boldsymbol{a}_M^y = \boldsymbol{r}, \tag{2.6}$$

where:

$$\boldsymbol{r} = \begin{pmatrix} f(x_1, y_1) - \tilde{f}^{M-1}(x_1, y_1) \\ \vdots \\ f(x_{n_t}, y_{n_t}) - \tilde{f}^{M-1}(x_{n_t}, y_{n_t}) \end{pmatrix},$$

$$\mathbf{M}_x = \begin{pmatrix} (\boldsymbol{N}_M^y(y_1))^\top \boldsymbol{a}_M^y (\boldsymbol{N}_M^x(x_1))^\top \\ \vdots \\ (\boldsymbol{N}_M^y(y_{n_t}))^\top \boldsymbol{a}_M^y (\boldsymbol{N}_M^x(x_{n_t}))^\top \end{pmatrix},$$

$$\mathbf{M}_y = \begin{pmatrix} (\boldsymbol{N}_M^x(x_1))^\top \boldsymbol{a}_M^x (\boldsymbol{N}_M^y(y_1))^\top \\ \vdots \\ (\boldsymbol{N}_M^x(x_{n_t}))^\top \boldsymbol{a}_M^x (\boldsymbol{N}_M^y(y_{n_t}))^\top \end{pmatrix}.$$

If Eqs. (2.5) and (2.6) are solved in the Ordinary Least Squares (OLS) sense:

$$\boldsymbol{a}_M^x = (\mathbf{M}_x^\top \mathbf{M}_x)^{-1} \cdot \mathbf{M}_x^\top \boldsymbol{r}, \tag{2.7}$$

$$\boldsymbol{a}_M^y = (\mathbf{M}_y^\top \mathbf{M}_y)^{-1} \mathbf{M}_y^\top \boldsymbol{r} \tag{2.8}$$

which give us the usual matrix equations in the OLS context.

The *rs*-PGD is based on putting a penalty term when solving (2.5) and (2.6) with the objectives of

(i) reduce overfitting and (ii) deal with strong multicollinearity, namely when the OLS regression problem is ill-posed.

Note that the overfitting problem can easily arise in the *s*-PGD context when high-order approximations (that separated representations exacerbate) are employed because of the usual unstructured low data regime used to train the model. This issue strongly affects the model's ability to perform on new, unseen sets. We illustrate this in the Results section. We notice this effect in the corresponding *s*-PGD results. Therefore, the idea of using the penalty term consists in improving the model's ability to perform on new samples at the cost of increasing the bias or the error model in the training set for a given set of basis functions.

Different regularizations can be envisaged depending on the properties of the problem such as the Tikhonov regularization or the Elastic Net regularization. For the sake of simplicity but without loss of generality, we start introducing the ridge regression regularization (a special case of the Tikhonov regularization) that will be generalized later to lead to the Elastic Net regularization.

For this purpose, we first rewrite Eqs. (2.7) and (2.8):

$$\boldsymbol{a}_M^x = (\mathbf{M}_x^\top \mathbf{M}_x + \lambda \mathbf{I})^{-1} \mathbf{M}_x^\top \boldsymbol{r} \tag{2.9}$$

$$\boldsymbol{a}_M^y = (\mathbf{M}_y^\top \mathbf{M}_y + \lambda \mathbf{I})^{-1} \mathbf{M}_y^\top \boldsymbol{r}, \tag{2.10}$$

where λ is the penalty factor and $\mathbf{I}$ is the identity matrix. In this case, both dimensions are equally penalized but different penalty factors could be considered depending on the considered dimension.

The regularized problems associated to Eqs. (2.9) and (2.10) are:

$$\boldsymbol{a}_M^x = \underset{\boldsymbol{a}_M^{x*}}{\arg\min} \left\{ \left\| \boldsymbol{r} - \mathbf{M}_x \boldsymbol{a}_M^{x*} \right\|_2^2 + \lambda \left\| \boldsymbol{a}_M^{x*} \right\|_2^2 \right\}, \tag{2.11}$$

$$\boldsymbol{a}_M^y = \underset{\boldsymbol{a}_M^{y*}}{\arg\min} \left\{ \left\| \boldsymbol{r} - \mathbf{M}_y \boldsymbol{a}_M^{y*} \right\|_2^2 + \lambda \left\| \boldsymbol{a}_M^{y*} \right\|_2^2 \right\}, \tag{2.12}$$

where the problem is divided in solving a ridge regression problem for each dimension when computing $\boldsymbol{a}_M^x$ and $\boldsymbol{a}_M^y$ during the alternate direction fixed point strategy.

The interpretation of employing Eqs. (2.11) and (2.12) during the PGD iterative scheme can be thought of as an attempt of solving the following problem within the PGD rationale:

$$\tilde{f}^M(\boldsymbol{a}_M^x, \boldsymbol{a}_M^y) = \underset{\boldsymbol{a}_M^{x*}, \boldsymbol{a}_M^{y*}}{\arg\min} \left\{ \left\| f - \tilde{f}^M(\boldsymbol{a}_M^{x*}, \boldsymbol{a}_M^{y*}) \right\|_2^2 + \lambda \left\| \boldsymbol{a}_M^{x*} \right\|_2^2 + \lambda \left\| \boldsymbol{a}_M^{y*} \right\|_2^2 \right\}, \tag{2.13}$$

where $\|\cdot\|_2$ is the Euclidean norm, and $\tilde{f}^M$ is the function defined in (2.4) where the new M-th order term of the model is sought.

As the terminology used in this section shows, a regularization problem is formulated at each enrichment step. Thus, we are looking for the best penalty factor at each updating stage, adapting the regularization whenever the approach is enriched. Other possibilities can be envisaged but this one seems the one which offers the best results according to our numerical experiments.

A null intercept term was assumed for $\boldsymbol{a}_M^x$ and $\boldsymbol{a}_M^y$ in the deduction of Eqs. (2.9), (2.10), (2.11) and (2.12). If this term is going to be included, it can be treated as in standard ridge procedures when solving the corresponding linear regularized regression problem for each dimension during the alternating direction strategy.

As we are generally looking for the mode with best predictive abilities in each enrichment, the proposed criterion to choose λ is to perform a k-fold cross-validation and select the value of λ that minimizes the cross-validated sum of squared residuals (or some other

measure). It is also possible to use the "one-standard error" rule (heuristic) with cross-validation, in which we choose the most penalized model whose error is no more than one standard error above the error of the best model. Such a rule acknowledges the fact that the tradeoff curve is estimated with error, and hence takes a conservative approach [14].

If enough data is available, the split of the training set in two subgroups is equally a reasonable option to select λ and in addition, computationally less demanding. In this case, one subgroup is employed for constructing the model and the other one to evaluate the predictive ability and then to select λ accordingly.

The Elastic Net regularization results of including a L1-norm regularization, from which Eqs. (2.11)–(2.12) and Eq. (2.13) become:

$$\boldsymbol{a}_M^x = \underset{\boldsymbol{a}_M^{x*}}{\arg\min} \left\{ \left\| \boldsymbol{r} - \mathbf{M}_x \boldsymbol{a}_M^{x*} \right\|_2^2 + \lambda \left[(1-\alpha) \left\| \boldsymbol{a}_M^{x*} \right\|_2^2 + \alpha \left\| \boldsymbol{a}_M^{x*} \right\|_1 \right] \right\}, \tag{2.14}$$

$$\boldsymbol{a}_M^y = \underset{\boldsymbol{a}_M^{y*}}{\arg\min} \left\{ \left\| \boldsymbol{r} - \mathbf{M}_y \boldsymbol{a}_M^{y*} \right\|_2^2 + \lambda \left[(1-\alpha) \left\| \boldsymbol{a}_M^{y*} \right\|_2^2 + \alpha \left\| \boldsymbol{a}_M^{y*} \right\|_1 \right] \right\}, \tag{2.15}$$

and

$$\begin{aligned} \tilde{f}^M(\boldsymbol{a}_M^x, \boldsymbol{a}_M^y) = \underset{\boldsymbol{a}_M^{x*}, \boldsymbol{a}_M^{y*}}{\arg\min} \Bigg\{ & \left\| f - \tilde{f}^M(\boldsymbol{a}_M^{x*}, \boldsymbol{a}_M^{y*}) \right\|_2^2 \\ & + \lambda \left[(1-\alpha) \left(\left\| \boldsymbol{a}_M^{x*} \right\|_2^2 + \left\| \boldsymbol{a}_M^{y*} \right\|_2^2 \right) + \alpha \left(\left\| \boldsymbol{a}_M^{x*} \right\|_1 + \left\| \boldsymbol{a}_M^{y*} \right\|_1 \right) \right] \Bigg\}, \end{aligned} \tag{2.16}$$

respectively, where $\alpha \in [0, 1)$ and λ are the penalty factors. These coefficients could be also different for the different dimensions, and the lambda coefficients also different for the norm L2 and L1. The limit cases $\alpha = 0$ and $\alpha = 1$ result in the Ridge and Lasso regressions respectively.

It is worth highlighting the fact that the elastic net procedure is used in case ridge does not achieve the desired performance, in which case the hyperparameter alpha would be added. In addition, alpha is selected using the state-of-the-art machine learning tools to tune hyperparameters. In the first test, we recommend ridge regression because: 1. It is faster. 2. It has not the hyperparameter alpha.

s^2-PGD

For the ease of the exposition and representation but without loss of generality, let us continue by assuming the same two-dimensional unknown function discussed in Section *rs*-PGD.

Here, we are dealing with a solution which admits a sparse solution for a certain basis using the PGD separated form (2.1). In this case, the goal is to identify the correct non-zero coefficients at each enrichment step in order to guide the approach to the correct separated representation.

Without a roadmap to select these nonzero coefficients, the traditional *s*-PGD fails to capture the true relationship between the model's features as well as its final response. Furthermore, if high-order terms appear in the searched function, these issues become even worse leading to serious overfitting issues.

Let us consider the theory discussed in the previous section but now considering the L1 regularization with the idea to promote sparsity in the overall solution of the nonlinear regression problem:

$$\tilde{f}^M(\boldsymbol{a}_M^x, \boldsymbol{a}_M^y) = \underset{\boldsymbol{a}_M^{x*}, \boldsymbol{a}_M^{y*}}{\arg\min} \left\{ \left\| f - \tilde{f}^M(\boldsymbol{a}_M^{x*}, \boldsymbol{a}_M^{y*}) \right\|_2^2 + \lambda \left\| \boldsymbol{a}_M^{x*} \right\|_1 + \lambda \left\| \boldsymbol{a}_M^{y*} \right\|_1 \right\}. \tag{2.17}$$

This formulation is convenient because the nonlinear problem can be solved using the PGD constructor [1,10], with an alternate direction fixed point strategy, where just a LASSO regression problem is considered in each dimension.

Therefore, the regression problems for the iterative scheme will be:

$$\boldsymbol{a}_M^x = \arg\min_{\boldsymbol{a}_M^{x*}} \left\{ \left\| \boldsymbol{r} - \mathbf{M}_x \boldsymbol{a}_M^{x*} \right\|_2^2 + \lambda \left\| \boldsymbol{a}_M^{x*} \right\|_1 \right\}, \tag{2.18}$$

$$\boldsymbol{a}_M^y = \arg\min_{\boldsymbol{a}_M^{y*}} \left\{ \left\| \boldsymbol{r} - \mathbf{M}_y \boldsymbol{a}_M^{y*} \right\|_2^2 + \lambda \left\| \boldsymbol{a}_M^{y*} \right\|_1 \right\}. \tag{2.19}$$

That consists of solving a LASSO regression problem for each dimension when computing $\boldsymbol{a}_M^x$ and $\boldsymbol{a}_M^y$ within the alternate direction fixed point strategy. Moreover, as previously discussed, in the present case again, both dimensions are equally penalized but different penalty factors could be envisaged.

As we are iteratively solving a LASSO problem in each direction, we will end up with sparse solutions for each one-dimensional function choosing the right penalty factor. Again, a null itercept term was assumed.

In case of looking for sparsity just in the x dimension, only Eq. (2.18) applies for computing coefficients $\boldsymbol{a}_M^x$, whereas coefficients $\boldsymbol{a}_M^y$ are calculated by invoking the standard s-PGD or the rs-PGD, addressed in the previous section.

To determine λ, we first refer the reader to the discussion of the previous section. Then, the following considerations applied in the case of the doubly sparse PGD:

- Before selecting a model according to the predictive criterion, a filter is considered taking only the models with a minimum sparsity criterion $\left\| \boldsymbol{a}_M^x \right\|_0 \leq \chi_x^{lim}$. If sparsity is also desired in y direction, χ_y^{lim} will be defined accordingly. Note: We define $\|\cdot\|_0$ by $\|\boldsymbol{x}\|_0 = \#\{i : \boldsymbol{x}_i \neq 0\}$. We consider this notation even if it is actually not a norm.
- Once model selection is performed, the OLS methodology is employed with the detected non-zero elements to obtain the correct update. The reason of this step is that LASSO regression terms are in general not accurate, and so it may be necessary to debias the obtained values. Remember that the LASSO shrinkage causes the estimates of the non-zero coefficients to be biased towards zero and in general they are not consistent [6,14].

If there is prior or physical knowledge about the solution, it can be used to decide the direction to penalize and, in fact, this often helps to successfully decide on the rigth dimension. If there is no prior knowledge, usual machine learning strategies to tune hyperparameters can be employed.

Finally, the enrichment procedure for the s^2-PGD strategy (where LASSO regularization is employed to promote sparsity, that is, $\alpha = 1$) is:

1 Compute different mode enrichments changing the penalty factor.
2 Select the best one considering the defined accuracy metric and if desired, the other commented robustness rules such as the χ_x^{lim} filter.
3 The selected mode is employed to identify the non-zero elements. Then, as previously indicated, the OLS methodology is used with the detected non-zero elements to obtain the correct update. For the non-sparse dimensions, the s-PGD MAS strategy is considered.

In the present work, the lasso problems are solved by employing the algorithm implemented in [26] where the Coordinate Descent and ADDM algorithms are used [5,13]

The ANOVA-based sparse-PGD

The ANOVA decomposition of a function $f(s^1, \dots, s^d) : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ is an orthogonal decomposition based on the analysis of variance, a statistical model designed for data analysis. Thus, the function $f(\boldsymbol{s})$ can be written as a sum of orthogonal functions:

$$f(\boldsymbol{s}) = f_0 + \sum_{i=1}^{d} f_i(s^i) + \sum_{i_1=1}^{d} \sum_{i_2=i_1}^{d} f_{i_1,i_2}(s^{i_1}, s^{i_2}) + \dots + f_{1,2,\dots d}(s^1, s^2, \dots, s^d), \tag{3.1}$$

satisfying

$$\mathbb{E}_i(f_{i_1,\dots,i_k}(s^{i_1}, \dots, s^{i_k})) = 0, \tag{3.2}$$

where $\mathbb{E}_i$ stands for the expectation with respect to any coordinate i in the set $(i_1, \dots, i_k)$, with $1 \leq k \leq d$. This property results in the orthogonality of functions involved in the previous decomposition.

To prove it, consider for example a simple 2D case with, $\boldsymbol{s} = (x, y), f(\boldsymbol{s}) \equiv f(x, y)$. Thus, with $\mathbb{E}_x(f_x(x)) = 0$, $\mathbb{E}_x(f_{x,y}(x, y)) = 0$ and $\mathbb{E}_y(f_{x,y}(x, y)) = 0$, we have $\mathbb{E}_{x,y}(f_{x,y}(x, y) f_x(x)) = \mathbb{E}_x\{\mathbb{E}_y(f_{x,y}(x, y)) f_x(x)\} = 0$.

The number of function involved in the decomposition (without considering the constant term) is $2^d - 1$, and they can be parametrized by the integer n, $n = 1, \dots, 2^d - 1$. The different functions involved in the ANOVA decomposition can be expressed from expectations according to:

$$\begin{cases} \mathbb{E}(f(\boldsymbol{s})) = f_0 \\ \mathbb{E}(f(\boldsymbol{s}|s^i)) = f_i(s^i) + f_0 \\ \mathbb{E}(f(\boldsymbol{s}|s^i, s^j)) = f_{i,j}(s^i, s^j) + f_i(s^i) + f_j(s^j) + f_0 \\ \vdots \end{cases} \tag{3.3}$$

where $\mathbb{E}(f(\boldsymbol{s}|s^i))$ refers to the integration on all the variables except s^i.

Sensitivity analysis: Sobol coefficients

The variance of $f(\boldsymbol{s})$, which we refer to as $\mathrm{Var}(f(\boldsymbol{s}))$, taking into account the orthogonality of the functions involved in the ANOVA decomposition, reads

$$\mathrm{Var}(f(\boldsymbol{s})) = \sum_{n=1}^{2^d-1} \mathbb{E}\left(f_n(\boldsymbol{s})\right)^2 = \sum_{n=0}^{2^d-1} \mathrm{Var}_n, \tag{3.4}$$

that allows defining the so-called Sobol sensitivity coefficients $\mathcal{S}_n$

$$\mathcal{S}_n = \frac{\mathrm{Var}_n}{\mathrm{Var}(f(\boldsymbol{s}))}. \tag{3.5}$$

The *anchored* ANOVA

Multidimensional settings imply expensive calculations for computing the multidimensional expectations. For alleviating those costly computations we introduce the so-called anchor point $\boldsymbol{c}$ such that $f_0 = f(\boldsymbol{c})$. Then, in the definition of the functions involved in the ANOVA decomposition, the expectations are replaced by $f(\boldsymbol{c}|\boldsymbol{s}_n)$, that is, the particularization of the function in the anchor point, except for those coordinates involved in $\boldsymbol{s}_n$.

Combining the *anchored*-ANOVA with the sparse PGD

A valuable strategy consists in: (i) first, using the standard *anchored*-ANOVA for evaluating the functions depending on each dimension $f_i(s^i)$, $i = 1, \ldots, d$, by using an adequate sampling, a sort of multidimensional cross centered at the anchor point $\boldsymbol{c}$.

Let $\boldsymbol{c} = (c_1, c_2, \ldots, c_d)$ and $q_i^1, q_i^2, \ldots, q_i^{n_i}$ be one-dimensional sampling points for the dimension s^i. In each dimension, $f_i(s^i)$ can be approximated by using any 1D interpolation method such as polynomial regression, kriging, piecewise polynomial interpolation, ...For instance the term f_1 can be approximated from the data collected at the points $\left(q_1^j, c_2, \ldots, c_d\right)$ such that $f_1\left(q_1^j\right) \approx f\left(q_1^j, c_2, \ldots, c_d\right) - f_0$. In this work, the f_i are approximated using a spline interpolation. Then, (ii) one could compute the residual $f'(\boldsymbol{s})$:

$$f'(\boldsymbol{s}) = f(\boldsymbol{s}) - f_0 - \sum_{i=1}^{d} f_i(s^i), \tag{3.6}$$

and finally, (iii) using the *rs*-PGD, or thes^2-PGD, for approximating that residual $f'(\boldsymbol{s})$ that contains the different correlations. In that case, an enhanced sparse sampling can be considered by increasing the density of the sampling points near the boundaries of the parametric domain.

Results

In this section, the results of using the above techniques are shown for different cases.

The *rs*-PGD examples are chosen to see how the *s*-PGD overfitting is reduced thanks to the proposed strategy. The s^2-PGD examples are selected to see how the proposed technique improve the *s*-PGD results due to a better model indentification. The ANOVA PGD aims at allying orthogonal hierarchical bases with a more favorable scaling (with respect to the SSL) of the amount of data with the approximation richness. The ANOVA PGD example is chosen to easily illustrate the proposed strategy and how it can improve the results.[3]

First, in Section Results for the *rs*-PGD approach, the error reduction is shown when using the *rs*-PGD comparing with the classical procedure (*s*-PGD). Then, in Section Checking the performance of s^2-PGD when addressing sparse solutions, sparse identification and error reduction is presented when using the s^2-PGD comparing with the standard sparse procedure (*s*-PGD). Finally, Section ANOVA-PGD numerical results employes the analysis of variance and combines it with regularized approximations to define an original and powerful regression methodology.

Results for the *rs*-PGD approach

The following examples considers the Elastic Net Regularization. For that purpose, an α parameter is employed for combining the Ridge and Lasso regression. The α parameter is selected by running the algorithm several times for different α values, and then choosing the one which has better predictive performances.

The error reduction depends on the nature of the sought function to be built into the PGD separated representation. For instance, if the sought function contains some sparsity, the lasso penalty or Elastic Net need with α close to 1 will produce a greater error reduction than ridge or Elastic Net with α close to 0.

[3] A part two of the present work, which constitutes a work in progress, will study the general workflow for allying the PGD-based techniques under the ANOVA procedure for the solution of a given problem.

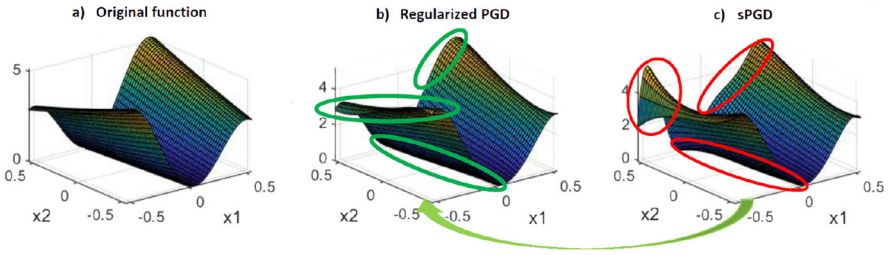

Fig. 3 Comparing the reference (Eq. (4.1)) and its associated *s*-PGD and *rs*-PGD regressions, at points $(x_1, x_2, x_3 = 0, x_4 = 0, x_5 = 0.7071)$

Generally, it is not known beforehand which regularization might work best for a given problem (it can also depend on the collected data, function properties, ...) Therefore, a hyperparemeter tuning job has to be performed for α: The α parameter is selected by running the algorithm several times for different values. Note that ridge and lasso are particular α values of the Elastic Net procedure. This is the reason why various values of α are tested. Therefore, if the ridge strategy gives, for instance, a better error reduction, it will be detected in the previous hyperparameter tuning job (selecting in this case $\alpha = 0$ as the best run).

A first example involving a five dimensional polynomial

In the first example, we are trying to approximate the five-dimensional function

$$f(x_1, x_2, x_3, x_4, x_5) = (8x_1^3 - 6x_1 - 0.5x_2)^2 + (4x_3^3 - 3x_3 - 0.25x_4)^2 + 0.1(2x_5^2 - 1). \tag{4.1}$$

The above function is to be reconstructed in the domain $\Omega = [-0.51, 0.51]^5$. The sampling for the training set contains 160 points. Therefore, only these points are used to construct the model either using the *s*-PGD or the *rs*-PGD methodology. In addition, the Latin hypercube sampling (LHS) is used to generate this set of data.

A testing set of 54,000 untrained points is considered to compare the results between techniques when predicting unseen scenarios. This second set will be used to study the predictive ability of both models once they are finally constructed.

A standard MAS employing up to 4th degree polynomials for both the s-PGD and the *rs*-PGD is considered. To measure the error of both methodologies in the testing set, the following error criterion is used:

$$\text{err}_{pgd} = \frac{\|\boldsymbol{z} - \boldsymbol{z}_{pgd}\|_2}{\|\boldsymbol{z}\|_2}; \qquad \text{err}_{rpgd} = \frac{\|\boldsymbol{z} - \boldsymbol{z}_{rpgd}\|_2}{\|\boldsymbol{z}\|_2};$$

where $\boldsymbol{z}$ is the vector containing the values of $f(x_1, x_2, x_3, x_4, x_5)$ in the testing set, $\boldsymbol{z}_{pgd}$ and $\boldsymbol{z}_{rpgd}$ are the vectors containing the prediction in the testing set of both methodologies (*s*-PGD and *rs*-PGD, respectively).

After employing the discussed techniques in the above conditions, we obtain in this example that the error is reduced by 52.38 % using the *rs*-PGD with $\alpha = 0.1$.

To perceive the improvements and the overfitting reduction, in Figure 3, we show a plot of the original function $f(x_1, x_2, x_3 = 0, x_4 = 0, x_5 = 0.7071)$. It can be noticed that the *rs*-PGD corrects the shape of the function in the indicated areas in Fig. 3, improving the performance of the regression.

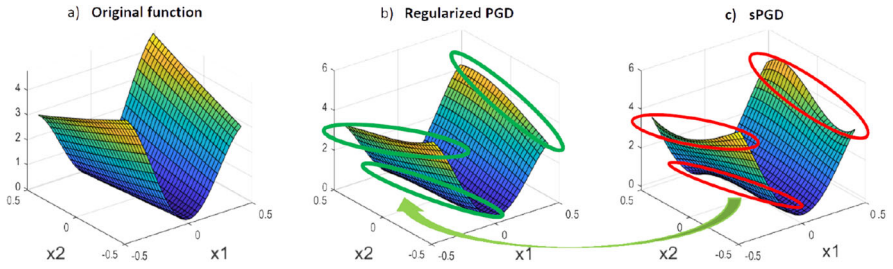

Fig. 4 Comparing the reference (Eq. (4.1)) and its associated *s*-PGD and *rs*-PGD regressions, at points $(x_1, x_2, x_3 = -0.17069, x_4 = -0.17069, x_5 = -0.015517)$

This improvement occurs over the whole five-dimensional domain. Other result is shown in Fig. 4 that depicts $f(x_1, x_2, x_3 = -0.17069, x_4 = -0.17069, x_5 = -0.015517)$.

A second example involving five dimensions with trigonometric and logarithmic functions

In this second example, we are trying to approximate the function:

$$f(x_1, x_2, x_3, x_4, x_5) = \cos(x_1 x_2)\Big[\big(\sin(2x_3) - 3.14\big)\log(3x_4 + 1.5)\cos(x_5) + \exp(x_4)\cosh(x_3)\sinh(x_5)\Big], \tag{4.2}$$

by using the *rs*-PGD with polynomials. The above function is intended to be reconstructed in the domain $\Omega = [-1, 1]^5$.

In this case, the sampling for the training set contains 390 points. Therefore, only these points are used to construct the model either by using the *s*-PGD or the *rs*-PGD methodology. In addition, the Latin hypercube sampling is used to generate this set of data.

A testing set of 2000 untrained points is available to compare the results when predicting unseen scenarios. Again a standard MAS is employed reaching 4th degree polynomials in both, the s-PGD and the *rs*-PGD. An error reduction of about 47% is accomplished with $\alpha = 0.5$.

Checking the performance of s^2-PGD when addressing sparse solutions

A first example involving sparsity in one dimension

In the first example of this Section, we are trying to approximate the function:

$$f(x_1, x_2, x_3) = (\sin(2x_1) - 3.14)T_5(x_2) + \exp(x_3)\cosh(x_1), \tag{4.3}$$

by using a Chebyshev basis for the one-dimensional functions of the PGD. The above function is intended to be reconstructed in the domain $\Omega = [-1, 1]^3$. Please note that in this work, we employ the terminology T_n to denote the Chebyshev polynomials of the first kind.

Moreover, the sampling for the training set is created using a sparse grid based on the Smolyak quadrature rule [3,21] of level 3 based on the Clenshaw-Curtis univariate quadrature rule. Therefore, only these points are used to construct the model either using the *s*-PGD or the s^2-PGD methodology. In Fig. 5, the mesh used for the training set is shown.

A testing set of 27,000 untrained points is available to compare the results between techniques when predicting unseen scenarios. This second set will be used to study the predictive ability of both models once they are finally constructed.

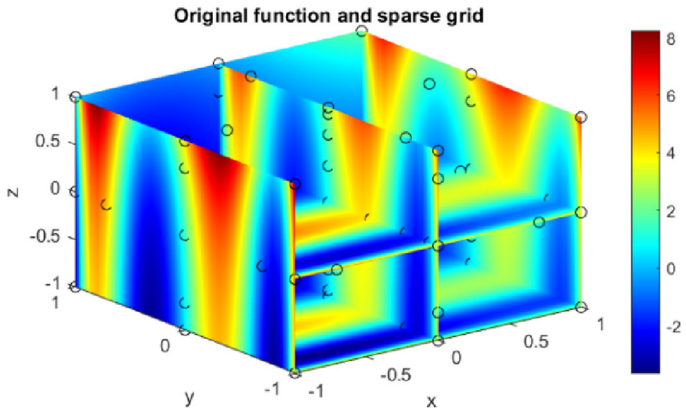

Fig. 5 Plot of the original function and the training set (circles) used to construct the PGD models

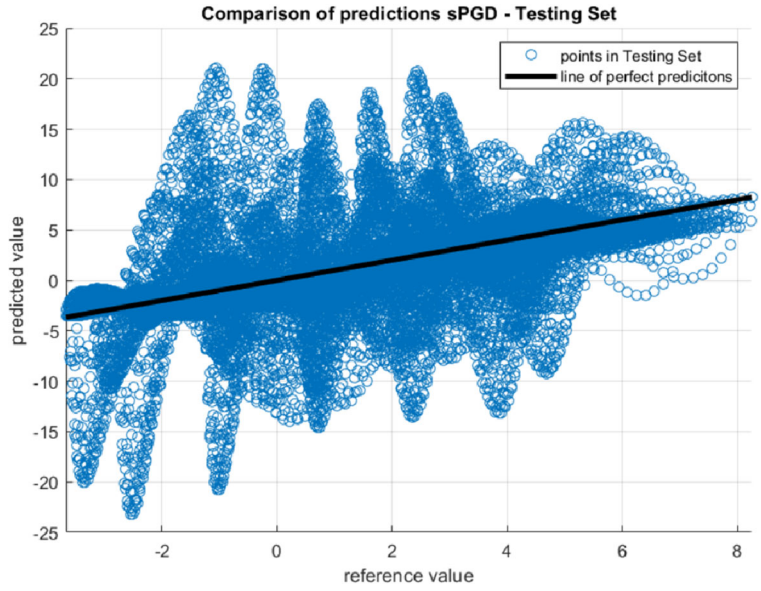

Fig. 6 Problem defined in Eq. (4.3): Comparison of predicted s-PGD values with the reference ones in the testing set (the black line represents a perfect prediction)

The conditions to employ the s^2-PGD in this example are the following. A basis reaching eighth-degree polynomials is chosen for the sparse dimension. Moreover, a standard MAS-based s-PGD is used, reaching 4th degree polynomials along the non-sparse dimensions.

In Figure 6, the results of the standard s-PGD are shown. In this case, we can see that the predictions are bad because this methodology completely fails in finding this type of sparse solutions. This is one of the problems that the s-PGD is facing and we propose to solve with the s^2-PGD.

In addition, if we observe the s-PGD solution we can see that all the possible elements are nonzero, so it fails in identifying the sparsity. To detect sparsity, three simulations of the s^2-PGD are carried out, penalizing a different dimension at each iteration. Consequently, the model with best predictive ability (out of the training set) will be the selected one. For instance, x_1 is supposed sparse in the first simulation, x_2 is supposed sparse in the second simulation, and so on. As expected, the chosen model is the one obtained when penalizing the x_2 dimension.

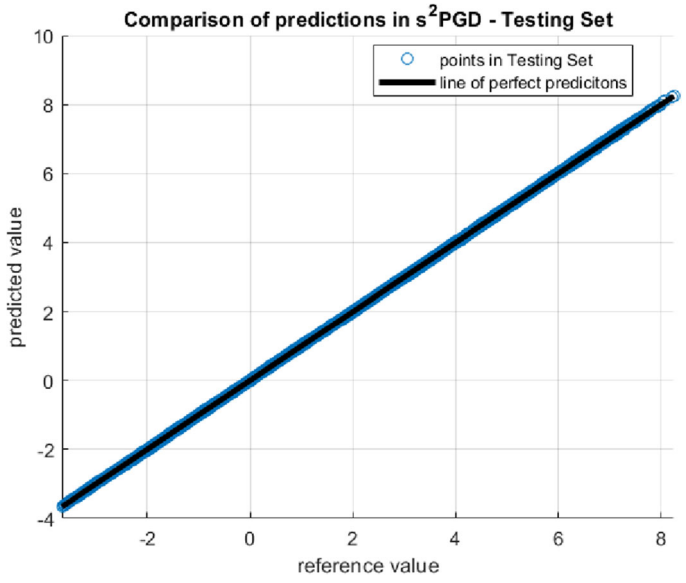

Fig. 7 Problem defined in Eq. (4.3): Comparison of predicted s^2-PGD values with the reference ones in the testing set (the black line represents a perfect prediction)

Table 1 Example of mode enrichment when constructing the s^2-PGD solution of problem defined in Eq. (4.3)

	x_1	x_2	x_3
T_0	0.0312	0.0016	0.6806
T_1	0.1307	0	7.78E−09
T_2	0.1184	0	0.0009
T_3	0.0097	0	1.26E−07
T_4	0	0	0
T_5	0	5.8792	0
T_6	0	0	0
T_7	0	0	0
T_8	0	0	0

As it can be observed, the method can correctly detect the non-zero elements in the sparse dimension of the separated representation

In Fig. 7, the results of the s^2-PGD are presented. As we can observe, predictions are almost perfect. In this case the solution is correctly identified using four modes, that is, four sums of the PGD decomposition. In Table 1 we can observe an example of mode enrichment where the correct non-zero elements are identified in the sparse dimension.

The errors concerning the s-PGD and the s^2-PGD solutions are respectively $\text{err}_{pgd} = 141$ % and $\text{err}_{s^2pgd} = 0.56$ %.

A second example involving sparsity in two dimensions

In this case we consider the approximations problem of function

$$\begin{aligned} f(x_1, x_2, x_3, x_4, x_5) = &\big[T_5(x_1) + 2T_1(x_1)\big]\big[T_2(x_2) + 2T_4(x_2)\big] \\ &\Big[\big(\sin(2x_3) - 3.14\big)\log(3x_4 + 1.5)\cos(x_5) \\ &+ \exp(x_4)\cosh(x_3)\sinh(x_5)\Big] \end{aligned} \tag{4.4}$$

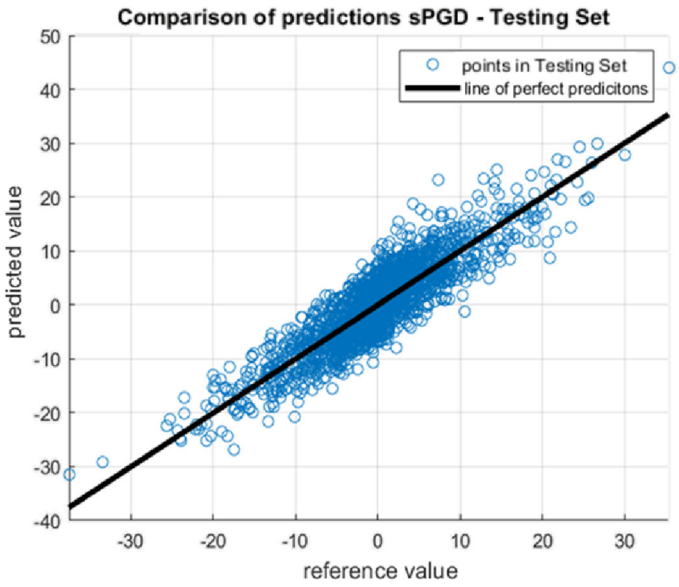

Fig. 8 Problem defined in Eq. (4.4): Comparison of predicted *s*-PGD values with the reference ones in the testing set (the black line represents a perfect prediction)

by using a Chebyshev approximation basis for the one-dimensional functions involved in the PGD constructor.

The above function is intended to be reconstructed in the domain $\Omega = [-1, 1]^5$. The sampling for the training set contains 490 points. In addition, the Latin hypercube sampling is used to generate this random set of data.

A testing set of 2000 untrained points is available to compare the results between techniques when predicting unseen scenarios. As in the previous examples, this second set will be used to study the predictive ability of both models once they are finally constructed.

Concerning the s^2-PGD a basis reaching sixth-degree polynomials is chosen for the sparse dimensions. Moreover, a standard MAS is used, up-to 4th degree polynomials, in the non-sparse dimensions.

In Fig. 8, the results of the standard *s*-PGD are shown. In this case, we can see that the predictions are bad. This is due to the wrong identification of the non-zero elements in the separated representation, which causes overfitting problems. This is a proof of the limitations that the *s*-PGD can find. The s^2-PGD is designed to address that.

To detect sparsity, five different simulations of the s^2-PGD are carried out, penalizing one different dimension each time. In other words, the algorithm is employed five times but changing the dimension to penalize to seek sparsity. Consequently, the model with best predictive ability (out of the training set) will be the selected one. For instance, x_1 is supposed sparse in the first simulation, x_2 is supposed sparse in the second simulation, and so on. As expected, the chosen model is the one obtained when penalizing the x_1 dimension. The reason is that in this case, we observe that the correct non-zero terms for x_1 and x_2 are identified just penalizing x_1.

In Fig. 9, the results of the s^2-PGD are presented. An exellent agreement between the real function and the proposed approach is observed. Furthermore, if we examine the modes of the s^2-PGD solution, we can see that the model has correctly identified the non-zero elements in the two sparse dimensions. For instance, see Table 2 as a mode

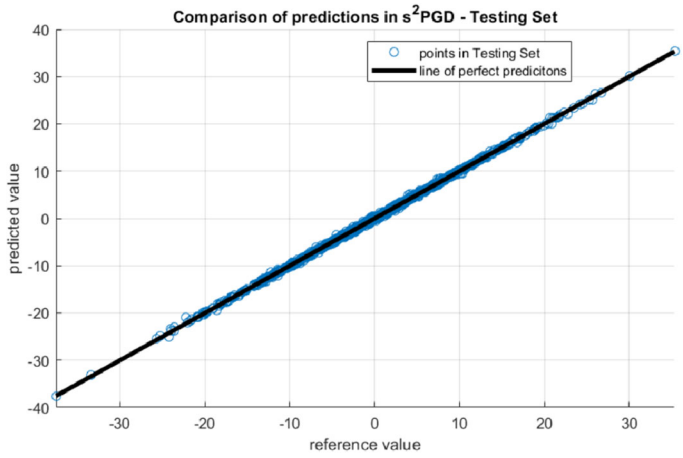

Fig. 9 Problem defined in Eq. (4.4): Comparison of predicted s^2-PGD values with the reference ones in the testing set (the black line represents a perfect prediction)

Table 2 Example of mode enrichment when constructing the s^2-PGD solution of problem defined in Eq. (4.4)

	x_1	x_2	x_3	x_4	x_5
T_0	0	0	1.3218	0.1658	0.2213
T_1	0.369	0	−3.6109	0.5287	0.3541
T_2	0	−0.1236	4.387	0.1045	0.5085
T_3	0	0	−1.7597	0.7351	0.3519
T_4	0	0.0649	0	0	0
T_5	−0.1788	0	0	0	0
T_6	0	0	0	0	0

As it can be observed, the method can correctly detect the non-zero elements in the two sparse dimensions of the separated representation

example. In addition, this PGD solution needed 104 modes, that is, 104 sums of the PGD decomposition, solution that can be re-compacted by invoking again the PGD [10].

Finally, the errors concerning the s-PGD and the s^2-PGD solutions are respectively $\text{err}_{pgd} = 46.39\ \%$ and $\text{err}_{s^2pgd} = 2.4\ \%$.

A third example involving more dimensions

In the third example of this Section, we are trying to approximate a challenging function involving eight dimensions:

$$\begin{aligned} f(\vec{x}) = &\left(180x_3^3T_3(x_1)T_2(x_2) + 120x_3^3T_1(x_1)T_2(x_2) + 144x_3^2T_3(x_1)T_2(x_2)\right.\\ &+96x_3^2T_1(x_1)T_2(x_2) + 18x_3T_3(x_1)T_2(x_2) + 12x_3T_1(x_1)T_2(x_2)\\ &\left.-18T_3(x_1)T_2(x_2) - 12T_1(x_1)T_2(x_2)\right)\left(\sin(2x_4) - 3.14\right)\left(\log(3(x_5 + 1.5))\right)\\ &(6x_6^3 - 9x_6^2 - x_6 - 2)(-6x_7^3 + 9x_7^2 + 4x_7 - 2)(-6x_8^3 - 4x_8^2 + 2x_8 - 2), \end{aligned} \tag{4.5}$$

by using a Chebyshev basis for the one-dimensional functions of the PGD constructor. Note that $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$.

The sampling for the training set contains 2900 points. In addition, the Latin hypercube sampling is used to generate this random set of data.

A testing set of 3000 untrained points is available to compare the results between s-PGD and s^2-PGD techniques when predicting unseen scenarios. As in the previous examples,

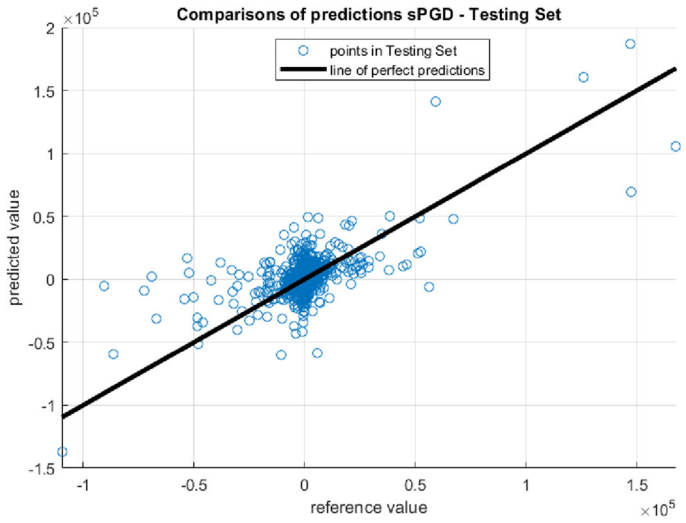

Fig. 10 Problem defined in Eq. 4.5: Comparison of predicted *s*-PGD values with the reference ones in the testing set (the black line represents a perfect prediction)

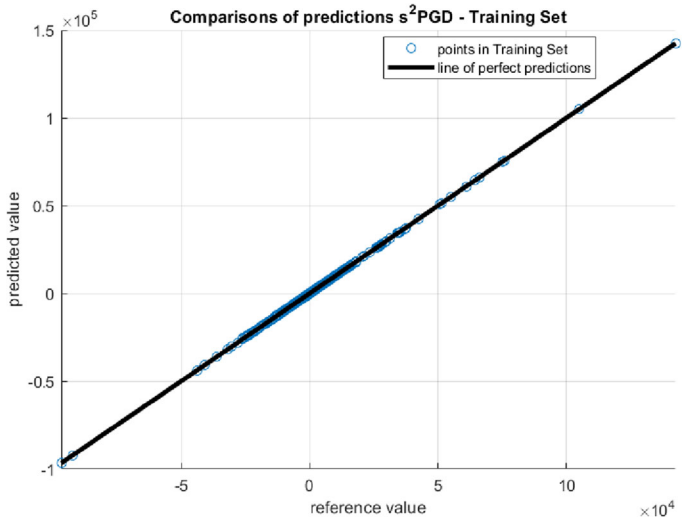

Fig. 11 Problem defined in Eq. 4.5: Comparison of predicted s^2-PGD values with the reference ones in the training set (the black line represents a perfect prediction)

this second set will be used to study the predictive ability of both models once they are finally constructed.

As in the previous examples, the standard MAS strategy is used for the *s*-PGD. On the other hand, dimensions x_1 and x_2 are penalized in the s^2-PGD algorithm to detect sparsity.

In Fig. 10, the results of the standard *s*-PGD are shown. In this case, we can see that the predictions are bad. This is due to the wrong identification of the non-zero elements in the separated representation. This is another proof of the limitations that the *s*-PGD can find in several scenarios. The s^2-PGD is designed to address that to achieve the correct sparse identification.

In Figs. 11, 12, the results of the s^2-PGD are presented. An exellent agreement between the real function and the proposed approach is observed. This is true for the training set

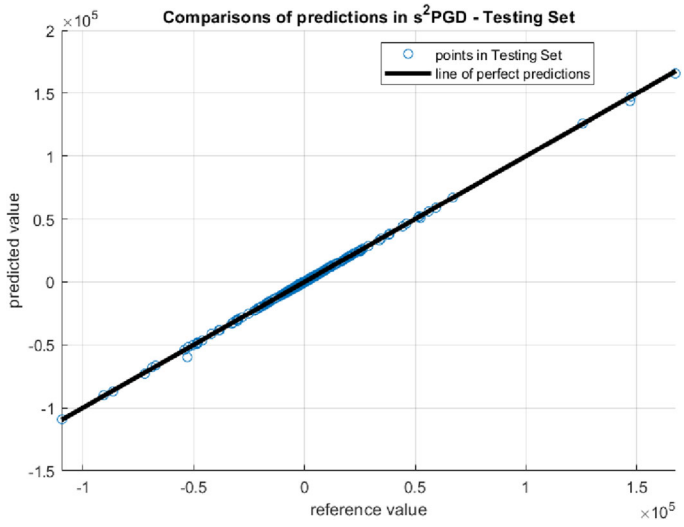

Fig. 12 Problem defined in Eq. 4.5: Comparison of predicted s^2-PGD values with the reference ones in the testing set (the black line represents a perfect prediction)

and also for unseen scenarios such as the untrained set of points employed to check the predictivie ability (Fig. 12).

Finally, the errors concerning the s-PGD and the s^2-PGD solutions are respectively $\text{err}_{pgd} = 87.17$ % and $\text{err}_{s^2pgd} = 1.9$ %.

ANOVA-PGD numerical results

ANOVA-PGD regression consists of applying regression techniques (such as standard interpolation, s-PGD, rs-PGD or s^2-PGD) separately to the different terms (or groups of terms) in the ANOVA decomposition. This strategy suggests the MAS since it enforces some simplicity in the first modes, even if here richer approximations can be envisaged, but it also provides other benefits through the orthogonality of the decomposition and the opportunity to work in a low dimension setting, as previously expossed.

Here, we consider the numerical test related to the 2D function

$$f(x, y) = -2\cos(3x^{1.75}) + 10\log(y - 0.6)^4 + 6\cos(x)(y - 0.3y^2), \tag{4.6}$$

that perfectly fits the ANOVA structure, despite the functional complexity of the terms involving the coordinates x and y, $2\cos(3x^{1.75})$ and $10\log(y - 0.6)^4$ respectively, and the one coupling both coordinates, $6\cos(x)(y - 0.3y^2)$.

When considering the ANOVA-based sampling consisting of the center point of the parametric domain acting as the anchor $\boldsymbol{c} = (x_c, y_c)$, 10 additional points in the first dimension (of the form (x, y_c)) and 10 additional points in the second dimension (of the form (x_c, y)), functions $f_x(x)$ and $f_y(y)$ were calculated with a cubic spline interpolation. Then, a standard 2D nonlinear regression using basis functions of the form $(x - x_c)^m(y - y_c)^n$, $m, n \geq 1$ (due to the low dimensionality of the treated problem the employ of separated representations is not needed) was employed for calculating the term $f_{x,y}(x, y)$ using 4 sample points.

The constructed solution is depicted in Fig. 13 where it is compared with the exact solution as well as with the solution obtained by using the standard s-PGD (with a Latin Hypercube Sampling containing 25 points), while Figs. 14 and 15 compare the predictions

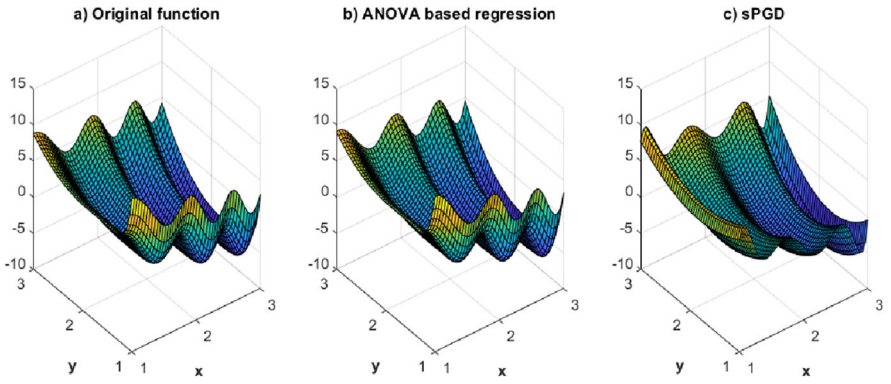

Fig. 13 Comparing *s*-PGD and ANOVA-PGD regressions

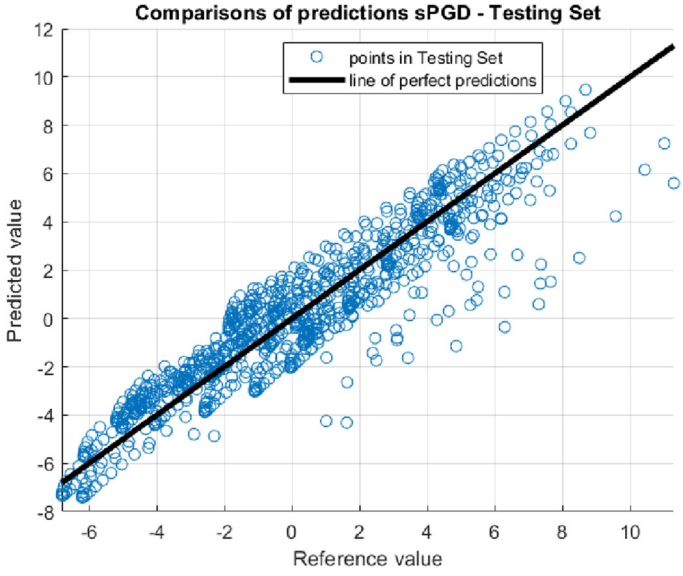

Fig. 14 Problem defined in Eq. (4.6): Comparison of predicted *s*-PGD values with the reference ones in the testing set (the black line represents a perfect prediction)

and the reference values. From all these results, excellent performances of the ANOVA-based regression can be stressed.

Conclusions

In this paper, three different data-driven regression techniques are introduced, the first two, the so-called *rs*-PGD and s^2-PGD, that consist of a regularization of the usual sparse PGD, and the third, that combines analysis of variance features with sparse separated representations. It has been shown and discussed, through different examples, how they can improve significantly the existing sparse *s*-PGD performance, reducing overfitting and achieving great explanatory predictive capabilities when dealing with unseen scenarios.

Furthermore, the s^2-PGD can be employed to sparse identification and variable selection when the *s*-PGD fails. The comparison of Figs. 6 and 7 is an example of the substantial improvements under this rationale. In what respecto to the ANOVA version just introduced, in Fig. 15 the clear improvement obtained with respecto to the s-PGD approach can be noticed at first sight.

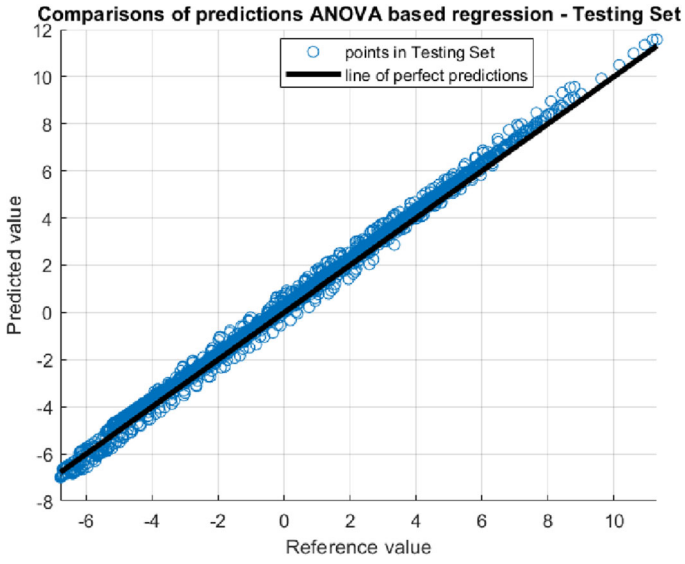

Fig. 15 Problem defined in Eq. (4.6): Comparison of predicted ANOVA-PGD values with the reference ones in the testing set (the black line represents a perfect prediction)

In addition, the suitability of the *s*-PGD to deal with the challenging scenarios concerning the low-data regime context and high-dimensional parametric functions was previously proved in [18, 20]. Therefore, the improvements carried out by these new techniques opens the door to construct better high-performance ROMs in this difficult context. Moreover, this is really appealing because of the increasing industrial interest of obtaining accurate models under these circumstances.

Our works in progress address specific industrial applications where the use of these techniques can be competitively advantageous. In addition, other penalties are being studied for its use in specific frameworks as well as different sampling strategies when they can be controlled, to maximize the ROM performance.

Author contributions

All the authors participated in the definition of techniques and algorithms.

Funding

The authors are grateful for the support of ESI Group through the ESI Chair at ENSAM Arts et Metiers Institute of Technology, and through the project 2019-0060 "Simulated Reality" at the University of Zaragoza. The support of the Spanish Ministry of Science and Innovation, AEI/10.13039/501100011033, through grant number CICYT-PID2020-113463RB-C31 and by the Regional Government of Aragon and the European Social Fund, Grant T24-20R, are also gratefully acknowledged.

Availability of data and materials

The interested reader can contact the authors to access the code.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 2 April 2022 Accepted: 31 December 2022
Published online: 09 March 2023

References

1. Ammar A, Mokdad B, Chinesta F, KEUNINGS R. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modelling of complex fluids. Part II: Transient simulation using space-time separated representations. J Non-Newtonian Fluid Mech. 2007;144(2–3):98–121.
2. Argerich C. Study and development of new acoustic technologies for nacelle products. PhD thesis, Universitat Politecnica de Catalunya; 2020.
3. Beddek K. Propagation d'incertitudes dans les modèles éléments finis en électromagnétisme : application au contrôle non destructif par courants de Foucault. PhD thesis, Ecole doctorale Sciences pour l'Ingenieur (Lille) - L2EP, 2012. Thèse de doctorat dirigée par Clénet, StéphaneLe Menach, Yvonnick et Moreau, Olivier Génie électrique. 2012.
4. Borzacchiello D, Aguado JV, Chinesta F. Non-intrusive sparse subspace learning for parametrized problems. Arch Comput Methods Eng. 2019;26(2):303–26.
5. Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends Mach Learn. 2011;3(1):1–122.
6. Brunton SL, Kutz JN. Data-driven science and engineering: machine learning, dynamical systems, and control. Cambridge: Cambridge University Press; 2019.
7. Brunton SL, Proctor JL, Nathan KJ. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc Nat Acad Sci. 2016;113(15):3932–7.
8. Chinesta F, Huerta A, Rozza G, Willcox K. Encyclopedia of Computational Mechanics, chapter Model Order Reduction. New York: Wiley; 2015.
9. Chinesta F, Cueto E, Abisset-Chavanne E, Duval JL, El Khaldi F. Virtual, digital and hybrid twins: a new paradigm in data-based engineering and engineered data. Arch Comput Methods Eng. 2020;27(1):105–34.
10. Chinesta F, Keunings R, Leygue A. The Proper Generalized Decomposition for Advanced Numerical Simulations: A Primer. Berlin: Springer Publishing Company; 2013.
11. Cueto E, Gonzalez D, Alfaro I. Proper Generalized decompositions: an introduction to computer implementation with Matlab. 1st ed. New York: Springer; 2016.
12. Forrester AIJ, Sobester A, Keane AJ. Engineering design via surrogate modelling: a practical guide. New York: Wiley; 2008.
13. Friedman JH, Hastie T, Tibshirani R. Regularization paths for generalized linear models via coordinate descent. J Stat Softw. 2010;33(1):1–22.
14. Hastie T, Tibshirani R, Friedman JH. The elements of statistical learning: data mining, inference, and prediction. New York: Springer; 2009.
15. Hernandez Q, Badias A, Gonzalez D, Chinesta F, Cueto E. Deep learning of thermodynamics-aware reduced-order models from data 2020. arXiv preprint arXiv:2007.03758.
16. Hernández Q, Badías A, González D, Chinesta F, Cueto E. Structure-preserving neural networks. J Comput Phys. 2021;426: 109950.
17. Ibanez R, Abisset-Chavanne E, Cueto E, Ammar A, Duval J-L, Chinesta F. Some applications of compressed sensing in computational mechanics: model order reduction, manifold learning, data-driven applications and nonlinear dimensionality reduction. Comput Mech. 2019;64(5):1259–71.
18. Ibanez R. Advanced physics-based and data-driven strategies. Theses: École centrale de Nantes; Universitat politécnica de Catalunya; 2019.
19. Ibanez R, Abisset-Chavanne E, Ammar A, González D, Cueto E, Huerta A, Duval JL, Chinesta Fra. A multidimensional data-driven sparse identification technique: the sparse proper generalized decomposition. Complexity, 2018.
20. Ibanez Pinillo R, Abisset-Chavanne E, Ammar A, González D, Cueto E, Huerta A, Louis Duval J, Chinesta F. A multidimensional data-driven sparse identification technique: The sparse proper generalized decomposition. Complexity. 2018;2018(1–11):11.
21. Kaarnioja V. Smolyak Quadrature. mathesis, University of Helsinki, 2013.
22. Kubicek M, Minisci E, Cisternino M. High dimensional sensitivity analysis using surrogate modeling and high dimensional model representation. Int J Uncertainty Quant. 2015;5:01.
23. Laughlin Robert B, Pines David. The theory of everything. Proc Nat Acad Sci USA. 2000;97(1):28.
24. Victor L, Xavier D, Emmanuel L, Rubén I, Clara A, Fatima D, Duval Jean L, Francisco C. Advanced model order reduction and artificial intelligence techniques empowering advanced structural mechanics simulations: application to crash test analyses. Mech Indus. 2019;20(8):804.
25. Ly HV, Tran HT. Modeling and control of physical processes using proper orthogonal decomposition. J Math Computer Model. 2001;33(1–3):223–36.
26. MathWorks. Documentation lasso function. https://uk.mathworks.com/help/stats/lasso.html#bvm6oqf, 2021. Accessed 29 Oct 2021.
27. Moya B, Alfaro I, Gonzalez D, Chinesta F, Cueto E. Physically sound, self-learning digital twins for sloshing fluids. PLoS One. 2020;15(6): e0234569.
28. Moya B, Badías A, Alfaro I, Chinesta F, Cueto E. Digital twins that learn and correct themselves. Int J Numer Methods Eng. 2020;67:89.
29. Moya B, González D, Alfaro I, Chinesta F, Cueto E. Learning slosh dynamics by means of data. Comput Mech. 2019;64(2):511–23.
30. Jiang PP, Zhou Q, Shao X. Surrogate Model-Based Engineering Design and Optimization. New York: Springer; 2020.
31. Papritz A, Stein A. Surrogate Model-Based Engineering Design and Optimization. In: Stein A, Van der Meer F, Gorte B, editors. Spatial Statistics for Remote Sensing Remote Sensing and Digital Image Processing, vol. 1. Dordrecht: Springer; 1999.
32. Sancarlos A, Cameron M, Abel A, Cueto E, Duval J-L, Chinesta F. From rom of electrochemistry to ai-based battery digital and hybrid twin. In: Archives of Computational Methods in Engineering, 2020; pp. 1–37.
33. Sancarlos A, Cueto E, Chinesta F, Duval JL. A novel sparse reduced order formulation for modeling electromagnetic forces in electric motors. SN Applied Sciences, 2021.

34. Sancarlos A, Pineda M, Puche R, Sapena A, Riera M, Martinez J, Perez J, Roger J. Application of the parametric proper generalized decomposition to the frequency-dependent calculation of the impedance of an ac line with rectangular conductors. Open Phys. 2017;15:12.
35. Shiffrin RM, Bassett DS, Kriegeskorte N, Tenenbaum JB. The brain produces mind by modeling. Proc Nat Acad Sci. 2020;117(47):29299–301.
36. Tang K, Congedo PM, Abgrall R. Sensitivity analysis using anchored anova expansion and high-order moments computation. Int J Numer Methods Eng. 2015;102:1554–84.
37. Udrescu S-M, Tan A, Feng J, Neto O, Wu T, Tegmark M. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. 2020. arXiv preprint arXiv:2006.10782.

Publisher's Note