

Non-intrusive surrogate modelling using sparse random features with applications in crashworthiness analysis

Maternus Herold^{1 2}✉, Anna Veselovska^{1 3 4}, Jonas Jehle², and Felix Krahmer^{1 3 4}

¹Department of Mathematics, Technical University of Munich

²BMW Group

³Munich Data Science Institute

⁴Munich Center for Machine Learning

✉ maternus.herold@tum.de

January 2, 2023

Abstract — Efficient surrogate modelling is a key requirement for uncertainty quantification in data-driven scenarios. In this work, a novel approach of using Sparse Random Features for surrogate modelling in combination with self-supervised dimensionality reduction is described. The method is compared to other methods on synthetic and real data obtained from crashworthiness analyses. The results show a superiority of the here described approach over state of the art surrogate modelling techniques, Polynomial Chaos Expansions and Neural Networks.

1 Introduction

Computer simulations are at the core of today’s engineering craft as they allow for the design and analysis of new products in silico, rather than building those artifacts with hardware. The automotive industry, to name one, is heavily relying on computer simulations to test the crashworthiness of its vehicles during development [8, 16–18]. Further, having an accurate computational model is valuable to gain a comprehensive understanding of how the object under observation behaves, e.g., a car’s body during impact. While such information is easily obtainable from the simulation, the results have to be treated with caution due to uncertainty in the parameters and the model design [1, 11, 32, 34, 35]. As the computer simulations tend to have a long runtime, combined with large input dimensions [17, 23], quantifying the uncertainty numerically is challenging. A best practice approach to overcome this difficulty is to learn a data-adaptive surrogate model approximating the simulations that is computationally cheaper, and hence allows for a larger number of evaluations with different parameter settings.

In this work, the relations between inputs and outputs are considered as black-box evaluations of the simulation, i.e. the functions of the generating simulation cannot be altered. A common strategy in such non-intrusive settings is to use readily available data from previously run simulations to learn the surrogate models, framing a data-driven approach [21, 35]. Due to the long run times, the available datasets are small, compared to the high dimensions, leading to the curse of dimensionality when following a data-driven approach.

State-of-the-art methods use the sparsity-of-effects principle to reduce the model’s basis [23, 35], apply dimensionality reduction in a self-supervised fashion [21] or use small neural networks, which implicitly incorporating the two aforementioned approaches [27, 39]. In this paper, we propose an alternative approach based on adapting sparse random feature expansions (SRFE) [13] to surrogate modelling. We compare this algorithm with state-of-the-art methods on computational benchmark data and real crashworthiness data. To the best of our knowledge, our study is the first successful demonstration of using Sparse Random Features in the field of data-driven surrogate modelling for uncertainty quantification (UQ).

1.1 Data-Driven Surrogate Modelling for Uncertainty Quantification

In the context of uncertainty quantification (UQ), a physical or computational model of a system can be seen as a black box that performs the mapping

$$\mathbf{Y} = \mathcal{M}(\mathbf{X})$$

arXiv:2212.14507v1 [cs.LG] 30 Dec 2022

where \mathbf{X} is a random vector that parameterizes the uncertainty of the input through a joint probability density function (PDF) $\rho_{\mathbf{X}}$ and \mathbf{Y} is the corresponding random vector of model response. The leading goal of UQ is to propagate the uncertainties from \mathbf{X} to \mathbf{Y} through \mathcal{M} , which may require several thousands of evaluations of the physical model \mathcal{M} for different realizations of \mathbf{x} of \mathbf{X} . However, most models used in applications have high computational costs per model run, and consequently cannot be used directly [21]. To alleviate the computational burden, surrogate modelling has become a driving tool in the area of UQ.

A surrogate model $\widehat{\mathcal{M}}_{\mathbf{h},c}$ is a computationally inexpensive approximation of the true model $\widehat{\mathcal{M}}$ such that

$$\mathcal{M}(\mathbf{X}) \approx \widehat{\mathcal{M}}_{\mathbf{h},c}(\mathbf{X}), \quad (1)$$

where the hyper-parameters \mathbf{h} encode fixed design features and the parameters c specify the configuration of the surrogate model; they are usually calibrated based on the evaluations of the full model on a training set $\mathcal{X}_{\text{exp}} := \{\mathbf{x}_1^{\text{exp}}, \dots, \mathbf{x}_n^{\text{exp}}\}$. The set \mathcal{X}_{exp} is called *experimental design* and the set of the corresponding evaluations $\mathcal{Y}_{\text{exp}} := \{\mathcal{M}(\mathbf{x}_1^{\text{exp}}), \dots, \mathcal{M}(\mathbf{x}_n^{\text{exp}})\}$ is referred to as *model response*.

In the literature, two types of surrogate models have received particular attention. Firstly, surrogate models based on polynomial chaos expansions take the form $\widehat{\mathcal{M}}_{\mathcal{A},c}(\mathbf{X}) = \sum_{\alpha \in \mathcal{A}} c_{\alpha} \xi_{\alpha}(\mathbf{X})$, where $\xi_{\alpha}(\mathbf{X})$ are multivariate polynomials orthonormal w.r.t. some distribution $\rho_{\mathbf{X}}$ and the hyper-parameters $\mathbf{h} = \mathcal{A} \subset \mathbb{N}^d$ characterize the model by specifying the active polynomial elements used in the expansion [21]. Secondly, surrogate models based on neural networks, \mathbf{h} encodes the number of layers and their widths, while c captures the data-adapted weights. In this paper, we will focus on a third type of surrogate models, the so-called *sparse random feature expansion*, which will be described in the following section.

The quality of a surrogate model can be measured by the relative generalization error E defined as

$$E = \mathbb{E} \left[(\mathbf{Y} - \widehat{\mathcal{M}}_{\mathbf{h}}(\mathbf{X}))^2 \right] / \text{Var}[\mathbf{Y}]. \quad (2)$$

To estimate this error for an unknown distribution, one considers the empirical relative generalization error on a *validation set* $\mathcal{X}_{\text{test}} = \{\mathbf{x}_1^{\text{test}}, \dots, \mathbf{x}_{N_{\text{test}}}^{\text{test}}\}$ with model response $\mathcal{Y}_{\text{val}} = \{\mathcal{M}(\mathbf{x}_1), \dots, \mathcal{M}(\mathbf{x}_{N_{\text{test}}})\}$, which, in line with machine learning terminology, is often referred to as *generalization error* and given by

$$\widehat{\epsilon}(\mathcal{X}_{\text{test}}, \mathcal{Y}_{\text{test}}, \widehat{\mathcal{M}}) = \frac{\sum_{i=1}^{N_{\text{test}}} (y_i^{\text{test}} - \widehat{\mathcal{M}}(\mathbf{x}_i^{\text{test}}))^2}{\sum_{i=1}^{N_{\text{test}}} (y_i^{\text{test}} - \mu_y)^2}. \quad (3)$$

Here $y_i^{\text{test}} := \mathcal{M}(\mathbf{x}_i^{\text{test}})$ and $\mu_y = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} y_i^{\text{test}}$ is the sample mean of the validation set for the model response. The corresponding quantity evaluated on the experimental design \mathcal{X}_{exp} , i.e., $\widehat{\epsilon}(\mathcal{X}_{\text{exp}}, \mathcal{Y}_{\text{exp}}, \widehat{\mathcal{M}})$, is often referred to as training error.

1.2 Outline

This paper is structured in the following manner. In Section 2, we describe computational tools for data-driven surrogate modelling for UQ. Then, we present the a novel method of self-supervised surrogate modelling with low order sparse random feature expansion (LOS-RFE) in Section 3, including a discussion of the used optimization components and implementation. In Section 4, the presented LOS-RFE method is experimentally compared to a combination of dimensionality reduction and polynomial chaos expansion [21] and neural networks, which are both state-of-the-art surrogate modelling techniques. The numerical comparison is done both on synthetic data and on real data obtained from studies on crashworthiness.

2 Background on Computational Tools

In this paper, we propose a new method for surrogate modelling in high dimensions that emerges through a combination of sparse random feature expansion and dimensionality reduction to self-supervised learning. In this section, we provide some background on each of these topics. After discussing sparse random feature expansions in Section 2.1, we review methods for dimensionality reduction in Section 2.2 and describe our optimization method of choice in Section 2.3.

2.1 Sparse Random Feature Expansion via the LASSO

The idea of the *sparse random feature expansion* (SRFE) [13] is to fit the target model \mathcal{M} by sparse linear combinations of elements of a sufficiently rich randomized function family. To be more precise, one fixes a bivariate function ϕ and considers the random collection of functions $\phi(\cdot; \omega_i)$, $i = 1, \dots, R$, with weight vectors $\omega_i \in \mathbb{R}^d$ drawn independently at random from some probability distribution $\rho(\omega)$. These functions $\phi(\cdot; \omega_1), \dots, \phi(\cdot; \omega_R)$ are called *random features* [29]. Random features give rise to surrogate models f^* of the form

$$f^*(\mathbf{X}) := \sum_{j=1}^R c_j \phi(\mathbf{X}; \omega_j). \quad (4)$$

Consequently, given a model \mathcal{M} , the aim is to find coefficients $\mathbf{c} := (c_1, \dots, c_R) \in \mathbb{R}^R$ such that the resulting *random feature expansion* f^* approximates \mathcal{M} [29], that is,

$$\mathcal{M}(\mathbf{X}) = f^*(\mathbf{X}) + \varepsilon, \quad (5)$$

with some sufficiently small error term ε . Popular choices for $\phi(\cdot; \omega)$ include

- random Fourier features: $\phi(\cdot; \omega) = \exp(i\langle \mathbf{x}, \omega \rangle)$
- random trigonometric features: $\phi(\cdot; \omega) = \cos(\langle \mathbf{x}, \omega \rangle)$ and $\phi(\cdot; \omega) = \sin(\langle \mathbf{x}, \omega \rangle)$
- random ReLU features: $\phi(\cdot; \omega) = \max\{\langle \mathbf{x}, \omega \rangle, 0\}$.

It was noted in [13] that the generalization ability of the surrogate model may improve when enforcing in addition that the coefficient vector \mathbf{c} is sparse, that is, it has only a small number of non-vanishing entries. This is also the approach that we will be following in this paper. More precisely, we consider the so-called *low order sparse RFE* (LOS-RFE) [13] that promotes *bi-level sparsity* in the expansion (5). The first sparsity level is motivated by the fact that in many applications \mathcal{M} is well-approximated by a small subset of the random features $\phi(\cdot; \omega_1) \dots \phi(\cdot; \omega_R)$. Therefore, one aims for \mathbf{c} with only few non-zero entries, which can be done by minimizing $\|\mathbf{c}\|_1$ with respect to the given model response \mathcal{Y} .

Secondly, we enforce the *sparsity-of-effects* or Pareto principle, which states that most real-world systems are dominated by a small number of low-complexity interactions. Consequently, we also promote sparsity for the feature weights ω . Mathematically, one can model this principle by the so-called *order- q functions*. We call a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ of order- q of at most K terms $q < d$, if there exist K functions $g_1, \dots, g_K : \mathbb{R}^q \rightarrow \mathbb{R}$ such that

$$g(x_1, \dots, x_d) = \frac{1}{K} \sum_{j=1}^K g_j(x_{j_1}, \dots, x_{j_q}). \quad (6)$$

Since for the target model \mathcal{M} , the set of active variables is not known in advance, to incorporate the potential coordinate sparsity into the weights ω_i , one randomly draws $R = n \binom{d}{q}$ q -sparse features ω_j with nonzero entries distributed w.r.t. $\rho_q(\omega)$ for each subset of size q from the index set $\{1, \dots, d\}$, for more details see [13].

To track the coefficient vector \mathbf{c} numerically, one can proceed as follows. Let $\mathbf{A} \in \mathbb{C}^{m \times R}$ be given by $a_{jk} = \phi(\mathbf{x}_k, \omega_j)$ and consider the vector $\mathbf{y} = (y_1, \dots, y_m)$ with $y_i = \mathcal{M}(\mathbf{x}_i)$ representing the vectorized data \mathcal{Y} . Then approximating \mathcal{M} by f^* as in (5) with sparse \mathbf{c} is equivalent to finding \mathbf{c} that minimizes the residual norm $\|\mathbf{A}\mathbf{c} - \mathbf{y}\|_2$ under the constraint that only a few entries of \mathbf{c} are non-zero. In [13], it has been proposed to track \mathbf{c} via the BP-denoising problem. We observed, however, that in the context of the problems studied in this paper, solving a BP-denoising problem takes too long to be used in a self-supervised loop where optimization is run several hundreds of times. Hence, a more efficient optimization is required, and to find the coefficient vector \mathbf{c} we propose to use the following LASSO formulation

$$\operatorname{argmin}_{\mathbf{c} \in \mathbb{R}^d} \lambda \|\mathbf{c}\|_1 + \|\mathbf{A}\mathbf{c} - \mathbf{y}\|_2^2. \quad (7)$$

Algorithm 1: Sparse Random Feature Expansion via the LASSO

1 input : observed data $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, parametric basis function $\phi_\omega(\mathbf{x}) = \phi(\langle \mathbf{x}, \omega \rangle)$, feature sparsity q , probably measure ρ and a stability parameter η
2 output: surrogate model: $f^*(\mathbf{x}) = \sum_{j=1}^R c_j^* \phi_{\omega_j}(\mathbf{x})$
 /* draw q -sparse feature weights */
3 $\Omega = \{\omega_j \sim \rho_q(\omega) \mid \forall j \in [1, \dots, R]\}$
 /* construct random feature matrix */
4 $\mathbf{A} \in \mathbb{R}^{m \times R}$ s.t. $a_{kj} = \phi_{\omega_j}(\mathbf{x}_k)$
 /* solve the Lasso problem to obtain the coefficients */
5 $\mathbf{c}^* = \underset{\mathbf{c}}{\operatorname{argmin}} \lambda \|\mathbf{c}\|_1 + \|\mathbf{A}\mathbf{c} - \mathbf{y}\|_2^2$

In this formulation, the hyperparameter λ balances the sparsity inducing regularizer and the data fidelity. Indeed, if λ is well chosen, (7) will reduce coefficients to zero, yielding a sparse solution.

From the theoretical perspective, it has been shown in [29] that if the model \mathcal{M} belongs to a reproducing kernel Hilbert space (associated with the choice of the feature functions), then the model can be uniformly approximated by the RFE as in (4) with any accuracy ε in the case when the number of random features R is chosen of order $1/\varepsilon^2$. For the LOS-RFE, it has been shown that low-order functions can be approximated with the accuracy ε with a significantly lower number of random features R , see [13] for more details. We believe that this improvement is key for obtaining an advantage in comparison to the neural-network-based methods and Polynomial Chaos Expansion (PCE).

2.2 Dimensionality Reduction via Kernel Principal Component Analysis

A common challenge in learning a surrogate model is that the experimental design $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ is rather high-dimensional, so a good surrogate model will need to have a large number of parameters, which leads to computational bottlenecks. At the same time, the experimental design is often intrinsically low-dimensional, that is, all data points lie close to a lower-dimensional manifold. A common strategy is to first identify this lower-dimensional structure and then learn the surrogate model only on this set of interest.

Mathematically, this dimensionality reduction (DR) corresponds to a non-linear mapping $\tau : \mathbb{R}^d \rightarrow \mathbb{R}^k$ with $k \ll d$. We denote the image of \mathcal{X} under this map by $\mathcal{Z} = \{\mathbf{z}_i = \tau(\mathbf{x}_i), i = 1, \dots, N\} \subset \mathbb{R}^k$. The aim is to choose the parameter k close to the dimension of the underlying manifold; consequently k is often referred to as the “intrinsic dimension” of \mathcal{X} .

In this work, we use *kernel principal component analysis* (KPCA) [4, 14, 24, 31], which is a non-linear variant of principal component analysis (PCA), where the data is first mapped to a higher dimensional or potentially even infinite-dimensional Hilbert space \mathcal{H} via a feature map $\Phi : \mathbb{R}^d \rightarrow \mathcal{H}$, and the PCA is applied.

Analogously to the classical PCA case, to directly compute the principal components of $\Phi(\mathcal{X})$ one would need to perform the eigen-decomposition of the sample covariance matrix $C_{\Phi(\mathcal{X})} = \frac{1}{N} \sum_{i=1}^N \Phi(\mathbf{x}_i) \Phi(\mathbf{x}_i)^*$. Such a direct computation, however, is, in general, not feasible and the dimension of \mathcal{H} is typically very large or even infinite. This problem is by-passed by observing that each eigenvector \mathbf{v}_k belongs to the span of the samples $\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_M)$, therefore scalar coefficients α_k^i exist [31], such that each eigenvector \mathbf{v}_m can be expressed as the following linear combination

$$\mathbf{v}_m = \sum_{i=1}^N \alpha_i^m \Phi(\mathbf{x}_i), \quad m = 1, \dots, N. \quad (8)$$

Consequently,

$$C_{\Phi(\mathcal{X})} \mathbf{v}_m = \sum_{j=1}^N \sum_{i=1}^N \alpha_i^m \Phi(\mathbf{x}_j) \left(\Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i) \right). \quad (9)$$

Thus, if one considers the kernel function $\kappa : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ given by

$$\kappa(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}), \quad (10)$$

one can apply the so-called kernel trick, which refers to the observation that, if the access to \mathcal{H} only takes place through inner products, then there is no need to explicitly compute Φ . Rather, the result of the inner product can be directly calculated using $\kappa(\cdot, \cdot)$. With this observation, one can reformulate the eigenvalue problem for the matrix $C_{\Phi(\mathcal{X})}$ as the following eigenvalue problem

$$\mathbf{K}\alpha^m = \lambda_i \alpha^m, \quad m = 1, \dots, N \quad (11)$$

for the kernel matrix \mathbf{K} given by $\mathbf{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ and the coefficient vector α^m in representation (8). The α^m can then be converted back to the eigenvectors \mathbf{v}_m .

This gives rise to a dimensionality reduction method (see, e.g., [31]) where each data point projected to the k leading principal axes $\mathbf{V}_k := \{\mathbf{v}_m, m = 1, \dots, k\}$. That is, each \mathbf{x}_i is represented by a vector $\mathbf{z}_i = (\mathbf{z}_i^1, \dots, \mathbf{z}_i^k)^T$ with entries

$$\mathbf{z}_i^m = \Phi(\mathbf{x}_i)^T \mathbf{v}_m = \sum_{j=1}^N \alpha_j^m \kappa(\mathbf{x}_i, \mathbf{x}_j). \quad (12)$$

Note that this method does not explicitly use the feature map and can hence be defined also for more general classes of kernels.

In this paper, we use KPCA with a nonisotropic Gaussian kernel parameterized by a vector $\theta \in \mathbb{R}^d$ as given by

$$\kappa_{\theta}(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2} \sum_{i=1}^d \frac{1}{\theta_i^2} (\mathbf{x}_i - \mathbf{y}_i)^2\right), \quad (13)$$

and denote the corresponding dimensionality reduction map by τ_{θ}^k with θ and k indicating the kernel parameters and the reduced dimension, respectively.

2.3 Particle Swarm Optimization

The aim of our approach is choose the parameter θ in the Gaussian kernel in a data-adaptive way. To optimize the choice of θ , we will use particle swarm optimization (PSO) [28], a gradient-free metaheuristic to minimize a given loss-function $\mathcal{L} : \mathcal{D} \subset \mathbb{R}^m \rightarrow \mathbb{R}$.

The idea behind the PSO metaheuristic, as the name suggests, is to use a large number of points in the domain \mathcal{D} called particles, a particle swarm, in order to explore the landscape of the loss function in an iterative way. As a starting point, the particles are randomly initialized, and then each particle explores the co-domain of the function driven by the difference between the values of \mathcal{L} at its own position and at the positions of the other particles. After a certain number iterations, the method outputs the minimum point ever explored by this swarm of particles.

More rigorously, assume that we have P particles and the position of a particle $p \in \{1, \dots, P\}$ at iteration t is denoted by $\hat{\theta}_p(t) \in \mathbb{R}^m$ and its velocity is denoted by $V_p(t) \in \mathbb{R}^m$. This velocity is recomputed in each iteration step as a combination of three contributions

- The velocity $V_p(t)$ of the previous iteration, weighted by a learning rate $r \in [0, 1]$.
- A cognitive term driving the particle towards the location of minimal loss $\hat{\theta}_{i,p}^{pbest}$ it has previously encountered, weighted by a constant $c_{cognitive}$ and a random positive rescaling $u_1(t) \sim \mathcal{U}[0, \xi_1]^d$ for some $\xi_1 \in \mathbb{R}$.
- A social term driving the particle towards the location of minimal loss $\hat{\theta}_i^{gbest}$ that has been previously encountered by any particle in the swarm, weighted by a constant c_{social} and a random positive rescaling $u_2(t) \sim \mathcal{U}[0, \xi_2]^d$ for some $\xi_2 \in \mathbb{R}$.

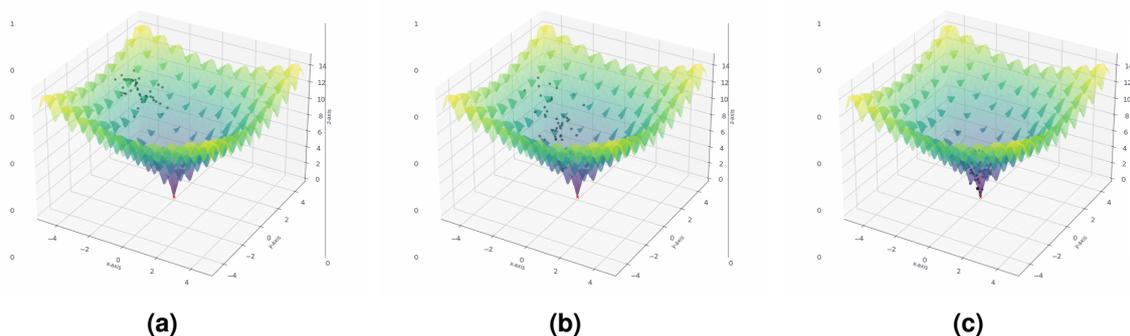


Figure 1 The PSO algorithm is applied to the popular Ackley function. The different particles used are marked as black dots. Fig. 1a shows locations of the particles shortly after the start of the optimization. Fig. 1b indicates the positions of particles after a couple of iterations, and Fig.1c shows the final iteration stage. As one can see, the particles first spread out to explore the function landscape and then move closer and closer to the global minimum further in the optimization.

In formulas, this update reads as

$$V_p(t+1) = r \cdot V_p(t) + c_{\text{cognitive}} \cdot u_1(t) [\hat{\theta}_{i,p}^{pbest} - \hat{\theta}_p(t)] + c_{\text{social}} \cdot u_2(t) [\hat{\theta}_i^{gbest} - \hat{\theta}_p(t)]. \quad (14)$$

The random weights u_1, u_2 are often called acceleration coefficients [28] and the underlying parameters $\xi_{1,2}$ determining the magnitude of the random force is usually set equal to 2 [28]. With this velocity vector, the position $\hat{\theta}_p(t)$ of particle p is updated as

$$\hat{\theta}_p(t+1) = \hat{\theta}_p(t) + V_p(t+1). \quad (15)$$

We will denote the output of the PSO algorithm with starting points $\{\hat{\theta}_i\}_{i=1}^P$ and loss function \mathcal{L} by **PSO**($\mathcal{L}, \{\hat{\theta}_i\}_{i=1}^P$).

The behavior of the PSO algorithm is sketched in Figure 1, where PSO is used to optimize over the Ackley function used for benchmarking. We observe numerous local minima, in which the algorithm could get stuck, but it proceeds to the global minimum.

For the sake of completeness, it is to mention that there are a number of related optimizers also based on interacting particle systems, which have gained lots of popularity in recent years as they yield stronger theoretical guarantees with respect to their convergence rate. In particular *consensus-based optimization* models the movement of the particles using Brownian motion, which is in favour of the exploration capabilities of the particles [5–7, 9, 10, 38]. While motivated by implementation efficiency of the PSO algorithm we consider only the PSO method for the purpose of this paper, we find it a very interesting research direction to explore the potential of alternative optimization methods for our problem both in theory and practice.

3 Self-Supervised Sparse Random Features for Surrogate Modelling

The method we propose in this paper combines the three aforementioned approaches to design and calibrate a surrogate model. The dimensionality is reduced via KPCA with parameters θ optimized via PSO and a surrogate model for the reduced dimension is designed as a sparse random feature expansion (LOS-RFE). Key to our approach is that these steps are not performed sequentially but the method is self-supervised – the parameters are constantly updated in view of the success of the combined method.

The approach of enhancing a model's performance via self-supervision is especially common in the field of computer vision [12, 15, 25] and has been just recently applied to surrogate modelling by Lataniotis et al., [21]. Such techniques are particularly useful when the available data is of insufficient resolution or in the case of scarce data, which is in the focus of this work. Further, a self-supervised approach eases the process of fitting a model in a data-driven setting as the incorporated algorithms feedback themselves.

To implement our self-supervised method, we split the experimental design $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ and the corresponding model response $\mathcal{Y} = \{\mathcal{M}(\mathbf{x}_1), \dots, \mathcal{M}(\mathbf{x}_n)\}$ into two parts, a training set \mathcal{X}_{train} used to learn the coefficients of the random features and a validation set \mathcal{X}_{val} used to supervise the learning process, i.e., to update iteratively the KPCA parameters. Note that the validation set is different from the test set \mathcal{X}_{test} ; the latter is never used in the learning process. That is, the data set is now divided into three parts of cardinalities $N_{val} + N_{train} + N_{test} = n$.

A challenge we are facing is that it is not clear a priori which intrinsic dimension k^* to choose. Consequently, we explore multiple potential intrinsic dimensions in a candidate set $D \subset \mathbb{N}$. For each $k \in D$, we use PSO to simultaneously optimize the parameter θ for the KPCA dimensionality reduction map τ_θ^k and the coefficients c of the LOS-RFE surrogate model f_c for the data set in the reduced dimension.

In the initialization step, $t = 0$, of the PSO algorithm, we randomly generate $P \in \mathbb{N}$ particles, $\{\theta_p \in \mathbb{R}^k, p = 1, \dots, P\}$ representing the initial KPCA parameter vector θ and set $V_p(0) = 0$. We then iterate over t and repeat the following steps. For each θ_p , the training set $\mathcal{X}_{train} \subset \mathbb{R}^d$ is reduced to $\mathcal{Z}_{\theta_p, train} = \tau_{\theta_p}^k(\mathcal{X}_{train}) \subset \mathbb{R}^k$ by the respectively parameterized KPCA map $\tau_{\theta_p}^k$. For the reduced training set $\mathcal{Z}_{\theta_p, train}$ and the corresponding model response \mathcal{Y}_{train} , we optimize the parameters $c_p = (c_{1,p}, \dots, c_{R,p})$ in a surrogate model f_{c_p} of the form

$$f_{c_p}(\mathbf{z}) = \sum_{j=1}^R c_{j,p} \phi(\mathbf{z}; \boldsymbol{\omega}_j), \quad (16)$$

via a LASSO-inspired method (see below for details). Here, the $\phi(\mathbf{z}; \boldsymbol{\omega}_j)$ are fixed realizations of random features with sparse Gaussian weights $\boldsymbol{\omega}_j \in \mathbb{R}^k$ as discussed above.

Now, again for each $p = 1, \dots, P$, the key paradigm of PSO is used to update the parameter θ_p on the reduced validation set $\mathcal{Z}_{\theta_p, val} = \tau_{\theta_p}^k(\mathcal{X}_{val}) \subset \mathbb{R}^k$ based on the loss

$$\tilde{e}(\mathcal{Z}_{\theta_p, val}, \mathcal{Y}_{val}, f_c) = \frac{\sum_{i=1}^{N_{val}} (y_i^{val} - f_c(\mathbf{z}_i))^2}{\sum_{i=1}^{N_{val}} (y_i^{val} - \mu_y)^2}. \quad (17)$$

associated with the surrogate model f_{c_p} learned in the first step. That is, we compute a velocity $V_p(t+1)$ as a combination of the previous velocity $V_p(t)$ with a cognitive and a social term, (see Section 2.3 for details). We then use this velocity to update the KPCA parameter θ_p via

$$\hat{\theta}_p(t+1) = \hat{\theta}_p(t) + V_p(t+1). \quad (18)$$

If the PSO algorithm converges (which we observed to be the case in most instances), we obtain a pair of KPCA parameters $\theta \in \mathbb{R}^k$ and LOS-RFE parameters $c = (c_1, \dots, c_R)$ for each $k \in D$.

It remains to describe how we compute random feature coefficients c .

As the number of surrogate models to fit per dimension k is equal to the product of the swarm size and the number of iterations (the optimization is run for), solving for the optimal sparse coefficients c^* via the LASSO, as proposed in Section 2.1, in each iteration is computationally quite demanding. To overcome this computational bottleneck, we observe that exactly solving the LASSO is of little value during the initial stage of the PSO algorithm, as first we need to find a rough direction for the swarm of particles to be moved to. To find such a direction for the swarm $\{\theta_p \in \mathbb{R}^k, p = 1, \dots, P\}$ in a cheaper way, we use for the initial PSO steps the following ridge formulation

$$\operatorname{argmin}_{\mathbf{c} \in \mathbb{R}^d} \lambda \|\mathbf{c}\|_2 + \|\mathbf{A}\mathbf{c} - \mathbf{y}\|_2^2 \quad (19)$$

for finding the coefficients c in the surrogate model f_c instead of the LASSO. Such a ridge formulation can be solved more efficiently, decreasing the running time per optimization step. Once the particles start to converge, i.e. the particle position's change is less than a certain threshold $\eta \in \mathbb{R}$

$$\|\theta_{\text{step}, p} - \theta_{\text{step}+1, p}\|_2 \leq \eta, \quad \forall p = 1, \dots, P \quad (20)$$

where $\theta_{\text{step},p}$ describes the position of particle p in the current iteration, the switch is made to the LASSO formulation

$$\underset{\mathbf{c} \in \mathbb{R}^d}{\operatorname{argmin}} \lambda \|\mathbf{c}\|_1 + \|\mathbf{A}\mathbf{c} - \mathbf{y}\|_2^2, \quad (21)$$

as described in Section 2.1, in order to find sparse random feature expansion for the computational model \mathcal{M} .

At the end, as soon as for dimension $k \in D$ optimal parameters θ and c have been found by the algorithm, we choose the composition of $\tau_{\theta^*}^{k^*}$ and f_{c^*} , which enjoy the best performance among all $k \in D$, as the target surrogate model.

The approach just described is summarized in Algorithm 2 with the following list parameters

- $P \in \mathbb{N}$ is the number of particles used for particle swarm optimization
- $p \in \mathbb{N}$ indicates particle index with $p \in \{1, 2, \dots, P\}$,
- $R \in \mathbb{N}$ is the number of random features in the LOS-SRF model f_c ,
- $D \subset \mathbb{N}$ is the set of potential intrinsic dimensions; D has to be either set by a user or has to be defined by a hyper-parameter search¹
- $n_{\text{iterations}} \in \mathbb{N}$ indicates the number of iterations to optimize over the projection into a lower dimensional space of dimension k .

Moreover, for the purpose of this paper, we use only different real-valued basis functions s.a. \cos , \sin and $ReLU$, reducing the required memory drastically, as the targets are solely real-valued as well.

As a general note, the PSO method was chosen in the proposed setup as it only requires a notion of the current best particle in the swarm as well as the position for which the current particle obtained its best value. In contrast to that, popular gradient-based methods require the gradient of a function at a certain position and sometimes even the second derivative. While those methods can be very efficient for large problems, as e.g. with Neural Networks containing thousand of parameters, they are not used in the current setting as they would interfere the modularity of the approach². Further, PSO is well suited for non-convex optimization problems as the interplay between the particles helps to prevent the optimizer getting stuck in small local minima, without having to adapt the update parameters during optimization, as it is for example done with learning rate schedulers for gradient descent. Thus, the choice of a metaheuristic for optimization is in alignment with the data-driven mentality of tackling surrogate modelling.

We note that our method is designed for a medium-sized number of observations. For a larger number N of observations we advise to use different dimensionality reduction method due to the complexity of $\mathcal{O}(N^2)$ arising in KPCA when computing the kernel matrix. Due to the modularity of our approach this can be done without interfering with other parts of the algorithm.

Finally, the remaining hyper-parameters required in Algorithm 2, such as R, q , can be found via hyper-parameter search. Common methods for hyper-parameter search include simple (but inefficient) grid-search, Bayesian search or random search [2, 3, 22, 40]. In the experiments of Section 4 we use Bayesian search.

¹A possible method to find a suitable set D could be a coarse to fine approach. Starting with a few samples from a set with larger steps should yield single dimensions $d \in \mathbb{N}$ around which more dimensions $d' = d \pm \{1, 2, \dots\}$ would be sampled to test.

²While presenting a specific instance of a self-supervised algorithm, the aim should be to ease changes in the algorithms used for dimensionality reduction and surrogate modelling.

Algorithm 2: Self-supervised Sparse Random Feature Expansion for Surrogate Modelling in High Dimensions

```
1 input :  $\{(\mathbf{x}, y)_i\}_{i=1}^N$  where  $\mathbf{x} \in \mathbb{R}^d$ ;  $n, n_{\text{iterations}}, q, R, P \in \mathbb{N}$ ;  $\eta \in \mathbb{R}$ ;  
           $D = \{k_1, \dots, k_m | k_j \in \mathbb{N}, k_j \ll d\}$   
2 output: optimal params. for dim. reduction, surrogate modelling and final loss:  $\theta^*, \mathbf{c}^*, \hat{\epsilon}^*$   
3 ridge-reg  $\leftarrow$  True  
4 ref. alg for KPCA and ref. alg. for PSO /* initialize parameters */  
5  $\theta_{0,p} \leftarrow \mathcal{U}[0, 1]^d$ , for all  $p \in \{1, 2, \dots, P\}$ ;  
  /* iterate over possible lower dimensions */  
6 for  $k$  in  $D$  :  
  /* optimize reduction and fit model in reduced dim. */  
7 for step in range( $n_{\text{iterations}}$ , for all  $p \in \{1, \dots, P\}$ ) :  
  /* reduce the data according to current parameters */  
8  $\mathbf{x}'_{\text{train}}, \mathbf{x}'_{\text{val}} \leftarrow \tau_{\theta_{\text{step},p}}^k(\mathbf{x}_{\text{train}}, \mathbf{x}_{\text{val}})$ ,  $\forall \mathbf{x}_{\text{train}} \in \mathcal{X}_{\text{train}}$  and  $\forall \mathbf{x}_{\text{val}} \in \mathcal{X}_{\text{val}}$ ;  
  /* fit sparse random feature expansion acc. to Algorithm 1 */  
9 if ridge-reg is True then  
10 |  $f^* \leftarrow \psi_{R, \text{Ridge}}^q(\mathbf{x}'_{\text{train}}, \mathbf{y}_{\text{train}})$   
11 else  
12 |  $f^* \leftarrow \psi_{R, \text{LASSO}}^q(\mathbf{x}'_{\text{train}}, \mathbf{y}_{\text{train}})$   
13 end  
14  $\hat{\mathbf{y}}_{\text{val}} \leftarrow f^*(\mathbf{x}'_{\text{val}})$ ;  
  /* compute loss between prediction from low dimensional data and ground truth */  
15  $\hat{\epsilon}_{\text{step},p} \leftarrow \mathcal{L}(\mathbf{y}_{\text{val}}, \hat{\mathbf{y}}_{\text{val}})$ ;  
  /* optimize reduction parameters, yielding config. for upcoming iteration */  
16  $\theta_{\text{step}+1,p} \leftarrow \mathbf{PSO}(\hat{\epsilon}_{\text{step}}, \{\theta_{\text{step},p}\}_{p=1}^P)$ ;  
17 if  $\|\theta_{\text{step},\cdot} - \theta_{\text{step}+1,\cdot}\|_2 \leq \eta$  then  
18 | ridge-reg  $\leftarrow$  False  
19 end
```

4 Numerical Experiments

In this section, we test the proposed surrogate modelling approach based on LOS-RFE numerically and compare the results to state-of-the-art surrogate modelling techniques based on neural networks and polynomial chaos expansions.

In our numerical tests, we use both realistic and synthetic data enabling a thorough analysis of the methods with regards to practical applicability as well as their ability of correctly conceiving the data structure. The first dataset is generated from a function given in a closed form, which allows to compare the output of the surrogate model with the ground-truth data generated by this function. The second dataset has been provided by the Department of Vehicle Safety at BMW Group and consists of crash test simulations for real world settings. Both of the presented datasets have unstructured inputs and scalar outputs. While the intrinsic dimension is approximately known for the first dataset, it is unknown for the second one.

In the following, the three different methods are compared against each other with respect to the empirical generalisation error (3), both on training and test data. We start by describing the data sets and also the methods in some more detail.

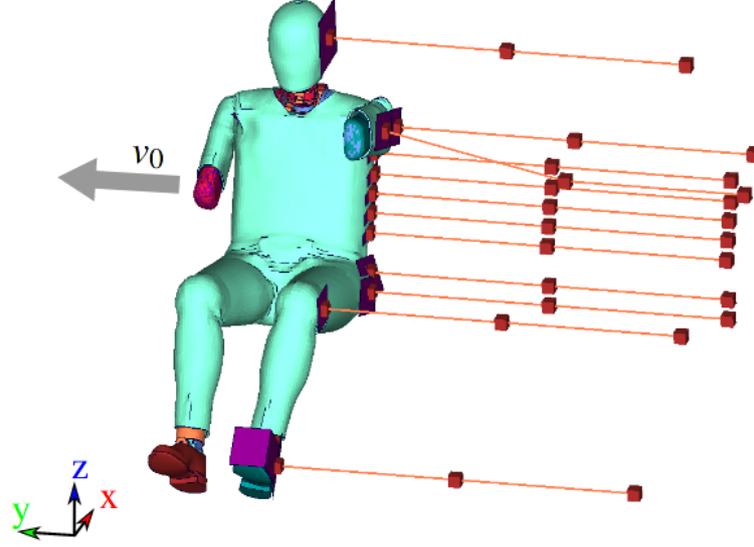


Figure 2 Snapshot of the data-generating simulation showing the full-scale dummy experiencing a lateral force in its initial position. Image taken from [16, 17].

4.1 Description of Datasets

First Dataset. The Sobol function, often also called the G-function [21, 30, 37], is common for benchmarking in the context of UQ. As the properties of the function, such as the shape or approximate intrinsic dimension, are known and can be configured via an appropriate choice of parameters, the function yields a rich class of test cases. In general, for some $K \in \mathbb{N}$, the Sobol function is defined as

$$f_{\text{Sobol}}(\mathbf{X}) = \prod_{i=1}^K \frac{|4\mathbf{X}_i - 2| + u_i}{1 + u_i}, \quad u_i \in \mathbb{R}_+ \quad (22)$$

where $\mathbf{X} = \{\mathbf{X}_1, \dots, \mathbf{X}_K\}$ are independent random input variables drawn from the uniform distribution $\mathcal{U}[0, 1]$, which in the setting of UQ hold the uncertainty of the system. The vector of parameters $\mathbf{u} = \{u_1, \dots, u_k\} \in \mathbb{R}_+^K$ controls the intrinsic dimension as it weights \mathbf{X}_j 's effect on the output value of f_{Sobol} . To be more precise, the effect of each input variable \mathbf{X}_j to the output $f_{\text{Sobol}}(\mathbf{X})$ is inversely proportional to the value of u_i : relatively small values of u_i result in a relative importance of \mathbf{X}_i for $f_{\text{Sobol}}(\mathbf{X})$. Finally, the parameter K defines the input dimension and allows to scale to extremely large dimensions.

For our first dataset, we aim to choose the vector \mathbf{u} to yield an intrinsic dimension of about 6 with an input dimension of $K = 20$. For this purpose, following [19–21], the vector \mathbf{u} is set to

$$\mathbf{u} = \{1, 2, 5, 20, 50, 100, 500, \dots, 500\}. \quad (23)$$

Thus, the first 6 variables can provide a compressed representation of the data, while a few extra dimensions might be valuable to capture the information provided by dimensions 7-20. Picking the exact number of intrinsic dimensions as target dimension would be called expert's knowledge. However, such information is usually not available, which is why we explore multiple potential intrinsic dimensions.

In our experiments, the input data $\mathbf{X}_i \in \mathbb{R}^d$ for (22) were sampled from $[0, 1]^d$ using the Sobol sampling schema [33], allowing for a better coverage of the domain compared to pure pseudo-random number generators.

Second Dataset. Here we use a dataset provided by the Department of Passive Safety at the BMW Group, which consists of crash test data obtained from computational simulations of a highly accurate Finite-Element (FE) model. As the evaluation of the FE model tends to run for several hours for one setting of input parameters, UQ directly on the data-generating model is infeasible. Therefore, the provided dataset places us in a realistic setting where a highly accurate but cheap to evaluate surrogate model is needed to approximate the original FE simulation.

Each BMW FE simulation is configured by a uniformly distributed random vector $\mathbf{X}_i \in \mathbb{R}^d$, for $d = 36$ and $i = 1, \dots, 600$, where the different components of the \mathbf{X}_i correspond to certain attributes such as car speed, impact angle etc. While the precise nature of this distribution cannot be revealed in this work due to intellectual property constraints [17], we stress that we have used the same data for all three methods. The simulations give rise to target values (data) \mathcal{Y} describing a head-injury value in the setting illustrated in Figure 2. In contrast to the first dataset, increasing the number of observations is a bottleneck as it requires highly expensive computations.

4.2 Description of Surrogate Modelling Techniques Used for Comparison

The first method chosen for comparison, described by Lataniotis et al. [21], is self-supervised similar to the method proposed in this paper. As a key building block in addition to dimensionality reduction, it uses a polynomial chaos expansion. More precisely, a genetic algorithm is used to find a suitable set of parameters for dimensionality reduction, and then the reduced data is iteratively approximated by a polynomial chaos expansion.

As the second method, we chose neural networks as comparison given that they inherently incorporate dimensionality reduction if one reduces the number of nodes layer by layer. Further, they have the ability to model complex relationships due to the nonlinear activation in each node. Neural networks have proven themselves to be the model of choice in several disciplines [26,27,36,39,41]. However, they tend to require a lot of data to optimize their parameters. In the setting of Uncertainty Quantification, it is usually the case that there are only a limited number of samples available for fitting a model, so the data scarce setting deserves particular attention.

The PCE-based approach was fitted according to the description given by Lataniotis et al. [21], i.e. iterating over the set of possible latent dimensions and optimizing the dimensionality reduction’s parameters with respect to the surrogate’s performance in that dimension. The neural networks were constructed by using fully connected layers with the ReLU activation function. The number of nodes per layer was chosen to minimize the training error using cross-validation as well as minimizing the generalisation gap, i.e. the difference between errors on training and validation data. The final network for the first dataset, for example, had the following number of nodes per hidden layer: $20 \rightarrow 10 \rightarrow 8 \rightarrow 4 \rightarrow 1$.

4.3 Numerical Comparison

4.3.1 Performance on the Sobol Dataset

To test different surrogate models from the theoretical prospective, we use the Sobol function f_{Sobol} with parameters \mathbf{u} as in (23) to generate the dataset consisting of $N_{\text{train}} = 800$ samples for training, $N_{\text{val}} = 1200$ for validation, and another $N_{\text{test}} = 2000$ for testing. Such a division of the training and test data was motivated by a more detailed error analysis of the models’ ability for generalization to unseen data points.

For these data sets, firstly, the above-described PCE-based surrogate model was trained and obtained according to [21] for the set of potential intrinsic dimensions $D = \{2, 4, 6, 8, 10, 12, 14\}$. For each latent dimension $k \in D$, the data was projected into the lower k -dimensional space, using Kernel PCA with an anisotropic Gaussian kernel with parameters learned in a self-supervised process with respect to the model’s performance during validation. The highest order of multivariate polynomials was chosen to be three as this showed to be the best trade-off between model complexity and ability to generalize in the given setting, see [21] for more details. The method’s performances over all intrinsic dimensions at the final stage of training process is illustrated in Figure 3 with the best performing dimensions $k = 8$. Such combination of dimensionality reduction and PCE, results in the empirical generalization errors (3), for the best-performing dimensions $\hat{\epsilon}_{d=8} = 0.0171^3$. A detailed view on the predictions vs. ground truth values

³While Lataniotis [21] reports a performance of $\hat{\epsilon} \approx 0.008$, it was not possible to reproduce those results. Also, the narrative of this subsection does not change with respect to the optimal performance of the PCE-based method. Taking into account the optimal performance from the reference, the neural network would then yield the lowest performance, and the LOS-RFE method would still perform best, as shown in Table 1 .

for the best performing dimension $k = 8$ on the validation set and the test set is depicted in Figure 4a and Figure 4d, respectively. As one can see, the PCE-based surrogate performs almost in the same manner for on both sets, which means that the model has learn fairly well the underlying model f_{Sobol} .

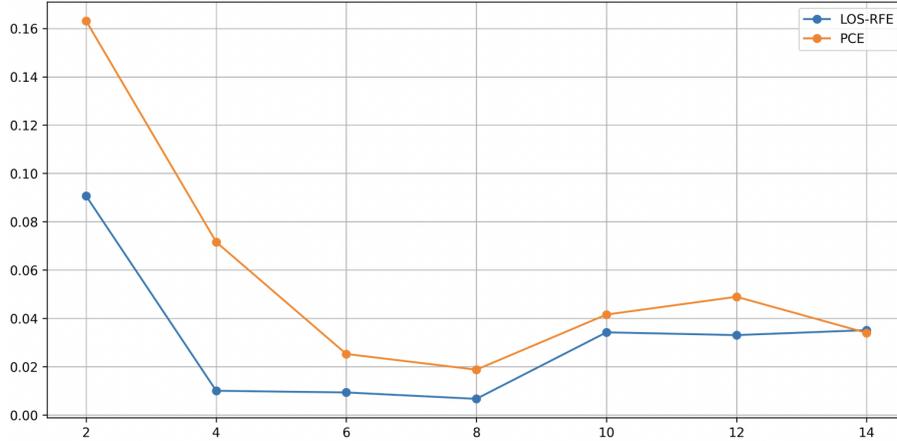


Figure 3 The figure shows the performance history of the PCE-based approach and the proposed LOS-RFE approach on the Sobol data per latent dimensions in D (x-axis) in terms of validation error (y-axis) in the final stage of the self-supervised learning process. As one can observe for latent dimensions $k \in \{6, 8\}$, the LOS-RFE approach and PCE-based approach shows similar performance, but LOS-RFE approach enjoys lower validation error than the PCE-based one. The neural network approach is not included here as it does not have an explicit dimensionality reduction step.

Secondly, a neural network is fit on the same training data and evaluated on the same hold-out test data. The neural network’s performance on the training data and in the test data is shown in Figure 4b and Figure 4e, respectively. As an advantage, the model does not need a self-supervised projection to yield a suitable representation of the data as it is capable of learning this mapping within its layers. The neural network shows a superiority over the PCE-based approach as the network’s error is roughly half of the PCE-based’s error.

In order to generate the LOS-RFE-based surrogate model, we choose sparsity order $q = 2$ and generate $R = 10000$ random \cos -features as described in Section 3. Although here we have even lower order of iterations between variables than for PCE, random features allow to generate a richer family of functions for surrogate modelling. This is also reflected in numerical experiments leading to a better model fit on average, see Figure 3. For the LSO-RFE model as for the PCE-based one, the latent dimension $k = 8$ was found to be optimal in the Sobol setting. However, the LSO-RFE method in combination with dimensionality reduction yields far lower errors than the PCE-based model as well as the neural network, both on validation and test data see Figure 4c and Figure 4f. This indicates that the LOS-RFE model was able to generalize to the unseen data without overfitting, leading to test error $\hat{\epsilon} = 0.0062$.

Not surprisingly, in the large data setting neural networks have a clear advantage over the two other methods. For example, if the data is enlarged to $N_{\text{train}} = 4800$, a neural network performs better the LOS-RFE, as it is shown on Figure 5. This confirms that our method is particularly competitive in the scarce data regime, which often occurs in the surrogate modelling setting.

4.3.2 Performance on the Crash Test Dataset

As mentioned above, the crash test dataset resulted from $N_{\text{total}} = 600$ executions of a crash test FE simulation [16, 17]. The simulation itself was mainly focused on the mannequin inside the car rather than the deformations happening to the chassis. Due to the emergence of the dataset, the latent dimension is not known - differently from the Sobol dataset.

For the comparison, the same three types of models as for the Sobol data are determined on the new data sets with $N_{\text{train}} = 150$ elements for training and $N_{\text{val}} = 150$ for validation, $N_{\text{test}} = 300$ for testing,

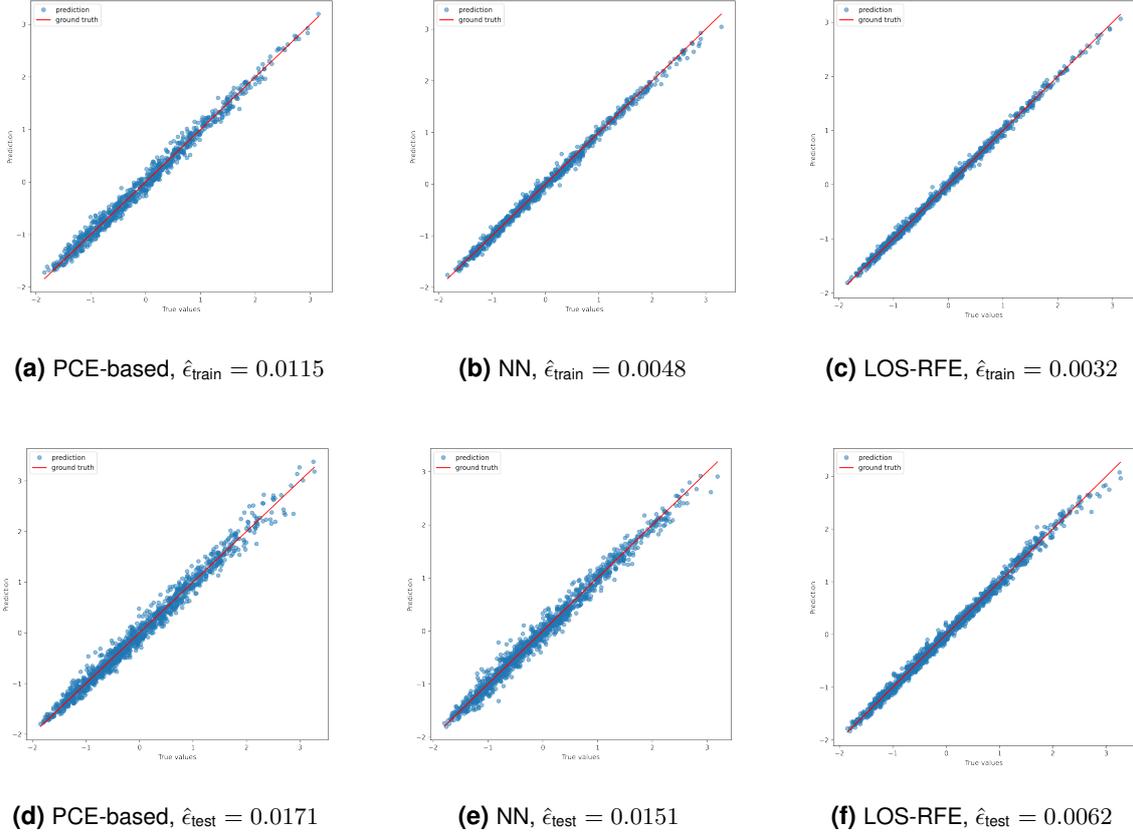


Figure 4 The performance of different surrogate modelling approaches on the training ($N_{\text{train}} = 800$ and $N_{\text{val}} = 1200$) as well as the test data ($N_{\text{test}} = 2000$) generated by the Sobol function parameterized by \mathbf{u} as in (23). Figure 4a-4d correspond to the PCE-based approach [21] with a best performing latent dimension of $k = 8$, figure 4b-4e depicts results for a neural network, and figure 4c-4f shows the performance of the LSO-RFE for $k = 8$ as well.

allowing to mimic a data-scarce scenario while holding enough observations back, to get a good error estimate.

The PCE-based model was examined first, showing a preferred latent dimension larger than 10, as Figure 6 shows. While the optimization seems to have plateaued for the $k \in \{12, 14, 18, 20\}$, the optimization for $k = 16$ might have reached the maximum number of iterations rather than a local minimum. As the results for the dimensions before and after $k = 16$ seem to be far lower, this can be assumed to be an outlier rather than a single dimension yielding a bad fit. The best performance was obtained for $k = 22$. The performance of the best PCE-configuration on the test data is shown in Figure 7d, achieving the generalization error $\hat{\epsilon} = 0.0.0468$ on the test set. Note, by looking at the training performance, the overfitting of the training data by the PCE-model can be identified. This is most likely the reason for the poor performance on the test data, compared to the good training performance.

Second, the neural network with the same configuration as before is fit on the crash dataset. In contrast to the first experiment on Sobol data, the training dataset is far smaller leading to difficulties generalizing to unseen data. As Figure 7b shows, the training of the network has converged without overfitting. However, while the model seems to be robust, the evaluation on the hold-out test dataset reveals difficulties to predict values especially at the lower end of the range, see Figure 7e. This time, the performance of the neural network decreased with a minimum error of $\hat{\epsilon} = 0.0539$ on the test data.

Finally, the proposed method LOS-RFE with $R = 10000$ random features and sparsity level $q = 2$ is determined using the training and validation sets of the crash test data. Similar to the PCE-based method, a favorable latent dimension resides in the last quarter in the range, taken into consideration. The dimension yielding the lowest error is $k = 14$. An overview of the model's performance with respect to the

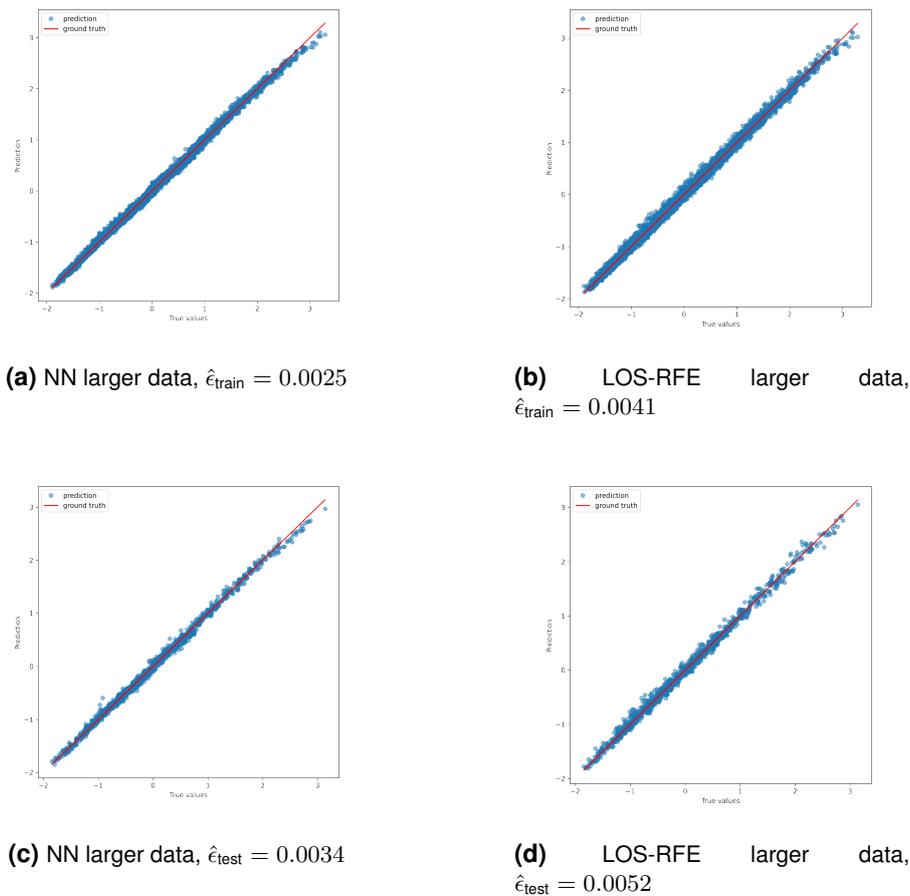


Figure 5 The performance of the neural network and the LOS-RFE on an increased training dataset $N_{\text{train}}^* = 4800$ compared to $N_{\text{train}} = 800$ as used to train the models shown in Figure 4. The neural network approach has a clear advantage in this large data regime. The PCE-based approach is not included here due to computational limitations arising from its longer run times.

different latent dimensions is given in Figure 6. The model resulting from the best configuration achieved an empirical generalization error of $\hat{\epsilon} = 0.0186$ on test data, which is again lower than the errors for the neural networks as well as the PCE-based method.

We would like to note that the proposed LOS-RFE can have a very high complexity by choosing a large number of random features. This can, similar to too large neural networks, lead to overfitting the data. To demolish the overfitting effect, special care has to be taken how the hyper-parameters R , the number of random features, sparsity q and λ , the scale of the regularizer, are chosen.

It shall also be noted, that the LOS-RFE method is more resource-demanding as it requires more memory when creating the large matrices, holding the random features. Also, due to the large coefficient vector, the time to find a solution to the LASSO problem usually takes more time than solving the minimization problem for the PCE variant. However, the resource consumption and compute time when fitting a PCE depends directly on the maximum order of the monomials used for the expansion.

However, a great amount of computation is still spent on finding a suitable lower dimensional representation of the data, which is shared by both methods, PCE-based and the proposed approach. Incorporating methods for estimating the latent dimension would decrease the runtime drastically, as the optimizer for the reduction method would only have to be run once.

Finally, the overall results on the test data for experiments carried in this section are summarized in Table 1.

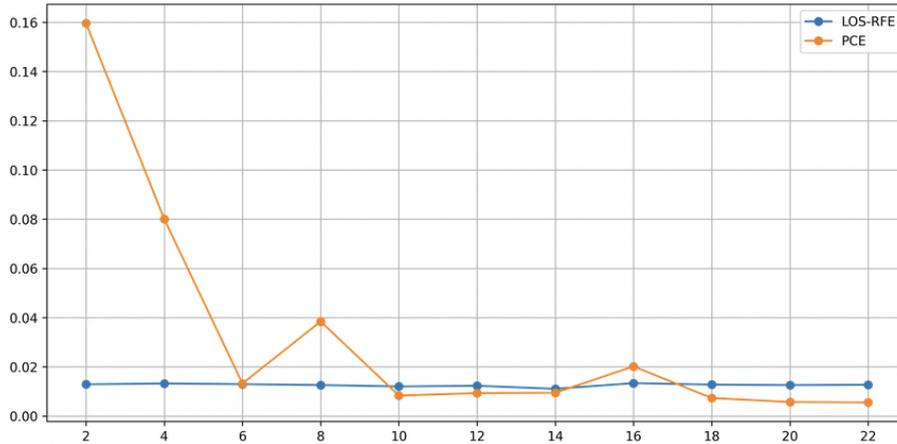


Figure 6 Similar to Figure 3, the figure shows the performances for the PCE-based method and the LOS-RFE on the crash data per latent dimensions in D (x-axis) in terms of validation error (y-axis) in the final stage of the self-supervised learning process. Regarding the PCE-based method, there is no a clear minimum but a trend towards higher dimensions. The ragged graph hints an unfinished optimization for $k = 16$. Treating the observation at $k = 16$ as outlier due to an unfinished optimization, the latent dimensions of $k = 22$ seems to provide a good representation of the data. Similar to the PCE-based approach, the favorable dimensions for our approach are larger than for the Sobol data. In this case, the lowest validation error for LOS-RFE was attained for $k = 14$.

Model	Test error $\hat{\epsilon}$		
	Sobol data	inc. Sobol data	Crash test data
PCE-based	0.0171	.	0.0468
Neural Net.	0.0151	0.0034	0.0539
LOS-SRF	0.0062	0.0052	0.0186

Table 1 Final results obtained from the experiments described above, where the fitted model was evaluated on test data, which was reduced to the most suitable latent dimension using the according θ .

4.4 Summary

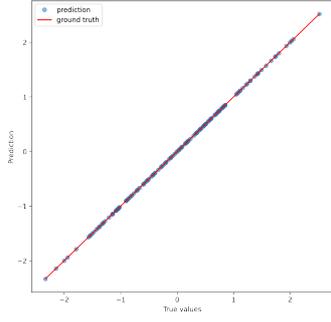
The examples above have shown the high accuracy of the proposed method of a self-supervised combination of dimensionality reduction and LOS-RFE. Also, the advantage of a self-supervised loop is observable by the streamlined process with no further interaction by the applicant than providing the input data. Compared to state-of-the-art techniques for surrogate modelling, such as PCE-based methods and neural networks, the proposed LOS-RFE method has shown clear advantages in the scarce data setting.

While the networks learn the parameters for all its nodes, the LOS-RFE approach learns to use only the most informative features. This is the reason why surrogate modelling based on random features has an advantage in data-scarce settings, compared to neural networks, which are data hungry as it was also mentioned in [13].

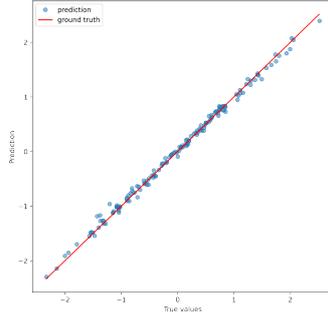
5 Conclusion

In the discipline of UQ, the goal is to measure the uncertainty within a simulation or computational model. As most such computational models are too complex to be run numerous times to construct a proper distribution of the output, conditioned on the input, surrogate models are used as a fast-to-evaluate replacement. This also implies that only few evaluations from the computational model are available to design such surrogates. For the surrogate models, it is essential to approximate the computational model with high accuracy.

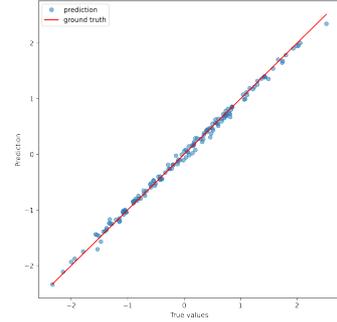
In this work, a method of construction of high-fidelity surrogates in data-scarce settings has been proposed. Namely, it has been shown that low-order sparse random feature expansion (LOS-RFE) can be



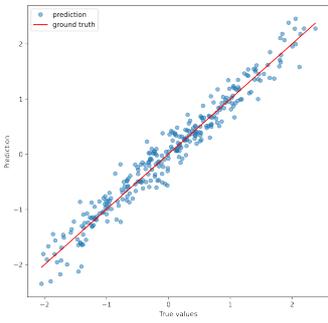
(a) PCE-based, $\hat{\epsilon}_{\text{train}} = 6.7397 \cdot 10^{-28}$



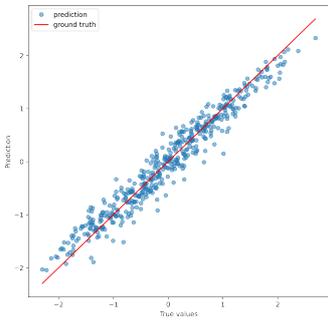
(b) NN, $\hat{\epsilon}_{\text{train}} = 0.0054$



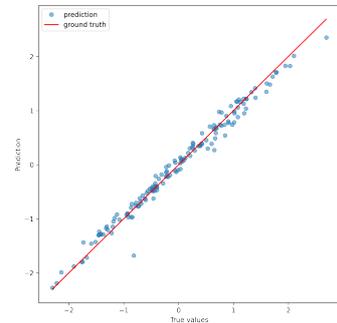
(c) LOS-RFE, $\hat{\epsilon}_{\text{train}} = 0.0036$



(d) PCE-based, $\hat{\epsilon}_{\text{test}} = 0.0468$



(e) NN, $\hat{\epsilon}_{\text{test}} = 0.0539$



(f) LOS-RFE, $\hat{\epsilon}_{\text{test}} = 0.0186$

Figure 7 The predictions depicted against the true values for the crash dataset for the PCE-based model (Figure 7d), LOS-RFE (Figure 7f) and the neural network (Figure 7e). The loss of the best PCE-based model on the test data, with a latent dimension of $k = 22$, is $\hat{\epsilon} = 0.0468$ on the test data. For the proposed LOS-RFE model, the latent dimension was chosen to be $k = 14$ with a loss of $\hat{\epsilon} = 0.0186$. LOS-RFE gives rise to the best generalization error among the three methods.

used to surrogate models of higher accuracy compared to state-of-the-art surrogate modelling techniques. Further, a self-supervised process including dimensionality reduction has been established with the aforementioned model by using the model's performance as feedback for the reduction method. This allows for optimizing over the parameters of the reduction as well as the surrogate's parameters with respect to data representation. In addition, such representation of the data in lower dimensions was learned to be best-suited for the LOS-RFE surrogate model, possibly increasing the surrogate model's performance.

As an object for future work, we see further analysis of different metaheuristics used to optimize the reductions parameter's as well as the reduction method itself. Regarding the reduction method, the here used Kernel PCA benefits from the small dataset and is thus very performant. In the case of larger datasets, KPCA will turn to be a bottleneck due to increased memory requirements. Finally, it could also be interesting to see how the method performs on larger datasets as the sparsity-of-effects principle is often stated for such as well.

Last but not least, it has to be mentioned that the method of LOS-RFE requires a certain amount of memory. This is due to the large matrices holding several features per observation. Thus, given a dataset of fixed size N , the memory requirement grows with the order of the random features $\mathcal{O}(R \cdot N)$. Therefore, when using LOS-RFE one should mainly focus on scarce data settings, which was also the main motivation for our work. While most machines allow running the application using multithreading or even multiprocessing, the available memory is the bottleneck. This is in contrast to other surrogate models such as PCE, which require more resources on the CPU side. However, the datasets in the described setting

of UQ are relatively small and thus memory size of today's high-end personal computers should suffice in most of the applications.

Concluding, we have shown in this work how a method such as LOS-RFE, inspired by compressive sensing, can be successfully used for surrogate modelling. When further coupled with dimensionality reduction in a self-supervised manner, the presented algorithm yields very good performance, with a minimal set of hyper-parameters to be determined.

Acknowledgments. FK and AV acknowledge support by the German Science Foundation (DFG) in the context of the Emmy Noether junior research group KR 4512/1-1 and the collaborative research center TR-109 as well as by the Munich Data Science Institute and Munich Center for Machine Learning. MH and JJ acknowledge support by the BMW Group in the context of the ProMotion program. The authors would like to thank Marco Rauscher for valuable discussions related to the numerical experiments.

References

- [1] Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., ... & Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76, 243-297.
- [2] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).
- [3] Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24.
- [4] Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4, No. 4, p. 738). New York: Springer.
- [5] Carrillo, J. A., Totzeck, C., & Vaes, U. (2021). Consensus-based optimization and ensemble Kalman inversion for global optimization problems with constraints. *arXiv preprint arXiv:2111.02970*.
- [6] Carrillo, J. A., Jin, S., Li, L., & Zhu, Y. (2021). A consensus-based global optimization method for high dimensional machine learning problems. *ESAIM: Control, Optimisation and Calculus of Variations*, 27, S5.
- [7] Carrillo, J. A., Choi, Y. P., Totzeck, C., & Tse, O. (2018). An analytical framework for consensus-based global optimization method. *Mathematical Models and Methods in Applied Sciences*, 28(06), 1037-1066.
- [8] Eichmueller, G., & Meywerk, M. (2020). On computer simulation of components for automotive crashworthiness: validation and uncertainty quantification for simple load cases. *International journal of crashworthiness*, 25(3), 263-275.
- [9] Fornasier, M., Klock, T., & Riedl, K. (2021). Consensus-based optimization methods converge globally in mean-field law. *arXiv preprint arXiv:2103.15130*.
- [10] Fornasier, M., Pareschi, L., Huang, H., & Sünnen, P. (2021). Consensus-Based Optimization on the Sphere: Convergence to Global Minimizers and Machine Learning. *J. Mach. Learn. Res.*, 22(237), 1-55.
- [11] Ghanem, R., Higdon, D., & Owhadi, H. (Eds.). (2017). *Handbook of uncertainty quantification* (Vol. 6). New York: Springer.

- [12] Gojcic, Z., Litany, O., Wieser, A., Guibas, L. J., & Birdal, T. (2021). Weakly supervised learning of rigid 3D scene flow. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 5692-5703).
- [13] Hashemi, A., Schaeffer, H., Shi, R., Topcu, U., Tran, G., & Ward, R. (2023). Generalization bounds for sparse random feature expansions. *Applied and Computational Harmonic Analysis*, 62, 310-330.
- [14] Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). The elements of statistical learning: data mining, inference, and prediction (Vol. 2, pp. 1-758). New York: springer.
- [15] Hur, J., & Roth, S. (2020). Self-supervised monocular scene flow estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7396-7405).
- [16] Jehle, J. S., Lange, V. A., & Gerdts, M. (2022). Proposing an Uncertainty Management Framework to Implement the Evidence Theory for Vehicle Crash Applications. *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, 8(2).
- [17] Jehle, J. S. Uncertainty Management Framework for Automotive Crash Applications (Doctoral dissertation, Dissertation, Neubiberg, Universität der Bundeswehr München, 2022)
- [18] Jehle, J. S., Lange, V. A., & Gerdts, M. (2021). Enabling the evidence theory through non-intrusive parametric model order reduction for crash simulations. REC2021.
- [19] Kersaudy, P., Sudret, B., Varsier, N., Picon, O., & Wiart, J. (2015). A new surrogate modeling technique combining Kriging and polynomial chaos expansions—Application to uncertainty analysis in computational dosimetry. *Journal of Computational Physics*, 286, 103-117.
- [20] Konakli, K., & Sudret, B. (2016). Global sensitivity analysis using low-rank tensor approximations. *Reliability Engineering & System Safety*, 156, 64-83.
- [21] Lataniotis, C., Marelli, S., & Sudret, B. (2020). Extending classical surrogate modeling to high dimensions through supervised dimensionality reduction: a data-driven approach. *International Journal for Uncertainty Quantification*, 10(1).
- [22] Liashchynskiy, P., & Liashchynskiy, P. (2019). Grid search, random search, genetic algorithm: a big comparison for NAS. arXiv preprint arXiv:1912.06059.
- [23] Lüthen, N., Marelli, S., & Sudret, B. (2021). Sparse polynomial chaos expansions: Literature survey and benchmark. *SIAM/ASA Journal on Uncertainty Quantification*, 9(2), 593-649.
- [24] Mika, S., Schölkopf, B., Smola, A., Müller, K. R., Scholz, M., & Rätsch, G. (1998). Kernel PCA and de-noising in feature spaces. *Advances in neural information processing systems*, 11.
- [25] Mittal, H., Okorn, B., & Held, D. (2020). Just go with the flow: Self-supervised scene flow estimation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 11177-11185).
- [26] Papadopoulos, V., Soimiris, G., Giovanis, D. G., & Papadrakakis, M. (2018). A neural network-based surrogate model for carbon nanotubes with geometric nonlinearities. *Computer Methods in Applied Mechanics and Engineering*, 328, 411-430.
- [27] Pfrommer, J., Zimmerling, C., Liu, J., Kärger, L., Henning, F., & Beyerer, J. (2018). Optimisation of manufacturing process parameters using deep neural networks as surrogate models. *Procedia CiRP*, 72, 426-431.
- [28] Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm intelligence*, 1(1), 33-57.

- [29] Rahimi, A., & Recht, B. (2007). Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20.
- [30] Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., ... & Tarantola, S. (2008). *Global sensitivity analysis: the primer*. John Wiley & Sons.
- [31] Schölkopf, B., Smola, A., & Müller, K. R. (1997, October). Kernel principal component analysis. In *International conference on artificial neural networks* (pp. 583-588). Springer, Berlin, Heidelberg.
- [32] Smith, R. C. (2013). *Uncertainty quantification: theory, implementation, and applications* (Vol. 12). Siam.
- [33] Sobol', I. Y. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4), 784-802.
- [34] Soize, C. (2017). *Uncertainty quantification*. Springer International Publishing AG.
- [35] Sullivan, T. J. (2015). *Introduction to uncertainty quantification* (Vol. 63). Springer.
- [36] Sun, G., & Wang, S. (2019). A review of the artificial neural network surrogate modeling in aerodynamic design. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 233(16), 5863-5872.
- [37] Surjanovic, S., & Bingham, D. (2021) *Virtual Library of Simulation Experiments: Test Functions and Datasets*. Retrieved October 16, 2021, from <http://www.sfu.ca/~ssurjano>
- [38] Totzeck, C. (2022). Trends in consensus-based optimization. In *Active Particles, Volume 3* (pp. 201-226). Birkhäuser, Cham.
- [39] Tripathy, R. K., & Billionis, I. (2018). Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of computational physics*, 375, 565-588.
- [40] Wu, J., Chen, X. Y., Zhang, H., Xiong, L. D., Lei, H., & Deng, S. H. (2019). Hyperparameter optimization for machine learning models based on Bayesian optimization. *Journal of Electronic Science and Technology*, 17(1), 26-40.
- [41] Zhang, X., Xie, F., Ji, T., Zhu, Z., & Zheng, Y. (2021). Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 373, 113485.