Simplex-stochastic collocation method with improved scalability

W.N. Edeling [a,b,*], R.P. Dwight [b], P. Cinnella [a]

[a] Arts et Métiers ParisTech, DynFluid laboratory, 151 Boulevard de l'Hopital, 75013 Paris, France
[b] Delft University of Technology, Faculty of Aerospace Engineering, Kluyverweg 2, Delft, The Netherlands

ARTICLE INFO

Article history:
Received 25 November 2014
Received in revised form 14 August 2015
Accepted 15 December 2015
Available online 18 December 2015

Keywords:
Simplex-stochastic collocation method
Uncertainty quantification
Surrogate model
High-dimensional model reduction techniques
Uniform simplex sampling

ABSTRACT

The Simplex-Stochastic Collocation (SSC) method is a robust tool used to propagate uncertain input distributions through a computer code. However, it becomes prohibitively expensive for problems with dimensions higher than 5. The main purpose of this paper is to identify bottlenecks, and to improve upon this bad scalability. In order to do so, we propose an alternative interpolation stencil technique based upon the Set-Covering problem, and we integrate the SSC method in the High-Dimensional Model-Reduction framework. In addition, we address the issue of ill-conditioned sample matrices, and we present an analytical map to facilitate uniformly-distributed simplex sampling.

1. Introduction

In order to make reliable predictions of a physical system using a computer code it is necessary to understand what effect the uncertainties in the inputs have on the output Quantity of Interest (QoI). Attempts to do so while keeping the computational cost low can be found in [30,22,14], which rely on Smolyak-type sparse-grid stochastic-collocation methods. Whereas traditional collocation-type methods [4,18] use full-tensor product formulas to extend a set of one-dimensional nodes to higher dimensions, sparse-grid methods build a sparse interpolant using a constrained linear combination of one-dimensional nodes. This can provide a grid with a potential reduction in support nodes of several orders of magnitude.

Although computationally more efficient than full-tensor approaches, sparse-grid methods add points equally in all dimensions, irrespective of whether the response surface is locally smooth or discontinuous. Therefore further gains can be achieved through adaptive stochastic-collocation schemes which have been developed in recent years. For instance Ma et al. [19] proposed an Adaptive Sparse-Grid (ASG) collocation method where the probabilistic space is spanned by linear finite-element basis functions. During each iteration the probability space is refined locally through an error measure based upon the hierarchical surplus, defined as the difference between the interpolation of the previous level and the newly added code sample. Although the space is refined locally, unphysical oscillations can still occur due to the lack of sample points on some of the edges of the local support of the basis functions. The Simplex-Stochastic Collocation (SSC) method of Witteveen et al. [38] circumvents this problem by discretizing the domain into simplices by means of a Delaunay triangulation, and enforcing the so-called Local-Extremum Conserving limiter to suppress unphysical oscillations. Furthermore, it computes Essentially Non-Oscillatory (ENO) stencils [39] of the sample points which allows for high-order polynomial interpolation.

* Corresponding author at: Delft University of Technology, Faculty of Aerospace Engineering, Kluyverweg 2, Delft, The Netherlands.
E-mail addresses: W.N.Edeling@tudelft.nl (W.N. Edeling), R.P.Dwight@tudelft.nl (R.P. Dwight), P.Cinnella@ensam.eu (P. Cinnella).

http://dx.doi.org/10.1016/j.jcp.2015.12.034

Further features include randomized sampling, the ability to deal with non-hypercube probability spaces and it can be extended to perform interpolation with sub-cell resolution [40].

Besides schemes which efficiently sample the probabilistic space, there are other means of dealing with the curse of dimensionality, i.e. the exponential increase in the amount of computational cost with increasing dimension. Physical systems often have a low effective dimension, meaning that only a few coefficients are influential, and only low-order correlations between these input coefficients have a significant impact on the output. To capitalize on this behavior, High-Dimensional Model-Reduction (HDMR) techniques can be applied [27]. In HDMR a d-dimensional function is exactly represented as a hierarchical sum of 2^d component functions of increasing dimensionality. In the case of low effective dimension, the d-dimensional function can be approximated well by a truncated expansion. The main idea is to solve several low-dimensional sub-problems instead of one high-dimensional one, which greatly reduces the computational cost. A well-known member of this class of decompositions is the analysis of variance (ANOVA) decomposition. In [12], Foo et al. successfully coupled their Multi-Element Probabilistic Collocation method [13] with a HDMR–ANOVA decomposition to problems with up to 600 dimensions. Although they achieved a significant reduction in the computational cost compared to approaches with full-tensor products, the number of points required to sufficiently sample these extremely high-dimensional spaces is still of the order of millions or even more. Furthermore, in [20] Ma et al. coupled their previously mentioned ASG method with the so-called cut-HDMR technique of [27]. This approach is computationally more efficient than ANOVA–HDMR, as it does not require the evaluation of multi-dimensional integrals. Besides truncating the cut-HDMR expansion at a certain order, the authors of [20] also made their approach dimension adaptive through weights which identify the dimensions that contribute the most to the mean of the stochastic problem. They applied this approach to several easily-evaluated test problems of very high dimensionality, i.e. up to a stochastic dimension of 500. Again, their results represent a significant reduction in the required number of code samples for a certain error level compared to full tensor grids, but in absolute terms the number of samples is still very high.

In our view, the interest of a surrogate modelling technique is to apply it to some complex simulation code which is too expensive to integrate by simple Monte Carlo techniques. In this setting it will be intractable to sufficiently sample a probabilistic space of dimension $\mathcal{O}(100)$, regardless of the surrogate modelling technique used. Therefore we will investigate means to efficiently create surrogate models with a moderate number of uncertain input parameters, under the assumption that the QoI is the output of a computationally expensive code. Of course, the term “moderate dimensionality” is system dependent. So to clarify, we consider the dimensionality moderate when the number of inputs parameters falls within the range of 5 to 10. Many problems of engineering interest are simulated using codes with similar dimensionality, e.g. turbulence models [11,15], groundwater models [28] or thermodynamic equations-of-state [8,21].

Due to its adaptivity, high polynomial order and Runge-phenomenon free interpolation, the SSC method requires a relatively low number of code samples to attain a given convergence level for problems with moderate dimensionality. For example, the SSC method has been applied in [3] to optimization under uncertainty of a Formula 1 tire brake intake. After a one-dimensional perturbation analysis, 3 variables were selected for analysis. Furthermore, in [9,10] the SSC method is efficiently coupled with Downhill-Simplex optimization in a setting for robust design optimization. Several example problems are considered, but again the maximum number of uncertain variables is 3. We found that the SSC method itself, without considering the cost of sampling the code, can become prohibitively expensive when considering 5 uncertain parameters. Furthermore, due to excessive memory requirements we were unable to create surrogate models of dimension 6 or higher. In [38] the authors also note that the cost of the Delaunay triangulation becomes prohibitively large from a dimensionality of 5 onward. They suggested to replace the Delaunay triangulation with a scheme where simplices are formed by selecting the nearest points from randomly placed Monte Carlo (MC) samples. Using this approach the SSC method was applied to a continuous QoI with 15 uncertain parameters. However, in this case individual simplices can overlap and there is no guarantee that the entire probabilistic domain is covered.

This paper is aimed at identifying bottlenecks, and reducing the computational burden of the SSC method, while retaining the Delaunay triangulation. We investigate two separate techniques. First we propose the use of new alternative stencils based upon the set-covering problem [16]. The main idea is to use the fast increase in number of simplex elements with polynomial order to create a small set of stencils which covers the entire probabilistic domain. Afterwards, only this set is used for interpolation. This allows for a more efficient implementation of the SSC method. Our results show that these stencils are capable of reducing the computational cost up to 8 dimensions. We furthermore present a new method for avoiding problems with the ill-conditioning of the sample matrix, and we provide a new formula for placing uniformly distributed samples in a simplex of arbitrary dimension. Secondly, inspired by the work of Ma et al. [20], we integrate the SSC method into the cut-HDMR framework. This approach combines the advantages of SSC and cut-HDMR, and avoids the disadvantages related to the ASG method such as linear interpolation and the possible occurrence of the Runge phenomenon. Unlike the authors of [20], we apply our method to a complex computer code for which obtaining samples is expensive. For both proposed techniques we perform a detailed analysis of the error and we give a discussion on computational cost as a function of the number of input parameters.

This paper is organized as follows: in Section 2 we present the baseline SSC method as developed by Witteveen et al. Next, in Section 3 we describe the Set-Covering stencils, our method for avoiding singular sample matrices and the analytic mapping for uniformly-distributed simplex sampling. The following section describes the cut-HDMR approach and in Section 5 we present the obtained results and the discussion. Finally, we give our conclusions in Section 6.

2. Simplex-stochastic collocation method

In this section we give a general outline of the Simplex-Stochastic Collocation (SSC) method as developed by Witteveen et al. For a more detailed description we refer to [41,38,39,37].

2.1. General outline baseline SSC method

The SSC method was introduced as a non-intrusive method intended for robust and efficient propagation of uncertainty through computer models. It differs from traditional collocation methods, e.g. [4,18], in two main ways. First, for multi-dimensional problems it employs the Delaunay triangulation to discretize the probability space into simplex elements, rather than relying on the more common tensor product of one-dimensional abscissas [24]. Using a multi-element technique has the advantage that mesh adaptation can be performed, such that only regions of interest are refined. Secondly, the SSC method is capable of handling non-hypercube probability spaces [38].

The response surface of the QoI $u(\boldsymbol{\xi})$ is denoted by $w(\boldsymbol{\xi})$ and it is constructed using a set of n_s samples from the computational model, $\mathbf{v} = \{v_1, \cdots, v_{n_s}\}$. Here, $\boldsymbol{\xi}$ is a vector of d random input parameters $\boldsymbol{\xi}(\boldsymbol{\omega}) = (\xi_1(\omega_1), \cdots, \xi_d(\omega_d)) \in \Xi \subset \mathbb{R}^d$. Furthermore, we define Ξ to be the parameter space and $\boldsymbol{\omega} = (\omega_1, \cdots, \omega_d) \in \Omega \subset \mathbb{R}^d$ is a vector containing realizations of the probability space $(\Omega, \mathcal{F}, P)$ with $\mathcal{F}$ the σ-algebra of events and P a probability measure. The variables in $\boldsymbol{\omega}$ are distributed uniformly as $\mathcal{U}(0,1)$, and the input parameters can have any distribution $f_{\boldsymbol{\xi}}$, although for the sake of simplicity we restrict ourselves in this paper to $f_{\boldsymbol{\xi}} = \mathcal{U}(\xi_i^a, \xi_i^b)$, with the bounds ξ_i^a and ξ_i^b. We perform all our analysis on the unit hypercube $K_d := [0,1]^d$, and we use a linear map in order to go from K_d to the parameter domain $\boldsymbol{\xi}$. Our goal is to propagate $f_{\boldsymbol{\xi}}$ through the computational model in order to assess the effect of $f_{\boldsymbol{\xi}}$ on the m-th statistical moment of $u(\boldsymbol{\xi}(\boldsymbol{\omega}))$, i.e. we wish to compute

$$\mu_u^{(m)} := \int_{\Xi} u(\boldsymbol{\xi})^m f_{\boldsymbol{\xi}}(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}. \tag{1}$$

Note that u can also be a function of a physical coordinate $\mathbf{x}$ or other deterministic explanatory variables, but for brevity we omit $\mathbf{x}$ from the notation. Since the SSC method discretizes the parameter space Ξ into n_e disjoint simplices $\Xi = \Xi_1 \cup \cdots \cup \Xi_{n_e}$, the mth statistical moment is approximated as

$$\mu_u^{(m)} = \int_{\Xi} u(\boldsymbol{\xi})^m f_{\boldsymbol{\xi}}(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \approx \mu_w^{(m)} := \sum_{j=1}^{n_e} \int_{\Xi_j} w_j(\boldsymbol{\xi})^m f_{\boldsymbol{\xi}}(\boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\xi}. \tag{2}$$

Here, w_j is a local polynomial function of order p_j associated with the j-th simplex Ξ_j such that

$$w(\boldsymbol{\xi}) = w_j(\boldsymbol{\xi}), \quad \text{for } \boldsymbol{\xi} \in \Xi_j, \tag{3}$$

and the interpolation condition requires

$$w_j(\boldsymbol{\xi}_{k_{j,l}}) = v_{k_{j,l}}. \tag{4}$$

The subscript $k_{j,l} \in \{1, \cdots, n_s\}$ is a global index which refers to the k-th added computational sample, while j refers to a certain simplex element. Furthermore, $l = 0, \cdots, N_j$ is a local index used to count the number of samples from $\mathbf{v}$ involved in the construction of w_j. The $N_j + 1$ number of points needed for d-dimensional interpolation of order p_j is given by

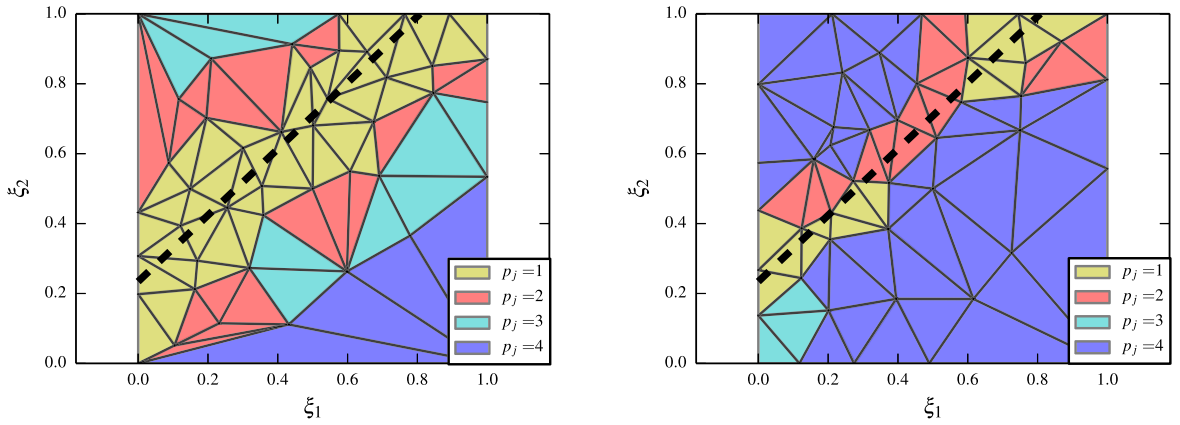
$$N_j + 1 = \frac{(d + p_j)!}{d! p_j!}, \tag{5}$$

and the local interpolation function w_j itself is given by the expansion

$$w_j(\boldsymbol{\xi}) = \sum_{l=0}^{N_j} c_{j,l} \boldsymbol{\Psi}_{j,l}(\boldsymbol{\xi}). \tag{6}$$

The choice of basis polynomials $\boldsymbol{\Psi}_{j,l}$, and the determination of the interpolation coefficients $c_{j,l}$ is dealt with in Section 2.2.1. Note that for a given d, the maximum allowable order p_j based on the number of samples n_s can be inferred from (5). The particular choice of p_j will depend on the smoothness of the response, with the objective of avoiding the Runge phenomenon.

Which $N_j + 1$ points are used in (6) is determined by the interpolation stencil S_j. The stencil can be constructed based on the nearest-neighbor principle [38]. In this case the first $d + 1$ points are the vertices $\{\boldsymbol{\xi}_{k_{j,0}}, \cdots, \boldsymbol{\xi}_{k_{j,d}}\}$ of the simplex Ξ_j,



(a) Delaunay triangulation color coded by p_j with nearest neighbor stencils.

(b) Delaunay triangulation color coded by p_j with ENO stencils.

Fig. 1. Delaunay triangulation for two stencil types with a discontinuity running along the dotted line. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

which would suffice for $p_j = 1$. For higher degree interpolation, neighboring points $\boldsymbol{\xi}_k$ are added based on their proximity to the center of simplex Ξ_j, i.e. based on their ranking according to

$$\|\boldsymbol{\xi}_k - \boldsymbol{\xi}_{\text{center},j}\|_2, \tag{7}$$

where those $\boldsymbol{\xi}_k$ of the current simplex Ξ_j are excluded. The simplex center $\boldsymbol{\xi}_{\text{center},j}$ is defined as

$$\boldsymbol{\xi}_{\text{center},j} := \frac{1}{d+1}\sum_{l=0}^{d} \boldsymbol{\xi}_{k_{j,l}}. \tag{8}$$

The nearest neighbor stencil (7) leads to a p_j distribution that can increase quite slowly when moving away from a discontinuity. An example of this behavior can be found in Fig. 1(a), which shows the Delaunay triangulation with a discontinuity running through the domain. An alternative to nearest-neighbor stencils are stencils created according to the Essentially Non-Oscillatory (ENO) principle [39]. The idea behind ENO stencils is to have higher degree interpolation stencils up to a thin layer of simplices containing the discontinuity. For a given simplex Ξ_j, its ENO stencil is created by locating all the nearest-neighbor stencils that contain Ξ_j, and subsequently selecting the one with the highest p_j. This leads to a Delaunay triangulation which captures the discontinuity better than its nearest-neighbor counterpart. An example can be found in Fig. 1(b). Unless otherwise stated, for all subsequent baseline SSC surrogate models we will use ENO-type stencils.

The initial samples, at least in the case of hypercube probability spaces, are located at the 2^d corners of the hypercube K_d. Furthermore, one sample is placed in the middle of the hypercube. Next, the initial grid is adaptively refined based on an appropriate error measure. This error measure can either be based on the hierarchical surplus between the response surface of the previous iteration and new a sample v_k, or on the geometrical properties of the simplices. The latter option is more reliable in multiple stochastic dimensions as it is not based on the hierarchical surplus in a single discrete point [37]. The geometrical refinement measure is given by

$$\bar{e}_j := \bar{\Omega}_j \bar{\Xi}_j^{2O_j}. \tag{9}$$

It contains the probability and the volume of simplex Ξ_j, i.e.

$$\bar{\Omega}_j = \int_{\Xi_j} f_{\boldsymbol{\xi}}(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi} \quad \text{and} \quad \bar{\Xi}_j = \int_{\Xi_j} \mathrm{d}\boldsymbol{\xi}, \tag{10}$$

where $\bar{\Xi} = \sum_{j=1}^{n_e} \bar{\Xi}_j$. The probability $\bar{\Omega}_j$ can be computed by Monte-Carlo sampling and $\bar{\Xi}_j$ via the relation

$$\bar{\Xi}_j = \frac{1}{d!}|\det(D)|, \quad D = \left[\boldsymbol{\xi}_{k_{j,1}} - \boldsymbol{\xi}_{k_{j,0}} \quad \boldsymbol{\xi}_{k_{j,2}} - \boldsymbol{\xi}_{k_{j,0}} \quad \cdots \quad \boldsymbol{\xi}_{k_{j,d+1}} - \boldsymbol{\xi}_{k_{j,0}}\right] \in \mathbb{R}^{d\times d}. \tag{11}$$

Finally, the order of convergence O_j is given by [37]

$$O_j = \frac{p_j + 1}{d}. \tag{12}$$

The simplex with the highest $\bar{e}_j$ is selected for refinement. To ensure a good spread of the sample points, only randomly-selected points inside a simplex sub-element Ξ_{sub_j} are allowed. The vertices of this sub-element are defined as

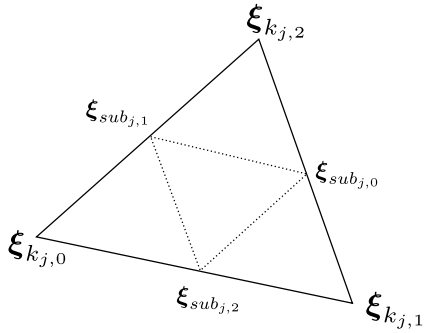


Fig. 2. The sub simplex (dotted line) of a two-dimensional simplex. Upon refinement one sample is placed at a randomly selected location inside the sub simplex in order to avoid clustering of points.

$$\boldsymbol{\xi}_{sub_{j,l}} := \frac{1}{d} \sum_{\substack{l^*=0 \\ l^* \neq l}}^{d} \boldsymbol{\xi}_{k_{j,l^*}}, \tag{13}$$

see Fig. 2 for a two-dimensional example. In order to place random samples uniformly in an arbitrary simplex we derive an analytical map $M_d : K_{n_\xi} \rightarrow \Xi_j$, see Section 2.2.2. The SSC algorithm can be parallelized by selecting the $N < n_e$ simplices with the N largest $\bar{e}_j$ for refinement.

Note that by using (13) only simplex interiors will be refined (see again Fig. 2), and the boundaries of the hypercube will never be sampled outside the initial 2^d points. As a consequence, discontinuities that cross a hypercube border cannot be captured accurately at that border. To avoid this, we do not use (13) if a simplex element located at the boundary is selected for refinement. Instead, we randomly place samples at the longest simplex edge which is at the boundary, $\pm 10\%$ from the edge center.

Note that (9) is probabilistically weighted through Ω_j and that it assigns high $\bar{e}_j$ to those simplices with low p_j since in general $\bar{\Xi}_j \ll 1$. Effectively this means that (9) is a solution-dependent refinement measure which refines simplices near discontinuities since the SSC method automatically reduces the polynomial order if a stencil S_j crosses a discontinuity. It achieves this by enforcing the so-called Local Extremum Conserving (LEC) limiter to all simplices Ξ_j in all S_j. The LEC condition is given by

$$\min_{\boldsymbol{\xi} \in \Xi_j} w_j(\boldsymbol{\xi}) = \min \mathbf{v}_j \wedge \max_{\boldsymbol{\xi} \in \Xi_j} w_j(\boldsymbol{\xi}) = \max \mathbf{v}_j, \tag{14}$$

where $\mathbf{v}_j = \{v_{k_{j,0}}, \cdots, v_{k_{j,d}}\}$ are the samples at the vertices of Ξ_j. If w_j violates (14) in one of its $\Xi_j \in S_j$, the polynomial order p_j of that stencil is reduced, usually by 1. Since polynomial overshoots occur when trying to interpolate a discontinuity, p_j is reduced the most in discontinuous regions. Interpolating a function on a simplex with $p_j = 1$ and $\mathbf{v}_j$ located at its vertices always satisfies (14) [37]. This ensures that $w(\boldsymbol{\xi})$ is able to represent discontinuities without suffering from the Runge phenomenon. In practice, (14) is enforced for all Ξ_j in all S_j via random sampling of the w_j. If for a given w_j (14) is violated for any of the randomly placed samples $\boldsymbol{\xi}_j$, the polynomial order of the corresponding stencil is reduced. Again, how we sample the d-dimensional simplices is described in Section 2.2.2. The computational cost of enforcing (14) is investigated in Section 5.

The procedure of enforcing the LEC condition, computing a refinement measure and subsequently refining certain selected simplices is either repeated for a maximum of I iterations, $n_{s_{max}}$ samples or halted when a sufficient level of accuracy is obtained. This level of accuracy can be estimated through an error measure based upon the hierarchical surplus [37]. As mentioned, this is the difference between the response surface w_j and the newly added code sample $v_{k_{j,ref}}$ at the refinement location $\boldsymbol{\xi}_{k_{j,ref}}$, i.e.

$$\epsilon(\boldsymbol{\xi}_{k_{j,ref}}) := w_j(\boldsymbol{\xi}_{k_{j,ref}}) - v_{k_{j,ref}}. \tag{15}$$

This is a point estimate of the error, located at what will be a vertex in the new refined Delaunay grid. To assign error estimates to the simplices rather than to vertices, the error $\tilde{\epsilon}_j$ is introduced. For each Ξ_j, $\tilde{\epsilon}_j$ is simply the absolute value of (15) of its most recently added vertex $\boldsymbol{\xi}_{k^*}$. Since adding vertices will change the Delaunay discretization we relate the error of the previous simplex to the new one via

$$\hat{\epsilon}_j \approx \tilde{\epsilon}_j \left(\frac{\bar{\Xi}_j}{\bar{\Xi}_{k^*,ref}} \right)^{\mathcal{O}_j} \tag{16}$$

[38]. The ratio $\bar{\Xi}_i / \bar{\Xi}_{k^*,ref}$ represents the change in volume from its old size $\bar{\Xi}_{k^*,ref}$, i.e. the volume of the simplex which was refined by $\boldsymbol{\xi}_{k*}$, to its new size $\bar{\Xi}_j$. Finally, each individual $\hat{\epsilon}_j$ is combined in a global error estimate via the following root mean square (RMS) error norm

$$\hat{\epsilon}_{rms} = \sqrt{\sum_{j=1}^{n_e} \Omega_j \hat{\epsilon}_j^2}. \tag{17}$$

The complete baseline SSC method is given in pseudo code in Appendix A.

2.2. Improvements on the baseline SSC method

Before discussing our new stencil selection technique in Section 3.1, we introduce two improvements to the baseline SSC method not discussed in the original references [41,38,39,37].

2.2.1. Poised sample sequence

The authors of [35] write (6) in matrix form, constraining $\mathbf{\Psi}_{j,l}$ to the class of monomials, and subsequently solve explicitly for the coefficients $c_{j,l}$. They note that although they had no difficulties in solving this system, the matrix could have a high condition number. This poses no real problem for $d \leq 3$, but for higher dimensions it can become problematic. To cope with this we impose an additional condition on the construction of the stencils S_j such that the interpolation problem is *poised*, meaning that the sample matrix is non-singular [23]. In the following discussion we drop the subscript j until further notice to make the notation more concise.

To construct the interpolating monomials, let us define the collection consisting of $N+1$ d-dimensional multi-indices $\bar{i} := (i_1, \cdots, i_k, \cdots, i_d)$, where for all $\bar{i}$ we have $|\bar{i}| := i_1 + \cdots + i_d \leq p_j$ and each i_k is an integer between 0 and d. Furthermore, for a given vertex $\boldsymbol{\xi}_l = (\xi_{1,l}, \cdots, \xi_{d,l})$ belonging to stencil S, let us define its $\bar{i}$-th power to be $\boldsymbol{\xi}_l^{\odot\bar{i}} := \xi_{1,l}^{i_1} \times \cdots \times \xi_{d,l}^{i_d}$. The sample matrix Ψ, a multi-dimensional Vandermonde matrix, can then be written as

$$\Psi = \begin{bmatrix} \boldsymbol{\xi}_0^{\odot\bar{0}} & \boldsymbol{\xi}_0^{\odot\bar{1}} & \cdots & \boldsymbol{\xi}_0^{\odot\bar{N}} \\ \boldsymbol{\xi}_1^{\odot\bar{0}} & \boldsymbol{\xi}_1^{\odot\bar{1}} & \cdots & \boldsymbol{\xi}_1^{\odot\bar{N}} \\ \vdots & & \vdots & \vdots \\ \boldsymbol{\xi}_N^{\odot\bar{0}} & \boldsymbol{\xi}_N^{\odot\bar{1}} & \cdots & \boldsymbol{\xi}_N^{\odot\bar{N}} \end{bmatrix} \in \mathbb{R}^{(N+1)\times(N+1)}. \tag{18}$$

As an example, the l-th row of (18) in lexicographical order for $p_j = 2$ will look like $[\,1 \quad \xi_{1,l} \quad \xi_{2,l} \quad \xi_{1,l}^2 \quad \xi_{1,l}\xi_{2,l} \quad \xi_{2,l}^2\,]$. The coefficients c_l in (6) can now be obtained by solving the system

$$\begin{bmatrix} \boldsymbol{\xi}_0^{\odot\bar{0}} & \boldsymbol{\xi}_0^{\odot\bar{1}} & \cdots & \boldsymbol{\xi}_0^{\odot\bar{N}} \\ \boldsymbol{\xi}_1^{\odot\bar{0}} & \boldsymbol{\xi}_1^{\odot\bar{1}} & \cdots & \boldsymbol{\xi}_1^{\odot\bar{N}} \\ \vdots & & \vdots & \vdots \\ \boldsymbol{\xi}_N^{\odot\bar{0}} & \boldsymbol{\xi}_N^{\odot\bar{1}} & \cdots & \boldsymbol{\xi}_N^{\odot\bar{N}} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_N \end{bmatrix}, \tag{19}$$

where $\{v_0, \cdots, v_N\}$ are the code samples belonging to stencil S. Once the c_l are known, we can interpolate to any point $\boldsymbol{\xi}$ in the domain spanned by S.

We define $\Delta \equiv \det(\Psi)$, and note that the whole approach hinges on the well-poisedness condition $\Delta \neq 0$. This condition is relatively easy violated during the SSC procedure in higher dimensions. For instance, if for $d = 4$ we determine the maximum allowable p using (5) on the initial Delaunay grid we obtain $p_{max} = 2$. However, many stencils in this case will have $\Delta = 0$. Also situations where a stencil has too many vertices located in the same plane (e.g. due to edge refinement at the boundary of K_d), can lead to a zero determinant of (18). Thus, for $d > 1$ the poisedness condition $\Delta \neq 0$ imposes constraints on the geometrical distribution of the $\boldsymbol{\xi}_l$. From [23,7] we know

Theorem 1. *The $N+1$ vertices $\boldsymbol{\xi}_0, \cdots, \boldsymbol{\xi}_N \in \mathbb{R}^d$ are polynomially poised iff they are not a subset of any algebraic hypersurface of degree $\leq p$.*

An algebraic hypersurface in $\mathbb{R}^d$ is a $d-1$ dimensional surface embedded in a d-dimensional space constrained to satisfy an equation $f(\xi_1, \cdots, \xi_d) = 0$. The degree is given by f.

The authors of [7] devised a Geometric Characterization (GC) condition which allows us to detect if a set of vertices is poised, i.e.:

Definition 1. GC condition: For each $\boldsymbol{\xi}_l$ in a set of $N+1$ vertices in $\mathbb{R}^d$, there exists p distinct hyperplanes $G_{1,l}, \cdots, G_{p,l}$ such that i) $\boldsymbol{\xi}_l$ does not lie on any of these planes, and ii) all other $\boldsymbol{\xi}_k$, $k = \{0, \cdots, N\} \backslash \{l\}$ lie on at least one of these hyperplanes. Mathematically speaking i) and ii) amount to

$$\boldsymbol{\xi}_i \in \bigcup_{k=0}^{p} G_{k,l} \quad \text{if } i \neq l, \quad \forall i = 1, 2, \cdots, N. \tag{20}$$

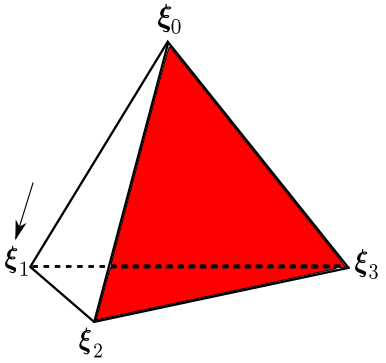


Fig. 3. When selecting node ξ_1, there exists one ($p = 1$) plane which contains all other points except ξ_1. This is true for all nodes in the simplex.

Theorem 2. *Let $\{\xi_l\}$ be a set of $N+1$ vertices in $\mathbb{R}^d$. If $\{\xi_l\}$ satisfies the GC condition, then $\{\xi_l\}$ admits a unique interpolation of degree $\leq p$ [7].*

Due to its geometrical configuration, a single simplex Ξ_j in $\mathbb{R}^d$ always satisfies the GC condition for $p = 1$, see Fig. 3 for a three-dimensional example. For a given vertex $\xi_l \in \Xi_j$, we always have one hyperplane containing the face of the simplex made up by all vertices except ξ_l. Thus, Theorem 2 implies that simplex Ξ_j will lead to a Ψ with $\Delta \neq 0$ and $p_j = 1$.

We use this result to obtain a set of well-poised ENO stencils $S_j \; \forall j = 1, \cdots, n_e$, in a way that is similar to the construction of the ENO-stencils as described in [39]. Only if during the enforcement of the LEC condition (14) we encounter a stencil S_j for which $\Delta = 0$, we collect a set of k candidate nearest-neighbor stencils $\{S_{j,i}\}_{i=1}^{k}$ which all contain simplex Ξ_j. We then select the S_j which has the highest p_j and $\Delta \neq 0$. In the worst case scenario we get $p_j = 1$, where S_j contains only the vertices of Ξ_j itself and for which $\Delta \neq 0$ is guaranteed by Theorem 2. If we have multiple S_j with $p_j > 1$ which satisfy these conditions, we select the one with the smallest average Euclidean distance to the cell-center $\xi_{\text{center},j}$.

2.2.2. Simplex sampling

Simplices are refined by randomly placing a point inside the sub-simplex (13). Also, to randomly sample the w_j during the LEC enforcement we need to place random points inside d-dimensional simplices. If we would like to uniformly sample a line section with the end points $[\xi_0, \xi_1]$ we would use the mapping

$$M_1 = \xi_0 + r_1(\xi_1 - \xi_0), \tag{21}$$

where $r_1 \sim \mathcal{U}[0,1]$. Generating points inside a triangle can be done with

$$M_2 = \xi_0 + r_2^{1/2}(\xi_1 - \xi_0) + r_2^{1/2} r_1 (\xi_2 - \xi_1) \tag{22}$$

which maps points $\{r_1, r_2\}$ inside the unit square K_2 to points inside a triangle described by the vertices $\{\xi_0, \xi_1, \xi_2\}$ [33]. The working principle of (22) is shown in Fig. 4(a). The parameter $r_2^{1/2}$ selects a line segment parallel to the edge $[\xi_0, \xi_1]$, while r_1 selects a point along the chosen line segment. The exponent $1/2$ ensures that uniformly distributed points in the square yield uniformly distributed points in the triangle. This can be shown by considering the length of the chosen line segment, which increases linearly when $r_2^{1/2}$ moves from ξ_0 to ξ_1. Since we require a uniform distribution of points, and considering $r_1 \sim \mathcal{U}[0,1]$, the pdf of $r_2^{1/2}$ should be linear as well. If we have the random variable $X = r^{1/\tau}$ with $r \sim \mathcal{U}[0,1]$ and $\tau \in \mathbb{N}_{>0}$, we find the cumulative distribution function (cdf) of X as

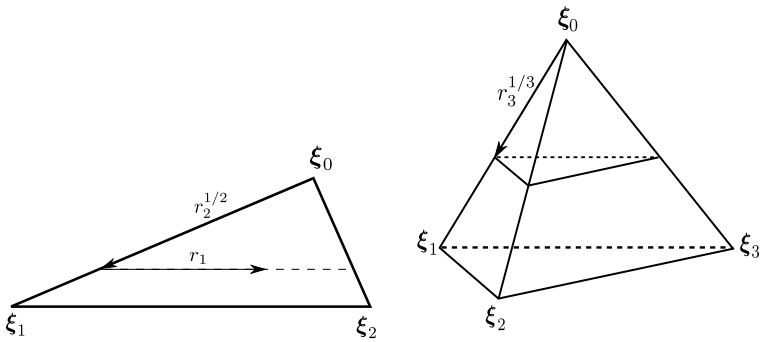
$$F_X(x) = \mathbb{P}(X \leq x) = \mathbb{P}\left(r^{1/\tau} \leq x\right) = \mathbb{P}\left(r \leq x^\tau\right) = x^\tau. \tag{23}$$

And thus we have the pdf $f_X(x) = \mathrm{d}F_X/\mathrm{d}x = \tau x^{\tau-1} \sim \text{Beta}(\tau, 1)$. Therefore, in order to have a linear pdf for $r^{1/\tau}$, we must set $\tau = 2$.

It is suggested in [33] that (22) can be extended to higher dimensions, although no specific formulas are given. Hence, we use the same principle to select uniformly distributed points inside a tetrahedron, see Fig. 4(b). Here, the parameter $r_3^{1/3}$ selects a triangle parallel to the base of the tetrahedron. From there we use $r_2^{1/2}$ and r_1 as before to select a point on this triangle. The exponent $1/3$ again ensures that the point distribution will be uniform. Note that the area of the selected triangles increases quadratically as $r_3^{1/3}$ moves from ξ_0 to ξ_1. Hence, it must be distributed as $\text{Beta}(3,1)$. We can now derive an expression for M_3 using the geometrical similarities between the base triangle and the selected parallel triangle, which gives us

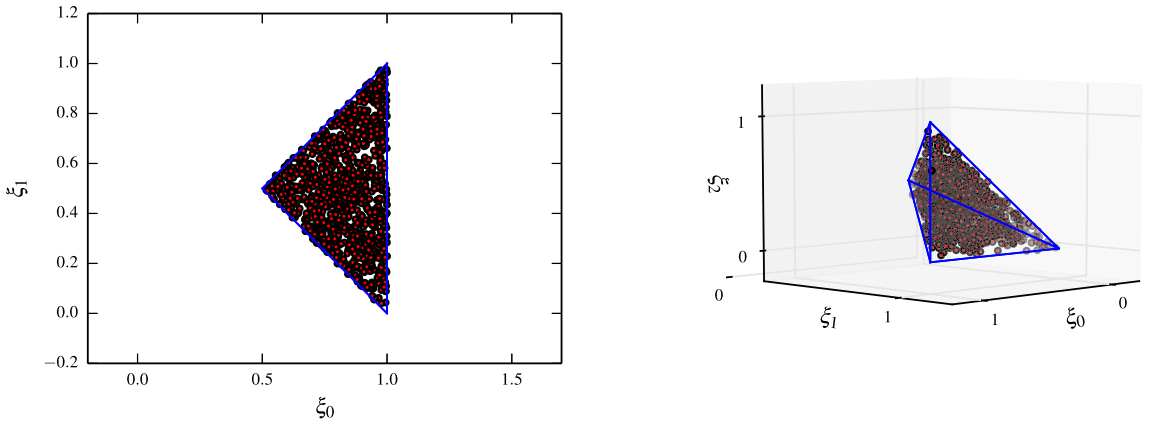
$$M_3 = \xi_0 + r_3^{1/3}(\xi_1 - \xi_0) + r_3^{1/3} r_2^{1/2} (\xi_2 - \xi_1) + r_3^{1/3} r_2^{1/2} r_1 (\xi_3 - \xi_2). \tag{24}$$

When comparing (21), (22) and (24) we see a pattern emerge which suggests that the map from a d-dimensional hypercube to a d-dimensional simplex with vertices $\{\xi_0, \cdots, \xi_d\}$ in $\mathbb{R}^d$ and uniform point distribution is



(a) Triangle. (b) Tetrahedron.

Fig. 4. Selecting a point inside a triangle and tetrahedron.



(a) Samples mapped from the unit square K_2 to a triangle. (b) Samples mapped from the unit cube K_3 to a tetrahedron.

Fig. 5. An example of the map (25) for $d = 2, 3$ and 1000 samples.

$$M_d = \boldsymbol{\xi}_0 + \sum_{i=1}^{d} \prod_{j=1}^{i} r_{d-j+1}^{\frac{1}{d-j+1}} (\boldsymbol{\xi}_i - \boldsymbol{\xi}_{i-1}), \tag{25}$$

where again the r_q are distributed as $\mathcal{U}[0, 1]$. Our proof that (25) produces uniformly distributed samples in the simplex can be found in Appendix D.

To numerically test (25) in 2 and 3 dimensions we can simply plot samples points, an example of which can be found in Fig. 5. We have performed similar tests up to 8 dimensions.

3. SSC Set-Covering method

In this section we describe alternative interpolation stencils, which results in a computational speed up in higher dimensions.

3.1. Set covering stencils

Section 5 will show that the enforcement of the LEC condition can become computationally expensive for high d and p_j. This is especially true for smooth response surfaces of the QoI. For many stencils of our discontinuous problem, the LEC condition is violated and p_j is reduced which in turn significantly lowers the total required number of surrogate model evaluations (n_w) needed to check (14). This does not happen very often when the response surface is smooth. As a consequence of the exponential nature of n_w (see Section 5.1.1), we also see an exponential increase in the computation

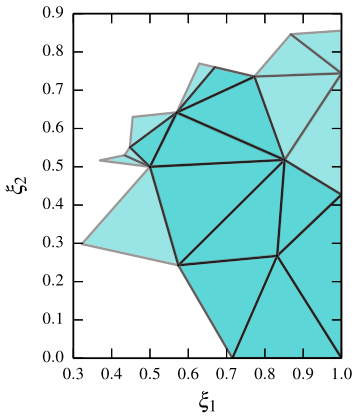


Fig. 6. Two stencils which overlap each other. The dark simplices are shared by both stencils.

time needed to construct the surrogate model. Note that this increase is due to the SSC procedure, and thus is additional to the time needed to sample the computer code.

However, the problem lies not only with the exponential increase of n_w, but also in the extremely large overlap of the stencils S_j. Note that the baseline SSC method enforces the LEC condition *for all simplices* Ξ_j *in all stencils* S_j. Hence, in each simplex Ξ_j, w_j is evaluated the same number of times as Ξ_j appears in all stencils S_j. For a two-dimensional example see Fig. 6. There are two stencils, denote them S_r and S_q, associated to two different simplices Ξ_r and Ξ_q. The dark colored simplices are the ones which appear in both stencils. Thus, when the LEC condition is checked for both stencils, w_r but also w_q is evaluated in the dark simplices. Moreover, since there are n_e stencils, the overlap will be large, and many different w_j will be evaluated in the same simplex element. This is no bottleneck for problems of low-dimensionality, but if the dimension increases this overlap will make the LEC condition very costly to enforce, see Section 5.1.1.

We propose an alternative technique for problems with higher d, using Set-Covering (SC) stencils based on the well-know set-covering problem [16], stated as follows in SSC terminology:

Set Covering problem. *Let* $X_j = \{\Xi_{j,0}, \cdots, \Xi_{j,K}\}$ *be the set of all simplices that are inside the domain spanned by the vertices of stencil* S_j*. Then, given the set* $\mathcal{X} = \{X_1, \cdots, X_{n_e}\}$*, and the set of all simplices* $\mathcal{U} = \{\Xi_1, \cdots, \Xi_{n_e}\}$*, find the smallest subset* $\mathcal{C} \subseteq \mathcal{X}$ *that covers* $\mathcal{U}$*, i.e. for which*

$$\mathcal{U} \subseteq \bigcup_{X_j \in \mathcal{C}} X_j$$

holds.

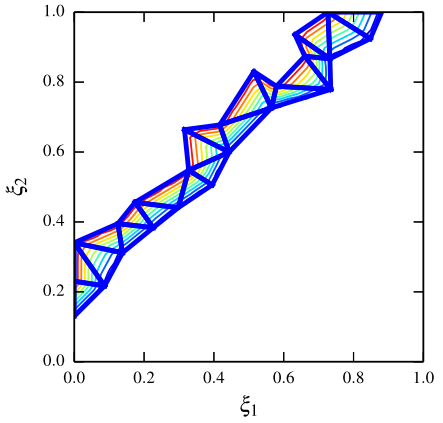
It is shown in [16] that the set-covering problem is NP-complete, and thus no fast solution is known. We could approximate $\mathcal{C}$ by the greedy algorithm, which at each step simply selects the X_j with the largest number of uncovered simplices. We then would have to check the LEC condition for all stencils in $\mathcal{S}_{sc}$, defined as the set of S_j corresponding to the $X_j \in \mathcal{C}$. For (high-dimensional) problems with a maximum polynomial order $p_{max} > 1$, the number of stencils in $\mathcal{S}_{sc}$ will be significantly lower than n_e. However, this approach would still require to construct all $X_j \in \mathcal{X}$. Also, many of the X_j could potentially cross a discontinuity, leading to a violation of the LEC condition and the subsequent reduction in size of X_j. When this happens the SC property of $\mathcal{C}$ can no longer be guaranteed. Thus, an iterative approach would be necessary which runs until $\mathcal{S}_{sc}$ satisfies both the SC and LEC property.

For reasons of computational efficiency, we want to avoid this iterative approach as much as possible, and thus not rely completely on the LEC condition to turn a set of nearest-neighbor stencils into a set of ENO stencils. Hence we will use the information contained in $\mathbf{v}$ regarding the discontinuity location to create a small set of SC stencils that also resemble ENO stencils, i.e. which do not cross a discontinuity. We will denote these stencils as SCENO stencils. Although more sophisticated approaches are available [40], for reasons of simplicity we identify the Ξ_j through which the discontinuity runs by simply imposing a threshold v_t on the maximum jump observed in $\mathbf{v}$ at each simplex. Then, the set of discontinuous simplices can be defined as

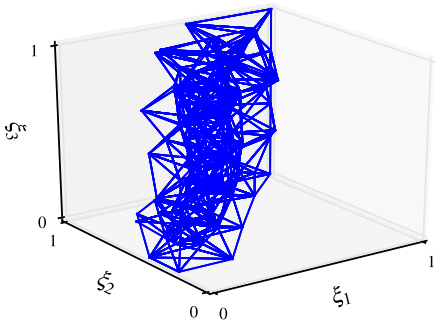
$$\mathcal{D} = \{\Xi_j \mid |\max \mathbf{v}_{k_{j,l}} - \min \mathbf{v}_{k_{j,l}}| \geq v_t, \;\; l = 0, \cdots, d, \;\; j = 1, \cdots, n_e\} \tag{26}$$

For the nozzle flow case we set the threshold value to $v_t = 1.0$. A two and three-dimensional visualization of the $\Xi_j \in \mathcal{D}$ can be found in Fig. 7. We furthermore redefine the set $\mathcal{C}$ as the set containing all the simplices Ξ_j that are currently covered by a stencil S_j, rather than the true smallest subset $\mathcal{C} \subseteq \mathcal{X}$ of the SC problem.

The general outline for constructing the SCENO stencils is now as follows. For the $\Xi_j \in \mathcal{D}$ we set $p_j = 1$ and $\mathcal{C} = \mathcal{C} \cup \mathcal{D}$, i.e. we add all discontinuous simplices to the set of covered simplices. Next, we specify the initial simplex Ξ_j^* as the simplex



(a) 2D discontinuous simplices plus contour lines from the QoI. Notice that the QoI is essentially flat outside the discontinuous simplices for this particular problem.



(b) 3D discontinuous simplices.

Fig. 7. Discontinuous simplices identified by (26).

from the set $\mathcal{U} \setminus \mathcal{C}$ with the largest volume. For the selected simplex we grow its stencil by adding neighboring Ξ_j which are not covered yet, i.e. which are not in $\mathcal{C}$. This will yield a set $\mathcal{C}$ where every simplex appears only once, i.e. a set with zero overlap. Note that to relax this condition one can easily allow for the addition of neighboring simplices which are in $\mathcal{C} \setminus \mathcal{D}$. In either case we continue growing the stencil until there are no more available neighbors or until S_j is large enough to allow interpolation of order p_{max}. We then move to the next Ξ_j^* and repeat until $\mathcal{C}$ covers the entire probabilistic space $\mathcal{U}$. For a graphical representation of the stencil construction we refer to Fig. 8. It is important to note that our main goal is to find a set $\mathcal{C}$ with a cardinality $|\mathcal{C}|$ significantly less that n_e, which is an easier task than approximating the true minimal $\mathcal{C}$ of the SC problem as closely as possible. In Appendix B the algorithm for constructing the SCENO stencils is displayed in pseudo code.

This approach assures that we have a relatively small set $\mathcal{S}_{sc}$ for which: i) $|\mathcal{S}_{sc}| \ll n_e$, ii) that not all n_e nearest-neighbor $X_i \in \mathcal{X}$ need be calculated, iii) that no X_j crosses a discontinuity, and iv) the $\Xi_j \in \mathcal{D}$ are interpolated linearly. The result is that the number of times the LEC condition needs to be checked is reduced significantly. Only for those S_j associated to the $X_j \in \mathcal{C} \setminus \mathcal{D}$ it is still necessary to check for interpolation overshoots, since the $\Xi_j \in \mathcal{D}$ are guaranteed to be LEC due to their linear interpolation. The property of SCENO stencils mentioned under iii) also means that the number of times the LEC condition is violated is reduced, although not always to zero due to reasons of ill conditioning of the sample matrix (18). This is especially true for high d. An approach as described in Section 2.2.1 would render some of the advantages mentioned under i)–iv) void. Reducing p_j for ill-conditioned stencils will increase the cardinality of $\mathcal{S}_{sc}$, and all $X_j \in \mathcal{X}$ should be calculated in order to look for alternative stencils. Instead we directly solve an ill-conditioned system (19) in the non-null subspace of the solution as described in [17]. This method utilizes Gauss–Jordan elimination with complete pivoting to identify the null subspace of a singular matrix Ψ, i.e. $\Psi_{null}\mathbf{c}_{null} = 0$. This partitions the linear system as depicted below,

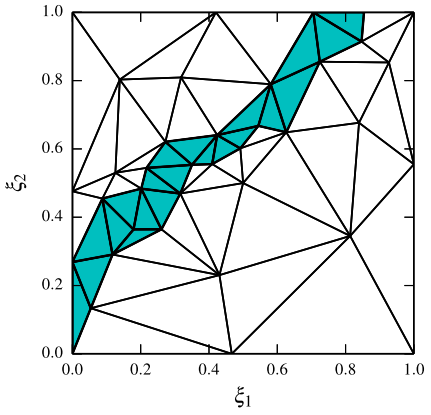
$$\begin{bmatrix} \Psi_{range} & \cdots \\ \cdots & \Psi_{null} \end{bmatrix} \begin{bmatrix} \mathbf{c}_{range} \\ \mathbf{c}_{null} \end{bmatrix} = \begin{bmatrix} \mathbf{v}' \\ \vdots \end{bmatrix}, \tag{27}$$

where $\Psi_{range}\mathbf{c}_{range} = \mathbf{v}'$ is the non-null subspace in which we can obtain accurate solutions. In the case of an ill-conditioned system, the null subspace is closely approximated by a space where the pivots ψ_{ii} are very small but not exactly equal to zero. The start of this 'near-null' subspace is identified by the first pivot ψ_{ii} for which the condition $|\psi_{ii}/\eta_c| < \epsilon$ holds, where η_c is the largest pivot of Ψ and ϵ is a very small parameter, which we set equal to 10^{-14}. In both the ill-conditioned and singular case the detrimental effect of Ψ_{null} on the solution is eliminated by a so-called zeroing operation, which basically replaces Ψ_{null} by an identity matrix of equal dimension and sets $\mathbf{c}_{null} = 0$. Thus, effectively speaking those coefficients $c_{j,l}$ which have been overwhelmed by round-off error are automatically cut out of the expansion (6). In our experiments we found that the dimension of Ψ_{null}, i.e. the nullity of Ψ, is small compared to the dimension of the full Ψ, see Table 1 for some typical examples at $d = 6$.

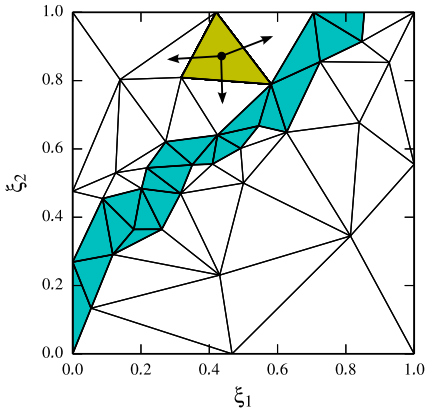
If the system of equations is well-posed, the algorithm amounts to regular Gauss–Jordan elimination with complete pivoting. In any case, the quality of the response surface is checked via the LEC condition.

4. High-Dimensional Model-Reduction

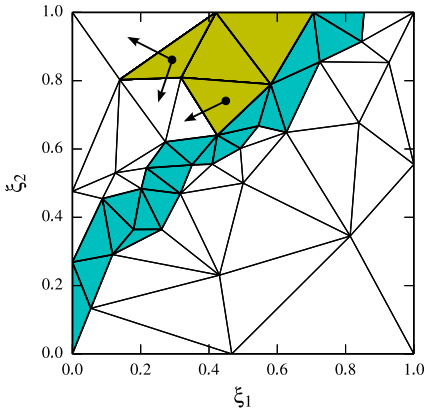
As will be shown in Section 5, the use of SCENO stencils makes the SSC method more computationally efficient within the range of dimensions where a Delaunay triangulation can be made. For problems of higher dimensionality a different



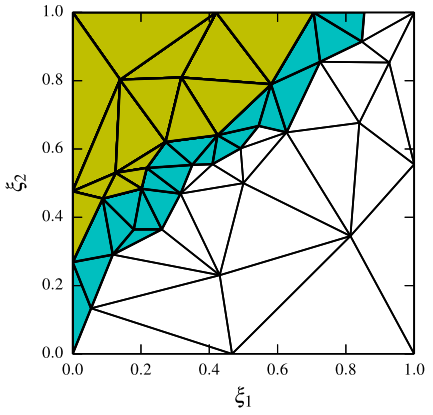
(a) Identify discontinuous simplices $\mathcal{D}$ and set their interpolation order to 1. Add all $\Xi_j \in \mathcal{D}$ to $\mathcal{C}$.



(b) Select uncovered simplex Ξ_j^* and add its neighbours to $\mathcal{C}$. Add corresponding nodes to S_j.



(c) Add uncovered neighbours of previously added simplices to $\mathcal{C}$. Add corresponding nodes to S_j.



(d) Iterate until the stencil size is large enough or until no more uncovered neighbours are available. Goto (b).

Fig. 8. A two-dimensional example of the SC stencil construction.

Table 1
Examples of ill-conditioned systems. We show the dimension d, the polynomial order of the stencil, the nullity and condition number of the sample matrix Ψ, and finally the condition number of the non-null Ψ_{range}.

d	p_j	Dimension Ψ	Nullity Ψ	Cond. Ψ	Cond. Ψ_{range}
6	2	28×28	1	1.36e+17	9.39e+3
6	3	82×82	1	1.31e+17	2.40e+4
6	3	84×84	2	2.85e+17	2.66e+3

approach is required. In physical systems it is often found that only a few parameters are influential, and only low-order correlations between the input parameters have a significant impact on the output. To capitalize on this behavior, High-Dimensional Model-Reduction techniques can be applied, see the references of Rabitz and Aliş [27,26]. Our QoI is represented by a d-dimensional function $f(\boldsymbol{\xi}, \mathbf{x})$ defined on the hypercube K_d, where $\mathbf{x}$ is a possible physical coordinate which we will again omit from the notation for the sake of brevity. Then, the HDMR expansion is an exact and finite hierarchical expansion of component functions of increasing dimension, given by

$$f(\boldsymbol{\xi}) = f_0 + \sum_i f_i(\xi_i) + \sum_{i_1<i_2} f_{i_1 i_2}(\xi_{i_1}, \xi_{i_2}) + \cdots + \sum_{i_1<\cdots<i_l} f_{i_1 \cdots i_l}(\xi_{i_1}, \cdots, \xi_{i_l}) + \cdots + f_{1\cdots d}(\xi_{i_1}, \cdots, \xi_{i_d}). \tag{28}$$

Here, the $i_1, \cdots, i_d$ are integers satisfying $1 \le i_1 < i_2 < \cdots < i_d \le d$. The zero-th order component function f_0 is a constant and represents the mean effect. The first-order function $f_i(\xi_i)$ is a univariate function, generally nonlinear, which represents the effect of independently varying input parameter ξ_i. Higher order functions represent the cooperative effects of increasing number of variables acting together on the output. If high-order correlations are weak, the physical system $f(\boldsymbol{\xi})$ can be ef-

ficiently represented by a truncated L-th order expansion, where $L < d$. This a called a problem with low *effective dimension*, which occurs frequently in problems of physical nature [12]. Thus, the general idea is to solve multiple low-dimensional subproblems in place of a single high-dimensional one. The resultant computational effort to determine the component functions will scale polynomially, rather than the traditional exponential increase with d [26].

A measure μ for the measure space $(K_d, \mathcal{B}(K_n), \mu)$, where $\mathcal{B}$ is the Borel σ-algebra on K_d, is defined as

$$\mathrm{d}\mu(\boldsymbol{\xi}) := \mathrm{d}\mu(\xi_1, \cdots, \xi_d) = \prod_{i=1}^{d} \mathrm{d}\mu_i(\xi_i), \quad \int_{K_1} \mathrm{d}\mu_i(\xi_i) = 1,$$

$$\mathrm{d}\mu(\boldsymbol{\xi}) = g(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi} = \prod_{i=1}^{d} g_i(\xi_i)\mathrm{d}\xi_i. \tag{29}$$

Here, $g(\xi_i)$ is the marginal density of the input ξ_i. It is the particular form chosen for the $g_i(\xi_i)$ that will determine the form of the component functions. In order to compute these functions, let us also define unconditional and conditional mean with respect to a group of input variables as

$$\mathbf{M}f(\boldsymbol{\xi}) := \int_{K_d} f(\boldsymbol{\xi}) d\mu, \qquad \mathbf{M}^{(i_1 \cdots i_l)} f(\boldsymbol{\xi}) := \int_{K_{d-l}} f(\boldsymbol{\xi}) \left[\prod_{j \notin \{i_1 \cdots i_l\}} d\mu_j(\xi_j) \right]. \tag{30}$$

Then, via a family of projection operators $P_{i_1 \cdots i_l} : K_d \to K_l$, the component functions are recursively defined as follows [26]:

$$\begin{aligned}
f_0 &:= P_0 f(\boldsymbol{\xi}) = \mathbf{M}f(\boldsymbol{\xi}) \\
f_i(\xi_i) &:= P_i f(\boldsymbol{\xi}) = \mathbf{M}^{(i)} f(\boldsymbol{\xi}) - P_0 f(\boldsymbol{\xi}) \\
f_{ij}(\xi_i, \xi_j) &:= P_{ij} f(\boldsymbol{\xi}) = \mathbf{M}^{(ij)} f(\boldsymbol{\xi}) - P_i f(\boldsymbol{\xi}) - P_j f(\boldsymbol{\xi}) - P_0 f(\boldsymbol{\xi}) \\
&\vdots \\
f_{i_1 \cdots i_l}(\boldsymbol{\xi}) &:= P_{i_1 \cdots i_l} f(\boldsymbol{\xi}) = \mathbf{M}^{(i_1 \cdots i_l)} f(\boldsymbol{\xi}) - \sum_{j_1 < \cdots < j_{l-1} \subset \{i_1 \cdots i_l\}} P_{j_1 \cdots j_{l-1}} f(\boldsymbol{\xi}) - \cdots - P_0 f(\boldsymbol{\xi})
\end{aligned} \tag{31}$$

The component functions $f_{i_1, \cdots i_l}$ and $f_{j_1 \cdots j_k}$ are independent and orthogonal, thus as long as one index between $\{i_1, \cdots i_l\}$ and $\{j_1 \cdots j_k\}$ differs we have

$$\int_{K_d} f_{i_1, \cdots i_l}(\xi_{i_1}, \cdots, \xi_{i_l}) f_{j_1 \cdots j_k}(\xi_{j_1}, \cdots, \xi_{j_k}) \mathrm{d}\mu = 0 \tag{32}$$

The correlation interpretation of $f_{i_1 \cdots i_l}$ is associated with the chosen form of the measure μ. If $g_i = 1$, $i = 1, \cdots, d$, the Lebesgue measure ($\mathrm{d}\mu = \mathrm{d}\xi_1 \mathrm{d}\xi_2 \cdots \mathrm{d}\xi_d$) is retrieved and (28) together with (31) becomes the well-know Analysis Of Variance (ANOVA) decomposition. Computing the component functions in the ANOVA decomposition involves evaluating multi-dimensional integrals, which can be done by for instance MC techniques [31]. An alternative which is more computationally tractable is the cut-HDMR decomposition proposed in [27,26]. In this case the measure is defined as

$$\mathrm{d}\mu = \prod_{i=1}^{d} \delta(\xi_i - \eta_i)\mathrm{d}\xi_i, \tag{33}$$

i.e. $g_i(\xi_i) = \delta(\xi_i - \eta_i)$, a Dirac measure located at the 'cut center' $\boldsymbol{\eta} = (\eta_1, \eta_2, \cdots, \eta_d)$. This choice removes the need for evaluating multi-dimensional integrals, and it expresses $f(\boldsymbol{\xi})$ as a superposition of its values along lines, planes and hyperplanes passing through the cut center $\boldsymbol{\eta}$. The component functions (31) now become

$$\begin{aligned}
f_0 &:= P_0 f(\boldsymbol{\xi}) = f(\boldsymbol{\eta}) \\
f_i(\xi_i) &:= P_i f(\boldsymbol{\xi}) = f^{(i)}(\xi_i) - P_0 f(\boldsymbol{\xi}) \\
f_{ij}(\xi_i, \xi_j) &:= P_{ij} f(\boldsymbol{\xi}) = f^{(ij)}(\xi_i, \xi_j) - P_i f(\boldsymbol{\xi}) - P_j f(\boldsymbol{\xi}) - P_0 f(\boldsymbol{\xi}) \\
&\vdots \\
f_{i_1 \cdots i_l}(\boldsymbol{\xi}) &:= P_{i_1 \cdots i_l} f(\boldsymbol{\xi}) = f^{(i_1 \cdots i_l)}(\xi_{i_1}, \cdots, \xi_{i_l}) - \sum_{j_1 < \cdots < j_{l-1} \subset \{i_1 \cdots i_l\}} P_{j_1 \cdots j_{l-1}} f(\boldsymbol{\xi}) - \cdots - P_0 f(\boldsymbol{\xi}).
\end{aligned} \tag{34}$$

Here, $f^{(i_1 \cdots i_l)}(\xi_{i_1}, \cdots, \xi_{i_l})$ is the conditional mean (30) taken with respect to measure (33), and thus it equals f with its inputs ξ_i set to η_i, except inputs $\xi_{i_1}, \cdots, \xi_{i_l}$. As an example, consider the univariate function $f^{(i)}(\xi_i) = f(\eta_1, \cdots, \eta_{i-1}, \xi_i, \eta_{i+1}, \cdots, \eta_{n_\xi})$.

The authors of [20] used the cut-HDMR framework coupled with their Adaptive Sparse-Grid (ASG) collocation method [19], where they chose $\boldsymbol{\eta}$ as the mean of the random input vector. Besides truncating (28) at a certain order, they also made their approach dimension adaptive based on weights which identify the important dimensions. Although their ASG method uses only a linear finite-element basis, interpolation overshoots can still occur. Thus, motivated by their work in [20] we will also employ a dimension adaptive cut-HDMR approach, except we will couple it with the SSC method utilizing the SCENO stencils to avoid the mentioned downsides of ASG.

If we define $\mathcal{K} := \{1, 2, \cdots, d\}$, the HDMR expansion (28) can be written in short-hand notation as [20]

$$f(\boldsymbol{\xi}) = \sum_{\mathbf{u}\subseteq\mathcal{K}} f_{\mathbf{u}}(\boldsymbol{\xi}_{\mathbf{u}}) = \sum_{\mathbf{u}\subseteq\mathcal{K}}\sum_{\mathbf{v}\subseteq\mathbf{u}} (-1)^{|\mathbf{u}|-|\mathbf{v}|} f^{(\mathbf{v})}(\boldsymbol{\xi}_{\mathbf{v}}), \tag{35}$$

where in the first equality we sum over the powerset of $\mathcal{K}$, i.e. over all possible subsets $\mathbf{u} \subseteq \mathcal{K}$. We furthermore set $f_{\emptyset} = f_0$. The second equality is obtained by expanding each component function $f_{\mathbf{u}}(\boldsymbol{\xi}_{\mathbf{u}})$ as indicated in (34). Notation wise, if for instance $\mathbf{v} = \{1, 4, 6\}$, then $f^{(\mathbf{v})}(\boldsymbol{\xi}_{\mathbf{v}}) = f^{(146)}(\xi_1, \xi_4, \xi_6)$. Each individual $|\mathbf{v}|$-dimensional subproblem $f^{(\mathbf{v})}(\boldsymbol{\xi}_{\mathbf{v}})$ can be approximated by a SSC surrogate (6). In that case (35) becomes

$$f(\boldsymbol{\xi}) \approx w(\boldsymbol{\xi}) = \sum_{\mathbf{u}\subseteq\mathcal{K}}\sum_{\mathbf{v}\subseteq\mathbf{u}} (-1)^{|\mathbf{u}|-|\mathbf{v}|} \sum_{j=1}^{n_e}\sum_{l=0}^{N_j} c_{jl}\boldsymbol{\Psi}_{jl}(\boldsymbol{\xi}_{\mathbf{v}}). \tag{36}$$

In order to assess the convergence of each individual $f^{(\mathbf{v})}(\boldsymbol{\xi}_{\mathbf{v}})$, the authors of [20] use the hierarchical surplus. This is also possible in the case of the SSC method, see (15). Alternatively, the RMS error estimate (17) can used for this purpose. Since (17) is a global error estimate and it also includes information from the distribution of the input parameters, we use the RMS error to assess the convergence.

Furthermore, the mean of each component function, defined as $J_{\mathbf{u}}$, can also be computed from the surrogate model

$$J_{\mathbf{u}} = \sum_{\mathbf{v}\subseteq\mathbf{u}} (-1)^{|\mathbf{u}|-|\mathbf{v}|} \sum_{j=1}^{n_e}\sum_{l=0}^{N_j} c_{jl}\mathbb{E}\left[\boldsymbol{\Psi}_{jl}(\boldsymbol{\xi}_{\mathbf{v}})\right]. \tag{37}$$

We compute (37) via random sampling, which can be performed quickly since it requires only sampling the surrogate model.

In order to identify the important dimensions, all first order component functions $f_i(\xi_i)$ are computed. Again, these are one-dimensional functions which measure the impact of a single independent input parameter on the output. Next, a weight is defined

$$\alpha_i = \frac{\|J_i\|_2}{\|f_0\|_2}, \tag{38}$$

which measures the contribution of each individual ξ_i on the mean of all first order component functions [20]. We always take the L_2 norm $\|\cdot\|_2$ over the spatial domain. Equation (38) can be considered as a sensitivity index, and only those dimensions for which (38) is larger than a user-prescribed error threshold ϵ_1 are considered important. All higher order $f_{\mathbf{v}}(\boldsymbol{\xi}_{\mathbf{v}})$ where $\mathbf{v}$ contains indices of dimensions which did not make the cut will not be computed. Consider e.g. a d-dimensional problem on K_d, where only $\mathbf{v} = \{1\}$ and $\mathbf{v} = \{2\}$ satisfy $\alpha_i > \epsilon_1$. The only higher-order component function that will be computed in this case is $f_{12}(\xi_1, \xi_2)$, regardless of the value of d.

The downside of (38) is that it is hard to choose ϵ_1 beforehand. One should first create the first-order HDMR expansion and decide on an appropriate value *a posteriori*. An alternative is to use a weight measuring the relative contribution of J_i with respect to the sum of all first-order means, i.e.

$$\alpha_i = \frac{\|J_i\|_2}{\sum_{k=1}^{d}\|J_k\|_2}. \tag{39}$$

Now one can *a priori* choose a $\epsilon_1 \in [0, 1]$, and select the smallest set of important dimensions for which the sum of their α_i is greater than ϵ_1.

Dimension adaptivity is extended to higher dimensions as well by defining a weight for $|\mathbf{u}| > 1$ as [20]

$$\alpha_{\mathbf{u}} = \frac{\|J_{\mathbf{u}}\|_2}{\|\sum_{\mathbf{v}\in\mathcal{V}_{comp},|\mathbf{v}|<|\mathbf{u}-1|} J_{\mathbf{v}}\|_2}. \tag{40}$$

Here, the set $\mathcal{V}_{comp}$ simply holds all the indices $\mathbf{v}$ that were computed. Furthermore, all subsets $\mathbf{v}$ of component functions which are important are added to a set $\mathcal{V}_{imp}$. That way, a higher-order important $\mathbf{u}$ is admissible if all $\mathbf{v} \subset \mathbf{u}$ required to compute (35) are also in $\mathcal{V}_{imp}$. This is the so-called admissibility condition, which is given by

$$\mathbf{u} \in \mathcal{V}_{imp} \text{ and } \mathbf{v} \subset \mathbf{u} \Rightarrow \mathbf{v} \in \mathcal{V}_{imp}. \tag{41}$$

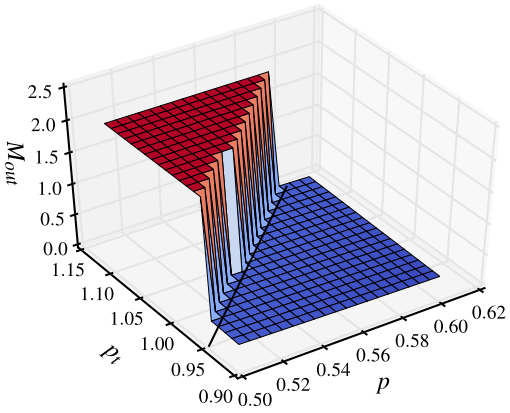


Fig. 9. M_{out} as function of p and p_t obtained by MC sampling, with the geometrical constants fixed to their nominal value.

Similar to the first-order case, we can define a relative counterpart of (40) as

$$\alpha_{\mathbf{u}} = \frac{\| J_{\mathbf{u}} \|_2}{\sum_{\mathbf{v} \in \mathcal{V}_{comp}, |\mathbf{v}|=|\mathbf{u}|} \| J_{\mathbf{v}} \|_2}, \tag{42}$$

such that the $\alpha_{\mathbf{u}}$ sum to one and we can choose a $\epsilon_1 \in [0, 1]$ *a priori*.

Finally, a relative error measure between two HDMR expansions of consecutive orders $p-1$ and p is defined as

$$\alpha_p = \frac{\| \sum_{|\mathbf{u}| \leq p} J_{\mathbf{u}} - \sum_{|\mathbf{u}| \leq p-1} J_{\mathbf{u}} \|_2}{\| \sum_{|\mathbf{u}| \leq p-1} J_{\mathbf{u}} \|_2}. \tag{43}$$

The algorithm stops when α_p becomes smaller than another used-defined threshold ϵ_2. An overview of the HDMR algorithm is depicted in Appendix C.

5. Results and discussion

5.1. Comparison ENO–SCENO stencils

We present the results obtained with the baseline SSC method with ENO stencils, versus the SSC method with the SCENO stencils. As a test case we use a quasi-1D nozzle case up to 5 dimensions and an algebraic test function up to $d = 8$.

5.1.1. Nozzle flow

As a test case we use the solver from [25], which computes the flow through a quasi-1D diverging nozzle. We prescribe the flow to be sonic at the nozzle inlet, i.e. $M_{in} = 1$. From fluid mechanics we know that the flow is driven by the pressure ratio, i.e. by the ratio between the total pressure p_t at the inlet and the static pressure p of the surroundings at the nozzle exit. Depending on the value of p_t/p, the flow can show very different behavior. If p_t/p exactly equals the adaptation value, the flow reaches the static pressure of the surroundings at the nozzle exit and the jet exhausts smoothly into the atmosphere. A stronger p_t/p will result in smooth flow through the nozzle, which is supersonic at the nozzle exit. In order to match the outside pressure p, the flow undergoes a supersonic expansion attached to the nozzle exit (under-expanded nozzle). A smaller p_t/p, but still above a threshold that depends on the ratio of the exit to the throat area, still results in smooth flow through the nozzle, but this is now over-expanded and is compressed to the outside pressure through an oblique shock attached to the nozzle exit. When p_t/p is equal to the threshold value, the flow is characterized by a normal shock located at the nozzle exit: upstream of the shock, the flow is smooth, and verifies adaptation conditions in the exit section; immediately downstream of it, the flow is subsonic and matches the outside pressure. Finally, when p_t/p is below the threshold value, a normal shock wave is formed somewhere inside the nozzle. This results in subsonic flow at the exit, and an exit pressure that is equal to p [2].

Given the pressure ratio, the flow is completely characterized by the shape of the nozzle [2]. As in [25], we consider the following hyperbolic tangent for the nozzle shape

$$f(x) = a + b \tanh(cx - d). \tag{44}$$

To test the SSC method, we specify two different ranges for the uncertain parameters such that two radically different response surfaces have to be created. First, we prescribe a wider range for p such that the QoI is highly discontinuous, see Fig. 9. In the second case we restrict p to a more narrow interval such that the QoI is smooth. More specifically, we prescribe the uniform input distributions for the 6 uncertain parameters described in Table 2. Furthermore, we choose M_{out} (the Mach numbers at the nozzle exit) as our quantity of interest, as it allows us to easy calculate other flow quantities via the

Table 2
Uncertain input parameters of the discontinuous (D) and smooth (S) case.

d	Parameter	Mean (D)	Range (D)	Mean (S)	Range (S)
1	p [bar]	0.55	[0.5, 0.6]	0.625	[0.60, 0.65]
2	p_t [bar]	1.0	[0.9, 1.1]	1.0	[0.9, 1.1]
3	a [–]	1.75	[1.575, 1.925]	1.75	[1.575, 1.925]
4	b [–]	0.7	[0.63, 0.77]	0.7	[0.63, 0.77]
5	c [–]	0.8	[0.72, 0.88]	0.8	[0.72, 0.88]
6	d [–]	4.0	[3.6, 4.4]	4.0	[3.6, 4.4]

Table 3
The computational cost of the discontinuous QoI.

Type	d [–]	n_s [–]	T [min]	LEC [%T]	S_j [%T]	$\mathbf{v}$ [%T]
Baseline	2	50	0.56	3.56	3.16	87.3
	3	100	2.09	24.39	11.46	39.32
	4	150	10.95	73.42	15.37	6.22
	5	200	119.29	85.21	11.26	0.58
SCC-SC	2	50	0.54	1.45	1.24	82.46
	3	100	1.33	1.2	2.33	54.75
	4	150	1.37	5.56	12.34	42.99
	5	200	4.75	4.88	17.2	11.47

Table 4
The computational cost of the smooth QoI.

Type	d [–]	n_s [–]	T [min]	LEC [%T]	S_j [%T]	$\mathbf{v}$ [%T]
Baseline	2	50	0.73	2.28	2.87	89.9
	3	100	2.52	20.37	16.42	42.07
	4	150	22.86	62.18	30.87	3.95
	5	200	731.5	58.31	40.99	0.13
SCC-SC	2	50	0.7	1.28	0.31	85.64
	3	100	1.65	4.0	0.43	61.14
	4	150	1.63	16.76	1.26	49.01
	5	200	4.68	13.62	1.41	15.45

isentropic relations once M_{out} is known [2]. When constructing the surrogate models, we will use a linear transformation for each input to map points from $[0, 1]$ in the stochastic domain to points in the physical domain with the range as specified in Table 2. This simplifies the construction of the surrogate models as it allows us to always work in the standard d-dimensional hypercube K_d.

For now, we will consider just the first 5 uncertain parameters of Table 2. In Tables 3 and 4 we show the computation time T in minutes versus the dimension d, in case of the discontinuous and smooth QoI for both the baseline and the method based on SCENO stencils. This is of course dependent upon the available computational resources, in our case a 24 core workstation. We use these cores to parallelize the LEC condition, code sampling and ENO stencils. Our algorithm for the construction of the SCENO stencils is not implemented in parallel, and uses just 1 core. We can see that T rises very quickly as d increases in the case of the baseline method, especially in the case of the smooth QoI. To explain which element is responsible for the high computation time, we also show the percentage of T that is spent on the LEC condition, construction of the stencils S_j, and QoI calculation.

Since the nozzle code is just a cheap test problem, Tables 3 and 4 show that computing the QoI samples $\mathbf{v}$ only takes up a significant portion of T for low d. For the baseline SSC method the construction of ENO-type stencils makes up a significant part of the computational cost, but the enforcement of the LEC condition is the most expensive component in higher dimensions. Thus, for the baseline method, most of the computational effort is put into enforcing the LEC condition. For that reason the computational cost of the LEC condition is investigated in more detail.

As explained in Section 2.1, the LEC condition (14) is enforced by a MC approach, for all simplices in S_j at all $j = 1, \cdots, n_e$. Thus, for the baseline SSC method the number of times the surrogate model is evaluated in each iteration i is bounded by

$$n_{w_i} = n_e \times n_{e,S_j}, \quad i = 1, \cdots, I \tag{45}$$

where n_{e,S_j} is the number of simplices in a single stencil S_j with $p = p_{max}$, and I is the total number or of iterations of the SSC algorithm. Here we assumed that per S_j, one sample is placed in each simplex using (25). The number of points in the Delaunay grid is given simply by (5), but estimating n_e for arbitrary d is not trivial. The worst-case number of simplices in a Delaunay triangulation is bounded by the so-called Upper-Bound theorem, which states that n_e is at most of $\mathcal{O}(n_s^{d/2})$. In the best-case scenario (points distributed uniformly at random inside the unit sphere) n_e scales as $\mathcal{O}(n_s)$ for any d, with

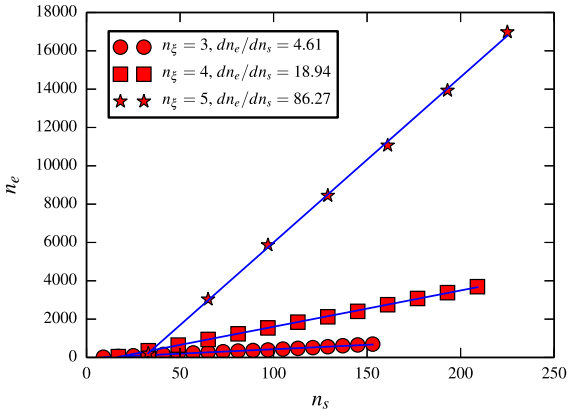
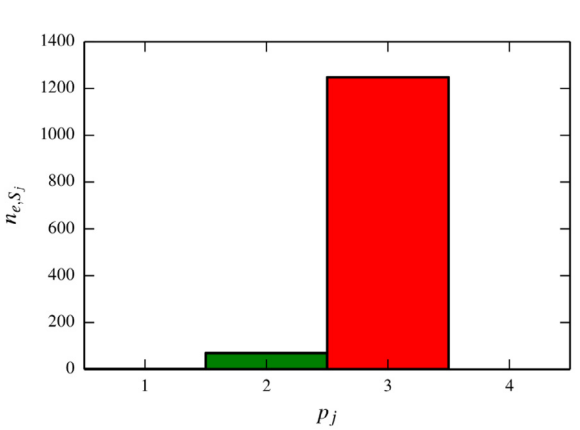
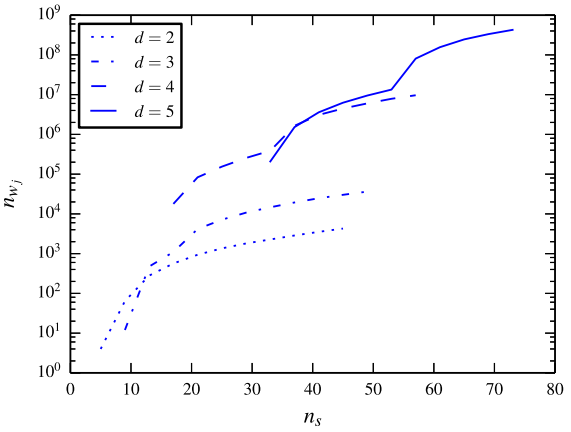


Fig. 10. n_e as function of n_s for the smooth QoI. The discontinuous QoI gives a similar figure. The slope dn_e/dn_s is computed via a least-square regression line.



(a) n_{e,S_j} versus p_j for $d = 5$, $n_s = 73$, $p_{max} = 3$.



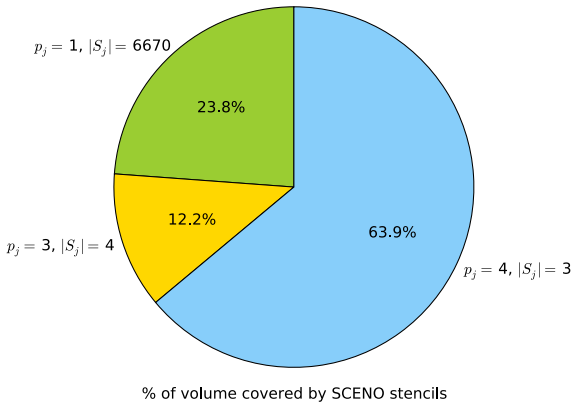
(b) The maximum number of surrogate model evaluations vs n_s. Here, $p_{max} = 4$ for $d = 2, 3$ and $p_{max} = 3$ for $d = 4, 5$. The jumps that can be observed occur when n_s is large enough to allow a higher p_j.

Fig. 11. Examples of the exponential growth of SSC components.

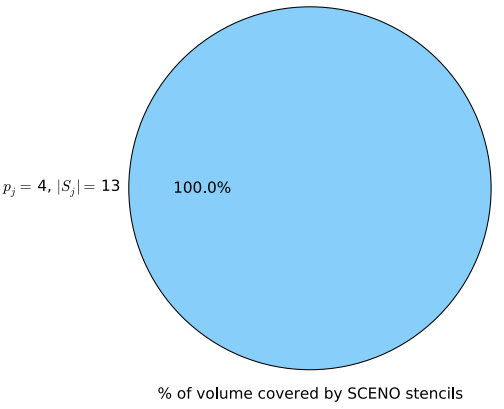
a constant factor that is exponential with the dimension [1]. To find out where in between these two bounds our specific problem resides, we plot n_e versus n_s in Fig. 10 for $d \in \{3, 4, 5\}$. These results indicate that we are close to the $\mathcal{O}(n_s)$ bound, since the $n_e(n_s)$ are described quite well by the linear regression also shown in Fig. 10. However, the exponential increase of dn_e/dn_s means that for a moderate number of samples we can still have a large number of simplices if d is high enough. Note that other than limiting the number of samples n_s, we have no means of controlling the magnitude of n_e.

The term n_{e,S_j} in (45) grows exponentially with p_j for a given d. This can be seen in Fig. 11(a), where we plot n_{e,S_j} versus the local polynomial order p_j for $d = 5$. Unlike n_e, we obviously have some control over the magnitude of n_{e,S_j} through the inclusion of a maximum allowable cutoff value for p_j. Note however that limiting p_j will affect the order of convergence (12). The upper bound (45), added over iterations i is plotted as a function of n_s in Fig. 11(b) for $d = 2, \cdots, 5$. It shows a rapid increase with both d and p_j.

By comparing the computational time T of the SSC method with that of the SSC-SC method (Tables 3–4), it is clear that the SSC-SC method is several orders of magnitude more efficient for the dimensions considered in this example. To clearly show why the SSC-SC method is computationally more efficient than the baseline method, consider Fig. 12. Here we display the fraction of the volume $\bar{\Xi}$ that is covered by SCENO stencils of different polynomial order p_j, for the discontinuous and smooth case with $d = 5$. Also the number of stencils $|S_j|$ per order is shown. Note that for the discontinuous QoI, just 7 high-order SCENO stencils (stencils with $p_j > 1$) already cover 76.1% of the domain. In the case of the baseline method the number of stencils (and thereby LEC iterations) equals n_e, which is 11 034 in this example. For the smooth QoI (Fig. 12(b)) we required 13 fourth-order stencils to cover the entire domain. With the baseline method we would have a set of 9451 stencils, likely all of fourth order as well and therefore large and expensive.



(a) Results for $d = 5$ and $n_s = 200$ with the discontinuous QoI.



(b) Results for $d = 5$ and $n_s = 200$ with the smooth QoI.

Fig. 12. The volume coverage of SCENO stencils per polynomial order.

Table 5
The relative errors (46) of the discontinuous QoI for the baseline SSC and SSC-SC method.

Type	d	n_s	ϵ_μ	ϵ_σ	ϵ_w
Baseline	2	50	3.536e−02	3.669e−02	2.001e−01
	3	100	5.271e−02	7.475e−02	2.408e−01
	4	150	2.532e−02	1.425e−01	2.828e−01
	5	200	2.006e−02	2.320e−01	3.253e−01
SCC-SC	2	50	1.590e−02	4.397e−02	1.597e−01
	3	100	3.975e−03	7.452e−02	2.108e−01
	4	150	1.199e−03	1.329e−01	2.547e−01
	5	200	3.368e−03	1.803e−01	2.876e−01

Table 6
The relative errors (46) of the continuous QoI for the baseline SSC and SSC-SC method.

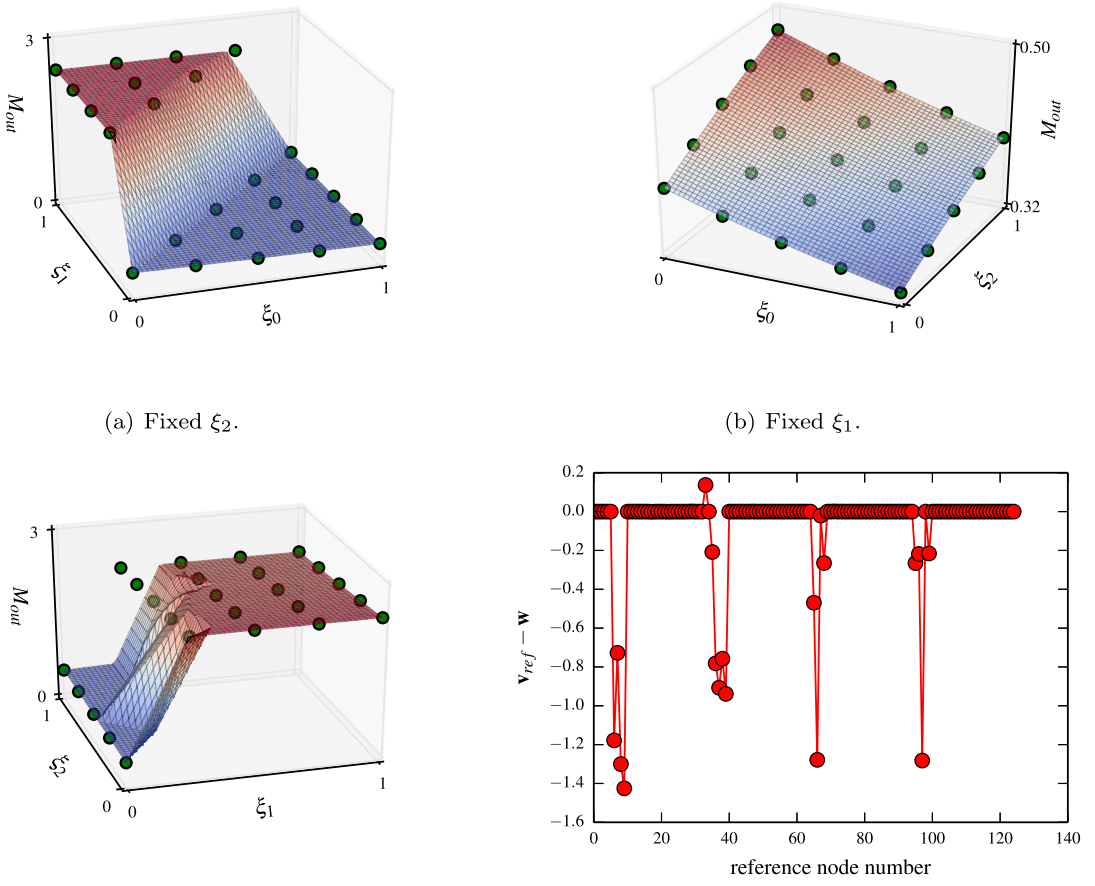
Type	d	n_s	ϵ_μ	ϵ_σ	ϵ_w
Baseline	2	50	1.131e−06	5.137e−06	1.088e−05
	3	100	7.276e−07	1.572e−05	1.416e−05
	4	150	1.015e−06	3.784e−06	2.566e−05
	5	200	2.446e−05	3.376e−04	2.228e−03
SCC-SC	2	50	8.042e−07	1.952e−05	1.916e−05
	3	100	8.734e−07	3.019e−07	7.075e−06
	4	150	1.006e−06	2.234e−06	2.104e−05
	5	200	3.034e−06	8.186e−06	8.218e−05

As stated in Section 2.1, our primary interest is computing the statistical moments of the QoI, in particular the mean and standard deviation. To assess the accuracy of the SSC method we used a reference solution for each considered dimension d. To compute the errors we define the following relative L_2 error measures for the mean, standard deviation and interpolation surface

$$\epsilon_\mu = \frac{\|\mu_w - \mu_{ref}\|_2}{\|\mu_{ref}\|_2}, \quad \epsilon_\sigma = \frac{\|\sigma_w - \sigma_{ref}\|_2}{\|\mu_{ref}\|_2}, \quad \epsilon_w = \frac{\|\mathbf{w}(\boldsymbol{\xi}_{ref}) - \mathbf{v}_{ref}\|_2}{\|\mathbf{v}_{ref}\|_2}. \tag{46}$$

Here, the subscript w denotes a quantity computed with the surrogate model, and *ref* is the exact value computed via random sampling. In the interpolation surface error, $\mathbf{v}_{ref}$ is a vector containing 10^4 MC code samples and $\mathbf{w}(\boldsymbol{\xi}_{ref})$ are the surrogate model outputs evaluated at the same MC locations $\boldsymbol{\xi}_{ref}$. The values of the error measures (46) for both QoIs and both surrogate models can be found in Tables 5–6. Note that the error levels are roughly the same for both surrogate models.

From Tables 5–6 we note that the errors of the discontinuous case are considerable higher than for the smooth case. This can be attributed to the smearing of discontinuities, i.e. the linear interpolation of a discontinuity over a simplex, which especially contributes to the error of the surrogate model in higher dimensions. See for instance Fig. 13, which depicts 2D projections of a 3D surrogate model along with reference data on an ordered uniform grid. Especially in Fig. 13(a) we can clearly identify regions where the smearing of the discontinuity contributes to the error. For this particular case, we plotted



(a) Fixed ξ_2.

(b) Fixed ξ_1.

(c) Fixed ξ_0.

(d) The difference $\mathbf{v}_{ref} - \mathbf{w}$ vs the reference node number for fixed ξ_0.

Fig. 13. 3D surrogate model displayed in 2 dimensions by fixing 1 dimension to a particular value. The green dots are reference data on an ordered uniform grid. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the difference between the surrogate model and the reference data in Fig. 13(b), which also identifies sharp regions of high error. This situation gets progressively worse as d increases.

In [40,36] Witteveen et al. apply a sub-cell resolution approach to the SSC method for the case when the discontinuity in the probabilistic space is a function of a physical discontinuity with random location. Our results indicate that for high d sub-cell resolution could prove to be beneficial, even if the physical location of the QoI is not random.

5.1.2. Computational cost at $d = 5$

Tables 3–4 show the computational cost at a fixed number of samples per dimension. In this section we investigate the cost of both the baseline and our SC approach as a function of the number of samples n_s. Specifically, we consider the smooth nozzle flow case at $d = 5$, and add 50 code samples at each iteration until an imposed maximum of 500 samples.

The results are depicted in Fig. 14. In light of previous results (Tables 3–4), we limited the maximum polynomial order to 4 in case of the baseline method in an attempt to suppress the cost. Still, it took 1734 minutes (roughly 29 hours) to create a surrogate model containing 333 code samples. Furthermore, attempts to create surrogates with more samples terminated prematurely due to the excessive memory requirements of the algorithm. With our SC approach we were able to reach the target of 500 samples in 20 minutes, without limiting the polynomial order.

5.2. Algebraic test function

To test the limitations of both the baseline and SSC-SC approach in dimensions higher than 5 we make use of the following algebraic test function from [38]

$$u(\boldsymbol{\xi}) = \arctan\left(\boldsymbol{\xi} \cdot \boldsymbol{\xi}^* + \xi_1^*\right). \tag{47}$$

Here, $\boldsymbol{\xi}$ is a d-dimensional vector of uniformly distributed random variables on $[0, 1]$, and $\boldsymbol{\xi}^* = \{0.5, \cdots, 0.5\} \in \mathbb{R}^d$.

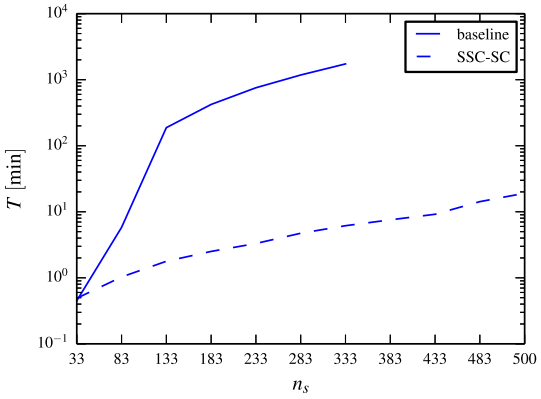


Fig. 14. The computational time T in minutes versus n_s at $d = 5$.

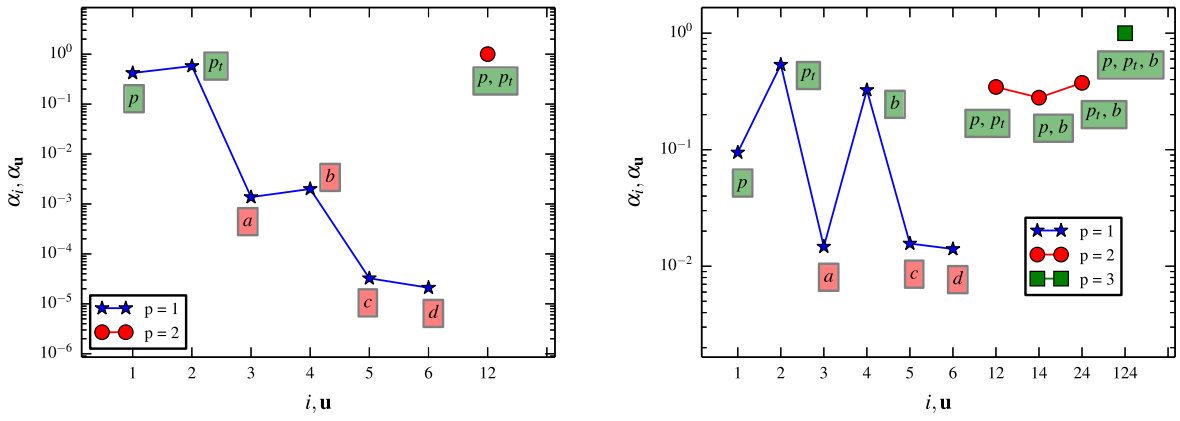
Table 7
The computational cost for the arctan test function. A '×' signifies a failed attempt.

Type	d [–]	n_s [–]	T [min]	LEC [%T]	S_j [%T]	$\mathbf{v}$ [%T]
Baseline	2	50	0.12	14.77	20.83	26.57
	3	100	1.7	24.77	24.09	4.03
	4	150	15.99	61.54	33.67	0.3
	5	200	473.87	57.95	41.04	0.01
	6	×	×	×	×	×
	7	×	×	×	×	×
	8	×	×	×	×	×
SCC-SC	2	50	0.07	13.14	3.26	35.34
	3	100	0.62	10.47	0.98	6.63
	4	150	0.84	39.3	2.7	3.63
	5	200	3.88	19.42	2.0	0.57
	6	250	11.58	70.28	4.03	0.14
	7	300	15.75	73.64	19.81	0.15
	8	350	186.11	51.51	45.17	0.02

The computational time is shown in Table 7. In the case of the baseline method we encountered the same behavior as in the preceding section, i.e. an exponential increase in the cost. Moreover, we were not able to create surrogates for $d > 5$ due to excessive memory requirements.

We were able to create surrogates up to $d = 7$ without difficulty with the SSC-SC approach. The limitations of the SSC-SC approach start to appear in the 8-dimensional case. Note that for the given number of samples in Table 7 we could still construct the surrogate in a reasonable amount of time, i.e. a run-time less than the expected cost of drawing 350 samples from some expensive simulation code. However, a sharp increase in the cost compared to the 7-dimensional case is still observed. The cause of this are the violations of the LEC limiter which occur despite the fact that (47) is a smooth QoI. As a consequence the SCENO stencils no longer cover the entire domain, and additional stencils must be computed for which the LEC limiter must also be enforced. Several iterations of the LEC and SCENO subroutines are required until the stencils are both LEC and set covering. A possible explanation for the LEC violations is that our monomial basis (18) is no longer suitable in these high-dimensional spaces, despite the fact that we solve our linear system as depicted in (27). Further research is required for spaces of 8 dimensions. Another bottleneck for higher dimensions is the ability to create Delaunay triangulations for $d > 8$, which is not supported by the Delaunay subroutines we used [5]. Moreover, the time complexity of the Delaunay triangulation in general for these dimensions is a known issue, see for instance [6] for a comparison between different triangulation algorithms up to a dimension of 6. The complexity bounds discussed in Section 5.1.1 and displayed in Fig. 10 will ultimately make the Delaunay triangulation unusable for our purpose, especially when considering problems of dimension $\mathcal{O}(10)$.

Thus, the use of the proposed set-covering approach is that it allows us to efficiently create SSC surrogate models, *within the range of dimensions where we could construct the Delaunay triangulation.* To apply the SSC method to higher dimensions, one option is to replace the Delaunay triangulation with a scheme where simplices are formed by selecting the nearest points from randomly placed MC samples as described in [38]. Another option is the use of a dimension adaptive approach such as cut-HDMR in order to avoid high-dimensional spaces altogether, the results of which are shown in the coming sections.



(a) Discontinuous QoI. (b) Smooth QoI

Fig. 15. The relative weights (39) and (42) for the discontinuous and smooth nozzle QoI. The green boxes are the dimensions which are added to V_{imp}. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 8
Relative error values vs the HDMR order p and n_s in the case of the discontinuous QoI.

HDMR order p	ϵ_1	n_s	ϵ_μ	ϵ_σ	ϵ_w
1	0.9	57	4.514e−01	3.601e−01	7.434e−01
2		89	3.926e−02	8.830e−02	2.196e−01

Table 9
Relative error values vs the HDMR order p and n_s in the case of the smooth QoI.

HDMR order p	ϵ_1	n_s	ϵ_μ	ϵ_σ	ϵ_w
1	0.9	37	4.933e−05	2.885e−04	5.035e−03
2		67	3.603e−05	6.457e−04	3.608e−03

5.3. cut-HMDR applied to nozzle flow

We now show the results obtained from using the cut-HDMR approach coupled with the SSC-SC method, applied to the nozzle flow case. Unlike in Section 5.1.1, we now consider all 6 uncertain coefficients from Table 2.

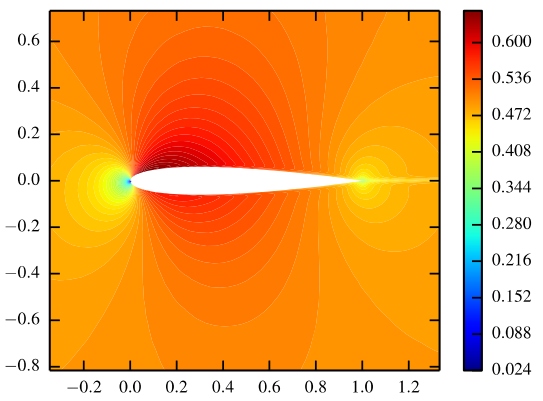
The first order weights α_i (39) are used to determine the important dimensions. Fig. 15 shows α_i with $i = 1, \cdots, 6$ corresponding to the 6 parameters of Table 2, as well as the subsequent higher order $\alpha_{\mathbf{u}}$. For the discontinuous QoI only the first 2 parameters are significant (p and p_t). Together they are responsible for 99.7% of the total first order mean. We have set $\epsilon_1 = 0.9$, such that those dimensions which make up 90% or more of the p-th order mean are added to $\mathcal{V}_{imp}$. For the smooth QoI we need p and p_t as well, but also the coefficient b in order to meet this constraint. All admissible subsequent dimensions are important as well.

The computational time for both QoIs, due to the fact that the nozzle code is quickly evaluated, is in the order of several minutes. Note that in the case of the baseline method we were unable to create a surrogate in a six-dimensional space.

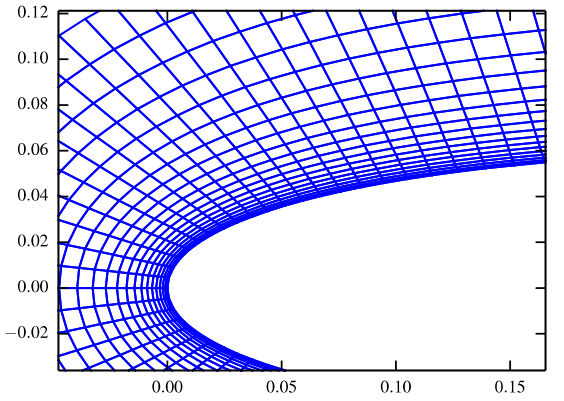
The values of error measures (46) for the discontinuous and smooth case are given in Tables 8–9. Notice that a first-order HDMR expansion is not sufficient for the discontinuous QoI, but already for $p = 2$ the relative errors in the statistical moments are of $\mathcal{O}(10^{-2})$. Table 9 shows the results for the smooth QoI. The errors, even for a first-order expansion, are of $\mathcal{O}(10^{-3})$ or below. Again, the higher errors in the discontinuous case can be attributed to the linear smearing of discontinuities in the response surface.

Compare the relative errors from Table 5 with those of Table 8, and likewise for Tables 6 and 9. The errors are of a similar order of magnitude, even though the HDMR method doesn't sample the full six-dimensional space. At the same time we gain information about the correlation between the input parameters, in the sense of their combined impact on the code output. Also, the cut-HDMR method could be applied to even higher dimensional spaces, provided that the effective dimension is low.

These results demonstrate the power of the cut-HDMR technique in terms of computational efficiency for problems with low effective dimension. The cost of computing 6 one-dimensional and 1–3 two-dimensional surrogate models is significantly less than computing 1 six-dimensional problem. Also, in the case of the SSC method, it can avoid problems with the bad scalability of the Delaunay triangulation with increasing d.



(a) A deterministic solution showing the distribution of the Mach number. The free-stream Mach number is 0.5 and the angle of attack is 5°.



(b) A zoom into the leading edge of the discretised airfoil. The first grid points are set at a distance of 10^{-6} from the wall.

Fig. 16. The symmetrical NACA0012 airfoil.

Table 10
Uniformly distributed closure coefficients of the SA model.

Parameter	Mean	Range
C_{b1}	0.1355	[0.0949, 0.1762]
C_{b2}	0.622	[0.435, 0.809]
C_{v1}	7.1	[4.97, 9.10]
σ	2/3	[0.467, 0.867]
C_{w2}	0.3	[0.210, 0.390]
C_{w3}	2	[1.40, 2.60]
κ	0.41	[0.287, 0.455]

5.4. cut-HDMR applied to airfoil flow

In this section we present the results of the cut-HDMR approach (again coupled to the SSC-SC method), when applied to a computationally expensive problem, i.e. the turbulent flow over a NACA0012 airfoil, see Fig. 16(a). The free-stream Mach number is 0.5 and the angle of attack is set to 5°. The Reynolds number based on the chord length is $1.2 \cdot 10^7$. The grid is a C-grid with 70.085 nodes and the first node is located at a distance of 10^{-6} from the wall in order to provide sufficient resolution, see Fig. 16(b). The computation time for one flow-field was roughly 50 minutes.

The governing equations are the Reynolds–Averaged Navier–Stokes (RANS) equations, coupled to the Spalart–Allmaras (SA) turbulence model [32]. This model contains 7 empirically determined closure coefficients [34], whose best-fit values are unknown *a-priori* [11]. Therefore, we treat all 7 inputs as uniformly distributed variables with the end points set at ±30% of their nominal values, see Table 10. The chosen QoI is the lift coefficient, defined as $c_l = L'/(\rho_\infty V_\infty^2/2)$, where ρ_∞ and V_∞ are the free-stream density and velocity respectively. The term L' is the two-dimensional lift force.

The values of the relative weights α_i and $\alpha_{\mathbf{u}}$, i.e. equations (39) and (42), are depicted in Fig. 17. The value of ϵ_1 was again set to 0.9. For the first-order HDMR expansion we need three coefficients, namely κ, C_{b1} and C_b2, to capture more than 90% of the total first-order mean. Further note that the constants C_{w2} and C_{w3} are completely unimportant for the computation of our QoI, as their weights are of $\mathcal{O}(10^{-14})$. For the second-order expansion, we only need the interactions of (C_{b1}, κ) and (C_{b2}, κ) to represent more than 90% of the second-order mean component functions. Although (C_{b1}, C_{b2}) is not added to $\mathcal{V}_{imp}$, the third-order interaction (C_{b1}, C_{b2}, κ) is still admissible according to the admissibility condition (41). However, in this particular simulation a second order expansion was enough to satisfy the error measure between two HDMR expansions of consecutive orders (43), which was set to $\epsilon_2 = 10^{-3}$. Hence, no third order interaction was computed.

Since emulating the RANS code itself is our objective, no reference solution is available. We therefore plot the convergence of the mean and standard deviation of c_l in Table 11. As can be expected from the weights in Fig. 17, there is little difference between the statistics of first and second order HDMR expansion. In this particular case even a first order expansion could be considered as converged.

Note that if we would have applied either the baseline or the SSC-SC approach to a full 7-dimensional space, the initial Delaunay grid alone would be comprised of 129 samples. As can be seen from Table 11, the HDMR approach requires significantly less samples. This difference can be expected to increase as the dimensionality increases, provided that the problem is one of low effective dimension. It should be noted however, that Witteveen and Iaccarino suggested a method in which the initial 2^d samples can be avoided [38], but this approach requires extrapolation towards the hypercube boundaries. In

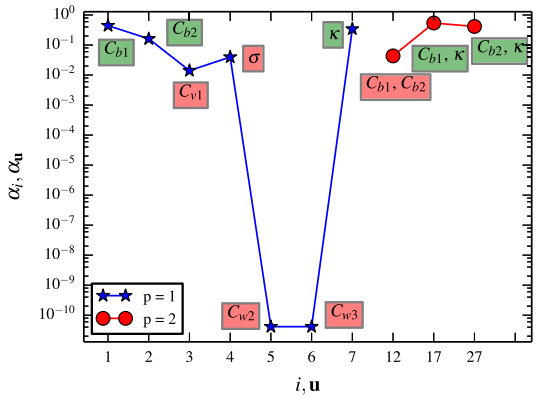


Fig. 17. The relative weights (39) and (42) for the NACA 0012 test case. The value of ϵ_1 was set to 0.9. The green boxes indicate the dimensions which are important. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 11
The convergence of the statistical moments of c_l as a function of the HDMR order.

HDMR order p	n_s	μ	σ
1	49	6.385509e−01	3.729915e−03
2	62	6.381699e−01	3.746544e−03

Table 12
The rankings from largest (1st) to smallest (7th) α_i for QoIs other than c_l. Below each coefficient its corresponding value of α_i is written.

QoI	1st	2nd	3rd	4th	5th	6th	7th
c_p	C_{b1} 2.604e−03	κ 1.339e−03	C_{b2} 8.784e−04	σ 5.510e−04	C_{v1} 6.866e−05	C_{w2} 3.674e−08	C_{w3} 3.655e−08
c_f	κ 1.764e−01	C_{b1} 8.835e−02	C_{b2} 8.676e−02	σ 2.072e−02	C_{v1} 6.819e−03	C_{w2} 9.976e−07	C_{w3} 9.875e−07
M	κ 3.658e−03	C_{b1} 1.951e−03	C_{b2} 1.498e−03	σ 6.591e−04	C_{v1} 1.233e−04	C_{w2} 3.958e−08	C_{w3} 3.951e−08
k	C_{b1} 8.708e−02	κ 2.982e−02	σ 2.726e−02	C_{b2} 2.529e−02	C_{v1} 5.735e−04	C_{w2} 2.650e−05	C_{w3} 2.591e−05

this case there is no guarantee that the LEC limiter is respected in the simplices where the extrapolation takes place, and only in the limit $n_e \to \infty$ full extremum-diminishing robustness is recovered for the entire domain Ξ.

Finally, for each code run we saved the results for a range of different physical quantities. Thus, we can *a posteriori* construct a surrogate model for each of these quantities. However, the samples were adaptively placed based on our chosen QoI c_l, and therefore might not be optimally distributed for another QoI. Moreover, the dimension adaptivity might cut out dimensions that are important for a QoI other than c_l. To perform a qualitative investigation, we therefore constructed a first-order HDMR expansion for the pressure coefficient c_p, skin-friction coefficient c_f, Mach number M and the turbulent kinetic energy k, all defined below.

$$c_p := \frac{p - p_\infty}{\frac{1}{2}\rho_\infty V_\infty^2}, \quad c_f := \frac{\tau_w}{\frac{1}{2}\rho_\infty V_\infty^2}, \quad M := \frac{u}{a}, \quad k := \frac{1}{2}\left(\overline{u_1'^2} + \overline{u_2'^2} + \overline{u_3'^2}\right) \tag{48}$$

Here, p and p_∞ are the static and free-stream pressure respectively. Furthermore, ρ_∞ and V_∞ are defined as before in c_l. The quantity τ_w is the wall shear stress, and in the expression for M, u is the local velocity, and a is the speed of sound. Finally, $\overline{u_i'^2}$ is the mean squared normal Reynolds stress in x_i direction [34].

In Table 12 we show the coefficients of the SA model sorted according to their value of the (non-relative) weight α_i (38) for all the QoIs of (48). Note that which coefficients are most influential does not change much from one QoI to another. For c_p, c_f and M the three most important coefficients are κ, C_{b1} and C_{b1}. In the case of the turbulent kinetic energy k, σ has taken third place, with again C_{b1} and κ as the most influential. Still, σ's value of α_i is close to the weight corresponding to C_{b2}. The ranking of the bottom three coefficients remains unchanged for all considered QoIs.

6. Conclusion

We have examined means to improve upon the scalability of the Simplex-Stochastic Collocation (SSC) method [39] for uncertainty quantification problems of moderate dimensionality, such that sampling an expensive simulation code will still be feasible. We found that creating a surrogate model using the baseline SSC method becomes restrictively expensive at 5 dimensions, and practically impossible at $d = 6$. To reduce this bad scalability, we needed to add some new features. First, for higher d we run the risk of obtaining a singular sample matrix. This can be circumvented by a method similar to the method used to construct the ENO stencils. If we encounter a stencil S_j for which the sample matrix is singular, we collect a set of candidate nearest-neighbor stencils which all contain the simplex Ξ_j associated to the j-th stencil S_j. We then select the stencil which has the highest polynomial order and which is non-singular. In the worst-case scenario we get a linear stencil, which is guaranteed to be non-singular irrespective of the dimension d.

Due to the exponential increase in the number of simplex elements with increasing dimension, enforcing the LEC condition becomes quickly very expensive for $d \geq 5$. As a first measure to combat this sharp increase in the computational burden, we have proposed an alternative technique for the stencil construction, based on the Set-Covering (SC) problem [16]. Unlike in the SSC method we do not construct a stencil for every simplex. Since n_e increases exponentially fast with d, we aim to find a relatively small set of stencils that covers all simplices in the probability domain. Due to the fact that the stencil size rises also exponentially, only a few high-order stencils are required to achieve this. We furthermore use the information contained in the code samples about the location of a possible discontinuity in the construction of the stencils. Using a simple measure based upon the maximum jump in code samples, we mark those simplices which contain a discontinuity and manually set their respective stencils to first order. Using only the remaining simplices as admissible candidates, we grow stencils by adding neighboring simplices. This approach assures that the number of stencils is significantly lower than n_e, and hence the number of times the LEC condition must be checked is reduced equally. Also, since the discontinuous simplices are removed as admissible candidates *a priori*, no stencil crosses a discontinuity. As a consequence the SC stencils resemble the ENO stencils in shape.

For dimensions 5 through 8 our SSC-SC method is significantly more computationally efficient as the baseline SSC method. However, at higher dimensions discretizing the probabilistic space using a Delaunay triangulation becomes a serious bottleneck. The SSC-SC method can therefore be viewed as a more efficient version of the baseline method within the range of dimensions where a Delaunay triangulation can be reasonably made. We examined another alternative, where we adapted the cut-HDMR method of [20] to the SSC method. Given a problem with low effective dimension, this approach circumvents the bad scalability of the Delaunay triangulation, while at the same time obtaining error estimates of similar order of magnitude compared to the full-dimensional baseline or SSC-SC method. Thus, this method is adaptive in both the stochastic domain as well as in the dimensions themselves, while retaining the advantages of the SSC method such as Runge-phenomenon free interpolation. We applied it to a computationally expensive flow case, i.e. the turbulent flow over an airfoil modeled with the Spalart–Allmaras eddy-viscosity model, which contains 7 imperfectly known closure coefficients. Instead of fully sampling a 7 dimensional space, and enforcing the LEC condition in this space, we could obtain a converged surrogate model with a second order HDMR expansion and 62 code samples.

Acknowledgements

The present work was supported by the French Agence Nationale de la Recherche (ANR) under contract ANR-11-MONU-008-002.

Appendix A. Baseline SSC algorithm

This appendix provides the general pseudo code of the baseline SSC algorithm with ENO stencils.

Compute initial d-dimensional Delaunay grid
Compute initial compute code samples $\mathbf{v} = (v_0, \cdots, v_k, \cdots, v_d)$ at the $2^d + 1$ grid points
Set initial hierarchical surplus errors to $\epsilon_k = -v_k$
$i \leftarrow 0$, choose max iterations I
while $i \leq I$ and $\hat{\epsilon}_{rms} >$ user-defined threshold **do**
  Determine p_{max} via (5)
  Compute nearest-neighbor stencils via (7)
  % Check (in parallel) the LEC condition:
  for $j = 1 \cdots, n_e$ **do**
    † Sample all $\Xi_j \in S_j$ via (25)
    At sample locations: compute w_j
    if (14) is violated in one or more sample locations **then**
      $p_j \leftarrow p_j - 1$
      Update S_j and goto †
    end if
  end for
  % Compute (in parallel) ENO stencils
  for $j = 1 \cdots, n_e$ **do**
    Collect the r nearest neighbor S_j that contain Ξ_j
    From these r stencils, select $S_j^* :=$ the S_j with the largest p_j
    if more than 1 S_j^* exists **then**
      Select the S_j^* with the smallest average Euclidean distance to the cell center of Ξ_j
    end if
  end for
  Compute probabilities Ω_j and volumes $\bar{\Xi}_j$ via (10)
  Compute refinement measures $\bar{e}_j$ via (9)
  Sample the sub-simplices (13) of the N simplices with the largest $\bar{e}_j$ via (25)
  Evaluate (in parallel) computer code at the N new sample locations, add code samples to $\mathbf{v}$
  Compute the N new hierarchical surplus values via (15)
  Refine the Delaunay grid by adding the N new sample locations
end while
Compute ENO stencils and check LEC condition from final iteration

Appendix B. SSC-SC algorithm

Below we give the detailed construction of SCENO stencils in pseudo code, and we repeat some definitions for convenience.

- $\mathcal{C}$ is the set of simplices which are currently covered by a stencil S_j.
- $\mathcal{C}_j$ is the set of simplices currently under construction which will be added to $\mathcal{C}$.
- $\mathcal{N}_j$ is the set of all neighboring simplices Ξ_k of simplex Ξ_j.
- $\mathcal{N}_{\mathcal{C}_j}$ is the set of all neighboring simplices Ξ_k of all simplices in $\mathcal{C}_j$.
- $\{\boldsymbol{\xi}_k\}$ is the set of $d+1$ vertices $\boldsymbol{\xi}_{k_{j,l}}$ that make up simplex Ξ_k.

% for each iteration of the SSC-SC algorithm do
compute $\mathcal{D}$ via (26)
$\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{D}$
%select the largest simplex in terms of volume or size
$\Xi_j^* \leftarrow \arg\max_{\Xi_j} \bar{\Xi}_j$

⊛*% loop while $\mathcal{C}$ does not cover $\mathcal{U}$*
while $\mathcal{U} \nsubseteq \bigcup_{X_j \in \mathcal{C}} X_j$ **do**
  % initialize all sets
  $\mathcal{C}_j \leftarrow \{\Xi_j^*\}$
  $S_j \leftarrow \{\boldsymbol{\xi}_{k_{j,l}} \mid \Xi_j \in \mathcal{C}_j\}$
  $\mathcal{N}_{\mathcal{C}_j} \leftarrow \{\Xi_k \mid |\{\boldsymbol{\xi}_k\} \cap \{\boldsymbol{\xi}_j\}| = d;\ \forall\, \Xi_j \in \mathcal{C}_j \wedge k = \{1, 2, \cdots, n_e\} \setminus \{j\}\}$

  % loop until no more uncovered neighbors are available or S_j is full
  while $\mathcal{N}_{\mathcal{C}_j} \setminus \{\mathcal{C} \cup \mathcal{C}_j\} \neq \emptyset$ and $|S_j| < N_j + 1$ **do**
    % update all sets
    $\mathcal{C}_j \leftarrow \mathcal{C}_j \cup \{\mathcal{N}_{\mathcal{C}_j} \setminus \{\mathcal{C} \cup \mathcal{C}_j\}\}$
    $S_j \leftarrow \{\boldsymbol{\xi}_{k_{j,l}} \mid \Xi_j \in \mathcal{C}_j\}$
    $\mathcal{N}_{\mathcal{C}_j} \leftarrow \{\Xi_k \mid |\{\boldsymbol{\xi}_k\} \cap \{\boldsymbol{\xi}_j\}| = d;\ \forall\, \Xi_j \in \mathcal{C}_j \wedge k = \{1, 2, \cdots, n_e\} \setminus \{j\}\}$
  end while

  sort $\mathcal{C}_j$ and S_j according to $\|\boldsymbol{\xi}_{center_k} - \boldsymbol{\xi}_{center_j}\|_2$

  if $|S_j| < N_j + 1$ **then**
    reduce p_j and S_j to new maximum as allowed by (5)
  end if

  $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_j$
  $\mathcal{S}_{sc} \leftarrow \mathcal{S}_{sc} \cup S_j$
end while

%check LEC condition for all $S_j \in \mathcal{S}_{sc}$
if LEC is violated in any S_j **then**
  $p_j \leftarrow p_j - 1$
  update $\mathcal{C}_j$, S_j
  update $\mathcal{C}$, $\mathcal{S}_{sc}$
  goto ⊛
end if

Appendix C. HDMR algorithm

The dimension-adaptive cut-HDMR of [20] coupled with the SSC method.

%Initialize sets
$\mathcal{V}_{imp} = \{\emptyset\}$, $\mathcal{V}_{comp} = \{\emptyset\}$, $\mathcal{R} = \{\emptyset\}$, $p = 1$

%Compute the component functions of order $p = 0$ and $p = 1$
Compute all sub problems $f^{(\mathbf{v})}$ in (36) using the SSC method. Stop when global RMS error measure (17) $< \epsilon$. Add all computed $\mathbf{u}$ to $\mathcal{V}_{comp}$.

Compute all first-order weights (38) α_i
if $\alpha_i > \epsilon_1$ **then**
  $\mathcal{V}_{imp} \leftarrow \mathcal{V}_{imp} \cup \{i\}$
end if
Add $\mathbf{u}$ with $|\mathbf{u}| = p$ and which satisfy admissibility condition to $\mathcal{R}$

while $\mathcal{R} \neq \{\emptyset\}$ and $\alpha_p > \epsilon_2$ **do**
  $p \leftarrow p + 1$
  Add $\mathbf{u}$ with $|\mathbf{u}| = p$ and which satisfy admissibility condition (41) to $\mathcal{R}$
  $\forall \mathbf{u} \in \mathcal{R}$, compute $f^{(\mathbf{v})}$ in (36) using SSC, stop when (17) $< \epsilon$, add all computed $\mathbf{u}$ to $\mathcal{V}_{comp}$.
  Compute weights $\alpha_{\mathbf{u}}$ (40)
  if $\alpha_{\mathbf{u}} > \epsilon_1$ **then**
    $\mathcal{V}_{imp} \leftarrow \mathcal{V}_{imp} \cup \{\mathbf{u}\}$
  end if
  $\mathcal{R} = \{\emptyset\}$, add $\mathbf{u}$ with $|\mathbf{u}| = p$ and which satisfy admissibility condition to $\mathcal{R}$
  Compute relative error measure α_p (43)
end while

Appendix D. Proof of uniform distribution

Equation (25), repeated below for convenience, is used to map points from the hypercube $K_d := [0,1]^d$ to an arbitrary simplex Ξ described by the (unique) points $\boldsymbol{\xi}_i \in \mathbb{R}^d$, $i = 1, \cdots, d+1$. The $R_j \in \mathbb{R}$ are d scalar i.i.d. random variables (r.v.'s) distributed uniformly as $\mathcal{U}[0,1]$, and describe a randomly picked point in K_d. This appendix contains the proof that (D.1) is distributed uniformly as well.

$$M_d = \boldsymbol{\xi}_0 + \sum_{i=1}^{d} \prod_{j=1}^{i} r_{d-j+1}^{\frac{1}{d-j+1}} (\boldsymbol{\xi}_i - \boldsymbol{\xi}_{i-1}). \tag{D.1}$$

From [29] we have the following theorem regarding the distribution of a transformation of random variables:

Theorem 3. *Consider the r.v.'s $R_1, \cdots, R_d$ with joint pdf $f_{R_1 \cdots R_d}$ positive and continuous on the set $A \subseteq \mathbb{R}^d$, and let $h_1, \cdots, h_d$ be real-valued transformations defined on A; that is, $h_1, \cdots, h_d \to \mathbb{R}$, and let B be the image of A under transformations $(h_1, \cdots, h_d)$. Suppose that $(h_1, \cdots, h_d)$ is one-to-one from A onto B. Thus, if we set $y_i = h_i(r_1, \cdots, r_d)$, we can uniquely solve for r_i, $i = 1, \cdots, d$: $r_i = h_i^{-1}(y_1, \cdots, y_d)$, $i = 1, \cdots, d$. Suppose further that the partial derivatives $\frac{\partial}{\partial y_j} h_i^{-1}$, $i, j = 1, \cdots, d$ exist and are continuous for $(y_1, \cdots, y_d) \in B$. Finally, suppose that the Jacobian*

$$J(y_1, \cdots, y_d) = \begin{bmatrix} \frac{\partial h_1^{-1}}{\partial y_1} & \cdots & \frac{\partial h_d^{-1}}{\partial y_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_1^{-1}}{\partial y_d} & \cdots & \frac{\partial h_d^{-1}}{\partial y_d} \end{bmatrix} \tag{D.2}$$

is $\neq 0$ on B. Then the joint pdf of the r.v.'s $Y_i = h_i(R_1, \cdots, R_d)$, $i = 1, \cdots, d$, $f_{Y_1 \cdots Y_d}$, is given by:

$$f_{Y_1 \cdots Y_d} = \begin{cases} |\det J(y_1, \cdots, y_d)| \cdot f_{R_1 \cdots R_d}\left(h_1^{-1}(y_1, \cdots, y_d), \cdots, h_d^{-1}(y_1, \cdots, y_d)\right), & (y_1, \cdots, y_d) \in B \\ 0 & \text{otherwise} \end{cases} \tag{D.3}$$

Table D.13
The absolute value of the determinant of J as a function of d.

d	1	2	3	4	5	6	7	8
$\lvert\det J\rvert$	1	2	6	24	120	720	5040	40 320

In our case A is the hypercube K_d and B is the target simplex Ξ. Also, we have uniform i.i.d. R_j such that $f_{R_1\cdots R_d} = f_{R_1} f_{R_2}\cdots f_{R_d} = 1$. Thus, in order for $f_{Y_1\cdots Y_d}$ to be uniform we need to show that $|J(y_1,\cdots,y_d)|$ is a constant. Furthermore, since (D.1) maps to a simplex this constant must be equal to the reciprocal of the volume of the simplex.

To simplify the analysis we will consider the standard simplex with nodes $\boldsymbol{\xi}_0 = (0,0,\cdots,0)$, $\boldsymbol{\xi}_1 = (1,0,\cdots,0)$, $\boldsymbol{\xi}_2 = (0,1,\cdots,0)$ etc. To demonstrate the structure of (D.1), we provide a four-dimensional example here:

$$M_4 = \begin{bmatrix} \sqrt[4]{r_4} - \sqrt[4]{r_4}\sqrt[3]{r_3} \\ \sqrt[4]{r_4}\sqrt[3]{r_3} - \sqrt[4]{r_4}\sqrt[3]{r_3}\sqrt{r_2} \\ \sqrt[4]{r_4}\sqrt[3]{r_3}\sqrt{r_2} - \sqrt[4]{r_4}\sqrt[3]{r_3}\sqrt{r_2}r_1 \\ \sqrt[4]{r_4}\sqrt[3]{r_3}\sqrt{r_2}r_1 \end{bmatrix}. \tag{D.4}$$

Each individual row of M_4 is a transformation function $y_i = h_i(r_1,\cdots,r_4)$, for which we can find the following inverse functions:

$$r_1 = \frac{y_4}{y_4+y_3}, r_2 = \frac{(y_4+y_3)^2}{(y_4+y_3+y_2)^2}, r_3 = \frac{(y_4+y_3+y_2)^3}{(y_1+y_4+y_3+y_2)^3}, r_4 = (y_1+y_4+y_3+y_2)^4. \tag{D.5}$$

Now we can compute the Jacobian matrix (D.2) as

$$J = \begin{bmatrix} 0 & 0 & -\frac{y_4}{(y_4+y_3)^2} & \frac{y_3}{(y_4+y_3)^2} \\ 0 & -2\frac{(y_4+y_3)^2}{(y_4+y_3+y_2)^3} & 2\frac{(y_4+y_3)y_2}{(y_4+y_3+y_2)^3} & 2\frac{(y_4+y_3)y_2}{(y_4+y_3+y_2)^3} \\ -3\frac{(y_4+y_3+y_2)^3}{(y_1+y_4+y_3+y_2)^4} & 3\frac{(y_4+y_3+y_2)^2 y_1}{(y_1+y_4+y_3+y_2)^4} & 3\frac{(y_4+y_3+y_2)^2 y_1}{(y_1+y_4+y_3+y_2)^4} & 3\frac{(y_4+y_3+y_2)^2 y_1}{(y_1+y_4+y_3+y_2)^4} \\ 4(y_1+y_4+y_3+y_2)^3 & 4(y_1+y_4+y_3+y_2)^3 & 4(y_1+y_4+y_3+y_2)^3 & 4(y_1+y_4+y_3+y_2)^3 \end{bmatrix}. \tag{D.6}$$

When we compute the determinant of J all the y_i terms drop out and we end up with $|\det J| = 24$. The values of $|\det J|$ for $d = 1,\cdots,8$ can be found in Table D.13. From these results it becomes clear that $f_{Y_1\cdots Y_d} = |\det J| = d!$. The volume $\bar{\Xi}$ of a simplex Ξ can be computed by

$$\bar{\Xi} = \frac{1}{d!}|\det(D)|, \quad D = \begin{bmatrix} \boldsymbol{\xi}_1 - \boldsymbol{\xi}_0 & \boldsymbol{\xi}_2 - \boldsymbol{\xi}_0 & \cdots & \boldsymbol{\xi}_{d+1} - \boldsymbol{\xi}_0 \end{bmatrix}, \tag{D.7}$$

which for the standard simplex reduces to $\bar{\Xi} = \frac{1}{d!}$. And thus we have

$$\int\cdots\int_{\Xi} f_{Y_1\cdots Y_d} \mathrm{d}Y_1\cdots\mathrm{d}Y_d = f_{Y_1\cdots Y_d}\int\cdots\int_{\Xi}\mathrm{d}Y_1\cdots\mathrm{d}Y_d = d!\cdot\frac{1}{d!} = 1, \tag{D.8}$$

which completes the proof.

References

[1] N. Amenta, D. Attali, O. Devillers, Complexity of Delaunay triangulation for points on lower-dimensional polyhedra, in: Proceedings of the Eighteenth Annual ACM–SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2007, pp. 1106–1113.

[2] J.D. Anderson, Fundamentals of Aerodynamics, vol. 2, McGraw-Hill, New York, 2001.

[3] J. Axerio-Cilies, G. Petrone, V. Sellappan, G. Iaccarino, Extreme ensemble computation for optimization under uncertainty, in: Evolutionary and Deterministic Methods for Design, Optimization and Control, 2011.

[4] I. Babuška, F. Nobile, R. Tempone, A stochastic collocation method for elliptic partial differential equations with random input data, SIAM J. Numer. Anal. 45 (3) (2007) 1005–1034.

[5] C.B. Barber, H. Huhdanpaa, The qhull software library, http://www.geom.umn.edu/software/qhull.

[6] Jean-Daniel Boissonnat, Olivier Devillers, Samuel Hornus, Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension, in: Proceedings of the Twenty-Fifth Annual Symposium on Computational Geometry, ACM, 2009, pp. 208–216.

[7] K.C. Chung, T.H. Yao, On lattices admitting unique Lagrange interpolations, SIAM J. Numer. Anal. 14 (4) (1977) 735–743.

[8] P. Cinnella, P.M. Congedo, V. Pediroda, L. Parussini, Sensitivity analysis of dense gas flow simulations to thermodynamic uncertainties, Phys. Fluids 23 (11) (2011) 116101.

[9] P.M. Congedo, J. Witteveen, G. Iaccarino, A simplex-based numerical framework for simple and efficient robust design optimization, Comput. Optim. Appl. 56 (1) (2013) 231–251.

[10] P.M. Congedo, J. Witteveen, G. Iaccarino, et al., Simplex–simplex approach for robust design optimization, in: EUROGEN 2011, International Conferences on Evolutionary Computing for Industrial Applications – ECCOMAS Thematic Conference, 2011.

[11] W.N. Edeling, P. Cinnella, R.P. Dwight, Predictive rans simulations via Bayesian model-scenario averaging, J. Comput. Phys. 275 (2014) 65–91.
[12] J. Foo, G.E. Karniadakis, Multi-element probabilistic collocation method in high dimensions, J. Comput. Phys. 229 (5) (2010) 1536–1557.
[13] J. Foo, X. Wan, G.E. Karniadakis, The multi-element probabilistic collocation method (me-pcm): error analysis and applications, J. Comput. Phys. 227 (22) (2008) 9572–9595.
[14] B. Ganapathysubramanian, N. Zabaras, Sparse grid collocation schemes for stochastic natural convection problems, J. Comput. Phys. 225 (1) (2007) 652–685.
[15] S. Guillas, N. Glover, L. Malki-Epshtein, Bayesian calibration of the constants of the k–ε turbulence model for a cfd model of street canyon flow, Comput. Methods Appl. Mech. Eng. 279 (2014) 536–553.
[16] R.M. Karp, Reducibility Among Combinatorial Problems, Springer, 1972.
[17] K.J. Kozaczek, S.K. Kurtzl, D.S. Kurtz, A direct algorithm for solving ill-conditioned linear algebraic systems, Adv. X-Ray Anal. 42 (2000).
[18] G.J.A. Loeven, J.A.S. Witteveen, H. Bijl, Probabilistic collocation: an efficient non-intrusive approach for arbitrarily distributed parametric uncertainties, in: Proceedings of the 45th AIAA Aerospace Sciences Meeting, vol. 6, 2007, pp. 3845–3858.
[19] X. Ma, N. Zabaras, An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations, J. Comput. Phys. 228 (8) (2009) 3084–3113.
[20] X. Ma, N. Zabaras, An adaptive high-dimensional stochastic model representation technique for the solution of stochastic partial differential equations, J. Comput. Phys. 229 (10) (2010) 3884–3915.
[21] X. Merle, P. Cinnella, Bayesian quantification of thermodynamic uncertainties in dense gas flows, Reliab. Eng. Syst. Saf. (2014).
[22] F. Nobile, R. Tempone, C.G. Webster, A sparse grid stochastic collocation method for partial differential equations with random input data, SIAM J. Numer. Anal. 46 (5) (2008) 2309–2345.
[23] P.J. Olver, On multivariate interpolation, Stud. Appl. Math. 116 (2) (2006) 201–240.
[24] R. Pecnik, J.A.S. Witteveen, G. Iaccarino, Uncertainty quantification for laminar-turbulent transition prediction in rans turbomachinery applications, AIAA Pap. 2011 (2011) 1–14.
[25] M. Pini, P. Cinnella, Hybrid adjoint-based robust optimization approach for fluid-dynamics problems, in: 15th Non-Deterministic Approaches Conference, American Institute of Aeronautics and Astronautics, 2013.
[26] H. Rabitz, Ö.F. Aliş, General foundations of high-dimensional model representations, J. Math. Chem. 25 (2–3) (1999) 197–233.
[27] H. Rabitz, Ö.F. Aliş, J. Shorter, K. Shim, Efficient input–output model representations, Comput. Phys. Commun. 117 (1) (1999) 11–20.
[28] R. Rojas, Sa. Kahunde, L. Peeters, O. Batelaan, L. Feyen, A. Dassargues, Application of a multimodel approach to account for conceptual model and scenario uncertainties in groundwater modelling, J. Hydrol. 394 (3) (2010) 416–435.
[29] G.G. Roussas, An Introduction to Probability and Statistical Inference, Academic Press, 2003.
[30] S.A. Smolyak, Quadrature and interpolation formulas for tensor products of certain classes of functions, Dokl. Akad. Nauk SSSR 4 (1963) 240–243.
[31] I.M. Sobol, Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates, Math. Comput. Simul. 55 (1–3) (2001) 271–280.
[32] P.R. Spalart, S.R. Allmaras, A one-equation turbulence model for aerodynamic flows, Rech. Aérosp. 1 (1) (1994) 5–21.
[33] G. Turk, Graphics Gems, Academic Press Professional, Inc., San Diego, CA, USA, 1990, pp. 24–28 (chapter Generating random points in triangles).
[34] D.C. Wilcox, Turbulence Modeling for CFD, vol. 3, DCW Industries La, Canada, CA, 2006.
[35] J.A.S. Witteveen, G. Iaccarino, Simplex elements stochastic collocation in higher-dimensional probability spaces, in: 12th AIAA Non-Deterministic Approaches Conference, Orlando, Florida, in: AIAA, vol. 2924, 2010.
[36] J.A.S. Witteveen, G. Iaccarino, Introducing essentially non-oscillatory stencil selection with subcell resolution into uncertainty quantification, in: Annual Research Briefs, Center for Turbulence Research, 2011, pp. 169–180.
[37] J.A.S. Witteveen, G. Iaccarino, Refinement criteria for simplex stochastic collocation with local extremum diminishing robustness, SIAM J. Sci. Comput. 34 (3) (2012) A1522–A1543.
[38] J.A.S. Witteveen, G. Iaccarino, Simplex stochastic collocation with random sampling and extrapolation for nonhypercube probability spaces, SIAM J. Sci. Comput. 34 (2) (2012) A814–A838.
[39] J.A.S. Witteveen, G. Iaccarino, Simplex stochastic collocation with eno-type stencil selection for robust uncertainty quantification, J. Comput. Phys. 239 (2013) 1–21.
[40] J.A.S. Witteveen, G. Iaccarino, Subcell resolution in simplex stochastic collocation for spatial discontinuities, J. Comput. Phys. 251 (2013) 17–52.
[41] J.A.S. Witteveen, A. Loeven, H. Bijl, An adaptive stochastic finite elements approach based on Newton–Cotes quadrature in simplex elements, Comput. Fluids 38 (6) (2009) 1270–1288.