



Automatic selection of basis-adaptive sparse polynomial chaos expansions for engineering applications

Nora Lüthen, Stefano Marelli, Bruno Sudret

► To cite this version:

Nora Lüthen, Stefano Marelli, Bruno Sudret. Automatic selection of basis-adaptive sparse polynomial chaos expansions for engineering applications. *International Journal for Uncertainty Quantification*, 2022, 10.1615/Int.J.UncertaintyQuantification.2021036153 . hal-02936436v2

HAL Id: hal-02936436

<https://hal.science/hal-02936436v2>

Submitted on 12 Aug 2021

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

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

AUTOMATIC SELECTION OF BASIS-ADAPTIVE SPARSE POLYNOMIAL CHAOS EXPANSIONS FOR ENGINEERING APPLICATIONS

N. Lüthen, S. Marelli and B. Sudret



Data Sheet

Journal:	International Journal for Uncertainty Quantification
Report Ref.:	RSUQ-2020-011C
Arxiv Ref.:	https://arxiv.org/abs/2009.04800 [cs.NA, stat.CO, stat.ML]
DOI:	-
Date submitted:	July 23, 2021
Date accepted:	-

Automatic selection of basis-adaptive sparse polynomial chaos expansions for engineering applications

Nora Lüthen¹, Stefano Marelli¹, and Bruno Sudret¹

¹*Chair of Risk, Safety and Uncertainty Quantification, ETH Zürich, Stefano-Franscini-Platz 5, 8093 Zürich, Switzerland*

July 23, 2021

Abstract

Sparse polynomial chaos expansions (PCE) are an efficient and widely used surrogate modeling method in uncertainty quantification for engineering problems with computationally expensive models. To make use of the available information in the most efficient way, several approaches for so-called basis-adaptive sparse PCE have been proposed to determine the set of polynomial regressors (“basis”) for PCE adaptively.

The goal of this paper is to help practitioners identify the most suitable methods for constructing a surrogate PCE for their model. We describe three state-of-the-art basis-adaptive approaches from the recent sparse PCE literature and conduct an extensive benchmark in terms of global approximation accuracy on a large set of computational models. Investigating the synergies between sparse regression solvers and basis adaptivity schemes, we find that the choice of the proper solver and basis-adaptive scheme is very important, as it can result in more than one order of magnitude difference in performance. No single method significantly outperforms the others, but dividing the analysis into classes (regarding input dimension and experimental design size), we are able to identify specific sparse solver and basis adaptivity combinations for each class that show comparatively good performance.

To further improve on these findings, we introduce a novel solver and basis adaptivity selection scheme guided by cross-validation error. We demonstrate that this automatic selection procedure provides close-to-optimal results in terms of accuracy, and significantly more robust solutions, while being more general than the case-by-case recommendations obtained by the benchmark.

1 Introduction

Surrogate modeling techniques are a popular tool in applied sciences and engineering, because they can significantly reduce the computational cost of uncertainty quantification analysis for costly real-world computational models. Here, the computational model is approximated by a cheaper-to-evaluate function, which is created based on a small number of model evaluations, the so-called experimental design. One well-known and popular surrogate modeling technique is

polynomial chaos expansion (PCE), which approximates the output of a computational model by a spectral expansion in terms of an orthonormal polynomial basis in the input random variables (Xiu and Karniadakis, 2002). PCE is particularly well suited for surrogating smooth models in low to medium dimension, and for the efficient computation of moments and Sobol’ indices (Sudret, 2008; Le Gratiet et al., 2017). Engineering models are often challenging due to their computational cost: complex models often depend on a large number of input parameters, but we can only afford a few tens or hundreds of model evaluations. This is the so-called low-data regime. Sparse PCE techniques, which aim to compute an expansion involving only few terms, have proven especially powerful and cost-efficient for real-world engineering problems such as, among many others, surrogate-assisted robust design optimization (Chatterjee et al., 2019), hybrid simulation for earthquake engineering (Abbiati et al., 2021), dam engineering (Guo et al., 2019; Hariri-Ardebili and Sudret, 2020), and wind turbine design (Slot et al., 2020). Note that real-world applications are typically not exactly sparse; however, sparse regression-based PCE is a useful tool to find good approximations at low computational cost. The aim in this context is not sparsity, but the accurate approximation of the computational model. In particular, we are interested in global approximation accuracy, as measured by the relative mean-squared error. Applications with specific requirements on accuracy, such as optimization or reliability analysis, might rely instead on more specialized surrogate techniques.

In the last decade, a large amount of literature on sparse PCE has been published, proposing methods that make sparse PCE more accurate, efficient and applicable to high-dimensional problems. However, it is often not obvious how well these methods perform when compared to and combined with each other, especially on real-world engineering problems. In an attempt to fill this gap, the authors recently conducted a literature survey on sparse PCE and a classification of the available methods into a general framework, as well as a benchmark of selected methods on a broad class of computational models (Lüthen et al., 2021). This benchmark extensively compared sparse regression solvers and experimental design sampling schemes, using a fixed polynomial basis to focus on the effect of those two classes of methods. We found that the choice of sparse solver and sampling scheme can make a difference of up to several orders of magnitude in the resulting approximation error, and that different methods are preferable in different regimes of ED size. The performance of solvers and sampling schemes seemed to be mostly independent from one another.

The goal of the present paper is to build on and complement this earlier benchmark by exploring the promising field of *basis-adaptive sparse PCE*, in which the basis is iteratively augmented and refined. So far, novel basis-adaptive methods have been proposed in isolation and not been compared to one another. We want to help practitioners choose the most suitable methods for constructing a PCE surrogate for their applications by answering the following questions: (1) Is there a significant difference between different combinations of sparse solvers and basis adaptivity strategies, and does the proper choice matter in actual applications? (2) In the case that no combination clearly emerges as superior, is there a smart strategy to automatically select a good combination? To answer the first question, we describe three basis-adaptive schemes from the sparse PCE literature in detail, namely, degree and q-norm adaptivity (Blatman and

(Sudret, 2011; Marelli et al., 2021), forward neighbor degree adaptivity (Jakeman et al., 2015), and anisotropic degree basis adaptivity (Hampton and Doostan, 2018). We then evaluate the performance and synergies of combinations of these basis-adaptive schemes with several sparse regression solvers in terms of validation error¹ on a library of 11 computational models of varying complexity, representative of a broad class of engineering models. These range from three- to 100-dimensional and include both analytical and numerical models. Since no combination of solver and basis adaptivity significantly outperforms the others, we address the second question by introducing an additional selection step, choosing one among several candidate PCEs computed by different combinations of methods on the same experimental design, using a cross-validation estimate for the generalization error.

The paper is structured as follows. In Section 2, we recall the definition of (sparse) PCE and the computing framework introduced in Lüthen et al. (2021). We discuss various estimators for the generalization error, sparse regression solvers, and basis adaptivity, and introduce automatic selection. The associated benchmark results for basis adaptivity and automatic selection are presented in Section 3. Finally, a discussion and summary of our results is provided in Section 4. Additional information and results can be found in the Appendix.

2 Sparse polynomial chaos expansions

Let \mathbf{X} be a d -dimensional random vector with mutually independent components and probability density function $f_{\mathbf{X}}$. Denote by $\mathcal{D}_{\mathbf{X}}$ the domain of the random vector \mathbf{X} . Define the space $L^2_{f_{\mathbf{X}}}(\mathcal{D}_{\mathbf{X}}) = \{h : \mathcal{D}_{\mathbf{X}} \rightarrow \mathbb{R} \mid \text{Var}_{\mathbf{X}}[h(\mathbf{X})] < +\infty\}$ of all scalar valued models with finite variance under $f_{\mathbf{X}}$. Under very general conditions on the input distribution $f_{\mathbf{X}}$, there exists a polynomial basis $\{\psi_{\alpha} : \alpha \in \mathbb{N}^d\}$ of $L^2_{f_{\mathbf{X}}}(\mathcal{D}_{\mathbf{X}})$ (Xiu and Karniadakis, 2002; Ernst et al., 2012). Since the components of \mathbf{X} are mutually independent, each polynomial basis function can be built as a product of univariate polynomials in X_1, \dots, X_d and characterized by the multi-index $\alpha \in \mathbb{N}^d$ whose entries are equal to the degrees of the univariate terms.

For a computational model $\mathcal{M} \in L^2_{f_{\mathbf{X}}}(\mathcal{D}_{\mathbf{X}})$, let $Y = \mathcal{M}(\mathbf{X})$ denote the output random variable. Y can be cast as the following spectral expansion:

$$Y = \mathcal{M}(\mathbf{X}) = \sum_{\alpha \in \mathbb{N}^d} c_{\alpha} \psi_{\alpha}(\mathbf{X}). \quad (1)$$

In practice, a finite, *truncated polynomial chaos expansion*

$$Y = \mathcal{M}(\mathbf{X}) \approx \mathcal{M}^{\text{PCE}}(\mathbf{X}) = \sum_{\alpha \in \mathcal{A}} c_{\alpha} \psi_{\alpha}(\mathbf{X}) \quad (2)$$

¹We focus on global approximation accuracy instead of the computational cost of training, since the main cost in surrogate modeling for engineering applications is typically incurred by the model evaluations themselves. Indeed, the training of each of the surrogates presented in this work requires between a few seconds and a few minutes on a standard business laptop. Even orders of magnitude difference in the training costs of each surrogate do not matter as long as the resulting accuracy is appreciably better.

is computed, where $\mathcal{A} \subset \mathbb{N}^d$ is the truncation set defining the basis elements used in the expansion. The accuracy of the expansion depends on \mathcal{A} and the coefficient values $(c_\alpha)_{\alpha \in \mathcal{A}} =: \mathbf{c} \in \mathbb{R}^P$, with $P = \text{card}(\mathcal{A})$.

Among several methods for computing the coefficient vector \mathbf{c} for a given truncation set, *sparse regression* is a particularly powerful and efficient method (Doostan and Owhadi, 2011; Blatman and Sudret, 2011). In this approach, the model is evaluated at a number of points $\mathcal{X} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\} \subset \mathcal{D}_X$ called the *experimental design* (ED), yielding the vector of model responses $\mathbf{y} = (\mathcal{M}(\mathbf{x}^{(1)}), \dots, \mathcal{M}(\mathbf{x}^{(N)}))^T$. Let $\{\alpha_j\}_{j=1}^P$ be an arbitrary ordering of the multi-indices in the truncation set and define the regression matrix $\Psi \in \mathbb{R}^{N \times P}$ with entries $\Psi_{ij} = \psi_{\alpha_j}(\mathbf{x}^{(i)})$. Sparse regression methods determine a vector \mathbf{c} that minimizes the residual norm $\|\Psi\mathbf{c} - \mathbf{y}\|_2$ under the constraint that it has only few nonzero entries i.e., it is *sparse*. This is usually achieved by regularized regression, resulting e.g. in the LASSO formulation

$$\hat{\mathbf{c}} = \min_{\mathbf{c} \in \mathbb{R}^P} \|\Psi\mathbf{c} - \mathbf{y}\|_2^2 \quad \text{s.t.} \quad \|\mathbf{c}\|_1 \leq \tau, \quad (3)$$

where τ is a parameter regulating the sparsity of \mathbf{c} . A PCE with a sparse coefficient vector \mathbf{c} is called *sparse PCE*. Provided that the regression matrix fulfills certain properties (Candès and Wakin, 2008; Bruckstein et al., 2009; Candès and Plan, 2011), sparse regression can find robust solutions to underdetermined systems of linear equations, which means that the experimental design can be smaller than the number of unknown coefficients.

The quality of the solution depends on the choice of the basis \mathcal{A} , on the experimental design \mathcal{X} , and on the method used for computing the coefficients. For each of these components, many different methods have been proposed in recent years, including iterative methods which adaptively determine \mathcal{A} , or the experimental design \mathcal{X} . These methods were recently surveyed and classified into the framework shown in Fig. 1 (modified from Lüthen et al. (2021)). In the present contribution, we focus on the question of how to determine a suitable basis \mathcal{A} . To this end, we benchmark several basis-adaptive approaches and explore their interplay with sparse regression solvers.

2.1 Error estimation and model selection

Model selection is applied on several levels in the sparse PCE framework:

- Within the sparse solver to select its hyperparameter (see Section 2.2)
- Within the basis adaptivity scheme to select a basis (see Section 2.3)
- Finally, between solvers and basis adaptivity schemes, to automatically select a combination that is close to best (see Section 2.4).

Our main quantity of interest is the generalization error, in the form of the relative mean squared error normalized by the model variance

$$E_{\text{gen}} = \frac{\mathbb{E}_{\mathbf{X}} \left[(\mathcal{M}(\mathbf{X}) - \mathcal{M}^{\text{PCE}}(\mathbf{X}))^2 \right]}{\text{Var}_{\mathbf{X}} [\mathcal{M}(\mathbf{X})]}. \quad (4)$$

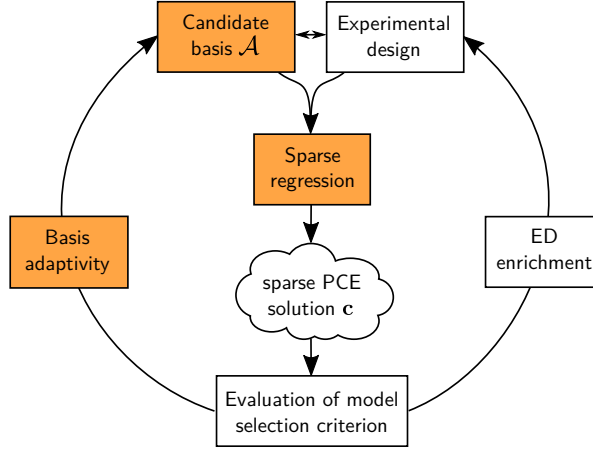


Figure 1: Framework for computing sparse PCE introduced by Lüthen et al. (2021), who conducted a literature survey and a benchmark for the central components “Experimental design” and “Sparse regression” (adapted from Lüthen et al. (2021)). In the present work, we discuss and benchmark the components marked in orange. In particular, we discuss the component “Basis adaptivity” and explore its relationship to sparse regression solvers.

It can be approximated by the validation error in the form of the relative mean squared error (MSE)

$$\text{RelMSE} = \frac{\sum_{i=1}^{N_{\text{val}}} \left(\mathcal{M}(\mathbf{x}_{\text{val}}^{(i)}) - \mathcal{M}^{\text{PCE}}(\mathbf{x}_{\text{val}}^{(i)}) \right)^2}{\sum_{i=1}^{N_{\text{val}}} \left(\mathcal{M}(\mathbf{x}_{\text{val}}^{(i)}) - \frac{1}{N_{\text{val}}} \sum_{j=1}^{N_{\text{val}}} \mathcal{M}(\mathbf{x}_{\text{val}}^{(j)}) \right)^2} \quad (5)$$

computed on a large validation set $\{\mathbf{x}_{\text{val}}^{(i)}\}_{i=1}^{N_{\text{val}}} \sim_{\text{i.i.d.}} f_{\mathbf{X}}$. We only consider model selection criteria that are estimates of the generalization error.

To get an accurate estimate of the generalization error, without using an additional validation set, a widely used method is *cross-validation* (Vapnik, 1995). Here, the available data is repeatedly divided into two disjoint sets, one for computing the solution (training) and the other for evaluating the error (validation). Aggregating the error estimates from the different splits, we get a cross-validation estimate of the generalization error.

One cross-validation strategy is to divide the data randomly into k disjoint, roughly equally large parts, and use each of the parts in turn as validation set. This is called *k-fold cross-validation*. If k is chosen to be equal to the size of the experimental design, the strategy is known as *leave-one-out cross-validation* (LOO). This is closest to using all data points to compute a solution, since in each iteration, only one point is left out from the training set.

In general, LOO can be quite costly, since for an experimental design of size N , the method under consideration has to be applied N times. A cheap approximation to the LOO error, which requires only one application of the method, is available for certain sparse regression solvers, namely for those which in their last step recompute the solution with ordinary least-squares (OLS) on the set of regressors with nonzero coefficient (called *active basis*) (Blatman and Sudret, 2010a). In particular, this is the case for the solvers hybrid least angle regression (LARS) (Blatman and Sudret, 2011), orthogonal matching pursuit (OMP) (Pati et al., 1993;

Jakeman et al., 2015), and subspace pursuit (SP) (Diaz et al., 2018) (see Section 2.2). For these solvers, the OLS-based LOO estimate coincides with the true LOO error if the following holds: regardless of which experimental design point is left out, the sparse regression solver consistently yields the same active basis.

Since the repeated use of LOO for model selection often results in the generalization error being underestimated, especially on small experimental designs, Blatman and Sudret (2011) proposed to use a modification factor originally developed for the empirical error (Chapelle et al., 2002), defined by

$$T = \frac{N}{N - P_{\text{active}}} \left(1 + \text{tr}((\Psi_{\text{active}}^T \Psi_{\text{active}})^{-1}) \right), \quad (6)$$

where P_{active} denotes the number of nonzero coefficients (the corresponding basis functions are called *active*), and Ψ_{active} denotes the regression matrix containing only the active regressors. The product of the modification factor T with the LOO error is called *modified LOO error*.

2.2 Sparse regression solvers

Various sparse regression solvers available for solving sparse PCE were described in detail in Lüthen et al. (2021). We give here only a short overview of the solvers used in our benchmark, and refer to Lüthen et al. (2021) for further details. These solvers are common choices in the sparse PCE literature.

- Hybrid least angle regression (LARS) (Blatman and Sudret, 2011): adding regressors one-by-one, following a least-angle strategy. The hybrid approach recomputes the final coefficient values by ordinary least squares (OLS) on the selected regressors.
- Orthogonal matching pursuit (OMP) (Pati et al., 1993; Jakeman et al., 2015): greedily adding orthogonal regressors one-by-one, computing their coefficients by OLS.
- Subspace pursuit (SP) (Diaz et al., 2018): searching iteratively for a solution with a certain ℓ^0 -norm by adding and removing regressors from the active set. Coefficients are computed by OLS. We include two variants of SP in this benchmark, one which determines its hyperparameter by 5-fold cross-validation similar to the implementation in (Diaz, 2018), which we denote by $\text{SP}_{k=5}$, and one where it is determined by OLS-based LOO, introduced in (Lüthen et al., 2021) and denoted by SP_{LOO} .
- Bayesian compressive sensing (BCS) (Babacan et al., 2010): using a Bayesian framework to enforce sparsity of the coefficients.
- SPGL1 (van den Berg and Friedlander, 2008): a convex optimization solver following the Pareto front of the residual-sparsity trade-off.

Each of the solvers features at least one hyperparameter whose value needs to be determined via cross-validation in order to get a good solution. For LARS, OMP, $\text{SP}_{k=5}$, and SP_{LOO} , this hyperparameter is the number of active basis functions (nonzero coefficients) of the final

solution. For BCS and SPGL1, it is the bound on the residual norm in the sparse regression formulation.

The benchmark in Lütthen et al. (2021) of sparse regression solvers on a non-adaptive polynomial basis showed that BCS and SP_{LOO} generally outperform other sparse solvers for low-dimensional models, while for high-dimensional models, BCS is by far the best sparse solver.

2.3 Basis adaptivity

The sparse solver and the experimental design, which were benchmarked in (Lütthen et al., 2021), are not the only ingredients to a sparse PCE. The choice of the *candidate basis*, from which the sparse solver determines the *active basis* (i.e., the set of regressors with nonzero coefficient), is another important ingredient: if the candidate basis is chosen too small, important terms might be missing, which might lead to a large model error. On the other hand, if the candidate basis is too large, the ratio of the number of experimental design points to the number of coefficients is small, which causes some properties of the regression matrix to deteriorate and can result in a large approximation error.

Of course, it is not possible to know a-priori the best choice of the candidate basis. Basis-adaptive schemes start with an initial candidate basis and adapt it iteratively, adding or removing basis functions in each iteration according to various heuristics. The performance of the bases in the different iterations is evaluated using a model selection criterion, typically an estimate of the generalization error.

Several procedures for basis-adaptive sparse PCE have been proposed in the literature. We discuss three approaches in detail, namely

- degree and q-norm (“p&q”) basis adaptivity, as implemented in UQLab (Marelli et al., 2021), see Section 2.3.1;
- forward neighbor basis adaptivity (Jakeman et al., 2015), see Section 2.3.2; and
- anisotropic degree basis adaptivity (Hampton and Doostan, 2018), from the software BASE_PC (Hampton and Doostan, 2017), see Section 2.3.3.

We also briefly mention several other approaches found in the literature.

2.3.1 Degree and q-norm (“p&q”) adaptivity

A typical choice for a PCE basis is the *basis of total degree p* defined by the set of multi-indices

$$\mathcal{A}^p = \{\boldsymbol{\alpha} \in \mathbb{N}^d : \sum_{i=1}^d \alpha_i \leq p\}. \quad (7)$$

Furthermore, *hyperbolic truncation* (Blatman and Sudret, 2011) uses the q-(quasi-)norm

$$\|\mathbf{x}\|_q = \left(\sum_{i=1}^d |x_i|^q \right)^{\frac{1}{q}} \quad (8)$$

with $q \in (0, 1]$ to truncate a total-degree basis further:

$$\mathcal{A}^{p,q} = \{\boldsymbol{\alpha} \in \mathbb{N}^d : \|\boldsymbol{\alpha}\|_q \leq p\}. \quad (9)$$

Hyperbolic truncation has the effect of excluding some of the basis functions with high degree and high interaction order. This is particularly effective for high-dimensional problems.

A simple basis-adaptive scheme is *degree adaptivity* (Blatman and Sudret, 2011), which computes a number of PCEs on a sequence of total-degree candidate bases of increasing size, and returns the PCE minimizing a certain error estimate as final solution. Analogously, a q-norm adaptive scheme can be developed, and easily be combined with degree adaptivity (Marelli et al., 2021), yielding *degree and q-norm ($p\&q$) adaptivity*. Degree and q-norm adaptivity is solution-agnostic, i.e., it does not take the solution computed in the previous iteration into account.

2.3.2 Forward neighbor basis adaptivity

Jakeman et al. (2015) suggest a basis-adaptive algorithm based on a graph view of the PCE regressors (see also Gerstner and Griebel (2003); Narayan and Jakeman (2014)). Since the input random vector is assumed to have independent components, the basis functions have tensor product structure. The basis functions can be considered nodes of a directed graph constructed as follows (see also Fig. 2): two regressors are considered neighbors if their multi-index of degrees differs only in one dimension by 1, i.e., there is a directed edge from $\psi_{\boldsymbol{\alpha}}$ to $\psi_{\boldsymbol{\beta}}$ iff $\boldsymbol{\gamma} := \boldsymbol{\beta} - \boldsymbol{\alpha}$ is a multi-index with $\gamma_i = 1$ for one $i \in \{1, \dots, d\}$ and $\gamma_j = 0, j \neq i$. *Forward neighbors* of a regressor are regressors reachable by an outgoing edge, and *backwards neighbors* are regressors connected by an incoming edge.

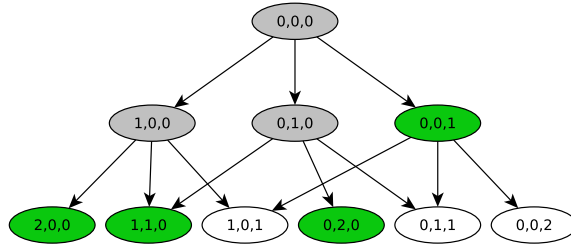


Figure 2: Illustration of forward neighbor basis adaptivity (Jakeman et al., 2015) in $d = 3$ dimensions. The regressors are seen as nodes in a graph, where a directed edge connects regressors whose multi-index differs by 1 in exactly one dimension. Assume that the set of active regressors $\mathcal{A}^{\text{active}}$ initially consists of the gray nodes. All admissible nodes are highlighted in green. Together with the gray nodes, they constitute the set $\mathcal{A}^{(1)}$ and would be added to the model in the next expansion step of forward neighbor basis adaptivity (Step 3 in Algorithm 1).

In the context of basis-adaptive PCE, this construction is used to determine a number of candidate regressors to be added to the currently active basis, starting from an initial basis of small total degree. Assume that an important high-degree or high-order regressor is not yet part of the candidate basis but some of its backwards neighbors are. The fact that it is missing should be

visible in the coefficients of its backwards neighbors, which can be expected to have a significant nonzero coefficient to compensate for the fact that the important high-degree regressor is not yet part of the basis.

This heuristic is the foundation of the algorithm whose pseudocode is summarized in Algorithm 1. In each iteration, the current set of active basis functions is determined (restriction step). All forward neighbors of these active basis functions are surveyed and added to the candidate basis if they are *admissible*, i.e., if all of their backwards neighbors are in the active set (expansion step). Jakeman et al. (2015) employ several expansion steps to prevent premature termination, and use cross-validation error estimates. We call this algorithm *forward neighbor basis-adaptive*.

Algorithm 1 Forward neighbor basis adaptivity (Jakeman et al., 2015)

- 1: Initial PCE (basis chosen a-priori, typically small total-degree basis)
 - 2: Restriction: retain only the active regressors $\mathcal{A}^{\text{active}}$
 - 3: Expansion: let $\mathcal{A}^{(0)} = \mathcal{A}^{\text{active}}$. For $t = 1, \dots, T$, obtain the set $\mathcal{A}^{(t)}$ by augmenting $\mathcal{A}^{(t-1)}$ by all its admissible forward neighbors.
 - 4: Compute a PCE and its error estimate for each augmented basis $\mathcal{A}^{(1)}, \dots, \mathcal{A}^{(T)}$.
 - 5: Choose the PCE with the lowest error estimate among the T candidates. Stop if this error estimate is larger than the previously obtained best error estimate. Else, continue iteration with restriction step (Step 2)
-

This algorithm is implemented in the software Dakota (Adams et al., 2014). We use our own Matlab-based implementation of the algorithm. Consistently with Jakeman et al. (2015), we set $T = 3$.

2.3.3 Anisotropic degree basis adaptivity

Hampton and Doostan (2018) propose an algorithm called BASE-PC, which combines a basis-adaptive scheme with sequential enrichment of a coherence-optimal experimental design. The two steps are executed alternatingly: based on the current ED, a suitable basis is chosen; and according to the currently active basis functions, the ED is enriched and the weights are updated.

In the present work, we solely consider the basis adaptivity part of the BASE-PC algorithm. The main idea of this algorithm is to use an *anisotropic degree basis*, defined by a *degree vector* $\mathbf{p} \in \mathbb{N}^d$. A related idea was explored earlier by Blatman and Sudret (2009). Similarly to a total-order basis, an anisotropic degree basis is defined by

$$\mathcal{A}^{\mathbf{p}} = \{\boldsymbol{\alpha} \in \mathbb{N}^d : \sum_{i=1}^d \frac{\alpha_i}{p_i} \leq 1\}. \quad (10)$$

If all entries of \mathbf{p} are the same, i.e., $p_1 = p_2 = \dots = p_d = p$, this definition reduces to a total-order basis of degree p . The equation $\sum_{i=1}^d \frac{\alpha_i}{p_i} = 1$ defines a hyperplane that cuts the i -th coordinate axis at $\alpha_i = p_i$.

In each iteration, the algorithm determines the current anisotropic degree vector based on the currently active basis. A new, larger candidate basis is then constructed by considering the anisotropic degree basis corresponding to a uniformly increased anisotropic degree vector.

We use our own, slightly modified implementation of the basis adaptivity part of BASE-PC, as summarized in Algorithm 2. The most costly operation is the computation of the anisotropic-degree basis (line 6). Hampton and Doostan (2018) developed a specialized efficient algorithm to generate the multi-indices of an anisotropic degree basis, which we utilize in our implementation. We call Algorithm 2 *anisotropic degree basis-adaptive*.

Algorithm 2 Anisotropic degree basis adaptivity (Hampton and Doostan, 2018)

```

1: Initial PCE (fixed basis of low order)
2: for  $o = 1, \dots, 10$  do ▷ outer loop
3:   Restriction: denote by  $\mathcal{A}^{\text{active}}$  the set of active regressors of the last selected PCE
4:   for  $i = 1, \dots, 10$  do ▷ inner loop
5:     Additional restriction: remove  $\frac{i-1}{10}$  regressors from the set  $\mathcal{A}^{\text{active}}$  starting with the
       ones with smallest-in-magnitude coefficients, obtaining a set  $\mathcal{A}^i \subseteq \mathcal{A}^{\text{active}}$ 
6:     Expansion: compute the dimension-wise maximal degree of the regressors in  $\mathcal{A}^i$ ,
       denoted by degree vector  $\mathbf{p}^{\text{max}}$ . Compute  $\mathbf{p}^{\text{new}} = (p_1^{\text{max}} + 1, p_2^{\text{max}} + 1, \dots, p_d^{\text{max}} + 1)$  and the
       associated anisotropic-degree basis  $\mathcal{A}^{i,\text{new}}$ 
7:     Compute a PCE on the basis  $\mathcal{A}^{i,\text{new}}$  and its error estimator  $e^i$ 
8:     If  $e^i \geq e^{i-1}$ , increase the so-called strike counter by 1. Break from the inner loop if
       the strike counter is  $\geq 3$ 
9:   end for
10:  From the 10 candidate PCEs, select the PCE with the lowest error estimate
11:  Break from the outer loop if the error estimate has increased three times in subsequent
     iterations (ExpansionEarlyStop; same idea as inner loop strike counter)
12: end for
13: Return the PCE with the lowest error estimate among all PCEs selected in the outer loop.
```

2.3.4 Other basis adaptivity algorithms

We briefly summarize other approaches for basis adaptivity for sparse PCE. These approaches will not be investigated in our benchmark.

Many algorithms which use stepwise regression to build up a sparse basis regressor-by-regressor can be classified both as sparse regression solvers (as done in Lüthen et al. (2021)) and as basis-adaptive algorithms. As an example, the approach by Blatman and Sudret (2010a) adds and removes regressors one-by-one based on the induced change in LOO error, thereby building up a small set of active basis functions which is sparse in a larger total-degree basis. In this case, the candidate basis coincides with the active basis. Mai and Sudret (2015) use the “principle of heredity” together with LARS to determine additional bivariate interaction terms to be added to the model, once a univariate term is identified as relevant. We do not consider these approaches,

since we are interested in algorithms modifying the candidate basis not only one regressor at a time, but at a larger scale.

Alemazkoor and Meidani (2017) propose a basis-adaptive algorithm relying on sparsity and step-by-step augmentation of the basis. In each step, the candidate basis is a total-order basis for a subset of input random variables, while the remaining input random variables are considered constant. The initial basis consists only of the constant term. In each iteration, either one of the constant dimensions is activated, or the total degree of the candidate basis is increased by one (active dimensions only), depending on the resulting error estimate or sparsity of the solution. The coefficients are solved by OLS until a pre-defined threshold of residual norm is reached. Then, the sparse regression solver SPGL1 is used, which identifies the sparsest solution whose residual norm is smaller than the given threshold. We do not consider this method due to its high computational cost and its strong tie to the solver SPGL1, making it less effective when paired with other sparse regression solvers.

Loukrezis and De Gersem (2019) propose a basis-adaptive algorithm for interpolating PCE on Leja-based sparse grids. Their algorithm is a more cautious version of forward neighbor basis adaptivity (Section 2.3.2): after adding all admissible forward neighbors to the basis and computing their coefficients, all of the recently added regressors are removed again, except for the one that achieved the largest-in-magnitude coefficient. Thapa et al. (2020) suggest a basis-adaptive procedure that relies on total-order bases of increasing degree. In contrast with degree adaptivity (Section 2.3.1), the basis functions of degree $p + 1$ are not added all at once, but in chunks of a certain size dependent on the dimension and the degree p . After adding a chunk of basis functions, the PCE is recomputed and regressors with a coefficient smaller than a certain threshold are removed from the basis. We do not consider these two approaches because they are similar to the previously presented approaches while being more costly.

2.4 Automatic selection of a sparse PCE solution from several candidate solutions

For realistic simulators used in engineering and applied sciences, evaluating the computational model is the expensive part of the surrogate modeling process. Once the model evaluations are obtained, all further computations are post-processing steps, and are computationally cheap in comparison to the model evaluations. Thus, it is feasible to apply several adaptive sparse PCE methods and choose the best one.

We therefore propose the use of an additional layer of selection which we call here *automatic selection*. It can be seen as a simple case of *ensemble modeling* (Sagi and Rokach, 2018) where a single metamodel (“strong learner”) is chosen based on its cross-validation performance – or, in other words, model selection. For a given experimental design, we compute several sparse PCEs through different combinations of sparse solvers and basis adaptivity schemes. From the resulting PCE solutions, we choose one based on the value of a suitable estimate of generalization error as model selection criterion. There are several possibilities. One possible choice is the estimator already used for selecting the hyperparameter of the solver and for basis adaptivity,

i.e., (modified) LOO for LARS, OMP and SP_{LOO} , and k -fold cross-validation for $\text{SP}_{k=5}$, BCS and SPGL1. However, this estimator might not be consistent between solvers, which is however necessary for such an automatic selection. Furthermore, this estimate might be biased due to its repeated use.

A second class of estimators is given by the so-called *hybrid cross-validation error estimators*. The word *hybrid* is a reference to Efron et al. (2004), who created the hybrid version of least-angle regression (LARS) which uses LARS only to identify the set of active basis functions, and then recomputes the coefficients by ordinary least-squares (OLS) on this active set. To compute the hybrid leave-one-out (LOO) or hybrid k -fold cross-validation error estimate for any PCE solution, the same procedure is used: first, the active basis functions are identified using the whole experimental design. Then, the coefficients are recomputed several times using OLS, following the chosen cross-validation framework. This requires solving a linear system of equations k times in the case of k -fold cross-validation. In case of LOO, only one linear system of equations needs to be solved (Blatman and Sudret, 2010a). Furthermore, we can make use of the modification factor of Eq. (6) to compute the hybrid modified LOO error estimate.

As baseline cases, we will also select a solution 1) randomly from a set of generally well-performing methods, and 2) corresponding to the best combination identified in the benchmark of basis adaptivity schemes (Section 3.1).

3 Numerical results

In this benchmark, we compare the performance of various combinations of sparse regression solvers with basis-adaptive schemes. We use the following methods and associated implementations. The sparse solvers (wrapped to fit into our benchmark framework) are:

- Hybrid-LARS: UQLab (Marelli et al., 2021; Marelli and Sudret, 2014)
- OMP: UQLab (Marelli et al., 2021; Marelli and Sudret, 2014)
- $\text{SP}_{k=5}$: own implementation of Diaz et al. (2018), available in UQLab (Marelli et al., 2021; Marelli and Sudret, 2014)
- SP_{LOO} : own adaptation of Diaz et al. (2018), available in UQLab (Marelli et al., 2021; Marelli and Sudret, 2014)
- BCS: own implementation of FastLaplace (Babacan et al., 2010), available in UQLab (Marelli et al., 2021; Marelli and Sudret, 2014)
- SPGL1: SPGL1 v1.9² (van den Berg and Friedlander, 2008; Van den Berg and Friedlander, 2015)

²Our benchmark is performed with SPGL1 v1.9. A new version of SPGL1, v2.1, is available since June 2020. In our tests, the new version (with “hybrid” mode) did not perform significantly better than the old version. The numerical results show therefore results for SPGL1 v1.9 with default parameters.

The basis adaptivity schemes are:

- p&q adaptivity: UQLab (Marelli et al., 2021; Marelli and Sudret, 2014)
- forward neighbor basis adaptivity: own implementation of the algorithm based on the description in Jakeman et al. (2015)
- anisotropic degree basis adaptivity: own implementation of an algorithm adapted from the basis-adaptive part of BASE-PC (v1) (Hampton and Doostan, 2018), using the function `basis_anisotropic_total_order.m` from Hampton and Doostan (2017) to generate the multi-indices of an anisotropic degree basis (see Section 2.3.3)

To reduce the complexity of our benchmark, we choose Latin hypercube sampling with maximin distance optimization to generate the experimental design (ED) since Lüthen et al. (2021) demonstrated that the choices of sparse solver and sampling scheme are mostly independent from each other.

We use the following model selection methods:

- For the selection of the hyperparameters of the sparse regression solvers, we use
 - modified OLS-based LOO for the solvers LARS, OMP and SP_{LOO}
 - k -fold CV for the solvers $SP_{k=5}$ ($k = 5$), BCS ($k = 10$) and SPGL1 ($k = 10$)
- The basis adaptivity schemes use the same criterion as the respective solver uses.
- We investigate in Section 3.2 which criterion is suited best for the final model selection.

I.e., instead of prescribing fixed values for the hyperparameters, we let the parameters be determined adaptively. In this sense, we give each method equal opportunity to produce the best possible solution (assuming that the CV error is a good proxy for solution quality).

For our benchmark, we use 11 benchmark models ranging from low-dimensional analytical models to high-dimensional differential equations. All of these models have previously been used as numerical examples for surrogate modeling or reliability analysis. None of these models has an exactly sparse representation. Note that we do not aim at benchmarking the ability of the methods to recover “true” sparse solutions, but instead their approximation capabilities on engineering models, which are typically not exactly sparse but *compressible*, meaning that the magnitude of their PC coefficients decays rapidly. Table 1 provides an overview of the benchmark models. For a more detailed presentation, we refer the interested reader to the respective publications (see column "Reference" of Table 1).

³<https://www.uqlab.com/sensitivity-high-dimension>

Table 1: Overview of the 11 computational models used in our benchmark. Finite element models are marked in italic font, all other models are analytical. The column “Reference” provides the literature in which the models and their input are described in detail. The column “ED sizes” contains the two sizes of experimental design (small and large) used in the basis adaptivity benchmark.

Model	Dimension	Distributions	Reference	ED sizes (small, large)
Ishigami function	3	uniform	Blatman and Sudret (2011)	50, 150
Undamped oscillator	6	Gaussian	Echard et al. (2013)	60, 120
Borehole function	8	Gaussian, lognormal, uniform	Morris et al. (1993)	100, 250
Damped oscillator	8	lognormal	Dubourg (2011)	150, 350
Wingweight function	10	uniform	Forrester et al. (2008)	100, 250
<i>Truss model</i>	10	lognormal, Gumbel	Blatman and Sudret (2011)	100, 250
Morris function	20	uniform	Blatman and Sudret (2010b)	150, 350
<i>Structural frame model</i>	21	lognormal, Gaussian; dependent input variables	Blatman and Sudret (2010a)	150, 350
<i>2-dim diffusion model</i>	53	Gaussian	Konakli and Sudret (2016)	100, 400
<i>1-dim diffusion model</i>	62	Gaussian	Fajraoui et al. (2017)	100, 400
100D function	100	uniform	UQLab example ³	400, 1200

3.1 Basis adaptivity

We benchmark the sparse solvers LARS, OMP, $\text{SP}_{k=5}$, SP_{LOO} , BCS, and SPGL1 combined with the basis-adaptive schemes described in Section 2.3:

1. degree and q-norm (p&q) adaptivity (abbreviation: PQ)
2. forward neighbor basis adaptivity (FN)
3. anisotropic degree basis adaptivity (AD)

As a base case, we include a *static basis* following the rule $P \approx \frac{10}{3}N$ (abbreviation: ST), where we use hyperbolic truncation with $q = 0.5$ for high-dimensional models ($d \geq 20$).

The benchmark is performed on all 11 models presented in Table 1. The experimental design (ED) is created by Latin hypercube sampling (LHS) with optimized maximin distance. We investigate one “small” and one “large” ED size per model (see last column of Table 1), which correspond to the second-smallest and second-largest experimental design size, respectively, of an earlier benchmark (Lüthen et al., 2021) dedicated to investigating solvers and sampling schemes on a static basis. The small ED size is chosen to be at the lower end of the range of reasonable ED sizes. The large ED size represents the “highly informative” regime (in the engineering sense), which for costly engineering models is in the order of a few hundred model evaluations. Here we restrict ourselves to two ED sizes to control the complexity of the results; however, two ED sizes are needed since the earlier benchmark showed that the solver behavior is different in the two regimes. Since almost no engineering model is exactly sparse in the PCE basis, adaptivity can be expected to help identify relevant higher-order terms in all regimes of ED size – as soon as the ED is large enough to contain some information about the model. For each model and experimental design size, we repeat the analysis $R = 30$ times to account for the stochasticity of the sampling method. Due to their excessive computational cost⁴, we omit SPGL1 and anisotropic degree basis adaptivity from the benchmark for high-dimensional models ($d \geq 20$). More details on the settings for the basis adaptivity schemes (e.g., initial basis and investigated degree ranges) can be found in A.

3.1.1 Boxplots of results for the Ishigami function

The results from this benchmark for the Ishigami function are displayed in Fig. 3 in the form of boxplots. Results for the remaining models are shown in B.1. The boxplots visualize the results for all combinations on 30 independent ED realizations. The star-shaped markers denote for one (arbitrarily selected) ED realization the respective validation errors of each of the combinations, highlighting one set of data points which is also contained in the boxplots. We observe that

⁴Anisotropic degree basis adaptivity increases the degree of the basis uniformly by 1 in each dimension, which for high-dimensional models often results in infeasibly large bases. SPGL1, or the implementation we are using, takes considerably longer than the other solvers for an increasing number of basis elements P . Since our benchmark setup requires several thousand runs of the sparse solver for the case of high-dimensional models, the computational cost of the full benchmark is infeasible given our computational resources.

the ranking based on statistics (e.g., median as denoted by the white circle) and the ranking based on the actually attained error on a specific ED can be quite different (see e.g. $SP_{k=5}$ & FN vs. BCS & FN in Fig. 3a). From the star-shaped markers, it is obvious that the solvers and basis-adaptive schemes do not exploit the available information in the same ways: while some combinations show their best performance on the selected ED, others perform average or worse on the same ED. We want to find the “best” method, i.e., a solver-basis adaptivity combination that consistently, on each different model and ED realization, achieves a close-to-optimal error. In other words, in this comparative study we are less interested in the absolute values of the error (since we assume that the tested methods have been validated before in the associated literature), but rather in the *relative performance* of the methods. The boxplots alone do not give the full picture, since they do not show which errors correspond to the same ED realization. Also, looking at the results for this and other benchmark models in B.1, it is difficult to visually extract information about the overall performance of the methods.

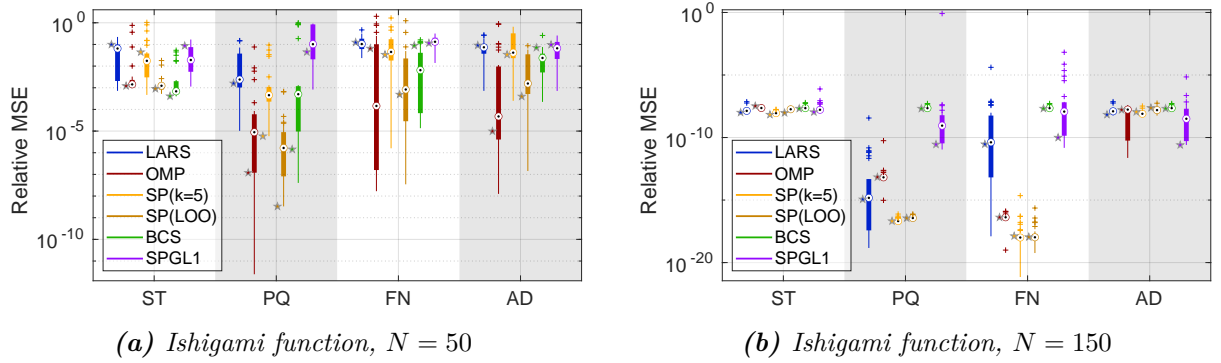


Figure 3: Comparison of all possible combinations of solvers and basis adaptivity schemes for the Ishigami function. Abbreviations of basis adaptivity schemes: ST – static basis, PQ – degree- and q -norm, FN – forward neighbor, AD – anisotropic degree. We show validation errors attained by all 24 combinations of methods on 30 realizations of experimental design. The boxplots visualize the attained errors across all 30 realizations. The star-shaped markers denote for one arbitrarily selected ED realization the errors that the different combinations attain (highlighting one set of data points which is also contained in the boxplots). Left: small ED; right: large ED.

3.1.2 Bar plots – results aggregated over models and replications, separately per model dimensionality and ED size

Therefore, we propose an aggregated visualization of the results as displayed in Fig. 4 and described in the following. For every model and ED size, we determine on each unique ED realization which combination attained the smallest error ϵ^* (for example, for the ED realization visualized in Fig. 3a by the star-shaped markers, SP_{LOO} & PQ attains rank 1). We also determine *on the same ED* for each combination whether it came within a factor of 2, 5, or 10 of ϵ^* . For example, for the ED realization visualized in Fig. 3a by the star-shaped markers, the best error is attained by SP_{LOO} & PQ. All other combinations achieve an error that is more than a magnitude

larger, and would therefore not get a count for this ED realization.

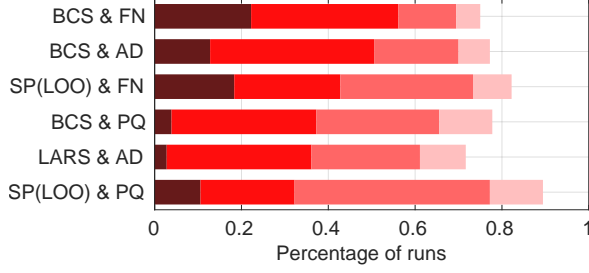
We then aggregate the counts over all repetitions and all models in four different classes: low dimensionality ($d \leq 10$) and small experimental design, low dimensionality and large experimental design, and their corresponding high dimensionality counterparts ($d \geq 20$). In other words, for each of the four classes we count how often each combination achieved the best error or came within a factor of 2, 5, or 10 of the best error, for each unique experimental design. These counts, scaled as percentages of all runs⁵, are then visualized in the form of overlapping horizontal bars in Figs. 4a–4e as follows: The dark red bar shows the percentage of runs in which the particular combination of sparse solver and basis-adaptive scheme found the smallest relative MSE ϵ^* . The other three bars in hues of red illustrate how often the respective combination was within a factor of 2, 5, or 10 of ϵ^* *on the same ED*.

These bars are therefore a measure of the dispersion of the accuracy of each method in terms of distance to the best attainable accuracy. This measure is more interesting than the absolute scatter of a method, since a large variability does not matter as long as the method always provides a close-to-optimal solution; and small variability is of no value if the method is always orders of magnitude away from the best possible solution. We only show the six combinations of solver and basis adaptivity scheme whose relative MSE was most often within two times the best relative MSE (denoted by the bright red bar), and sort the combinations according to this criterion. (For the full list of combinations, see Figs. 8–11 in the appendix.)

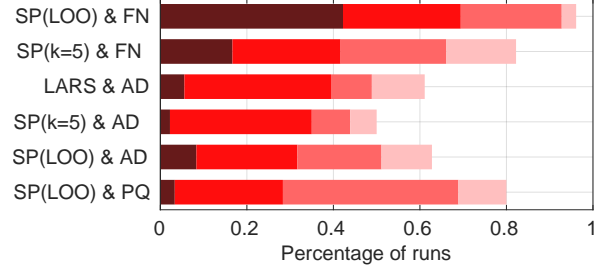
Comparing the subplots 4a–4d, we observe that the results for small and large ED sizes and low- and high-dimensional models are indeed quite different, which justifies analyzing the results separately. As detailed in the previous paragraph, the plots show both which combinations of solver and basis adaptivity scheme attain the smallest relative MSE how often (length of the dark red bar), and how robust the combination is (lengths of the other three bars), i.e., how often the solution was within a small factor of the best solution. We observe that depending on the criterion considered to assess the combinations of solver and basis adaptivity scheme, different combinations turn out to perform best. In the following, we summarize the performance of solver-basis adaptivity combinations with three numbers X - Y - Z , where X denotes the percentage of runs where the respective combination turned out best (i.e., the length of the dark red bar), Y denotes the percentage of runs that were within a factor of two of the smallest error on that ED (length of the bright red bar), and Z denotes the percentage of runs that were within 1 order of magnitude of the smallest error (light red bar). The numbers are rounded to integer values. Italic numbers indicate that this is the best value achieved among all combinations. The enumeration tags refer to the panels in Figure 4.

- (4a) *Low-dimensional models, small ED*: BCS together with forward neighbor basis adaptivity (22-56-75) achieves the smallest error more often than any other combination. However, this combination is not the most robust: in $100 - 75 = 25\%$ of runs, its error is more than

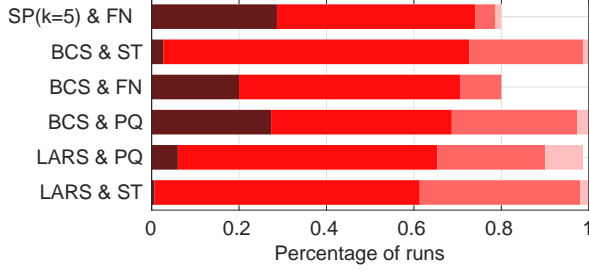
⁵There are 6×30 realizations for the two low-dimensional cases (Figs. 4a and 4b), and 5×30 realizations for the two high-dimensional cases (Figs. 4c and 4d). Fig. 4e aggregates over all 660 ED realizations.



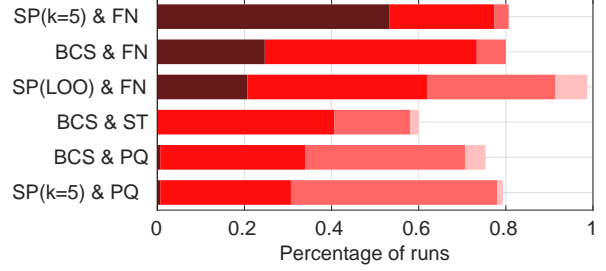
(a) low-dimensional models, small ED (6×30 runs)



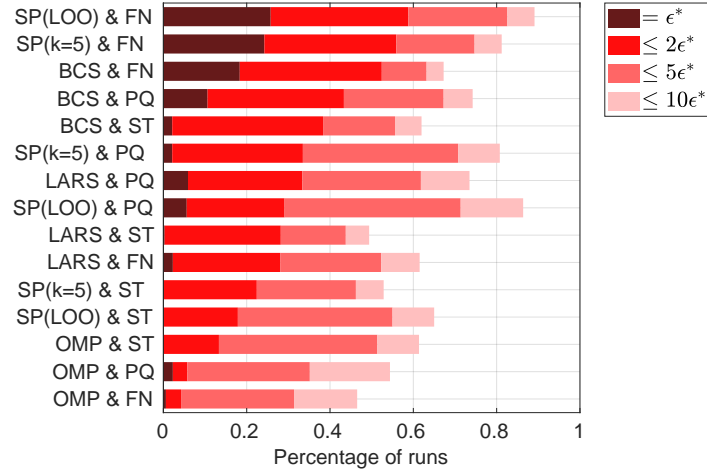
(b) low-dimensional models, large ED (6×30 runs)



(c) high-dimensional models, small ED (5×30 runs)



(d) high-dimensional models, large ED (5×30 runs)



(e) Aggregated results for all 11 models and both ED sizes ($11 \times 2 \times 30$ runs)

Figure 4: Aggregated results of our comparative study. For each of the 11 models, 2 ED sizes, and 30 replications (i.e., ED realizations), we have run each of the combinations of sparse solver and basis adaptivity scheme and computed the corresponding relative mean-squared errors (MSE) (see Fig. 3 for results for the Ishigami function). We determine the relative performance of the method combinations separately on each ED realization: denoting the best attained relative MSE on a specific ED by ϵ^* , we record which of the method combinations reached this error or came within a factor of $\{2, 5, 10\}$ of ϵ^* on the same ED.

We then aggregate the counts over all replications and models, separating by low/high dimensionality and small/large ED size (panels 4a–4d). This results in 6×30 runs for the low-dimensional cases, and 5×30 runs for the high-dimensional cases (per method combination). For readability, in panels 4a–4d only the six best combinations are shown (for the full results, see B.2). The results are displayed in the form of overlapping bars, representing percentages. The dark red bar represents the percentage of runs where the method combination reached the best relative MSE on the specific ED realization. The other bars in shades of red denote the percentages of runs where the error of the particular method combination was within a factor of $\{2, 5, 10\}$ of the best error on the same ED realization. The method combinations are sorted by the length of the bright red bar (number of runs with an error $\leq 2\epsilon^*$).

In panel 4e, we display the relative performance of the method combinations aggregated over all models, ED sizes and replications (i.e., $11 \times 2 \times 30$ runs).

one magnitude worse than the best error. SP_{LOO} together with p&q adaptivity (11-32-89) is the most robust instead.

- (4b) *Low-dimensional models, large ED*: by far the best combination in all three categories is SP_{LOO} together with forward neighbor basis adaptivity (42-69-96).
- (4c) *High-dimensional models, small ED*: there are two combinations that outperform the others: $\text{SP}_{k=5}$ with forward neighbor basis adaptivity (29-74-80) and BCS with a static basis (3-73-100). A close third is BCS with p&q adaptivity (27-69-100).
- (4d) *High-dimensional models, large ED*: the two best combinations are $\text{SP}_{k=5}$ with forward neighbor basis adaptivity (53-77-81) and SP_{LOO} with forward neighbor basis adaptivity (21-62-99).

We observe from this enumeration that only the solvers BCS, $\text{SP}_{k=5}$ and SP_{LOO} are found among the best combinations. Considering the best six combinations (based on the second number Y) as in Fig. 4, only LARS is joining the list. OMP does not perform well in any of the cases (see also Appendix, Figs. 8–11). This is likely due to its tendency to severely underestimate the generalization error, which might make the comparison between error estimates of different bases unreliable. Likewise, SPGL1 is never among the best six combinations for low-dimensional models.

Regarding basis adaptivity schemes, the static basis is not among the best six combinations (based on the second number Y) for low-dimensional models. However for high-dimensional models, especially for the small ED case, it is among the six best combinations several times and performs well.

Note that in the high-dimensional, small ED case (Figs. 4c and 9), the static basis shows the most robust behavior (in terms of the third number Z) while any solver with forward neighbor basis adaptivity is off by more than a magnitude in at least 20% of runs. This is due to its bad performance for the 100D function (see also Fig. 7i). The 100D function is also responsible for the peculiar results of the high-dimensional, large ED case (Figs. 4d and 11): only SP_{LOO} together with forward neighbor basis adaptivity is able to find a very accurate solution, reaching a validation error of 10^{-10} which is several orders of magnitude smaller than what the other combinations reach. However, several other combinations are within one order of magnitude of the smallest error for all other high-dimensional models, reaching overall 80% for Z , as can be seen from Fig. 11.

3.1.3 Bar plots – results aggregated over all models, ED sizes, and replications

To get a complete picture of solver and basis adaptivity scheme performance, we display in Fig. 4e the relative performance of all combinations of solvers and basis adaptivity schemes, averaged over all models and experimental designs regardless of their dimensionality and size. We exclude combinations involving SPGL1 and anisotropic degree basis adaptivity (AD), which were only tested for low-dimensional models. Therefore, Fig. 4e contains $3 \times 5 = 15$ combinations. We

see that SP_{LOO} together with forward neighbor basis adaptivity (26-59-89) performs best in all three categories. $\text{SP}_{k=5}$ together with forward neighbor basis adaptivity (24-56-81) is on the second place in terms of the first two criteria (X and Y), while SP_{LOO} together with p&q basis adaptivity (6-29-86) is the second-best solver in terms of achieving an error within one order of magnitude of the best error most often (Z). BCS together with any basis adaptivity scheme performs well, while all combinations involving OMP are on the bottom of the list. Combinations with a static basis are found more towards the end of the list, and those with forward neighbor basis adaptivity are found more towards the beginning of the list.

3.1.4 Conclusion of the comparative study

From these plots, we see clearly that there is no single combination of sparse solver and basis adaptivity scheme that always outperforms all others. While SP_{LOO} together with forward neighbor basis adaptivity shows superior performance when averaged over all models and ED sizes, we identify better-performing combinations when we differentiate by model dimension and ED size. In some cases, we are faced with a trade-off between accuracy and robustness: e.g., for low-dimensional models and small EDs, the choice of BCS & FN has a higher probability of yielding a near-optimal solution (within a factor of 2 to the best), while the choice of SP_{LOO} & PQ has a higher probability of not being more than a magnitude off from the optimal one.

It seems to be easier to identify trends regarding solvers than regarding basis adaptivity schemes. For example, BCS performs well when information is scarce (small ED sizes or high-dimensional models) while SP_{LOO} performs well for large ED sizes, as already observed in (Lüthen et al., 2021), and OMP generally underperforms. This might be because sparse regression solvers are based on quite different principles, from gradient descent over greedy stepwise regression to Bayesian reasoning, while the basis adaptivity schemes all work with variations of the same concept – namely, gradually increasing the degree of the basis. Basis-adaptive schemes generally outperform the static basis, most likely because they offer more basis elements to choose from. However, it seems to not matter as much how the additional basis elements are generated, since none of the three basis-adaptive methods always finds the best basis. Note that the selection of the bases is guided by the cross-validation error. If the cross-validation error does not correlate well with the real validation error, the scheme will select a suboptimal basis. OMP generally has an unreliable cross-validation error, which might explain its bad performance with basis-adaptive schemes.

3.2 Automatic selection of sparse solver and basis adaptivity scheme

As we saw in the previous section, there is no single best-performing combination of sparse solver and basis adaptivity scheme. In this section, we investigate the question: is there any criterion which can help us choose the best combination of solver and basis adaptivity scheme for a given problem? One such criterion could be a deterministic rule based on dimensionality of the model and ED size, relying on the results from the previous section. Another option could be to use

cross-validation-based model selection criteria as described in Section 2.4 (see also Section 2.1). We call this process of automatically choosing a well-performing combination from a number of candidate combinations using a model selection criterion *automatic selection*.

Due to the performance of the methods in the benchmark in the previous sections, we restrict our investigation to the solvers $SP_{k=5}$, SP_{LOO} and BCS, and to the basis adaptivity schemes

- p&q adaptivity (PQ), forward neighbor adaptivity (FN), and anisotropic degree adaptivity (AD) for low-dimensional models
- static basis (static), p&q adaptivity (PQ), and forward neighbor adaptivity (FN) for high-dimensional models

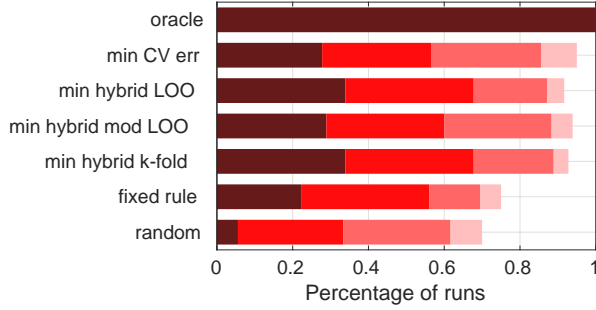
resulting in 9 possible solutions for each model, ED size and repetition.

We use the following model selection criteria (see also Section 2.4):

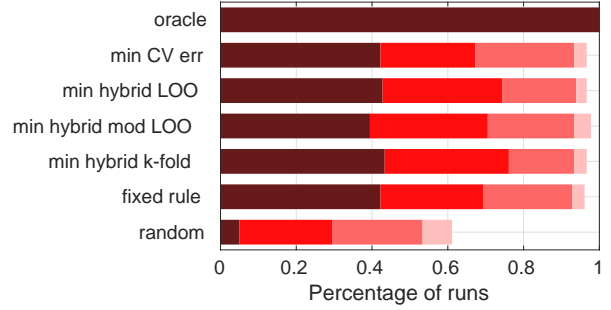
1. the oracle solution, i.e, the smallest relative MSE attained among *all* 24 or 15 combinations of methods (for low-dimensional or high-dimensional models, respectively) – i.e., among all methods tested in Section 3.1, not only among the 9 candidate solutions considered here – on each ED realization, as an ideal lower bound. Of course, this information is not available in practice.
2. the criterion used for basis and hyperparameter selection by the respective solver (see Section 2.1)
3. hybrid LOO, computed by OLS on the active basis functions only
4. hybrid modified LOO, computed by OLS on the active basis functions only, and using the correction factor from Eq. (6)
5. hybrid 10-fold cross-validation error, computed by OLS on the active basis functions only
6. A fixed rule dependent on dimensionality and ED size, according to the findings in Section 3.1, choosing the solver that achieved the smallest error most often (i.e., having the longest dark red bar, and coincidentally also the longest bright red bar):
 - low-dimensional models, small ED: BCS & FN
 - low-dimensional models, large ED: SP_{LOO} & FN
 - high-dimensional models, small ED: $SP_{k=5}$ & FN
 - high-dimensional models, large ED: $SP_{k=5}$ & FN

This implies that by design, the lengths of the bars for this selection criterion are identical with the results for the respective combination in Fig. 4.

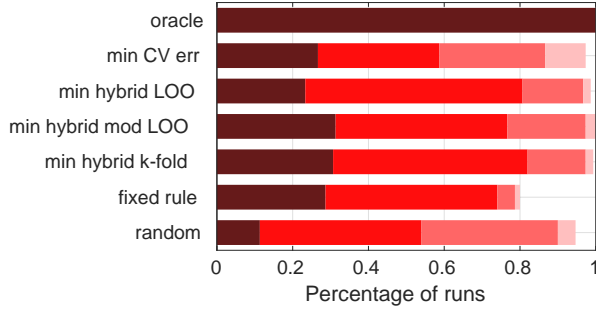
7. A randomly picked combination of solver and basis adaptivity scheme from the 9 available options as upper bound: any sensible model selection criterion must perform better than this.



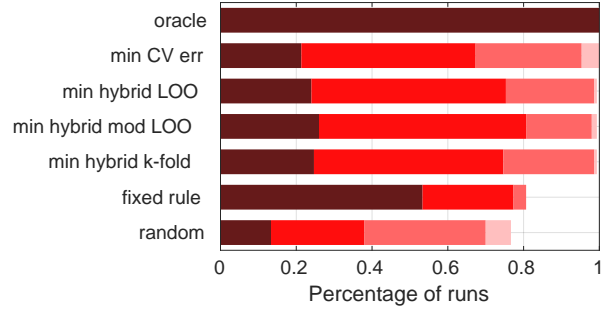
(a) low-dimensional models, small ED



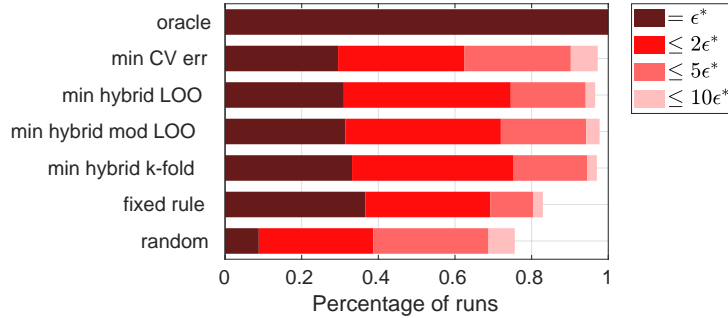
(b) low-dimensional models, large ED



(c) high-dimensional models, small ED



(d) high-dimensional models, large ED



(e) Aggregated over all 11 models and both ED sizes

Figure 5: Testing different automatic selection strategies. As before, the overlapping bar plot visualizes the percentage of runs where the respective strategy achieved the best error ϵ^* or was within a factor of $\{2, 5, 10\}$ of the best error ϵ^* among the seven selection strategies. The bar labels are explained in the enumeration of model selection criteria in the text.

The results are presented in Figure 5. Note that we do not display which sparse solver and basis adaptivity scheme was chosen – we only show how close the chosen solution comes to the best possible solution.

- By construction, the oracle selection performs best in all cases. (We only include it into the plot to emphasize that all other criteria are measured against the best attained error on each ED.)
- The random selection is by far the worst selection criterion, which shows that automatic selection provides a statistically significant improvement over random selection.
- All three hybrid selection criteria are in general more robust than the fixed rule, attaining an error within 2 times of the best solution (bright red bar) similarly or more often than the fixed rule, and being within one order of magnitude of the best error (light red bar) even more than 90% of runs (low-dimensional models) or 99% of runs (high-dimensional models). Averaging over all runs (Fig. 5e), the hybrid criteria attain the best error almost as often as the fixed rule, which was chosen for this reason, i.e., because it attained the best error most often. This shows that automatic selection is able to return a solution with almost the same accuracy as the best-performing method, while being more robust than a fixed rule or a random choice of methods.
- The criterion used for basis and hyperparameter selection, i.e., k-fold CV for $\text{SP}_{k=5}$ and BCS, and LOO for SP_{LOO} , performs slightly worse than the three hybrid selection criteria in all cases shown in Fig. 5. The slightly worse performance of this selection criterion might be explained by selection bias: since the criterion was already used twice for selection, it is likely that it underestimates the generalization error. Another reason might be that the criteria used by different solvers (OLS-based LOO vs. k-fold CV) might not be comparable among each other (inconsistency).

We conclude that automatic selection, i.e., using cross-validation to choose a PCE solution from a number of candidate solutions computed with different methods on the same experimental design, can achieve results that are as accurate as and more robust than results from a fixed rule for which combination of solver and basis adaptivity scheme to use, even if this rule is based on a thorough benchmark such as the one in Section 3.1. While automatic selection leads to increased computational cost due to the need of training the surrogate with different solvers and basis adaptivity schemes on the same experimental design, it is a case-independent strategy applicable to models of any dimensionality and experimental designs of any size, and it finally results in a PCE that is both accurate and more reliable.

4 Conclusion and discussion

Our goal was to provide guidance for the choice of sparse PCE methods in engineering applications, by answering the two questions: (1) Is there a significant difference between different

combinations of sparse solvers and basis adaptivity strategies, and does the proper choice matter in actual applications? (2) Is there a smart strategy to automatically select a good combination?

To answer these questions, we performed an extensive comparative study investigating several approaches for computing sparse PCE with the goal of surrogate modeling, using the relative mean squared error on a validation set as the main performance metric. In particular, we studied the performance of combinations of basis adaptivity schemes and sparse solvers in order to identify combinations that yield the smallest generalization error. Our investigations are based on 11 analytical and numerical benchmark models of varying input dimension and complexity, representative of a wide range of engineering problems. We considered the sparse solvers least angle regression (LARS), orthogonal matching pursuit (OMP), subspace pursuit based on k -fold cross-validation ($\text{SP}_{k=5}$), subspace pursuit based on LOO (SP_{LOO}), Bayesian compressive sensing/FastLaplace (BCS), and spectral projected gradient- ℓ^1 (SPGL1). The basis adaptivity schemes we compared were a fixed basis truncation scheme following the rule $N \approx \frac{10}{3}P$, degree- and q -norm adaptivity, forward neighbor basis adaptivity, and anisotropic degree basis adaptivity. We made a distinction between four cases, namely low- and high-dimensional models as well as small and large experimental design sizes.

The comparative study revealed that it is important to carefully select the strategy, since the difference in generalization error between different combinations of methods can be large, even more than an order of magnitude. No single solver or basis-adaptive scheme significantly outperformed the others. However, by dividing the analysis into classes (low-/high-dimensional models and small/large ED sizes), some significant patterns can be identified. The combinations that performed well always involved the solvers SP_{LOO} , $\text{SP}_{k=5}$, or BCS, but never SPGL1 or OMP. For the basis-adaptive schemes, the picture is less clear, except that combinations involving forward neighbor basis adaptivity achieved the best accuracy most often, and that the static basis (i.e., no basis adaptivity) was in nearly all cases outperformed by basis-adaptive schemes. This might imply that basis-adaptive schemes provide the opportunity to improve the solution by offering more regressors to choose from, but that none of them consistently generates the most suitable basis. The choice of the final basis is guided by the cross-validation error, which is computed based on the solution returned by the respective sparse solver. A good correlation of cross-validation and validation error is crucial for the scheme to make good choices – which for example is not the case for the solver OMP. Overall, there is no combination of solver and basis-adaptive scheme that was consistently best across all models and experimental design sizes, although averaging over all four cases, SP_{LOO} together with forward neighbor basis adaptivity outperformed all other combinations.

Since no clear best combination of solver and basis adaptivity scheme emerged, we introduced an automatic selection step. Based on a suitable error estimate, automatic selection chooses one PCE solution out of several candidate solutions computed by various combinations of solvers and basis adaptivity schemes. We found that automatic selection using any hybrid cross-validation error estimate performs better than the fixed rules obtained from the basis adaptivity benchmark: automatic selection attains the best possible relative MSE almost as often as the best solver-basis adaptivity combination, while being significantly more robust. An additional advan-

tage of automatic selection is that it is independent of model dimension or size of the available experimental design, unlike the proposed fixed rules, which rely on the somewhat arbitrary, albeit simple, classification we applied (low/high dimension and small/large experimental design).

These findings demonstrate that when building a sparse PCE for an expensive black-box computational model, it is worth it to carefully select a sparse solver, and to apply a basis-adaptive scheme, because the difference in relative MSE between different combinations of methods on the same experimental design can be larger than one order of magnitude. While we could identify a number of methods that generally perform well, and others that should be avoided, as we described above, a superior strategy is to compute several PCE solutions and perform a final model selection using one of the presented hybrid cross-validation error estimators.

Further research could investigate the use of true cross-validation for automatic selection instead of the hybrid estimates which we used here. Also, it might be possible to identify other problem characteristics besides model dimension and the size of the available experimental design to guide the choice of methods in the sparse PCE framework. A promising class of methods combines basis adaptivity with the sequential enrichment of the experimental design, adapted to the current candidate or active basis (Fajraoui et al., 2017; Diaz et al., 2018; Hampton and Doostan, 2018), which might be able to further improve on the results obtained here.

Acknowledgements

We thank John Jakeman (Sandia National Laboratories), Negin Alemazkoo (University of Virginia), Hadi Meidani (University of Illinois at Urbana-Champaign), and Jerrad Hampton (University of Colorado Boulder) for generously providing their code and explanations. We thank John Jakeman for his useful hints to relevant literature on basis adaptivity.

References

- Abbiati, G., S. Marelli, N. Tsokanas, B. Sudret, and B. Stojadinovic (2021). A global sensitivity analysis framework for hybrid simulation. *Mechanical Systems and Signal Processing* 146, 106997.
- Adams, B. M., L. E. Bauman, W. J. Bohnhoff, K. R. Dalbey, M. S. Ebeida, J. P. Eddy, M. S. Eldred, P. D. Hough, K. T. Hu, J. D. Jakeman, J. A. Stephens, L. P. Swiler, D. M. Vigil, , and T. M. Wildey (2014). *Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.0 User’s Manual*. Sandia National Laboratories. Technical Report SAND2014-4633 (Updated November 2015 (Version 6.3)).
- Alemazkoo, N. and H. Meidani (2017). Divide and conquer: An incremental sparsity promoting compressive sampling approach for polynomial chaos expansions. *Comput. Methods Appl. Mech. Engrg.* 318, 937–956.

- Babacan, S., R. Molina, and A. Katsaggelos (2010). Bayesian compressive sensing using Laplace priors. *IEEE Trans. Image Process.* 19(1), 53–63.
- Blatman, G. and B. Sudret (2009). Anisotropic parcimonious polynomial chaos expansions based on the sparsity-of-effects principle. In H. Furuta, D. Frangopol, and M. Shinozuka (Eds.), *Proc. 10th Int. Conf. Struct. Safety and Reliability (ICOSSAR'2009), Osaka, Japan*.
- Blatman, G. and B. Sudret (2010a). An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis. *Prob. Eng. Mech.* 25, 183–197.
- Blatman, G. and B. Sudret (2010b). Efficient computation of global sensitivity indices using sparse polynomial chaos expansions. *Reliab. Eng. Syst. Safety* 95, 1216–1229.
- Blatman, G. and B. Sudret (2011). Adaptive sparse polynomial chaos expansion based on Least Angle Regression. *J. Comput. Phys* 230, 2345–2367.
- Bruckstein, A. M., D. L. Donoho, and M. Elad (2009). From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review* 51(1), 34–81.
- Candès, E. J. and Y. Plan (2011). A probabilistic and RIPless theory of compressed sensing. *IEEE Trans. Inform. Theory* 57(11), 7235–7254.
- Candès, E. J. and M. B. Wakin (2008). An introduction to compressive sampling: A sensing/sampling paradigm that goes against the common knowledge in data acquisition. *IEEE Signal Process. Mag.* 25(2), 21–30.
- Chapelle, O., V. Vapnik, and Y. Bengio (2002). Model selection for small sample regression. *Machine Learning* 48(1), 9–23.
- Chatterjee, T., S. Chakraborty, and R. Chowdhury (2019). A critical review of surrogate assisted robust design optimization. *Arch. Comput. Method. E.* 26(1), 245–274.
- Diaz, P. (2018). DOPT_PCE. https://github.com/CU-UQ/DOPT_PCE. [Online; accessed 14-May-2020].
- Diaz, P., A. Doostan, and J. Hampton (2018). Sparse polynomial chaos expansions via compressed sensing and D-optimal design. *Comput. Methods Appl. Mech. Engrg.* 336, 640–666.
- Doostan, A. and H. Owhadi (2011). A non-adapted sparse approximation of PDEs with stochastic inputs. *J. Comput. Phys.* 230(8), 3015–3034.
- Dubourg, V. (2011). *Adaptive surrogate models for reliability analysis and reliability-based design optimization*. Ph. D. thesis, Université Blaise Pascal, Clermont-Ferrand, France.
- Echard, B., N. Gayton, M. Lemaire, and N. Relun (2013). A combined importance sampling and Kriging reliability method for small failure probabilities with time-demanding numerical models. *Reliab. Eng. Syst. Safety* 111, 232–240.

- Efron, B., T. Hastie, I. Johnstone, and R. Tibshirani (2004). Least angle regression. *Ann. Stat.* 32, 407–499.
- Ernst, O., A. Mugler, H.-J. Starkloff, and E. Ullmann (2012). On the convergence of generalized polynomial chaos expansions. *ESAIM: Math. Model. Numer. Anal.* 46(02), 317–339.
- Fajraoui, N., S. Marelli, and B. Sudret (2017). Sequential design of experiment for sparse polynomial chaos expansions. *SIAM/ASA J. Unc. Quant.* 5(1), 1061–1085.
- Forrester, A., A. Sobester, and A. Keane (2008). *Engineering design via surrogate modelling: a practical guide*. Wiley.
- Gerstner, T. and M. Griebel (2003). Dimension-adaptive tensor-product quadrature. *Computing* 71, 65–87.
- Guo, X., D. Dias, and Q. Pan (2019). Probabilistic stability analysis of an embankment dam considering soil spatial variability. *Comput. Geotech.* 113, 103093.
- Hampton, J. and A. Doostan (2017). BASE_PC. https://github.com/CU-UQ/BASE_PC. [Online; accessed 14-May-2020].
- Hampton, J. and A. Doostan (2018). Basis adaptive sample efficient polynomial chaos (BASE-PC). *J. Comput. Phys.* 371, 20–49.
- Hariri-Ardebili, M. and B. Sudret (2020). Polynomial chaos expansion for uncertainty quantification of dam engineering problems. *Engineering Structures* 203(#109631).
- Jakeman, J. D., M. S. Eldred, and K. Sargsyan (2015). Enhancing ℓ_1 -minimization estimates of polynomial chaos expansions using basis selection. *J. Comput. Phys.* 289, 18–34.
- Konakli, K. and B. Sudret (2016). Global sensitivity analysis using low-rank tensor approximations. *Reliab. Eng. Sys. Safety* 156, 64–83.
- Le Gratiet, L., S. Marelli, and B. Sudret (2017). *Metamodel-based sensitivity analysis: polynomial chaos expansions and Gaussian processes*, Chapter 8, Handbook on Uncertainty Quantification (Ghanem, R. and Higdon, D. and Owhadi, H. (Eds.)). Springer.
- Loukrezis, D. and H. De Gersem (2019). Adaptive sparse polynomial chaos expansions via Leja interpolation. *arXiv preprint arXiv:1911.08312*.
- Lüthen, N., S. Marelli, and B. Sudret (2021). Sparse polynomial chaos expansions: Literature survey and benchmark. *SIAM/ASA J. Unc. Quant.* 9(2), 593–649.
- Mai, C. V. and B. Sudret (2015). Hierarchical adaptive polynomial chaos expansions. In M. Papadrakakis, V. Papadopoulos, and G. Stefanou (Eds.), *1st Int. Conf. on Uncertainty Quantification in Computational Sciences and Engineering (UNCECOMP)*, Crete, Greece.

- Marelli, S., N. Lüthen, and B. Sudret (2021). UQLab user manual – Polynomial chaos expansions. Technical report, Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland. Report # UQLab-V1.4-104.
- Marelli, S. and B. Sudret (2014). UQLab: A framework for uncertainty quantification in Matlab. In *Vulnerability, Uncertainty, and Risk (Proc. 2nd Int. Conf. on Vulnerability, Risk Analysis and Management (ICVRAM2014), Liverpool, United Kingdom)*, pp. 2554–2563.
- Morris, M. D., T. J. Mitchell, and D. Ylvisaker (1993). Bayesian design and analysis of computer experiments: use of derivatives in surface prediction. *Technometrics* 35, 243–255.
- Narayan, A. and J. D. Jakeman (2014). Adaptive Leja sparse grid constructions for stochastic collocation and high-dimensional approximation. *SIAM Journal on Scientific Computing* 36(6), A2952–A2983.
- Pati, Y. C., R. Rezaifar, and P. S. Krishnaprasad (1993). Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proc. of 27th Asilomar Conf. on signals, systems and computers*, pp. 40–44. IEEE.
- Sagi, O. and L. Rokach (2018). Ensemble learning: A survey. *WIREs Data Mining Knowl Discov* 8(4), e1249.
- Slot, R. M., J. D. Sorensen, B. Sudret, L. Sverningsen, and M. L. Thogersen (2020). Surrogate model uncertainty in wind turbine reliability assessment. *Renewable Energy* 151, 1150–1162.
- Sudret, B. (2008). Global sensitivity analysis using polynomial chaos expansions. *Reliab. Eng. Sys. Safety* 93, 964–979.
- Thapa, M., S. B. Mulani, and R. W. Walters (2020). Adaptive weighted least-squares polynomial chaos expansion with basis adaptivity and sequential adaptive sampling. *Comput. Meth. Appl. Mech. Eng.* 360, 112759.
- van den Berg, E. and M. P. Friedlander (2008). Probing the Pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comput.* 31(2), 890–912.
- Van den Berg, E. and M. P. Friedlander (2015). SPGL1 - A Matlab solver for sparse optimization. <https://friedlander.io/software/spgl1/>. [Online; accessed 28-August-2019].
- Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.
- Xiu, D. and G. E. Karniadakis (2002). The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM J. Sci. Comput.* 24(2), 619–644.

A Details on the settings for the basis adaptivity schemes in the benchmark

In Table 2, we list the basis adaptivity settings for each of the 11 models in our benchmark.

- The rule for the static basis (ST) is to choose a total-degree basis with p so that the number of basis functions P is closest to $\frac{10}{3}N$, where N is the number of ED points. Here, for low-dimensional models, the q-norm is chosen as $q = 1$, while for high-dimensional models, we use $q = 0.5$.
- For degree and q-norm basis adaptivity (PQ), we choose the ranges for degree and q-norm as large as possible while still keeping the number of basis functions computationally manageable (rule of thumb $\lesssim 10^4$).
- For forward neighbor degree adaptivity (FN), the degree of the initial basis is chosen so that the size of the basis is closest to $10N$ (as recommended in (Jakeman et al., 2015)), while we set the q-norm to the maximum of the q-norm-range for PQ basis adaptivity.
- Finally, for anisotropic degree basis adaptivity, which is only used for low-dimensional models, we use $q = 1$ and $p = \lceil \frac{p_{\max}}{2} \rceil$, where p_{\max} is the maximum of the degree range for PQ basis adaptivity.

B Additional results

B.1 Basis adaptivity benchmark: raw data

Figs. 6 and 7 show the raw data of the basis adaptivity benchmark: for each model and ED size, we display the boxplots of resulting relative MSE for each combination of sparse solver and basis-adaptive scheme. The star-shaped markers visualize the attained error for one specific ED realization. They illustrate that the resulting errors are not always well correlated: on the same ED, some methods achieve one of their smallest errors while others produce one of their largest. Therefore, we continue in Section 3.1 by investigating the relative performance of the different methods.

Table 2: Details on the initial bases and degree and q -norm ranges for the various basis adaptivity schemes

Model	dim d	static basis	PQ range	FN initial	AD initial
Ishigami function	3	$p = 8$ (small ED) / $p = 12$ (large), $q = 1$	$p \in [1, \dots, 25]$, $q \in [0.5, 0.6, \dots, 1]$	$p = 12$ (small ED) / $p = 19$ (large)	$p = 13$
Undamped oscillator	6	$p = 4$ / $p = 4$, $q = 1$	$p \in [1, \dots, 10]$, $q \in [0.5, 0.6, \dots, 1]$	$p = 5$ / $p = 6$	$p = 5$
Borehole function	8	$p = 4$ / $p = 4$, $q = 1$	$p \in [1, \dots, 10]$, $q \in [0.5, 0.6, \dots, 1]$	$p = 5$ / $p = 6$	$p = 5$
Damped oscillator	8	$p = 4$ / $p = 5$, $q = 1$	$p \in [1, \dots, 7]$, $q \in [0.5, 0.6, \dots, 1]$	$p = 5$ / $p = 6$	$p = 4$
Wingweight function	10	$p = 3$ / $p = 4$, $q = 1$	$p \in [1, \dots, 7]$, $q \in [0.5, 0.6, \dots, 1]$	$p = 4$ / $p = 5$	$p = 4$
Truss model	10	$p = 3$ / $p = 4$, $q = 1$	$p \in [1, \dots, 6]$, $q \in [0.5, 0.6, \dots, 1]$	$p = 4$ / $p = 5$	$p = 3$
Morris function	20	$p = 6$ / $p = 8$, $q = 0.5$	$p \in [1, \dots, 8]$, $q \in [0.4, 0.5, 0.6]$	$p = 6$ / $p = 8$, $q = 0.6$	–
Structural frame model	21	$p = 5$ / $p = 8$, $q = 0.5$	$p \in [1, \dots, 8]$, $q \in [0.4, 0.5, 0.6]$	$p = 6$ / $p = 8$, $q = 0.6$	–
2-dim diffusion model	53	$p = 3$ / $p = 4$, $q = 0.5$	$p \in [1, \dots, 6]$, $q \in [0.4, 0.5, 0.6]$	$p = 4$ / $p = 5$, $q = 0.6$	–
1-dim diffusion model	62	$p = 3$ / $p = 4$, $q = 0.5$	$p \in [1, \dots, 5]$, $q \in [0.4, 0.5, 0.6]$	$p = 3$ / $p = 4$, $q = 0.6$	–
100D function	100	$p = 3$ / $p = 4$, $q = 0.5$	$p \in [1, \dots, 5]$, $q = 0.5$	$p = 4$ / $p = 5$, $q = 0.5$	–

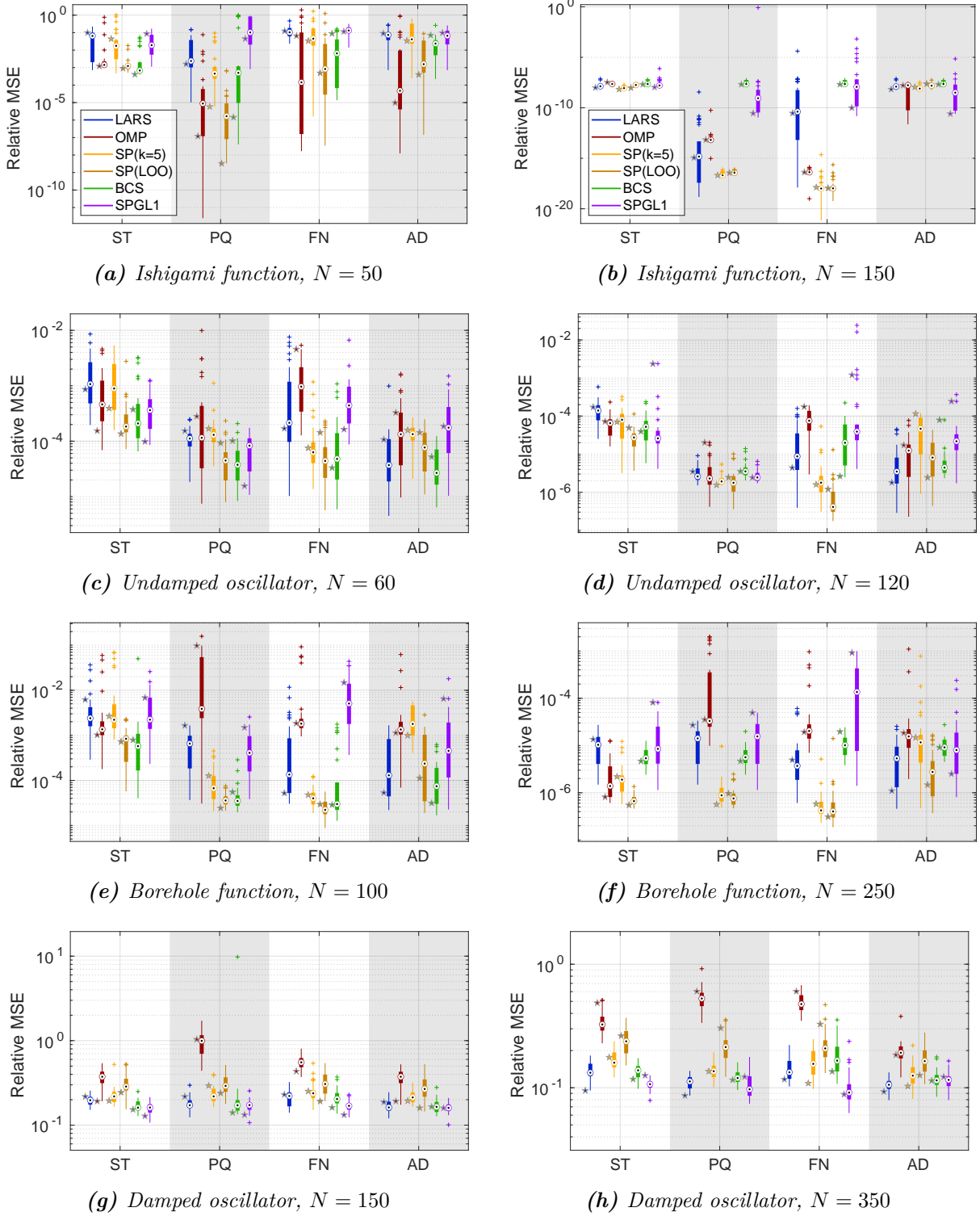
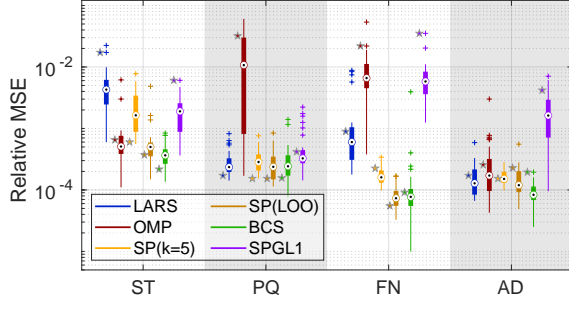
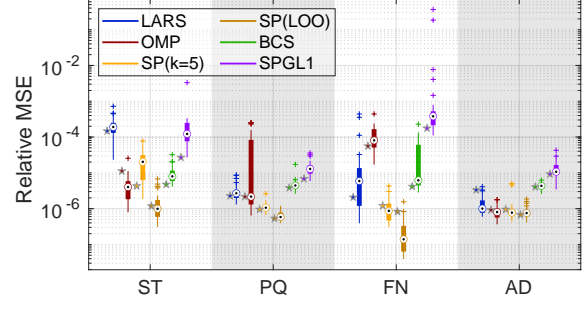


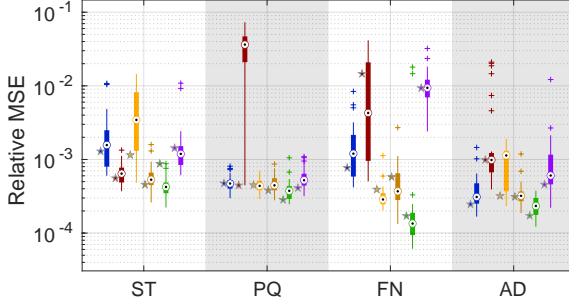
Figure 6: Comparison of different combinations of solvers and basis adaptivity schemes for **low-dimensional models**. We show validation errors attained by all 24 combinations of methods on 30 realizations of experimental design. The boxplots visualize the attained errors across all 30 realizations. The star-shaped markers denote the attained errors of all combinations for one selected ED realization. They highlight one set of data points which is also part of the larger set visualized by the boxplots. Left: small ED; right: large ED. Abbreviations of basis-adaptive schemes: ST: static basis; PQ: degree and q -norm; FN: forward neighbor; AD: anisotropic degree. Continued in next figure.



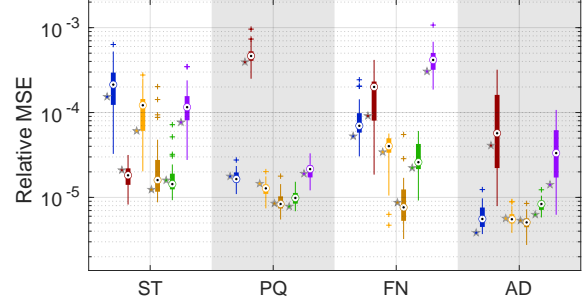
(i) Truss model, $N = 100$



(j) Truss model, $N = 250$



(k) Wingweight function, $N = 100$



(l) Wingweight function, $N = 250$

Figure 6: Continued. Comparison of different combinations of solvers and basis adaptivity schemes for **low-dimensional models**. We show validation errors attained by all 24 combinations of methods on 30 realizations of experimental design. The boxplots visualize the attained errors across all 30 realizations. The star-shaped markers denote the attained errors of all combinations for one selected ED realization. They highlight one set of data points which is also part of the larger set visualized by the boxplots. Left: small ED; right: large ED. Abbreviations of basis-adaptive schemes: ST: static basis; PQ: degree and q -norm; FN: forward neighbor; AD: anisotropic degree.

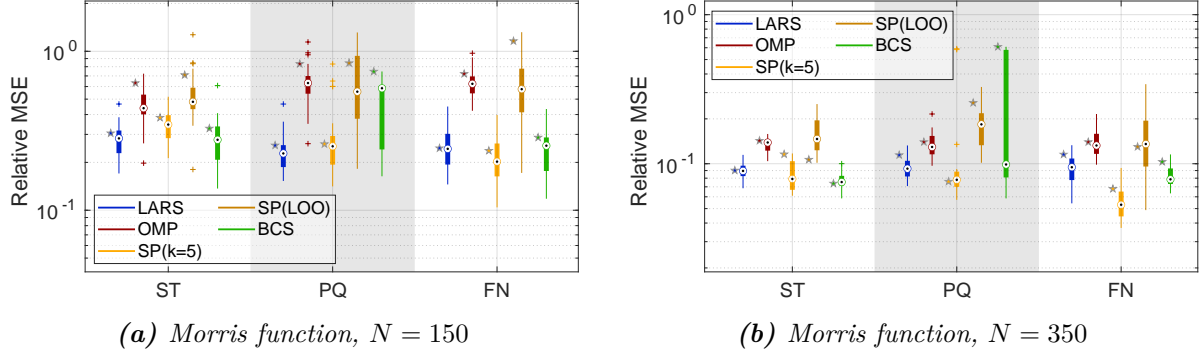
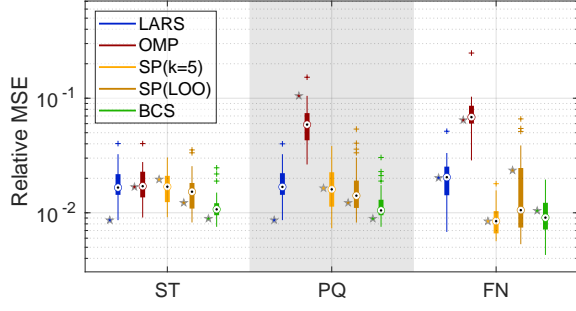


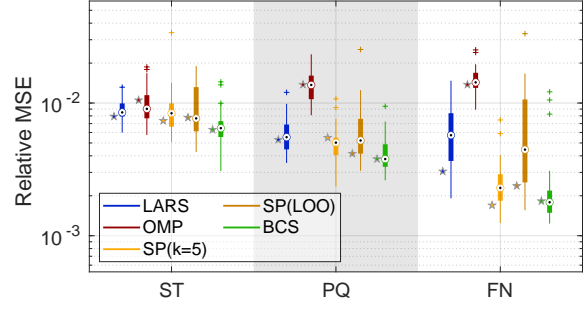
Figure 7: Comparison of different combinations of solvers and basis adaptivity schemes for **high-dimensional models**. We show validation errors attained by all 15 combinations of methods on 30 realizations of experimental design. The boxplots visualize the attained errors across all 30 realizations. The star-shaped markers denote the attained errors of all combinations for one selected ED realization. They highlight one set of data points which is also part of the larger set visualized by the boxplots. Left: small ED; right: large ED. Abbreviations of basis-adaptive schemes: *ST*: static basis; *PQ*: degree and *q*-norm; *FN*: forward neighbor. Continued in next figure.

B.2 Bar plots for all combinations of methods

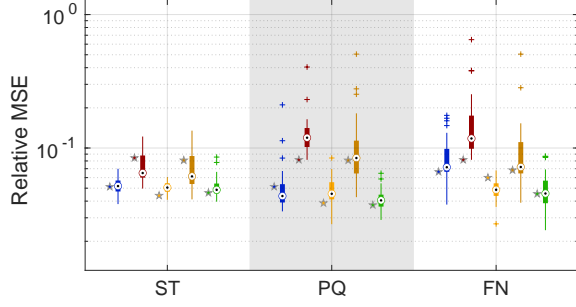
In Figs. 8-11, we display the aggregated results for all combinations of solvers and sampling schemes (while in Section 3.1, we only displayed the six best combinations). We sort the method combinations according to how often they achieved an error within two times the best relMSE (bright red bar). For a detailed description of how to read this plot, we refer to Section 3.1.



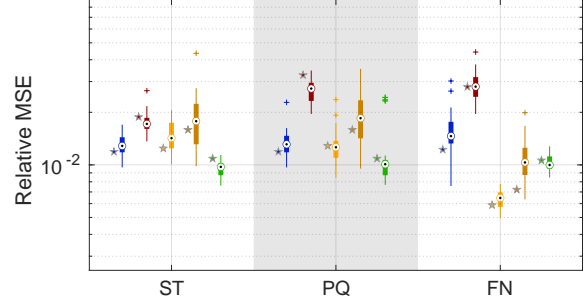
(c) Structural frame, $N = 150$



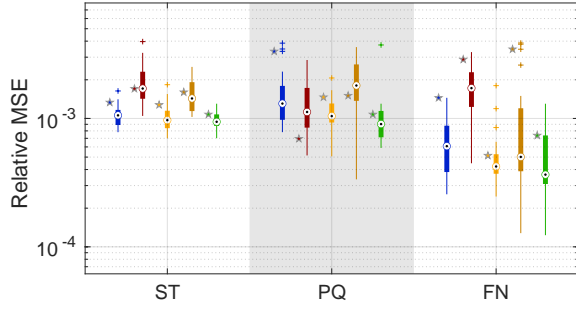
(d) Structural frame, $N = 350$



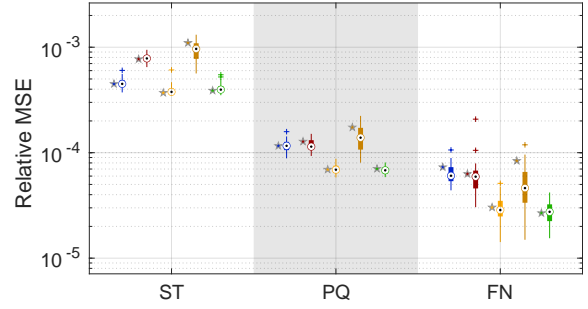
(e) Diffusion 2D, $N = 100$



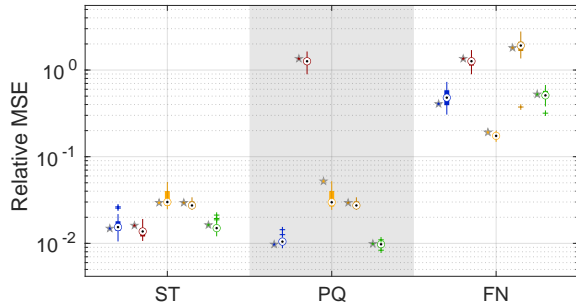
(f) Diffusion 2D, $N = 400$



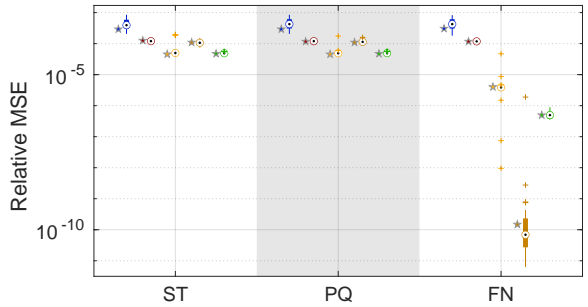
(g) Diffusion 1D, $N = 100$



(h) Diffusion 1D, $N = 400$



(i) 100D function, $N = 400$



(j) 100D function, $N = 1200$

Figure 7: Continued. Comparison of different combinations of solvers and basis adaptivity schemes for **high-dimensional models**. We show validation errors attained by all 15 combinations of methods on 30 realizations of experimental design. The boxplots visualize the attained errors across all 30 realizations. The star-shaped markers denote the attained errors of all combinations for one selected ED realization. They highlight one set of data points which is also part of the larger set visualized by the boxplots. Left: small ED; right: large ED. Abbreviations of basis-adaptive schemes: ST: static basis; PQ: degree and q -norm; FN: forward neighbor; AD: anisotropic degree.

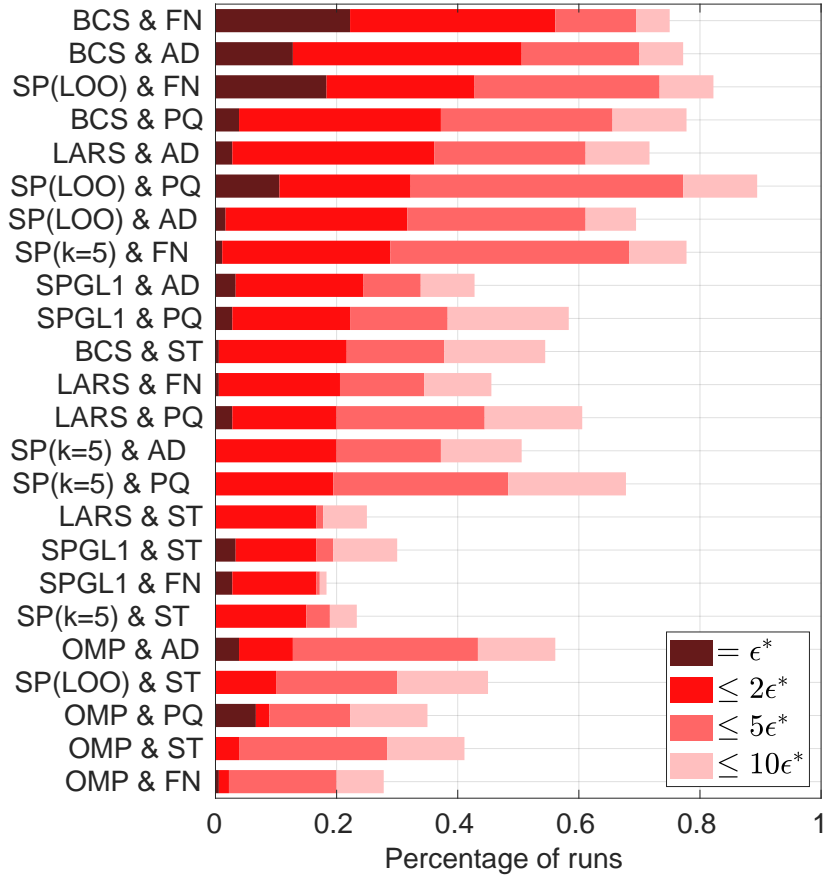


Figure 8: Small ED sizes. Aggregated results for *low-dim* models.

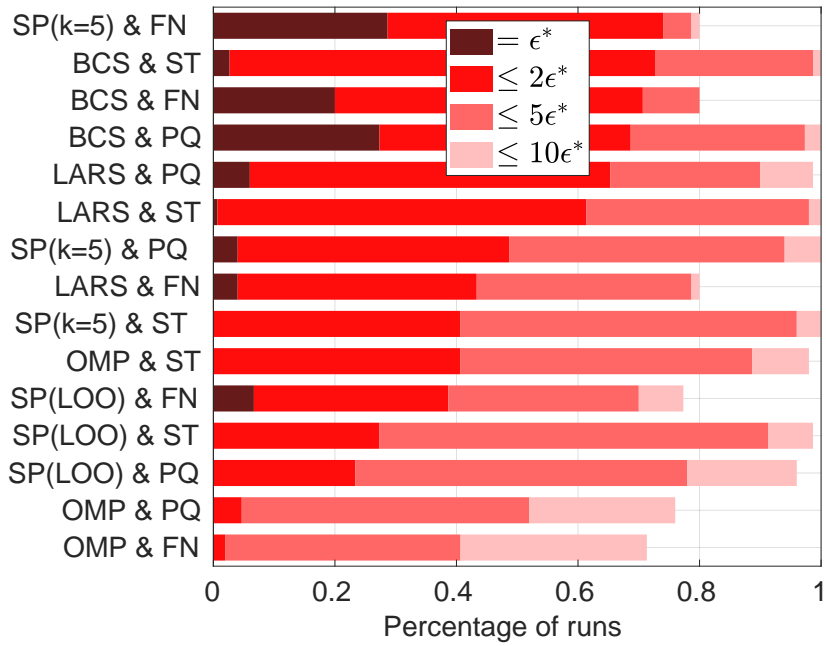


Figure 9: Small ED sizes. Aggregated results for *high-dim* models.

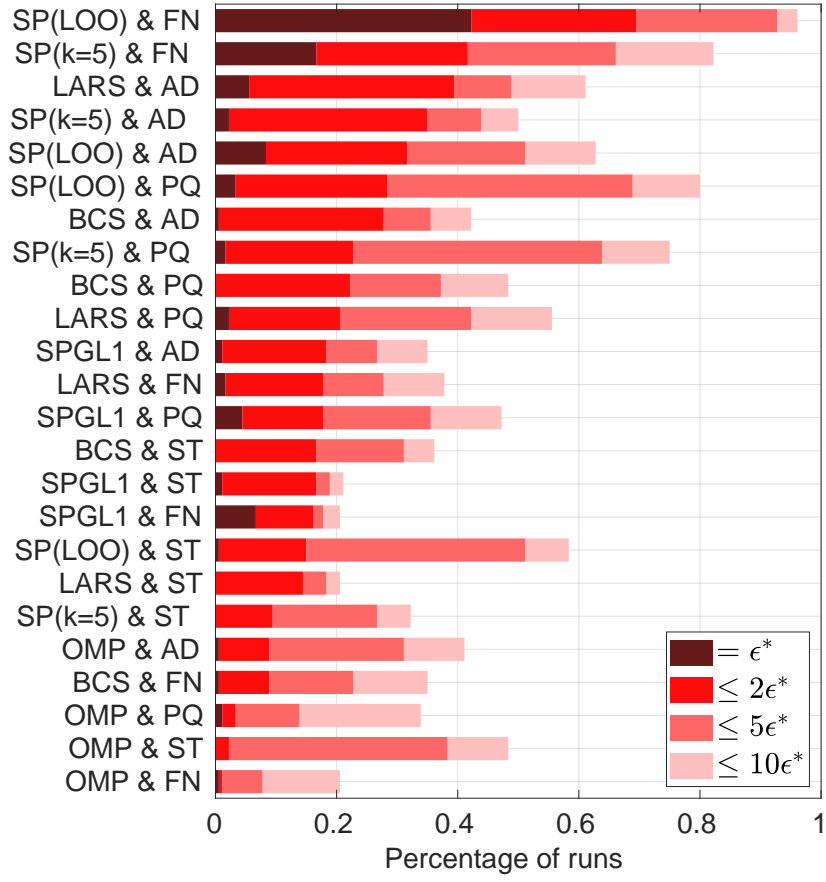


Figure 10: Large ED sizes. Aggregated results for *low-dim* models.

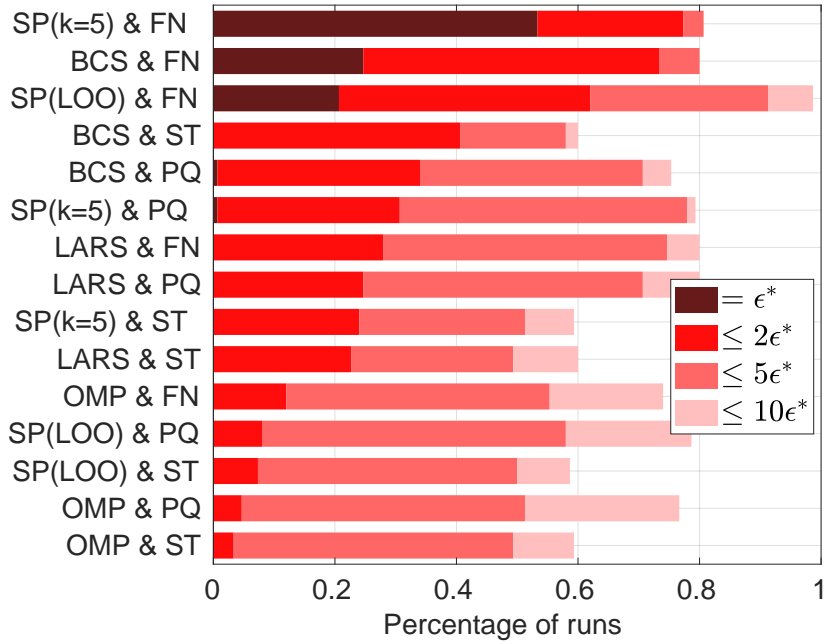


Figure 11: Large ED sizes. Aggregated results for *high-dim* models.