

On the use of derivatives in the polynomial chaos based global sensitivity and uncertainty analysis applied to the distributed parameter models

Igor Gejadze, Pierre-Olivier Malaterre, Victor Shutyaev

▶ **To cite this version:**

Igor Gejadze, Pierre-Olivier Malaterre, Victor Shutyaev. On the use of derivatives in the polynomial chaos based global sensitivity and uncertainty analysis applied to the distributed parameter models. Journal of Computational Physics, 2019, 381, pp.218-245. 10.1016/j.jcp.2018.12.023 . hal-02608782

HAL Id: hal-02608782
https://hal.inrae.fr/hal-02608782

Submitted on 29 Jul 2021

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

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



Distributed under a Creative Commons Attribution 4.0 International License

On the use of derivatives in the polynomial chaos based global sensitivity and uncertainty analysis applied to the distributed parameter models

I. Gejadze [a,*], P.-O. Malaterre [a], V. Shutyaev [b]

[a] *UMR G-EAU, IRSTEA Centre de Montpellier 361 Rue J.F. Breton, BP 5095, 34196, Montpellier, France*
[b] *Marchuk Institute of Numerical Mathematics, Russian Academy of Sciences (RAS), Federal State Budget Scientific Institution "Marine Hydrophysical Institute of RAS", Moscow Institute for Physics and Technology, 119333 Gubkina 8, Moscow, Russia*

Sensitivity Analysis (SA) and Uncertainty Quantification (UQ) are important components of modern numerical analysis. However, solving the relevant tasks involving large-scale fluid flow models currently remains a serious challenge. The difficulties are associated with the computational cost of running such models and with large dimensions of the discretized model input. The common approach is to construct a metamodel – an inexpensive approximation to the original model suitable for the Monte Carlo simulations. The polynomial chaos (PC) expansion is the most popular approach for constructing metamodels. Some fluid flow models of interest are involved in the process of variational estimation/data assimilation. This implies that the tangent linear and adjoint counterparts of such models are available and, therefore, computing the gradient (first derivative) and the Hessian (second derivative) of a given function of the state variables is possible. New techniques for SA and UQ which benefit from using the derivatives are presented in this paper. The gradient-enhanced regression methods for computing the PC expansion have been developed recently. An intrinsic step of the regression method is a minimization process. It is often assumed that generating 'data' for the regression problem is significantly more expensive that solving the regression problem itself. This depends, however, on the size of the importance set, which is a subset of 'influential' inputs. A distinguishing feature of the distributed parameter models is that the number of the such inputs could still be large, which means that solving the regression problem becomes increasingly expensive. In this paper we propose a derivative-enhanced projection method, where no minimization is required. The method is based on the explicit relationships between the PC coefficients and the derivatives, complemented with a relatively inexpensive filtering procedure. The method is currently limited to the PC expansion of the third order. Besides, we suggest an improved derivative-based global sensitivity measure. The numerical tests have been performed for the Burgers' model. The results confirm that the low-order PC expansion obtained by our method represents a useful tool for modeling the non-gaussian behavior of the chosen quantity of interest (QoI).

* Corresponding author.
E-mail addresses: igor.gejadze@irstea.fr (I. Gejadze), p-o.malaterre@irstea.fr (P.-O. Malaterre), victor.shutyaev@mail.ru (V. Shutyaev).

Introduction

Sensitivity Analysis (SA) and Uncertainty Quantification (UQ) are important components of modern numerical analysis [8]. In SA one studies the relative importance of different inputs on the model output, which is usually represented by the quantities of interest (QoI). The main purpose of SA is the design of optimal (and feasible) control, UQ and data assimilation (DA) algorithms. In UQ one focuses on quantifying the uncertainty in the QoI, given the uncertainty in the model inputs. The most straightforward UQ method is the direct Monte Carlo method, where the model is run in a loop with the perturbed inputs; then the sample of QoI is processed to obtain its statistical characterization (skill scores, for example). The basic Monte Carlo method has a number of more sophisticated variants, e.g. using the importance sampling [46], quasi-random sequences [43], etc. However, for computationally expensive fluid flow models such as those common in geophysics, these methods are not particularly suitable. While in the absence of a viable alternative the Monte Carlo method is presently used in weather and ocean forecasting (ensemble forecasting, [27]), the need for development of more efficient methods is obvious.

One possible approach is to use the 'local' sensitivities. In particular, the corresponding SA and UQ methods have been recently applied in the field of computational fluid dynamics [22,38,3,15]. Here, the adjoint (local) sensitivities are used to evaluate the variance of the uncertainty in QoI (or the covariance for a vector of QoI). The uncertainty in the input vector can also be either prior or posterior, i.e. before and after DA, respectively. The posterior covariance of the input uncertainty is approximated by the inverse Hessian of the auxiliary data assimilation cost function [16]. The adjoint approach is computationally very efficient, however its accuracy for nonlinear models depends on the validity of the tangent linear approximation for the error dynamics (case of weak nonlinearity and/or small uncertainties). This is a key assumption in the ensemble forecasting [27], since the method relies on computing the singular vectors of the propagator considered at the optimal solution point. Besides, in the case of strongly nonlinear dynamics, both the posterior input uncertainty and the uncertainty in QoI may have non-gaussian probability distributions, which means that knowing the variance/covariance is not sufficient. Thus, the global sensitivity based methods have to be considered.

An alternative option is to develop the reduced-order dynamical models, either using the proper orthogonal decomposition (POD) [48,28] or the dynamic mode decomposition (DMD) [39,35]. In the POD method, the distributed state variables are represented by the linear combinations of the spatial modes. These representations are substituted into the model equations and the Galerkin projection is applied. In this process the original model governed by the nonlinear partial differential equations is transformed into the system of nonlinear ordinary differential equations for the time dependent weights associated with the POD modes. The resulting 'surrogate' model is much less expensive than the original one, and it is generally suitable for sampling. The method is intrusive in the sense that the surrogate model have to be actually created, which means an additional development effort. This is not always easy as well, despite several successful examples being reported. In the DMD method, one looks for a (linear) Koopman operator (modes, eigenvalues and eigenfunctions), which defines the evolution model explicitly. Thus, the DMD method is non-intrusive, but relies on a linear approximation of the original propagator. It is not obvious that using this type of linearized model for UQ is better than just using the local sensitivities.

Another approach is to use a phenomenological model, i.e. a model which describes the relationship between the model inputs and outputs without attempting to explain why the variables interact the way they do. Such model can be based on the polynomial chaos (PC) expansion (for a concise review of the PC methods the reader may refer to [26]).

In the intrusive version of the PC-based UQ technique, called the 'direct nonsampling method', see [30], the uncertainty-loaded model variables are expressed using the PC expansions. These are substituted into the model equations and the Galerkin projection is applied. Technically this approach is somewhat similar to the one used with POD, however the resulting model has to be solved just once (no sampling required).

In our paper we focus on a **non-intrusive approach**, where the PC expansion describes the relationship between the model inputs and outputs directly. The task of constructing the continuous response surface, given a finite sample of model responses, is solved by the two main approaches: regression and projection. In the regression methods one has to minimize the appropriate cost function. A few types of regression methods are used: the least squares regression [5,11,21], l_1-minimization [33], and least angle regression [7], adaptive methods [1]. In the projection methods one benefits from orthogonality of the polynomial bases involved, which allows the PC expansion to be computed as multidimensional integrals [49,14,17,18]. In its original form, i.e. involving a full-tensor product quadrature, this method is not feasible for high-dimensional problems. The major upgrades of this method have been destined to improve the integration efficiency. They include, for example, the use of sparse grid sampling techniques [40], nested quadratures (Clenshaw–Curtis), quasi-random sequences [43] or Latin Hypercube sampling [9]. In all cases, however, the integrals over the model response function (QoI) are considered. As opposed to this classical approach, in the presented paper the integrals over the **derivatives** of the model response function are considered, which gives some essential benefits.

Variational data assimilation (DA) is a method based on the optimal control theory [25,29]. Presently, this method is preferred for weather and ocean forecasting in major operational centers around the globe, particularly in the form of the incremental 4D-Var [13], and in the form of the ensemble 4D-Var [12]. In general, the variational DA method belongs to the family of variational estimation methods, which are widely used in a variety of applications, including geophysics, astrophysics, bio-medical, mechanical and aerospace engineering. One useful bonus associated with the numerical models involved into variational estimation is the availability of the **adjoint model** which helps the gradient (first derivative) of the standard data mismatch cost-function to be calculated in a single model run. This adjoint model can be easily adapted to

compute the gradient of any other QoI being defined as a function of the state variables. Moreover, computing the Hessian (second derivative) of QoI is also possible, though more expensive, with a modification of the existing adjoint model into the second-order adjoint model [47]. Thus, at least two first derivatives can be available in the discussed framework for the specified group of models.

The idea of the gradient-enhanced methods for computing the PC expansion has been introduced and implemented almost simultaneously by several authors [2,34,32] in the regression framework. In this paper we propose a **derivative-enhanced projection** method. It has been mentioned that the major drawback of the projection methods is a slow convergence of the corresponding integrals. However, the use of the derivatives accelerates this convergence. For example, if the gradients are used, the convergence of the corresponding integrals for the first- and higher-order PC coefficients is accelerated by order of magnitude. Besides, we develop a filtering procedure which allows the sampling error to be further reduced. Another benefit of using the gradients is an easy way of accessing the 'total effect' global sensitivity indices [42]. These are used for ranking and defining the importance set, i.e. a presumably small subset of the full input including the most influential components. The proposed method is currently limited to the case of the third-order PC expansion. For the distributed parameter systems a large dimension of the full model input is expected. Moreover, the dimension of the importance set can still be large, in which case considering the PC expansion of the fourth- and higher orders does not look feasible anyway. While the method is devised for the gaussian inputs (i.e. using Hermit polynomials), it can be extended to different types of inputs using other polynomials of the Askey scheme.

The paper is organized as follows. In §1 we define QoI as a function of the model input vector (called 'design function' hereafter), and its PC expansion using the 'probabilistic' Hermit polynomials. In §2 we describe the regression approach, its gradient-enhanced version and the proposed Hessian-enhanced version. The derivative-enhanced projection method introduced in this paper is based on the explicit expressions for the PC expansion coefficients via the derivatives of arbitrary order, presented in §3 (see Theorem 1, proof – in Appendix 1). In §4 the relationship between the known derivative-based sensitivity measure [24] and the global sensitivity indices is analyzed (see Theorem 2, proof – in Appendices 2 and 3), and an improved derivative-based sensitivity estimate is suggested (see Corollary 2). The implementation details, including the description of the filtering algorithms, are presented in §5. Taking into account that the PC coefficients are estimated and represented in the limited-memory form, the appropriate expressions for the metamodels are derived in §6. The proposed method is validated using the Burgers' model: the model equation, the adjoint equation and the Hessian-vector product operator are defined in §7. The description of numerical experiments and the results of simulations are presented in §8. The main outcomes of the paper are summarized in the Conclusions, and major mathematical proofs and derivations are collected in the Appendixes.

1. Problem setup

Let us consider the numerical model describing behavior of a dynamical system in terms of its state variables $X \in \mathcal{X}$. Given a full set of the model inputs $U \in \mathcal{U}$, the model can be considered as a 'control-to-state' mapping $\mathcal{M} : \mathcal{U} \to \mathcal{X}$

$$X = \mathcal{M}(U), \tag{1}$$

where $\mathcal{X}$ and $\mathcal{U}$ are the state and control spaces, correspondingly.

For modeling the system behavior the true input set $\bar{U}$ must be specified. Under the 'perfect model' assumption the following can be postulated: $\bar{X} = \mathcal{M}(\bar{U})$. In reality, some components of $\bar{U}$ contain uncertainties $\bar{\xi} \in \mathcal{U}$. Thus, instead of $\bar{U}$ we use its best available approximation, either prior (background) or posterior (estimate):

$$U^* = \bar{U} + \bar{\xi}. \tag{2}$$

Because of the presence of $\bar{\xi}$, the predicted state $X|U^*$, that is, X evaluated (or conditioned) on U^*, also contains an error $\delta X = X|U^* - X|\bar{U}$.

In many practical situations the quantities of interest (QoI) are represented by the functional of the state $D(X) \in \mathcal{G}$, where $\mathcal{G}$ is the 'design' space and $D : \mathcal{X} \to \mathcal{G}$ is a linear or nonlinear mapping. Thus, one can define a generalized 'control-to-design' mapping $G : \mathcal{U} \to \mathcal{G}$, such that

$$D(\mathcal{M}(U)) = G(U). \tag{3}$$

The uncertainty $\bar{\xi}$ results into the error

$$\varepsilon = G(\bar{U} + \bar{\xi}) - G(\bar{U}). \tag{4}$$

From now on we shall call ε – the design function. Quantification of ε is an important task in a variety of applications.

Below we suppose that the input space is finite-dimensional, i.e. $\mathcal{U} = \mathbb{R}^N$, and $\bar{\xi} = (\bar{\xi}_1, \ldots, \bar{\xi}_N)^T$. Consider $\bar{\xi}$ is the form

$$\bar{\xi} = R(\xi), \tag{5}$$

where $\xi = (\xi_1, \ldots, \xi_N)^T$ with $\xi_i \sim N(0,1)$ being i.i.d. gaussian random variables with unit variance, and R is a transformation from the standard normal to a desired probability distribution. For example, if $R(\xi) = B^{1/2}\xi$, then one obtains $\bar{\xi} \sim N(0, B)$; if $R(\xi) = \mu + \sigma exp(\xi)$, one gets the log-normally distributed $\bar{\xi}$. Thus, the design function can finally be defined in the form

$$\varepsilon(\xi) = G(\bar{U} + R(\xi)) - G(\bar{U}). \tag{6}$$

Let us consider the polynomial expansion of $\varepsilon(\xi)$ using 'probabilistic' Hermite polynomials:

$$\varepsilon(\xi) = b_0 H_0 + \sum_{i_1=1}^{N} b_{i_1} H_1(\xi_{i_1}) + \sum_{i_1=1}^{N}\sum_{i_2=1}^{i_1} b_{i_1 i_2} H_2(\xi_{i_1}, \xi_{i_2}) + \sum_{i_1=1}^{N}\sum_{i_2=1}^{i_1}\sum_{i_3=1}^{i_2} b_{i_1 i_2 i_3} H_3(\xi_{i_1}, \xi_{i_2}, \xi_{i_3}) + \ldots +$$
$$\sum_{i_1=1}^{N}\sum_{i_2=1}^{i_1}\cdots\sum_{i_p=1}^{i_{p-1}} b_{i_1 i_2 \ldots i_p} H_p(\xi_{i_1}, \xi_{i_2}, \ldots, \xi_{i_p}), \tag{7}$$

where

$$H_0 = 1,\ H_1(\xi_{i_1}) = \xi_{i_1},\ H_2(\xi_{i_1}, \xi_{i_2}) = \xi_{i_1}\xi_{i_2} - \delta_{i_1 i_2},$$
$$H_3(\xi_{i_1}, \xi_{i_2}, \xi_{i_3}) = \xi_{i_1}\xi_{i_2}\xi_{i_3} - \xi_{i_1}\delta_{i_2 i_3} - \xi_{i_2}\delta_{i_1 i_3} - \xi_{i_3}\delta_{i_1 i_2},\ \ldots \tag{8}$$

The convergence of the series (7) for the gaussian entries ξ is proved in [10]. The total number of coefficients $\boldsymbol{b} = (b_0, \{b_{i_1}\}, \{b_{i_1 i_2}\}, \ldots, \{b_{i_1 \ldots i_p}\})^T$ is given by

$$M = \frac{(N+p)!}{N!\,p!}, \tag{9}$$

where p is the chosen order of polynomials. Representation (7) is often referred as the polynomial chaos (PC) expansion.

2. Implicit approach (least-square, regression)

According to (9), computing the PC expansion is feasible only for small N and p. Thus, one should define the inputs U' (and, correspondingly, ξ') having the most impact on $\varepsilon(\xi)$. These inputs constitute the importance set of size N', which is expected to be much smaller than N. For the importance set, the vector of PC coefficients is $\boldsymbol{b}'$. Below we deal with the importance set, keeping however the original notations (i.e. without $'$) for the sake of simplicity.

Let us represent (7) in the dot product form

$$\varepsilon(\xi) = \boldsymbol{H}(\xi)\boldsymbol{b},$$

where $\boldsymbol{H}(\xi)$ is a vector of weights (i.e. the values of the corresponding Hermit polynomials). In the least square regression method one looks for the minimum of the cost-function

$$J = \Omega(\boldsymbol{b}) + J_0, \tag{10}$$

with

$$J_0 = \sum_{l=1}^{L}\left(\varepsilon^*(\xi^l) - \boldsymbol{H}(\xi^l)\boldsymbol{b}\right)^2, \tag{11}$$

where $\varepsilon^*(\xi^l)$ is the computed value of the design function (6) at point ξ^l, and $\Omega(\boldsymbol{b})$ is a regularization term. The latter should be used if the system is under-determined and other types of regularization are not involved. In practice, in order to get a sensible estimate of $\boldsymbol{b}$ one needs an over-determined system $L = kM$, where $k > 2$ at least. Let us mention that the least square method is not the only suitable option for finding vector $\boldsymbol{b}$. For example, some authors prefer the l_1-minimization (see [32,33] and the references presented there).

The idea of using the first derivative (gradient) of $\varepsilon(\xi)$ in the UQ context comes from the optimal control theory [29], where the concept of 'adjoint equations' is introduced. The adjoint model allows the gradient of any function of the state variables to be computed at a cost comparable to the cost of computing the function itself. The gradient of (7) is a vector of size N with elements given by

$$\frac{\partial\varepsilon(\xi)}{\partial\xi_i} = \sum_{i_1=1}^{N} b_{i_1}\frac{\partial H_1(\xi_{i_1})}{\partial\xi_i} + \sum_{i_1=1}^{N}\sum_{i_2=1}^{i_1} b_{i_1 i_2}\frac{\partial H_2(\xi_{i_1}, \xi_{i_2})}{\partial\xi_i} + \sum_{i_1=1}^{N}\sum_{i_2=1}^{i_1}\sum_{i_3=1}^{i_2} b_{i_1 i_2 i_3}\frac{\partial H_3(\xi_{i_1}, \xi_{i_2}, \xi_{i_3})}{\partial\xi_i} + \ldots \tag{12}$$
$$i = 1, \ldots, N.$$

The above expression can be re-written in the dot product form

$$\frac{\partial \varepsilon(\xi)}{\partial \xi_i} = \frac{\partial \boldsymbol{H}(\xi)}{\partial \xi_i}\boldsymbol{b}_1,$$

where $\boldsymbol{b}_1 = (\{b_{i_1}\}, \{b_{i_1 i_2}\}, \ldots, \{b_{i_1 \ldots i_p}\})^T$ is a vector which includes all the PC coefficients except b_0. Now, the generalized cost-function looks as follows:

$$J = \Omega(\boldsymbol{b}) + J_0 + J_1, \tag{13}$$

with

$$J_1 = \sum_{l=1}^{L_1}\sum_{i=1}^{N}\left(\frac{\partial \varepsilon^*(\xi^l)}{\partial \xi_i} - \frac{\partial \boldsymbol{H}(\xi^l)}{\partial \xi_i}\boldsymbol{b}_1\right)^2, \tag{14}$$

where $\partial \varepsilon^*(\xi^l)/\partial \xi$ is the gradient of the design function (6) computed at point ξ^l. Thus, one run of the adjoint model simultaneously provides 'data' for N constraints. The earliest reports on the gradient-enhanced PC expansion method can be found in [2,34].

A natural extension to this method would be to consider the second derivative (Hessian) of (6). In particular, we know that the Hessian-vector product is given by the second-order adjoint model [47]. Using the Hessian-vector product, the eigenvalue decomposition $\{\lambda_s, U_s\}$, $s = 1, \ldots, S$ of the Hessian can be obtained using the Lanczos method. Then, the Hessian can be presented via its eigenpairs in the limited-memory form as follows:

$$\Theta := \frac{\partial^2 \varepsilon(\xi)}{\partial \xi^{\,2}} = \sum_{s=1}^{S}\lambda_s U_s U_s^T. \tag{15}$$

The second derivative of (7) is a symmetric $N \times N$ matrix with elements

$$\frac{\partial^2 \varepsilon(\xi)}{\partial \xi_i \partial \xi_j} = \sum_{i_1=1}^{N}\sum_{i_2=1}^{i_1} b_{i_1 i_2}\frac{\partial^2 H_2(\xi_{i_1}, \xi_{i_2})}{\partial \xi_i \partial \xi_j} + \sum_{i_1=1}^{N}\sum_{i_2=1}^{i_1}\sum_{i_3=1}^{i_2} b_{i_1 i_2 i_3}\frac{\partial^2 H_3(\xi_{i_1}, \xi_{i_2}, \xi_{i_3})}{\partial \xi_i \partial \xi_j} + \ldots \tag{16}$$

The above expression can be re-written in the dot product form

$$\frac{\partial^2 \varepsilon(\xi)}{\partial \xi_i \partial \xi_j} = \frac{\partial^2 \boldsymbol{H}(\xi)}{\partial \xi_i \partial \xi_j}\boldsymbol{b}_2,$$

where $\boldsymbol{b}_2 = (\{b_{i_1 i_2}\}, \ldots, \{b_{i_1 \ldots i_p}\})^T$. Now, the generalized cost-function looks as follows:

$$J = \Omega(\boldsymbol{b}) + J_0 + J_1 + J_2, \tag{17}$$

with

$$J_2 = \sum_{l=1}^{L_2}\sum_{i=1}^{N}\sum_{j=1}^{i}\left(\frac{\partial^2 \varepsilon^*(\xi^l)}{\partial \xi_i \partial \xi_j} - \frac{\partial^2 \boldsymbol{H}(\xi^l)}{\partial \xi_i \partial \xi_j}\boldsymbol{b}_2\right)^2, \tag{18}$$

where $\partial^2 \varepsilon^*(\xi^l)/\partial \xi^2$ is the Hessian of (6) computed at point ξ^l and presented in the form (15).

While the idea of using the Hessian computations in the context of the PC expansion looks attractive, its practicality depends on the circumstances. The Hessian is far more expensive object to compute than the gradient. For each Hessian-vector product the tangent linear and adjoint models have to be run in a sequence (see §7.2). The number of Lanczos iterations depends on the eigenvalue spectrum of the Hessian. If this spectrum contains a well separated subset of 'leading' eigenvalues, the number of Lanczos iterations is nearly equivalent to the size of this subset; it could be significantly larger otherwise. On the other hand, one computed Hessian simultaneously adds $N(N+1)/2$ constraints for each l.

There is one important aspect of the regression method which is often overlooked. It is usually assumed that running the model for data is much more expensive than solving the minimization problem for (10). In fact, this depends on N (size of the importance set). Let us consider, for example, the third-order PC expansion $p = 3$. In this case, computing $\boldsymbol{H}(\xi^l)\boldsymbol{b}$ for each l requires $N + N(N+1) + N(N+1)(N+2)/2$ multiplications. However, taking into account the sparsity of the derivatives of $\boldsymbol{H}$, computing $\partial \boldsymbol{H}(\xi^l)/\partial \xi_i \cdot \boldsymbol{b}_1$ requires $N + N(N+1)$ multiplications, and computing $\partial^2 \boldsymbol{H}(\xi^l)/(\partial \xi_i \partial \xi_j) \cdot \boldsymbol{b}_2$ – just N multiplications, for each l. Therefore, each equation from (14) and (18) is about N and N^2 times less expensive than the one from (11), respectively. So, the question here is what is more expensive: producing the 'data', i.e. the sample of $\partial \varepsilon^*/\partial \xi$, $\partial^2 \varepsilon^*/\partial \xi^2$, or resolving the subsequent minimization problem. Let us note that for distributed systems, the size of the importance set N could still be quite large.

While the Hessian-based enhancement of the regression methods can be useful in certain circumstances one needs more efficient tools allowing for larger N. Below we present an alternative method in which the derivatives are used, but minimization is not required. This method, designed for the 3rd-order PC expansion, i.e. $p \leq 3$, is related to the quasi-regression or projection methods [4,3,17,18]. The key elements of this method are the 2D and 3D filtering algorithms which allow the sampling error to be efficiently removed.

3. Explicit expressions for the PC coefficients

Let us denote by $E[\cdot]$ the expectation with respect to the standard normal distribution $N(0,1)$. The PC expansion coefficients $\boldsymbol{b}$ can be expressed via expectations of $\varepsilon(\xi)$ and its derivatives. For the high-dimensional distributed models in mind we limit ourselves by considering the third-order polynomials. The main results are summarized in the following theorem:

Theorem 1. *For the coefficients b_0, $\{b_i\}$, $\{b_{ij}\}$, $\{b_{ijk}\}$, $i = 1, \ldots, N$, $1 \leq j < i$, $1 \leq k < j$ of the expansion (7), the following expressions hold:*

$$b_0 = E[\varepsilon(\xi)], \tag{19}$$

$$\beta_i b_i = E[H_1(\xi_i)\varepsilon(\xi)] = E\left[H_0 \frac{\partial \varepsilon(\xi)}{\partial \xi_i}\right], \tag{20}$$

$$\beta_{ij} b_{ij} = E\left[H_2(\xi_i, \xi_j)\varepsilon(\xi)\right] = E\left[H_1(\xi_i)\frac{\partial \varepsilon(\xi)}{\partial \xi_j}\right] = E\left[H_0 \frac{\partial^2 \varepsilon(\xi)}{\partial \xi_i \partial \xi_j}\right], \tag{21}$$

$$\beta_{ijk} b_{ijk} = E\left[H_3(\xi_i, \xi_j, \xi_k)\varepsilon(\xi)\right] = E\left[H_2(\xi_i, \xi_j)\frac{\partial \varepsilon(\xi)}{\partial \xi_k}\right] = E\left[H_1(\xi_i)\frac{\partial^2 \varepsilon(\xi)}{\partial \xi_j \partial \xi_k}\right] = E\left[H_0 \frac{\partial^3 \varepsilon(\xi)}{\partial \xi_i \partial \xi_j \partial \xi_k}\right], \tag{22}$$

where

$$\beta_i = E\left[(H_1(\xi_i))^2\right] = 1,\ i = 1, \ldots, N, \tag{23}$$

$$\beta_{ij} = E\left[(H_2(\xi_i, \xi_j))^2\right] = \begin{cases} 2!, & i = j \\ 1 & i \neq j,\ i > j \end{cases}, \tag{24}$$

$$\beta_{ijk} = E\left[(H_3(\xi_i, \xi_j, \xi_k))^2\right] = \begin{cases} 3!\ i = j = k \\ 2!\ i = j \neq k \\ 2!\ i \neq j = k \\ 1\ \ i \neq j \neq k \end{cases}. \tag{25}$$

Proof of this theorem is presented in Appendix 1. Due to the super-symmetry, order of indexing in (21) and (22) can be changed. The expressions for $\boldsymbol{b}$ via $\varepsilon(\xi)$ (first equalities in the sequences (20)–(22)) are known in the 'projection' or 'quasi-regression' methods. However, expressions for $\boldsymbol{b}$ via the derivatives of $\varepsilon(\xi)$ presented here have not been previously reported in the literature. These expressions can be used as a basis for explicit evaluation of the PC coefficients using the derivatives of $\varepsilon(\xi)$.

Proposition 1. *One can prove by induction that*

$$E\left[H_p(\xi_{i_1}, \ldots \xi_{i_p})\varepsilon(\xi)\right] = E\left[H_{p-1}(\xi_{i_1}, \ldots \xi_{i_{p-1}})\frac{\partial \varepsilon(\xi)}{\partial \xi_{i_p}}\right] = \ldots = E\left[H_0 \frac{\partial^p \varepsilon(\xi)}{\partial \xi_{i_1} \ldots \partial \xi_{i_p}}\right]. \tag{26}$$

This implies that the chains of equalities (21) and (22)) can be generalized to the polynomials of arbitrary degree p, with corresponding weights β. Let us note, however, that for high-dimensional distributed models only computing the first two derivatives (gradient and Hessian) may be feasible.

Corollary 1. *The PC-representation (7) can be re-written via the derivatives of the design function $\varepsilon(\xi)$ as follows:*

$$\varepsilon(\xi) = E[\varepsilon(\xi)] + \sum_{i_1=1}^{N} E\left[\frac{\partial \varepsilon(\xi)}{\partial \xi_{i_1}}\right] H_1(\xi_{i_1}) + \sum_{i_1=1}^{N}\sum_{i_2=1}^{i_1} \frac{1}{\beta_{i_1 i_2}} E\left[H_1(\xi_{i_1})\frac{\partial \varepsilon(\xi)}{\partial \xi_{i_2}}\right] H_2(\xi_{i_1}, \xi_{i_2}) +$$

$$+ \sum_{i_1=1}^{N}\sum_{i_2=1}^{i_1}\sum_{i_3=1}^{i_2} \frac{1}{\beta_{i_1 i_2 i_3}} E\left[H_2(\xi_{i_1}, \xi_{i_2})\frac{\partial \varepsilon(\xi)}{\partial \xi_{i_3}}\right] H_3(\xi_{i_1}, \xi_{i_2}, \xi_{i_3}) + \ldots, \tag{27}$$

or

$$\varepsilon(\xi) = E[\varepsilon(\xi)] + \sum_{i_1=1}^{N} E\left[\frac{\partial \varepsilon(\xi)}{\partial \xi_{i_1}}\right] H_1(\xi_{i_1}) + \sum_{i_1=1}^{N}\sum_{i_2=1}^{i_1} \frac{1}{\beta_{i_1 i_2}} E\left[\frac{\partial^2 \varepsilon(\xi)}{\partial \xi_{i_1}\partial \xi_{i_2}}\right] H_2(\xi_{i_1}, \xi_{i_2}) +$$

$$+ \sum_{i_1=1}^{N}\sum_{i_2=1}^{i_1}\sum_{i_3=1}^{i_2} \frac{1}{\beta_{i_1 i_2 i_3}} E\left[H_1(\xi_{i_1}) \frac{\partial^2 \varepsilon(\xi)}{\partial \xi_{i_2}\partial \xi_{i_3}}\right] H_3(\xi_{i_1}, \xi_{i_2}, \xi_{i_3}) + \dots . \tag{28}$$

These expressions provide different forms of what can be referred as the **probabilistic Taylor expansion**.

4. Global sensitivity analysis

Global sensitivity analysis (GSA) aims to quantify the relative impact of the inputs ξ on the design function ε. The commonly used GSA method is the variance-based method of Sobol' sensitivity indices [41], also known as ANOVA decomposition. In this paper we limit ourselves by considering the first-order 'main effect' indices $\mathcal{S}_i$ and the 'total effect' indices $\mathcal{S}_i^T$, defined in [37] as follows:

$$\mathcal{S}_i = \frac{Var(E[\varepsilon(\xi)|\xi_i])}{Var(\varepsilon(\xi))} \tag{29}$$

$$\mathcal{S}_i^T = \frac{Var(\varepsilon(\xi)) - Var(E[\varepsilon(\xi)|\xi_{-i}])}{Var(\varepsilon(\xi))}, \tag{30}$$

where $\varepsilon(\xi)|\xi_i$ means the random value of ε with the input component ξ_i being fixed at its 'generic' value, ξ_{-i} stands for 'all, but i', and $Var(\varepsilon(\xi))$ is the total variance of $\varepsilon(\xi)$.

The unique relationship between the global sensitivity indices (GSI) and the PC coefficients has been derived in [6]. For the PC expansion in the form (7), which is different from the one used in [6], we can write (see Appendix 3 for detail):

$$\mathcal{S}_i \; Var(\varepsilon(\xi)) = \beta_i b_i^2 + \beta_{ii} b_{ii}^2 + \beta_{iii} b_{iii}^2 + \dots \tag{31}$$

$$\mathcal{S}_i^T \; Var(\varepsilon(\xi)) = \beta_i b_i^2 + \sum_{j=1}^{i} \beta_{ij} b_{ij}^2 + \sum_{j=1}^{i}\sum_{k=1}^{j} \beta_{ijk} b_{ijk}^2 + \dots \tag{32}$$

The derivative-based global sensitivity measures has been introduced in [42]. It has been proved that for ξ being a vector of random iid normal variables $\xi_i \sim N(0,1)$

$$\mathcal{S}_i^T \; Var(\varepsilon(\xi)) \le E\left[\left(\frac{\partial \varepsilon(\xi)}{\partial \xi_i}\right)^2\right]. \tag{33}$$

The above formula gives a convenient tool for computing the upper-bound for $\mathcal{S}_i^T$, to be used as a basis for ranking the inputs and, subsequently, for defining the importance set. If the final goal is to construct the probability density function (pdf) of $\varepsilon(\xi)$, using the upper-bound guaranties that all important variables are accounted for, however the corresponding importance set may contain some less important variables.

The upper-bound estimate can be improved if some of the PC coefficients are known. This follows from the theorem below:

Theorem 2. *The expectation of the squared gradient is given by the series*

$$E\left[\left(\frac{\partial \varepsilon(\xi)}{\partial \xi_i}\right)^2\right] = \tilde{\beta}_i b_i^2 + \sum_{j=1}^{i} \tilde{\beta}_{ij} b_{ij}^2 + \sum_{j=1}^{i}\sum_{k=1}^{j} \tilde{\beta}_{ijk} b_{ijk}^2 + \dots, \tag{34}$$

where

$$\tilde{\beta}_i = 1, \; i = 1, \dots, N, \tag{35}$$

$$\tilde{\beta}_{ij} = \begin{cases} 2 \times 2!, & i = j \\ 1 & i \neq j, \; i > j \end{cases}, \tag{36}$$

$$\tilde{\beta}_{ijk} = \begin{cases} 3 \times 3! & i = j = k \\ 2 \times 2! & i = j \neq k \\ 1 \times 2! & i \neq j = k \\ 1 & i \neq j \neq k \end{cases}. \tag{37}$$

Proof of this theorem is presented in Appendix 2. The Theorem essentially replicates some of the results from [44], but represents them in a more transparent and convenient form.

Corollary 2. *For any given order of polynomials, the improved derivative-based upper-bound for the 'total effect' GSI is given by the expression*

$$\mathcal{S}_i^T\ Var(\varepsilon(\xi)) \le E\left[\left(\frac{\partial\varepsilon(\xi)}{\partial\xi_i}\right)^2\right] - \sum_{j=1}^{i}\frac{\tilde{\beta}_{ij}-\beta_{ij}}{\beta_{ij}^2}\,E^2\left[H_1(\xi_i)\frac{\partial\varepsilon(\xi)}{\partial\xi_j}\right] -$$

$$-\sum_{j=1}^{i}\sum_{k=1}^{j}\frac{\tilde{\beta}_{ijk}-\beta_{ijk}}{\beta_{ijk}^2}\,E^2\left[H_2(\xi_i,\xi_j)\frac{\partial\varepsilon(\xi)}{\partial\xi_k}\right] - \ldots \tag{38}$$

Proof of Corollary 2 is fairly trivial. Comparing (32) and (34) we notice that

$$E\left[\left(\frac{\partial\varepsilon(\xi)}{\partial\xi_i}\right)^2\right] - \mathcal{S}_i^T\ Var(\varepsilon(\xi)) \ge \sum_{j=1}^{i}(\tilde{\beta}_{ij}-\beta_{ij})b_{ij}^2 + \sum_{j=1}^{i}\sum_{k=1}^{j}(\tilde{\beta}_{ijk}-\beta_{ijk})b_{ijk}^2 + \ldots$$

For the infinite series the above expression becomes an equality. Since $\tilde{\beta} \ge \beta$, all terms in the series to the right are non-negative. Therefore, the more terms in the series are truncated, the stronger the inequality becomes. Formula (33) corresponds to the case when all terms are truncated. That is why (38) must (at least in theory) provide less overestimated upper-bound for $\mathcal{S}_i^T$ than (33). Taking into account expressions for $b_{ij}, b_{ijk}, \ldots$ from (20)–(22) we obtain (38). It can be easily seen that, for the truncated series, formula (32) gives a lower-bound for $\mathcal{S}_i^T$, which may not be a reliable basis for defining the importance set.

In practice, given a sample of functions and gradients $\{\varepsilon^*(\xi^l), \partial\varepsilon^*(\xi^l)/\partial\xi\}$, $l = 1, \cdots, L$, instead of (38) one would use

$$\mathcal{S}_i^T\ \overline{Var}(\varepsilon(\xi)) < L^{-1}\sum_{i=1}^{L}\left(\frac{\partial\varepsilon^*(\xi^l)}{\partial\xi_i}\right)^2 - \sum_{j=1}^{i}(\tilde{\beta}_{ij}-\beta_{ij})b_{ij}^2 - \sum_{j=1}^{i}\sum_{k=1}^{j}(\tilde{\beta}_{ijk}-\beta_{ijk})b_{ijk}^2 - \ldots, \tag{39}$$

where

$$\overline{Var}(\varepsilon(\xi)) = \frac{1}{L-1}\sum_{l=1}^{L}\left(\varepsilon^*(\xi^l) - b_0\right)^2 \tag{40}$$

is the sample variance and $b_{ij}, b_{ijk}, \ldots$ are the estimated PC expansion coefficients based on this sample. Of course, for small samples there is no warranty that the upper-bound evaluated from (39) is better than the one without correction terms.

Let us finally note (considering (20) and (31)) that

$$\mathcal{S}_i\ Var(\varepsilon(\xi)) \ge E^2\left[\frac{\partial\varepsilon(\xi)}{\partial\xi_i}\right], \tag{41}$$

i.e. the squared expectation of the gradient gives a lower bound for the first-order 'main effect' indices.

As discussed above, given the sample of gradients one can evaluate the upper bound for the 'total effect' sensitivity indices $\mathcal{S}_i^T$, $i = 1, \ldots, N$ by (32). Then, the inputs U can be ranked accordingly, which means the mapping $i = \mathcal{I}(i')$ between the original and the ranked sequences of elements in U, as well as its inverse $i' = \mathcal{I}^{-1}(i)$, are defined. The importance set can be chosen by considering $N' < N$ elements (having the largest $\mathcal{S}_i^T$), where N' satisfies the condition

$$\sum_{i'=1}^{N'}\mathcal{S}_{\mathcal{I}(i')}^T \Big/ \sum_{i=1}^{N}\mathcal{S}_i^T \approx \epsilon,$$

where ϵ is a given threshold. Below we deal with the importance set of size N', which means that all indexes involved are i'. For the sake of simplicity, however, we keep the original notations, i.e. we use N instead of N', i instead of i', etc.

5. Finite sample estimates for the PC coefficients

5.1. Zero- and first-order coefficients

It has been already mentioned that the explicit ('projection' or 'quasi-regression') methods for estimating the PC coefficients have been previously reported, though without the derivatives of the design function being used. These methods are useful if computing a sufficiently large sample of model responses is feasible.

Note that equations (19)–(22) represent the statistical moments of a multivariate random function $\varepsilon^*(\xi)$ and its derivatives. In theory, the uncertainty (variance) in the sample mean, variance, skewness and kurtosis is proportional to $1/L$, $2/L$, $6/L$ and $24/L$, respectively, where L is the sample size [23]. These values are significantly larger in practice.

Let us assume that the following estimates of the PC coefficients have a sufficient level of statistical conditioning with $L = L^*$:

$$b_0 = L^{-1}\sum_{l=1}^{L}\varepsilon^*(\xi^l),\ b_i = L^{-1}\sum_{l=1}^{L}\frac{\partial\varepsilon^*(\xi^l)}{\partial\xi_i},\ b_{ij} = \beta_{ij}^{-1}L^{-1}\sum_{l=1}^{L}\frac{\partial^2\varepsilon^*(\xi^l)}{\partial\xi_i\partial\xi_j}. \tag{42}$$

For the estimates

$$b_i = L^{-1}\sum_{l=1}^{L}\xi_i^l\varepsilon^*(\xi^l),\ b_{ij} = \beta_{ij}^{-1}L^{-1}\sum_{l=1}^{L}\xi_i^l\frac{\partial\varepsilon^*(\xi^l)}{\partial\xi_j}, \tag{43}$$

the equivalent level of statistical conditioning can be achieved with the sample size $L > 2\times L^*$, and for

$$b_{ij} = \beta_{ij}^{-1}L^{-1}\sum_{l=1}^{L}(\xi_i^l\xi_j^l - \delta_{i,j})\varepsilon^*(\xi^l) \tag{44}$$

– with the sample size $L > 6\times L^*$.

Since the computational cost of the function and of its gradient are often comparable, computing the sample of the gradients of size L^* is feasible. Therefore, the second formula in (42) can be used for evaluating b_i. However, the computational cost of the Hessian is significantly (50–100 times) larger, which means that the sample of Hessians of size L^* may not be available. Therefore, $b_{i,j}$ have to be computed using the second formula from (43). This requires either a larger sample size to be consistent with the estimates of b_0 and b_i, or a special treatment to reduce the sampling error. The filtering procedures for computing b_{ij} and b_{ijk} are introduced below.

5.2. Second-order coefficients with filtering

The second-order coefficients b_{ij} can be computed via the gradient of $\varepsilon(\xi)$ using the second formula in (43). Let A be an $N\times N$ matrix with the elements

$$A_{i,j} = L^{-1}\sum_{l=1}^{L}\xi_i^l\frac{\partial\varepsilon^*(\xi^l)}{\partial\xi_j}, \tag{45}$$

where N is the importance set size. According to the last equality in (21), matrix A must be symmetric in the limit $L\to\infty$. For a finite L, however, it is not symmetric due to the sampling error, see Fig. 3(left). From this figure one can also notice that the presented 2D-field is smooth in the horizontal direction (index j), and oscillatory in the vertical direction (index i). Indeed, as it follows from (45), each row of A is a transposed weighted average of the gradient. The latter often is a sufficiently smooth function of the spatial coordinates (for distributed parameter systems).

The purpose of filtering is to retrieve the symmetric part from a noisy 2D 'image'. Let us define the matrix–vector product $w = A^T v$, where the summation is performed along the oscillatory direction i, as follows:

$$w_j = \sum_{i=1}^{N}A_{i,j}v_i = L^{-1}\sum_{l=1}^{L}\frac{\partial\varepsilon^*(\xi^l)}{\partial\xi_j^l}\left(\sum_{i=1}^{N}\xi_i^l v_i\right),\ j = 1,\ldots,N. \tag{46}$$

This matrix–vector product is used in the Lanczos algorithm to evaluate a number S_1 of eigenpairs (the largest magnitude eigenvalues λ_s and the corresponding eigenvectors U_s) of matrix A. Then, the limited-memory symmetric representation of A is given by the formula

$$\hat{A} = \sum_{s=1}^{S_1}\lambda_s U_s U_s^T, \tag{47}$$

and for the second-order coefficients we finally obtain

$$b_{ij} = \beta_{ij}^{-1}\hat{A}_{ij} = \beta_{ij}^{-1}\sum_{s=1}^{S_1}\lambda_s U_{s,i}U_{s,j}. \tag{48}$$

Two versions of this algorithm are possible. In the 'full-memory' version matrix A is explicitly computed (LN^2 multiplications) and stored in memory (N^2 numbers) before the eigenvalue analysis step. In this case computing the matrix–vector

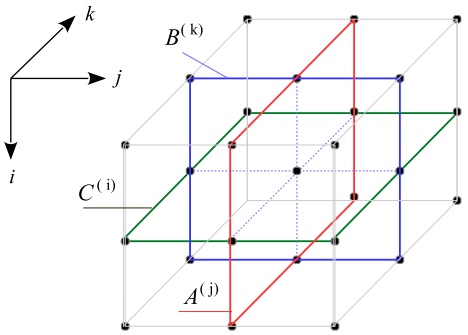


Fig. 1. Cube filtering procedure.

product costs N^2 multiplications. In the 'limited-memory' version formula (46) is used for computing the matrix–vector in the course of the Lanczos process. Here, the matrix–vector product costs $L(N+1)N$ multiplications, which is significantly more expensive in terms of the CPU time. However, in this case no additional storage is required.

Remark 1. Taking into account that the described procedure results into the limited-memory representation (48) for b_{ij} we note that the suggested second-order analysis might be affordable for the full input set, i.e. **before** defining the importance set. It means that the ranking can be performed on a basis of the improved measure (39) involving the second-order correction term. However, the third-order analysis is likely to be feasible for the importance set.

5.3. Third-order coefficients with filtering

The symmetrizer based on the eigenvalue decomposition represents a very efficient tool for removing the sampling error in the two-dimensional case (see Fig. 3(right) and §8.2 for details). Unfortunately, extension of this method to higher dimensions is not straightforward, since no direct analogue of the Lanczos algorithm for tensors is presently known. Here we suggest a simplified approach.

The third-order PC coefficients b_{ijk} can be evaluated via the gradient or Hessian of $\varepsilon(\xi)$ using one of the equalities from (22). Let us denote

$$\mathcal{A} = L^{-1} \sum_{l=1}^{L} H_2(\xi_i, \xi_j) \frac{\partial \varepsilon^*(\xi)}{\partial \xi_k}. \tag{49}$$

By definition b_{ijk}, $i, j, k = 1, \ldots, N$ must form a super-symmetric cube (see Fig. 1), however tensor $\mathcal{A}$ computed by (49) is not symmetric for a finite L due to the sampling error. Again, the gradient is a smooth function. Thus, tensor $\mathcal{A}$ has a smooth dimension k, and two oscillatory dimensions i and j. As in the 2D case, the goal is to retrieve the symmetric part from a noisy three-dimensional 'image'.

First, consider matrix $A^{(j)}$ – the slice of the cube by plane (i,k), $\forall j$, and define the matrix–vector product $w = A^{(j)} v$, where the summation is performed along the oscillatory direction i, as follows:

$$w_k = \sum_{i=1}^{N} A^{(j)}_{i,k} v_i = L^{-1} \sum_{l=1}^{L} \frac{\partial \varepsilon^*(\xi^l)}{\partial \xi_k} \left(\sum_{i=1}^{N} H_2(\xi^l_i, \xi^l_j) v_i \right), \; k = 1, \ldots, N. \tag{50}$$

This matrix–vector product is used in the Lanczos algorithm to evaluate a certain number of eigenpairs $\{\lambda^{(j)}_s, U^{(j)}_s\}$, $s = 1, \ldots, S_2$ of matrices $A^{(j)}$, $j = 1, \ldots, N$. Then, the limited-memory symmetric representation of $A^{(j)}$ is given by the formula

$$\hat{A}^{(j)} = \sum_{s=1}^{S_2} \lambda^{(j)}_s U^{(j)}_s U^{(j)T}_s. \tag{51}$$

Now, instead of the original 'unfiltered' cube $\mathcal{A}$, represented by a set of vectors ξ^l and $\partial\varepsilon(\xi^l)/\partial\xi_k$, we consider a set of matrices $\hat{A}^{(j)}$, $j = 1, \ldots, N$, each being compactly represented by its eigenpairs. These matrices form a partially filtered cube $\hat{\mathcal{A}}$, which has two smooth directions i, k and one oscillatory direction j. Vectors ξ^l and $\partial\varepsilon(\xi^l)/\partial\xi_k$ are no longer needed after that and could be removed from the memory if necessary. As in the 2D case, the full- and the limited-memory implementations are possible at this stage.

Secondly, consider matrix $B^{(k)}$ – the slice of the cube $\hat{\mathcal{A}}$ by plane (i,j), $\forall k$. We notice that $B^{(k)}_{i,j} = \hat{\mathcal{A}}^{(j)}_{i,k}$ and, therefore, define the matrix–vector product $w = B^{(k)} v$, where the summation is performed along the oscillatory direction j, as follows:

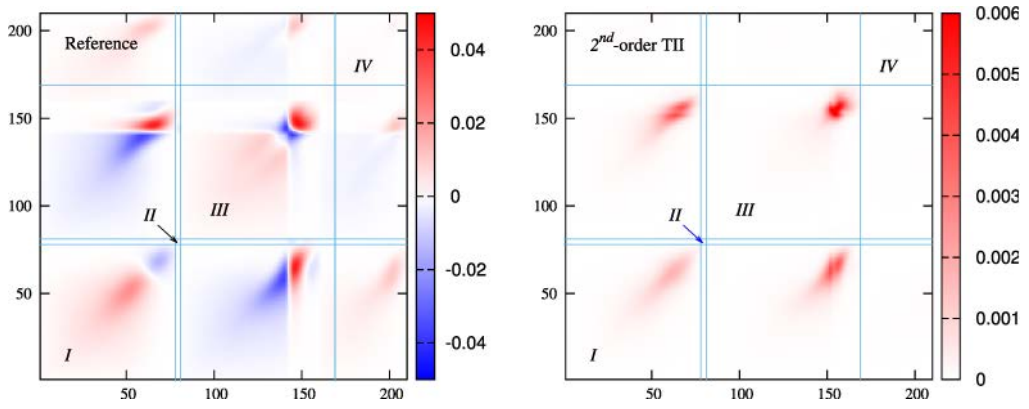


Fig. 2. Left: 'reference' expected Hessian (second derivative) using $E[\partial^2\varepsilon/\partial\xi^2]$, for sample size $L = 400$. Right: the total interaction indices (TII). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

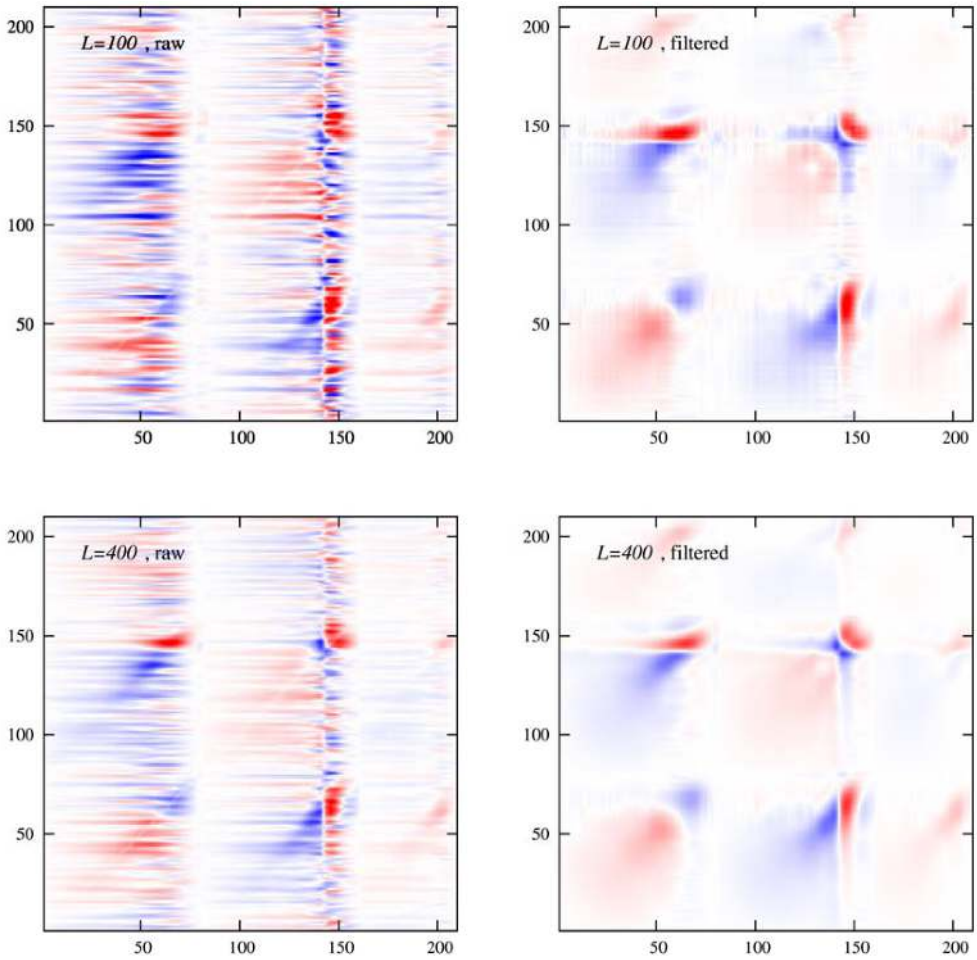


Fig. 3. Estimates of the expected Hessian (second derivative) using $E[\xi\partial\varepsilon/\partial\xi]$, for different sample sizes $L = 100$ (upper) and $L = 400$ (lower). Left – before filtering (raw), right – after filtering.

$$w_i = \sum_{j=1}^{N} B_{i,j}^{(k)} v_j = \sum_{i=1}^{N} \hat{A}_{i,k}^{(j)} v_j = \sum_{s=1}^{S_2} \sum_{j=1}^{N} \left(\lambda_s^{(j)} U_{s,i}^{(j)} U_{s,k}^{(j)} \right) v_j, \; i = 1, \ldots, N. \tag{52}$$

This matrix–vector product definition is used to evaluate a certain number of eigenpairs $\{\lambda_s^{(k)}, U_s^{(k)}\}$, $s = 1, \ldots, S_2$ of matrices $B^{(k)}$, $k = 1, \ldots, N$. Then, the limited-memory symmetric representation $\hat{B}^{(k)}$ of $B^{(k)}$ is given by the formula similar

to (51). Subsequently, instead of $\hat{\mathcal{A}}$ we consider a set of matrices $\hat{B}^{(k)}$, $k = 1, \ldots, N$, which form the cube $\hat{\mathcal{B}}$. Each matrix $\hat{B}^{(k)}$ is compactly represented by its own eigenpairs which substitute the eigenpairs of $\hat{\mathcal{A}}$ in memory.

Finally, consider matrix $C^{(i)}$ – the slice of the cube $\hat{\mathcal{B}}$ by plane (j,k), $\forall i$. We notice that $C^{(i)}_{j,k} = \hat{B}^{(k)}_{i,j}$ and, therefore, define the matrix–vector product $w = C^{(i)} v$, where the summation is performed along the direction k, as follows:

$$w_j = \sum_{k=1}^{N} C^{(i)}_{j,k} v_k = \sum_{k=1}^{N} \hat{B}^{(k)}_{i,j} v_k = \sum_{s=1}^{S_2} \sum_{k=1}^{N} \left(\lambda^{(k)}_s U^{(k)}_{s,i} U^{(k)}_{s,j} \right) v_k, \;\; j = 1, \ldots, N. \tag{53}$$

This matrix–vector product is used to evaluate a certain number of eigenpairs $\{\lambda^{(i)}_s, U^{(i)}_s\}$, $s = 1, \ldots, S_2$ of matrices $C^{(i)}$, $i = 1, \ldots, N$. Then, the limited-memory symmetric representation $\hat{C}^{(i)}$ of $C^{(i)}$ is given by the formula similar to (51). Let us note that even though the direction k was originally the smooth one, it could have been perturbed during the first two stages of the filtering procedure. Thus, the last (third) step is important to remove these possible perturbations. Finally, for the third-order coefficients we obtain

$$b_{ijk} = \beta^{-1}_{ijk} \hat{C}^{(i)}_{jk} = \beta^{-1}_{ijk} \sum_{s=1}^{S_2} \lambda^{(i)}_s U^{(i)}_{s,j} U^{(i)}_{s,k}. \tag{54}$$

Remark 2. If a sample of Hessians is available, the third-order PC coefficients can be computed by

$$b_{ijk} = \beta^{-1}_{ijk} L^{-1} \sum_{l=1}^{L} \xi_i \frac{\partial^2 \varepsilon^*(\xi)}{\partial \xi_j \partial \xi_k}, \tag{55}$$

then, a modified filtering procedure must be applied. Since the Hessian is given in the limited memory form via a certain number of its eigenpairs, it is a smooth two-dimensional field. Thus, the original cube has two smooth directions j and k, and one oscillatory direction i. The filtering procedure similar to the presented above can be easily built. In practice, computing the Hessian could be worthwhile if running the original nonlinear model is significantly more expensive than running the tangent linear and adjoint models (both are linear). This could be the case when the implicit time scheme is utilized, which implies that a few nonlinear iterations (fixed-point or Newton) has to be made at each time step.

6. Adaptation of metamodels

One can benefit from considering the PC coefficients in the form (48) and (54) when constructing the metamodel. In particular, the second-order term $\varepsilon(\xi)$ in (7) can be written as

$$\varepsilon_2(\xi) = \sum_{i=1}^{N} \sum_{j=1}^{i} b_{ij} (\xi_i \xi_j - \delta_{ij}) = \sum_{i=1}^{N} \left(\sum_{j=1}^{i-1} b_{ij} \xi_i \xi_j + b_{ii} (\xi_i^2 - 1) \right).$$

By using formula (48) to represent b_{ij} in the above expression and, also, taking into account that $U^{(i)}_{s,j}$ are the orthonormal vectors, we obtain

$$\varepsilon_2(\xi) = \sum_{s=1}^{S_1} \lambda_s \left[\sum_{i=1}^{N} d_{s,i} \left(\sum_{j=1}^{i-1} d_{s,j} + \frac{1}{2} d_{s,i} \right) - \frac{1}{2} \right], \; d_{s,i} = U_{s,i} \xi_i. \tag{56}$$

It is easy to see that computing $\varepsilon_2(\xi)$ by the above formula requires $2S_1 N$ instead of $N(N+1)$ multiplications by the original formula.

Similarly, the third-order term $\varepsilon(\xi)$ in (7) can be written as

$$\varepsilon_3(\xi) = \sum_{i=1}^{N} \sum_{j=1}^{i} \sum_{k=1}^{j} b_{ijk} (\xi_i \xi_j \xi_k - \xi_i \delta_{jk} - \xi_j \delta_{ik} - \xi_k \delta_{ij}) = \sum_{i=1}^{N} \sum_{j=1}^{i-1} \sum_{k=1}^{j-1} b_{ijk} \xi_i \xi_j \xi_k +$$
$$+ \sum_{i=1}^{N} \sum_{j=1}^{i-1} b_{ijj} \xi_i (\xi_j^2 - 1) + \sum_{i=1}^{N} \sum_{k=1}^{i-1} b_{iik} \xi_k (\xi_i^2 - 1) + \sum_{i=1}^{N} b_{iii} (\xi_i^3 - 3\xi_i).$$

By using formula (54) to represent b_{ijk} in the above expression and, also, taking into account that $U^{(i)}_{s,j}$ are the orthonormal vectors, we obtain

$$\varepsilon_3(\xi) = a_1 + a_2 + a_3, \tag{57}$$

where

$$a_1 = \sum_{i=1}^{N} \xi_i \sum_{s=1}^{S_2} \lambda_s^{(i)} \left\{ \left[\sum_{j=1}^{i-1} d_{s,j}^{(i)} \left(\sum_{k=1}^{(j-1)} d_{s,k}^{(i)} + \frac{1}{2!} d_{s,j}^{(i)} \right) \right] + \frac{1}{2!} (1 - (U_{s,i}^{(i)}))^2 \right\},$$

$$a_2 = \frac{1}{2!} \sum_{i=1}^{N} (\xi_i^2 - 1) \sum_{s=1}^{S_2} \lambda_s^{(i)} U_{s,i}^{(i)} \sum_{k=1}^{i-1} d_{s,k}^{(i)},$$

$$a_3 = \frac{1}{3!} \sum_{i=1}^{N} (\xi_i^3 - 3\xi_i) \sum_{s=1}^{S_2} \lambda_s^{(i)} (U_{s,i}^{(i)}))^2,$$

and $d_{s,j}^{(i)} = U_{s,j}^{(i)} \xi_j$. It is easy to see that computing $\varepsilon_3(\xi)$ by the above formula requires approximately $3S_2N^2/2$ instead of $N(N+1)(N+2)/2$ multiplications by the original formula.

7. Numerical model

7.1. Forward model

As a model we use the 1D generalized Burgers' equation

$$\frac{\partial \varphi}{\partial t} + (a + \varphi) \frac{\partial \varphi}{\partial x} = \frac{\partial}{\partial x} \left(d \frac{\partial \varphi}{\partial x} \right) + c\varphi + \eta, \varphi = \varphi(x,t),\ t \in (0,T],\ x \in (0,1), \tag{58}$$

where $a(x)$ is a convection coefficient, $c(x)$ is a reaction coefficient, $d(x)$ is a diffusion (viscosity) coefficient, $\eta(x,t)$ is a forcing term, along with the initial condition

$$\varphi(x,0) = u(x),$$

and the Neumann boundary conditions

$$(d\varphi/dx)|_{x=0} = (d\varphi/dx)|_{x=1} = 0.$$

Equation (58) is sometimes used in the data assimilation context as a simple model describing elements of the atmospheric motion.

For solving (58) its conservative representation and implicit time discretization are used as follows:

$$\frac{\varphi^j - \varphi^{j-1}}{h_t} + \frac{\partial}{\partial x} \left(\left(a + \frac{\varphi}{2} \right) \varphi^j - d \frac{\partial \varphi^j}{\partial x} \right) - \left(c + \frac{\partial a}{\partial x} \right) \varphi^j - \eta = 0,\ \ j = 1, ..., n_t,\ x \in (0,1), \tag{59}$$

where j is the time integration index, and $h_t = T/n_t$ is the time step. The spatial differential operator is discretized on a uniform grid (h_x is the spatial discretization step, $i = 1, ..., n_x$ is the node number, n_x is the total number of grid nodes) using the "power law" first-order scheme as described in [31]. For each time step we perform nonlinear iterations on φ, assuming initially that $\varphi = \varphi^{j-1}$, and iterating until (59) is approximately satisfied (i.e. the norm of the left-hand side in (59) becomes smaller than the threshold equal to $10^{-12} n_x^{1/2}$).

The design function is defined using a value of the state variable φ at a given point x^*, integrated over certain time period, i.e.

$$D(X) := D(\varphi) = \int_0^{t^*} \varphi(x^*, T - \tau) d\tau, \tag{60}$$

and the full input set is

$$U = (u(x), a(x), c(x), d(x))^T. \tag{61}$$

Thus, $G(U) = D(\varphi(U))$ and the design function is defined by the equation

$$\varepsilon(\xi) = G(\bar{U} + B^{1/2}\xi) - G(\bar{U}), \tag{62}$$

where $\bar{U} = (\bar{u}(x), \bar{a}(x), \bar{c}(x), \bar{d}(x))^T$ is the unperturbed ('true') input vector, B is the covariance matrix of the primal input uncertainty $\xi = (\xi_u(x), \xi_a(x), \xi_c(x), \xi_d(x))^T$, $\xi_i \sim N(0,1)$.

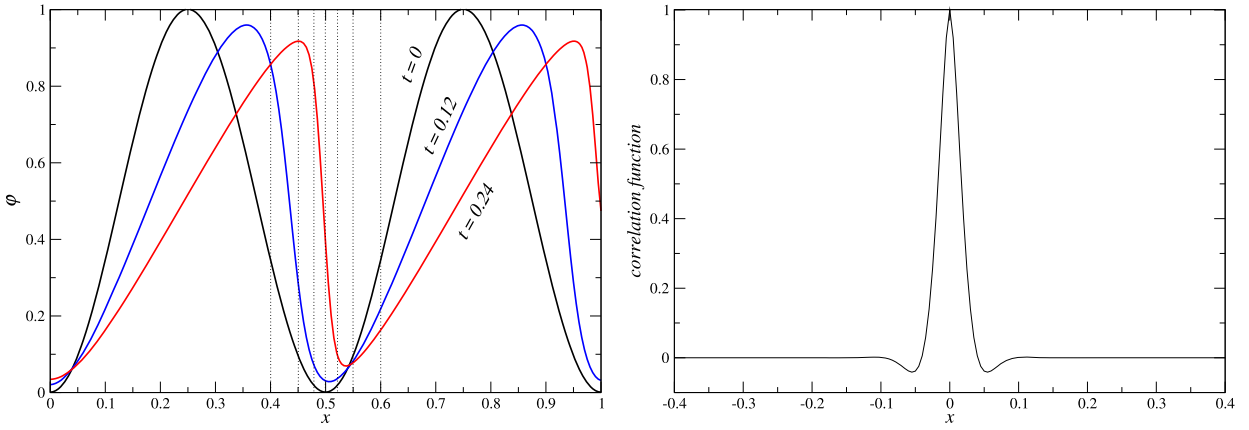


Fig. 4. Field evolution and correlation function for the input uncertainties.

The true initial state is given by the expression

$$\bar{u}(x) = 0.01 + 0.35\,[1 + \sin(4\pi x + 3\pi/2)]\,,\ x \in [0,1],$$

and the true values of coefficients in are: $\bar{a}(x) = 0$, $\bar{c}(x) = 0$ and $\bar{d}(x) = 0.002$. In computations we use the following parameters: simulation period $T = 0.24$, discretization steps $h_t = 0.004$, $h_x = 0.005$, number of spatial discretization nodes $n_x = 201$. Thus, the full size of the input set is $N = 804$.

Fig. 4(left) shows the initial state $u(x) = \varphi(x, 0)$ and the evolved state at times $t = T/2 = 0.12$ and $t = T = 0.24$. A general property of Burgers' solutions is that a smooth initial state evolves into a state characterized by areas of severe gradients (or even shocks in the inviscid case). The level of nonlinearity is, therefore, the strongest around the shock area at $x \approx 0.5$. This is where the most noticeable deviation from the gaussianity (for the pdf of $\varepsilon(\xi)$) has to be expected. Thus, we choose to consider (60) defined at points $x^* = \{0.4, 0.45, 0.48, 0.5, 0.52, 0.55, 0.60\}$. The covariance matrix B is generated by the method described in [16]. The correlation function for each variable in (61) is the same (see Fig. 4(right)). The standard deviations are uniform in space (apart from near the boundaries) with the following values: $\sigma_u = 0.2$, $\sigma_a = 0.2$, $\sigma_c = 1.0$, and $\sigma_d = 0.00033$.

7.2. Derivatives

Consider equation (58) with the input $U = \bar{U} + \xi$. Then, the gradient of $\varepsilon(\xi)$ (first derivative) is given by

$$\frac{\partial \varepsilon(\xi)}{\partial \xi} = \left(\varphi^*|_{t=0},\ -\varphi^* \frac{\partial \varphi}{\partial x},\ \varphi\varphi^*,\ -\frac{\partial \varphi}{\partial x}\frac{\partial \varphi^*}{\partial x} \right)^T, \tag{63}$$

where φ^* is the solution of the adjoint model:

$$-\frac{\partial \varphi^*}{\partial t} - \frac{\partial (a+\varphi)\varphi^*}{\partial x} = \frac{\partial}{\partial x}\left(d\,\frac{\partial \varphi^*}{\partial x} \right) + \left(c - \frac{\partial \varphi}{\partial x} \right)\varphi^* + \delta(x - x^*)|_{t>T-t^*}, \tag{64}$$
$$\varphi^*|_{t=T} = 0,\ \left((a+\varphi)\varphi^* + d\,\partial\varphi^*/\partial x \right)|_{x=0,1} = 0.$$

In turn, the Hessian (second derivative)-vector product is given by

$$\Theta \cdot v := \frac{\partial^2 \varepsilon(\xi)}{\partial \xi^{\,2}} \cdot v = \left(\tilde{\varphi}^*|_{t=0},\ -\tilde{\varphi}^* \frac{\partial \varphi}{\partial x} - \varphi^* \frac{\partial \psi}{\partial x},\ \psi\varphi^* + \varphi\tilde{\varphi}^*,\ -\frac{\partial \psi}{\partial x}\frac{\partial \varphi^*}{\partial x} - \frac{\partial \varphi}{\partial x}\frac{\partial \tilde{\varphi}^*}{\partial x} \right)^T, \tag{65}$$

where $v = (\delta u, \delta a, \delta c, \delta d)^T$, and ψ and $\tilde{\varphi}^*$ are the solutions of the sequence of the tangent linear and the (second-order) adjoint models as follows:

$$\frac{\partial \psi}{\partial t} + (a+\varphi)\frac{\partial \psi}{\partial x} = \frac{\partial}{\partial x}\left(d\,\frac{\partial \psi}{\partial x} \right) + \left(c - \frac{\partial \varphi}{\partial x} \right)\psi - \delta a \frac{\partial \varphi}{\partial x} + \delta c\,\varphi + \frac{\partial}{\partial x}\left(\delta d\,\frac{\partial \varphi}{\partial x} \right), \tag{66}$$
$$\psi|_{t=0} = v,\ (d\psi/dx)|_{x=0} = (d\psi/dx)|_{x=1} = 0;$$
$$-\frac{\partial \tilde{\varphi}^*}{\partial t} - \frac{\partial (a+\varphi)\tilde{\varphi}^*}{\partial x} = \frac{\partial}{\partial x}\left(d\,\frac{\partial \tilde{\varphi}^*}{\partial x} \right) + \left(c - \frac{\partial \varphi}{\partial x} \right)\tilde{\varphi}^* + \psi\frac{\partial \varphi^*}{\partial x}, \tag{67}$$
$$\tilde{\varphi}^*|_{t=T} = 0,\ \left((a+\varphi)\tilde{\varphi}^* + d\,\partial\tilde{\varphi}^*/\partial x \right)|_{x=0,1} = -\psi\tilde{\phi}^*|_{x=0,1}.$$

Equation (67) is often referred as the 'second-order adjoint' [47]. One can see that it is the usual adjoint equation (64) with a distributed source term (the last term in 67). Having a solver for the discretized equation (58) at hand, it is relatively easy to adjust it for solving problems (64), (66) and (67) by manipulating the coefficients and source terms. The resulting tangent linear and adjoint models are called 'inconsistent' models, since they are obtained using the 'optimize-then-discretize' approach. In some realistic large-scale applications the accuracy of the gradients produced by such models is considered as unsatisfactory. Alternatively, one can follow the 'discretize-then-optimize' approach, developing the models either manually or using the Automatic Differentiation (AD). The resulting adjoint model provides a precise gradient, however the model itself is often computationally inefficient and may suffer from instability. Since the **expectations of the derivatives** are sought in the UQ framework, the accuracy of the gradients here is less important than in optimization. Thus, the inconsistent models (if available) should be preferred because they could be far more efficient computationally. In this paper, however, we use the tangent-linear and adjoint models obtained by means of the AD engine TAPENADE [20], which has been applied to the Fortran routine implementing (59). This has been done to simplify the development of the routines computing the derivatives.

Let us denote by $forward(\bar{U}, \xi, \varepsilon(\xi))$ the subroutine which computes the design function. By differentiating this subroutine in 'forward' and 'backward' modes (scalar $\varepsilon(\xi)$ with respect to vector ξ) we obtain the tangent linear model $forward_d(\bar{U}, \xi, \xi_d, \varepsilon(\xi), \varepsilon_d(\xi))$ and the adjoint model $forward_b(\bar{U}, \xi, \xi_b, \varepsilon(\xi), \varepsilon_b)$, respectively. Here $\varepsilon_b = 1$ is a scaling constant and ξ_b is the sought gradient (63). Next, by differentiating $forward_d(\bar{U}, \xi, \xi_d, \varepsilon(\xi), \varepsilon_d(\xi))$ in the 'backward' mode (scalar $\varepsilon_d(\xi)$ with respect to vector ξ) we obtain $forward_d_b(\bar{U}, \xi, \xi_b, \xi_d, \varepsilon(\xi), \varepsilon_d(\xi), \varepsilon_d_b)$. Here $\varepsilon_d_b = 1$ is a scaling constant, ξ_d is the trial vector v and ξ_b is the sought Hessian-vector product (65).

Let us finally note that the development of the adjoint for a more complex model using AD may not necessarily be easier than the development of the 'inconsistent' versions (due to a number of reasons).

8. Numerical experiments

8.1. Global sensitivity indices (GSI)

Here we present the GSA results using the derivative-based measures associated with (33) and (41). First, we evaluate the reference values of the 'main-effect' (29) and 'total effect' (30) GSI using the direct Monte Carlo simulations. Numerical implementation of (29) and (30) for finite samples replicates the procedure presented in [37]. Next, these indices are compared to the derivative-based estimates.

Let $\xi^l = (\xi_1^l, \dots, \xi_N^l)^T$ and $\xi'^{(l)} = (\xi_1'^{(l)}, \dots, \xi_N'^{(l)})^T$, $l = 1, \dots, L_0$ be two independent samples of random vectors with elements $\xi_i^l \sim N(0,1)$. Introduce vectors

$$\psi_i = (\xi_1, \dots, \xi_{i-1}, \xi_i', \xi_{i+1}, \dots, \xi_N),\ \psi_{-i} = (\xi_1', \dots, \xi_{i-1}', \xi_i, \xi_{i+1}', \dots, \xi_N').$$

Then, the 'main-effect' and 'total effect' GSI are defined as follows:

$$\bar{S}_i = \frac{\bar{U}_i - \bar{E}_1^2(\varepsilon^*(\xi))}{\overline{Var}(\varepsilon^*(\xi))},\ \bar{S}_i^T = 1 - \frac{\bar{U}_{-i} - \bar{E}_2^2(\varepsilon^*(\xi))}{\overline{Var}(\varepsilon^*(\xi))}, \tag{68}$$

where

$$\bar{U}_i = \frac{1}{L_0 - 1} \sum_{l=1}^{L_0} \varepsilon^*(\xi^l)\varepsilon^*(\psi_{-i}^{(l)}),\ \bar{U}_{-i} = \frac{1}{L_0 - 1} \sum_{l=1}^{L_0} \varepsilon^*(\xi^l)\varepsilon^*(\psi_i^{(l)}),$$

$$\bar{E}_1^2(\varepsilon^*(\xi)) = L_0^{-1} \sum_{l=1}^{L_0} \varepsilon^*(\xi^l)\varepsilon^*(\xi'^{(l)}),\ \bar{E}_2^2(\varepsilon^*(\xi)) = \left(L_0^{-1} \sum_{l=1}^{L_0} \varepsilon^*(\xi^l) \right)^2,$$

and $\overline{Var}(\varepsilon^*(\xi))$ is the sample variance (40). A large sample size ($L_0 = 10^5$) is used for these computations.

The derivative-based measures are computed using a sample of gradients

$$(\partial\varepsilon^*(\xi^l)/\partial\xi_1, \dots, \partial\varepsilon^*(\xi^l)/\partial\xi_N)^T,\ l = 1, \dots, L_1,$$

where ξ^l is an element from the sample of random vectors ξ. These measures are defined as follows:

$$\bar{S}_i = \frac{1}{\overline{Var}(\varepsilon^*(\xi))} \left(L_1^{-1} \sum_{l=1}^{L_1} \frac{\partial\varepsilon^*(\xi^l)}{\partial\xi_i} \right)^2,\ \bar{S}_i^T = \frac{1}{\overline{Var}(\varepsilon^*(\xi))}\ L_1^{-1} \sum_{l=1}^{L_1} \left(\frac{\partial\varepsilon^*(\xi^l)}{\partial\xi_i} \right)^2. \tag{69}$$

Here, a relatively small sample size ($L_1 = 400$) is used.

Finally, the scaled local sensitivity based measure is introduced. It is computed at the reference (unperturbed) value of the control vector $\bar{U}$ as follows:

$$\hat{S}_i^0 = \frac{\beta}{\overline{Var}(\varepsilon^*(\xi))} \left(\frac{\partial \varepsilon^*(0)}{\partial \xi_i} \right)^2, \tag{70}$$

where

$$\beta = \sum_{i=1}^{N} \hat{S}_i^T / \sum_{i=1}^{N} \left(\frac{\partial \varepsilon^*(0)}{\partial \xi_i} \right)^2.$$

The scaling enables the shapes of $\hat{S}_i^0$ and $\hat{S}_i^T$ to be compared.

The GSA results are presented in Fig. 5 and Fig. 6. The panels in Fig. 5 show the 'main effect' and 'total effect' GSI – in dotted lines, and the derivative-based measures – in solid lines. The left, central and right panels demonstrate the indices associated with the initial condition $u(x)$, the convection coefficient $a(x)$ and the reaction coefficient $c(x)$, correspondingly. Each panel in the vertical direction corresponds to different values of x^* used in (60).

As expected, the numerical results conform with the theoretical predictions from [42] (presented in §4), asserting that: a) the mean squared gradient is the upper bound for the 'total effect' GSI; b) the squared mean gradient is the lower bound for the 'main effect' GSI. While the possibility of using the adjoint model is mentioned in [24], no relevant example for a distributed parameter high-dimensional model has been reported so far. Besides, from our numerical tests we can draw some interesting conclusions.

First, the original derivative-based measure S_i^T in (69) approximates the 'total effect' GSI exceptionally well and, therefore, can be directly used for ranking. As a result we could not find an example where the benefit of using the improved measure (39) is clearly demonstrated. This does not deny the importance of measure (39), which might be useful in different circumstances. It is interesting to note that such quality of approximation has been achieved despite the strong nonlinearity of the Burgers' model, manifesting itself in other ways (see e.g. Fig. 6 or results in §8.3).

Second, the 'strong impact' variables may cover significant areas of the model spatial domain. This depends on the transport mechanisms supported by the model and the simulation period. Thus the importance set may constitute a significant part of the full input. This is different from the practice of sensitivity analysis involving the lumped-parameter systems, where the existence of a small number of the most influential inputs is anticipated. These conclusions can likely be extrapolated to the more complex CFD models, such as described by the Shallow Water equations or the Primitive Equations of the Ocean for example, which play important role in geophysical CFD applications.

In addition, in Fig. 6 we also compare $\bar{S}_i^T$ with the local sensitivity based indices $\bar{S}_i^0$, defined by (70). Since the gradient can be obtained by a single run of the adjoint model, the local sensitivity based approach is naturally attractive for the high-dimensional geophysical models involved in variational estimation. The considered case corresponds to the first row of images in Fig. 5. One can observe the essential difference between $\bar{S}_i^T$ and $\bar{S}_i^0$, which means that the ranking based on $\bar{S}_i^0$ could be wrong. This example underlines the importance of GSA. Let us note that for a linear design function the global and local sensitivities are equivalent. Thus, the difference between the global and local sensitivities indicates the presence of the strong nonlinearity in the model equations.

8.2. Filtering procedure

The explicit expressions for computing the PC expansion using the first and second derivatives of $\varepsilon(\xi)$ (gradients and Hessians) has been introduced in §3. The key components of the method are the filtering procedures described in §5. Given a relatively small sample of gradients, these procedures allow the expected second and third derivatives (a basis for evaluating the second- and third-order PC coefficients, respectively) to be estimated with an acceptable level of the sampling error.

In order to illustrate the effect of filtering we check numerically formula (21). First, a sample of Hessians Θ^l, $l = 1, \ldots, L$ of size $L = 400$ is computed by the Lanczos method, using the exact definition of the Hessian-vector product (65). Each Hessian is represented in the limited-memory form (15), involving $S = 20$ eigenpairs. Then, the sample mean $\bar{\Theta}$ is evaluated as

$$\bar{\Theta} = L^{-1} \sum_{l=1}^{L} \Theta^l.$$

Second, we compute matrix A defined by formula (45). Given $A \cdot v$ in the form (46), the eigenvalue decomposition of A reveals its symmetric part $\hat{A}$ in the form (47), involving $S_1 = 10$ eigenpairs. The numerical results, which illustrate the performance of the filtering procedure, are presented in Fig. 2 and Fig. 3. The results correspond to the case $x^* = 0.50$. In particular, Fig. 2(left) shows $\bar{\Theta}$, Fig. 3(left) shows matrices A (i.e. before filtering) for different L, whereas Fig. 3(right) shows the corresponding matrices $\hat{A}$, i.e. symmetrized and filtered. These results confirm that the suggested filtering procedure removes the sampling error very efficiently, which has a crucial effect on the histogram estimates (see discussion in the next Section). Since Θ^l have been available in this experiment, we also present in Fig. 2(right) the total interaction indices (TII)

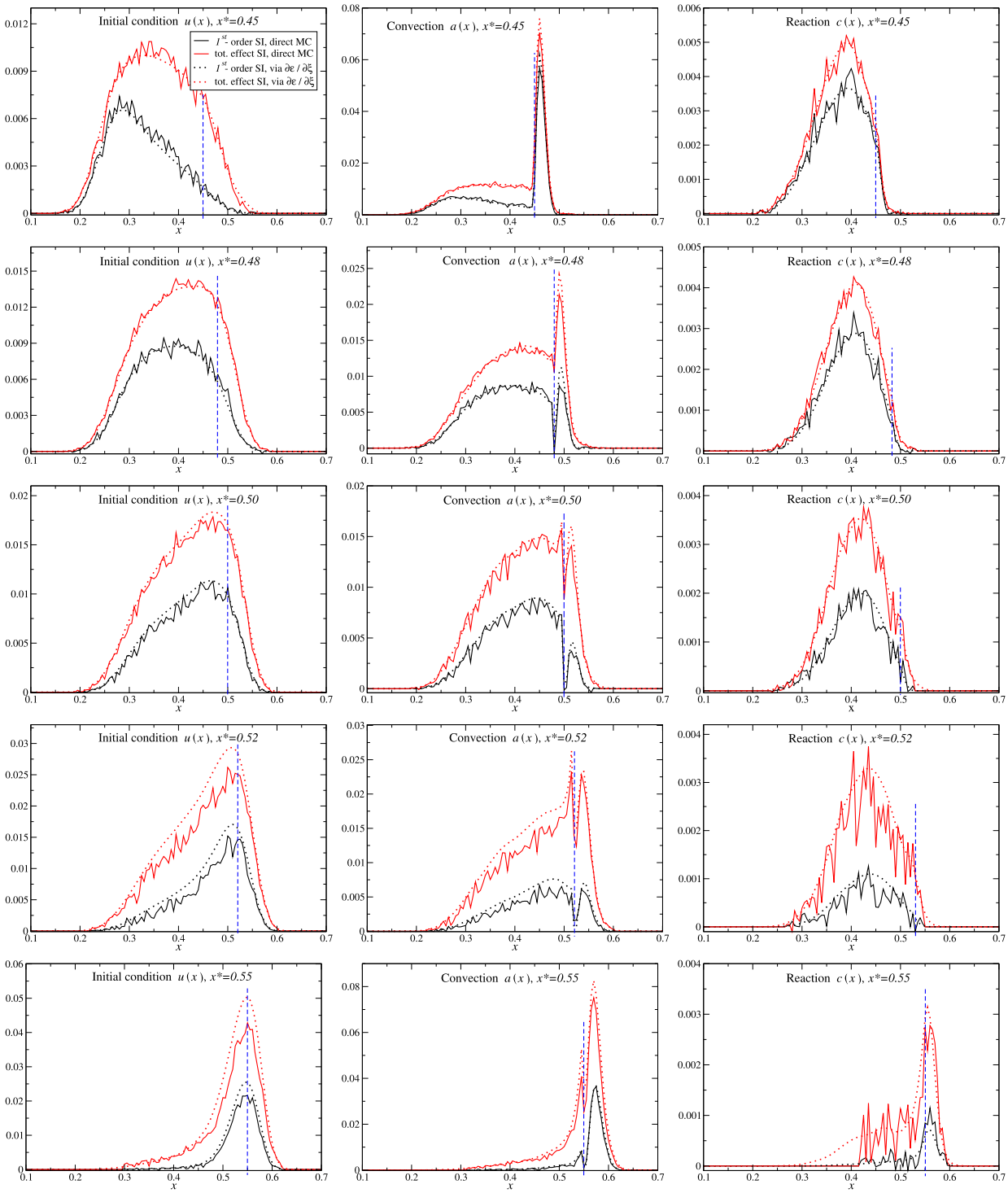


Fig. 5. First-order 'main-effect' and 'total-effect' GSI by direct Monte Carlo and using $\partial\varepsilon/\partial\xi$.

$$E\left[\Theta_{i,j}^2\right] \approx L^{-1}\sum_{l=1}^{L}\left(\Theta_{i,j}^{(l)}\right)^2,$$

which are important measures considered in the second-order interaction screening [36].

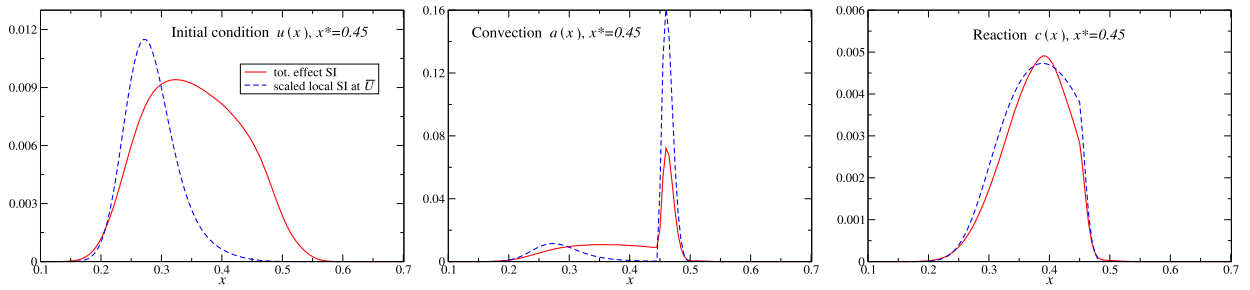


Fig. 6. 'Total-effect' global sensitivity indices using $\partial\varepsilon/\partial\xi$ and scaled local sensitivities (squared) at $U = \bar{U}$.

8.3. Probability density via histogram

The major outcome of GSA is the ranking of inputs which reveals the importance set. For this (relatively small) set the metamodels are constructed in the form (7), also taking into account expressions (56) and (57). By running the Monte Carlo simulations involving the metamodels the histograms are built. The histogram approximates the probability density function (pdf) of $\varepsilon(\xi)$, providing its most exhaustive stochastic description. For comparison, the reference histogram is built by running the direct Monte Carlo simulations involving the original model (58). The large sample size $L_0 = 10^5$ is used in all these simulations.

Given $\varepsilon(\xi^l)$, $l = 1, \cdots, L$, the sample variance $\overline{Var}(\varepsilon(\xi))$ can be estimated according to its definition (40). Alternatively, the total variance can be expressed as a sum of partial variances associated with the PC coefficients of a certain order as follows (e.g. [7]):

$$Var(\varepsilon(\xi)) := D_1 + D_2 + D_3 + \ldots = \sum_{i_1=1}^{N} \beta_{i_1} b_{i_1}^2 + \sum_{i_1=1}^{N} \sum_{i_2=1}^{i_1} \beta_{i_1 i_2} b_{i_1 i_2}^2 + \sum_{i_1=1}^{N} \sum_{i_2=1}^{i_1} \sum_{i_3=1}^{i_2} \beta_{i_1 i_2 i_3} b_{i_1 i_2 i_3}^2 + \ldots \tag{71}$$

The first-order metamodel includes first two terms of (7), involving coefficients b_0 and b_{i_1}. We introduce the scaling factor

$$\alpha_1 = \overline{Var}(\varepsilon(\xi))/D_1$$

and use $\alpha_1^{1/2} b_{i_1}$ in the metamodel instead of b_{i_1}. This guaranties that the variance of the pdf built with the first-order metamodel is exactly $\overline{Var}(\varepsilon(\xi))$. Thus, the first-order metamodel simply replicates the gaussian distribution with the mean equal to b_0 and the variance equal to $\overline{Var}(\varepsilon(\xi))$. The second-order metamodel includes three terms of (7) involving b_0, b_{i_1} and $b_{i_1 i_2}$. Now, we introduce the scaling factor

$$\alpha_2 = \overline{Var}(\varepsilon(\xi))/(D_1 + D_2),$$

and use $\alpha_2^{1/2} b_{i_1}$ and $\alpha_2^{1/2} b_{i_1 i_2}$ instead of b_{i_1} and $b_{i_1 i_2}$, respectively. This also guaranties that the variance of the pdf built with the second-order metamodel is exactly $\overline{Var}(\varepsilon(\xi))$. However, as opposite to the first-order metamodel, the shape of the pdf is not bound to be gaussian. The similar scaling is used for the third-order metamodel. Thus, all three metamodels would provide the pdf having the same mean and variance.

The results are presented in Fig. 7. Here, each panel corresponds to the corresponding x^* used in (60), and contains four graphs: the reference histogram – in wide pale line, the histogram using the first-order metamodel ($p = 1$) – in dotted line, the second-order metamodel ($p = 2$) – in dashed line and the third-order metamodel ($p = 3$) – in solid line. The corresponding first four distribution moments (mean, variance, skewness and kurtosis) are presented in Table 1.

First, we notice that for $x^* = 0.4$ and $x^* = 0.45$ the pdf remains close to the gaussian. This is despite strong non-linear effects, demonstrated in Fig. 6 for example. In these cases all three metamodels give close results. In cases $x^* = \{0.50, 0.52, 0.55, 0.6\}$ the second-order metamodel shows an impressive performance, allowing to catch the asymmetry of the sought pdf and the true location of its mode. Standing apart is the case $x^* = 0.48$, where the pdf has a strong kurtosis. Here, the second-order metamodel does not improve the results of the first-order metamodel, however using the third-order metamodel looks useful. The case $x^* = 0.50$ is another one where the impact of the third-order coefficients is essential. From these examples we conclude that the second-order metamodel allows the asymmetry to be described, which is usually associated with the skewness, whereas the third-order metamodel helps to describe the tailedness, associated with the kurtosis. Of course, the third-order metamodel is far more expensive both to generate and to run, and it may not be feasible if the importance set is large.

In the numerical experiments the PC coefficients for metamodels have been generated both by the proposed projection method, and by the least-square method described in §2. Naturally, the projection method is less expensive (both in terms of memory and the CPU time) since no minimization is required. However, the accuracy of histograms obtained by the two methods is interesting to compare. As 'data' we have used the samples $\{\varepsilon^*(\xi^l),\ \partial\varepsilon^*(\xi^l)/\partial\xi\}$, $l = 1, \ldots, L$, where $L = 400$.

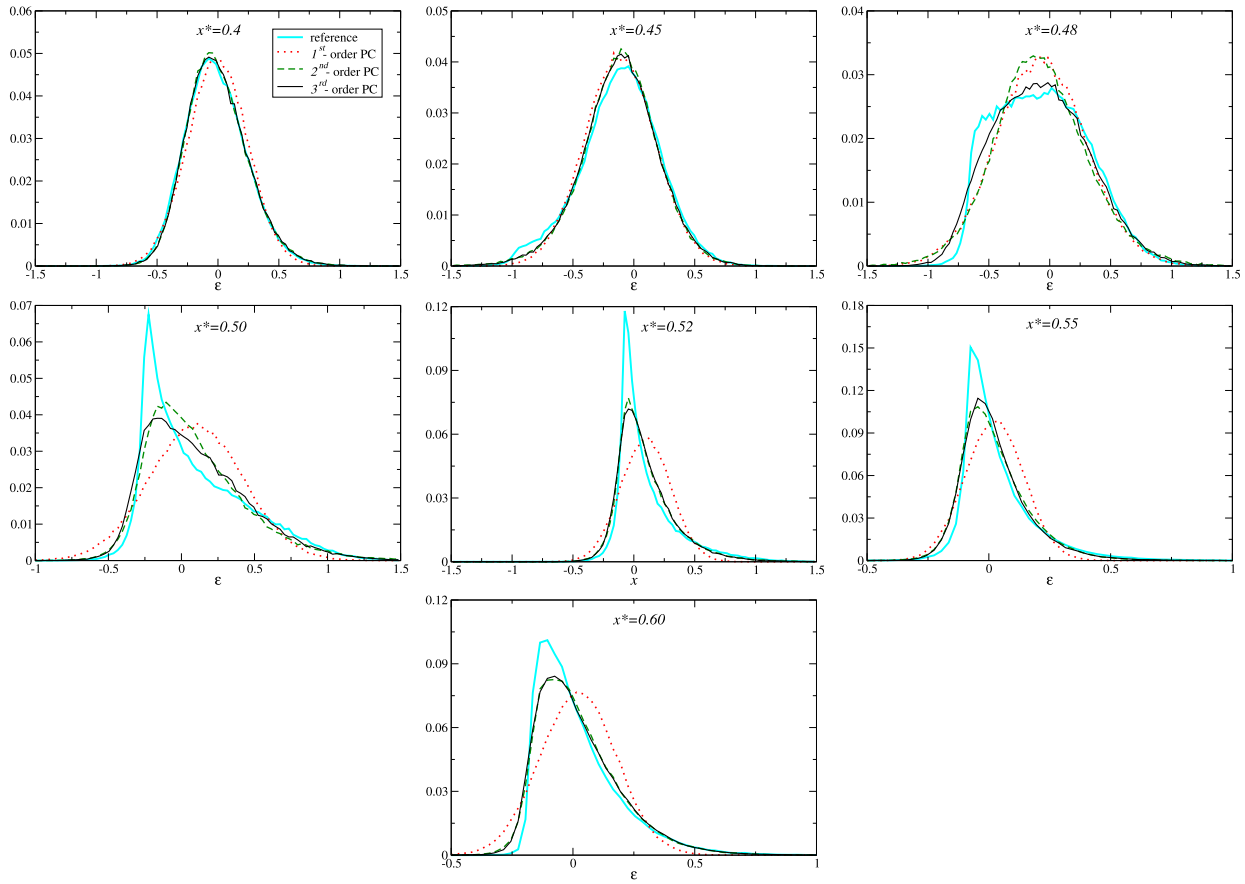


Fig. 7. Probability density function (histogram) of $\varepsilon(\xi)$, for $L = 400$.

In the least-square method the cost-function (13) without the regularization term is considered, and its gradient has been generated by the AD. For the second-order PC expansion, the least-square problem is overdetermined for $L = 400$, thus the minimization process is converging as long as the gradient accuracy allows. For the third-order PC expansion, the same least-square problem is under-determined, thus the minimization process is stopped by the L-curve criterion [19]. Analyzing the results we found that the histograms of a better quality are obtained by the projection method. In particular, the accuracy of the second-order metamodel is noticeably better in the case shown in Fig. 8(left). The performance of the third-order metamodel generated with the least-square method is unsatisfactory, since it is unable to catch the kurtosis anomaly of the histogram Fig. 8(right). At the same time, the third-order metamodel generated with the projection method performs satisfactorily in this case.

The impact of filtering on the estimated histograms is demonstrated in Fig. 9(left). Here, the histograms obtained without and with filtering are presented, alongside the reference one. In order to understand the mechanism of this impact, let us compare the corresponding partial variances D (see in (71)). That is, without filtering we obtain $D_1 = 4.652$, $D_2 = 4.561$ and $D_3 = 265.8(!)$, with filtering – $D_1 = 4.652$, $D_2 = 1.729$ and $D_3 = 0.332$. The difference in D_3 is particularly striking. No wonder that the third-order term obtained without filtering totally dominates the behavior of the metamodel, resulting into the erroneous histogram. Filtering reduces the norm of the second- and third-order terms by removing the sampling error. Thus, the corresponding histograms look far more sensible. Finally, Fig. 9(right) demonstrates how the accuracy of the projection method depends on the sample size. Here we observe that the histograms approach the reference one with increasing L. This is a sign of consistency in statistical computations.

8.4. CPU-time estimates

Here we present some estimates of the CPU time budget (a single processor implementation) required for different key computational steps. Of course, the attention should be paid to the time ratios. Let us remind that the size of the importance set involved is $N = 210$, the sample size is $L = 400$.

a) Computing second-order coefficients $b_{i,j}$ (Section 8.2): 'full memory'/'limited-memory' versions – 0.16 s/2.9 s;

b) Computing third-order coefficients $b_{i,j,k}$ (Section 8.3): 'full memory'/'limited-memory' versions – 61.2 s/725.2 s;

Table 1
Auxiliary information to histograms in Fig. 7.

$L = 400$	*mean*	*variance*	*skewness*	*kurtosis*
$x^* = 0.40,\ N = 167$				
reference	-1.154×10^{-2}	6.357×10^{-2}	0.3311	0.2230
2*nd*-order PC	-1.310×10^{-2}	6.348×10^{-2}	0.4488	0.5228
3*rd*-order PC	-1.310×10^{-2}	6.375×10^{-2}	0.4070	0.4817
$x^* = 0.45,\ N = 198$				
reference	-0.1152	0.1074	-0.2137	0.1289
2*nd*-order PC	-0.1160	9.909×10^{-2}	-0.3075	0.9873
3*rd*-order PC	-0.1160	9.960×10^{-2}	-0.2589	0.7740
$x^* = 0.48,\ N = 204$				
reference	-8.546×10^{-2}	0.1576	0.2286	-0.6636
2*nd*-order PC	-7.406×10^{-2}	0.1489	0.1838	0.5466
3*rd*-order PC	-7.400×10^{-2}	0.1496	0.2420	-0.2066
$x^* = 0.50,\ N = 208$				
reference	9.572×10^{-2}	0.1242	0.9233	0.3226
2*nd*-order PC	0.1026	0.1211	1.095	1.983
3*rd*-order PC	0.1022	0.1213	0.8900	0.7310
$x^* = 0.52,\ N = 210$				
reference	0.1016	6.286×10^{-2}	1.656	3.376
2*nd*-order PC	9.866×10^{-2}	5.829×10^{-2}	1.441	3.719
3*rd*-order PC	9.865×10^{-2}	5.886×10^{-2}	1.498	4.307
$x^* = 0.55,\ N = 210$				
reference	3.362×10^{-2}	2.293×10^{-2}	1.792	5.674
2*nd*-order PC	2.300×10^{-2}	1.946×10^{-2}	1.032	2.185
3*rd*-order PC	2.298×10^{-2}	1.987×10^{-2}	1.455	4.244
$x^* = 0.60,\ N = 100$				
reference	1.851×10^{-2}	2.760×10^{-2}	1.414	2.683
2*nd*-order PC	2.229×10^{-2}	2.868×10^{-2}	1.076	1.864
3*rd*-order PC	2.241×10^{-2}	2.920×10^{-2}	1.207	1.970

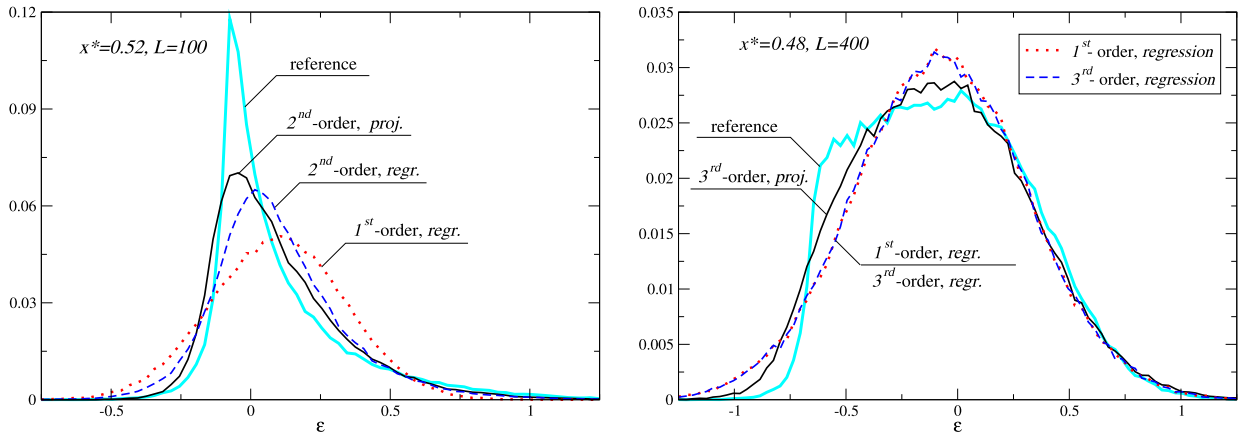


Fig. 8. Histogram of $\varepsilon(\xi)$ by the projection method and by the regression method (standard least-square).

c) One iteration of the gradient-enhanced regression method (Section 2) for the second-order PC expansion – 1.25 s, for the third-order PC expansion – 118.4 s;

d) One run of the metamodel: second-order – 1.51×10^{-3} s, third-order – 2.05×10^{-2} s;

e) One run of the adapted metamodel (Section 6): second-order – 1.42×10^{-4} s, third-order – 2.93×10^{-3} s;

f) One run of the forward model (Section 7.1): 2.7×10^{-2} s;

g) One run of the forward-adjoint model (Section 7.2): 4.9×10^{-2} s;

h) Time of computing the limited-memory Hessian (10 eigenpairs): 0.95 s.

The presented CPU-time budget confirms that the projection method provides a far more efficient tool for processing the computed samples $\{\varepsilon^*(\xi^l),\ \partial\varepsilon^*(\xi^l)/\partial\xi\}$. Indeed, the second-order metamodel is generated in 0.16 s. At the same time, just one iteration of the regression method requires 1.25 s, whereas a few tens of such iterations could be necessary. Besides,

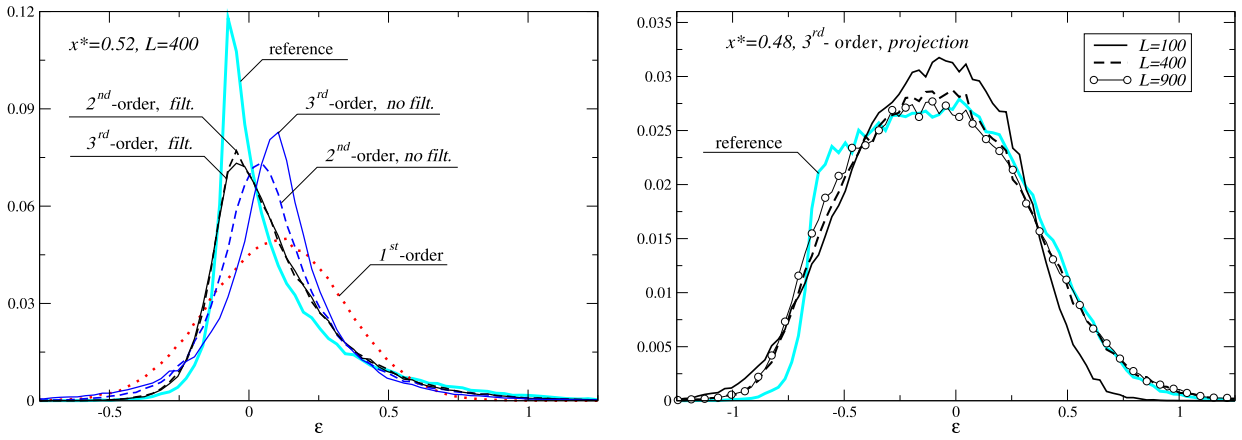


Fig. 9. Left: histogram of $\varepsilon(\xi)$, involving filtering and without filtering. Right: histogram of $\varepsilon(\xi)$ for different sample sizes $L = 100/400/900$, case $x^* = 0.48$.

the adapted metamodel generated by the projection method is about ten times faster than the metamodel obtained by the regression method. The third-order metamodel is generated in 61.2 s, while one iteration of the regression method requires 118.4 s. The number of iterations required to resolve the regression problem of size $M = 1587986$ is mounting to a few hundreds.

Conclusions

Performing Sensitivity Analysis and Uncertainty Quantification involving large-scale fluid flow models currently remains a serious challenge. One difficulty is the computational cost of running such models that limits the direct use of the Monte Carlo methods. Another difficulty is the high-dimensional discretized model input, which may not always be reduced to the importance set of a feasible size. At the same time, some of the models under consideration are involved in the process of variational estimation (data assimilation). This implies that the tangent linear and adjoint counterparts of such models are available, and, therefore, one can compute the gradient (first derivative) and the Hessian (second derivative) of any design function (QoI) defined over the state variables. These derivatives can be used both for SA and UQ.

The use of the polynomial chaos expansion is considered as a promising way of reducing the computational complexity of SA and UQ. A relatively recent developments in this direction are the gradient-based global sensitivity measures and the gradient-enhanced methods for computing the PC expansion. The latter have been implemented in the framework of the regression approach. In this paper we propose a general **derivative-enhanced projection method** for computing the PC expansion, and the **improved derivative-based sensitivity measure**, both suitable for high-dimensional and computationally expensive models. By 'derivative-enhanced' we mean that both the gradient and the Hessian information can be utilized. This method allows a high-level parallelization, e.g. the PC coefficients of different order can be computed independently. Being computationally more efficient than the gradient-enhanced least square regression methods (explicit filtering instead of minimization), the proposed method has demonstrated a better performance in terms of accuracy (using the same sample data).

Regarding SA, the numerical tests have demonstrated that the gradient-based global sensitivity measure approximates the total-effect GSI exceptionally well. Thus, it must be preferred to the adjoint (local) sensitivity-based measures, the use of which may result into a seriously wrong ranking and an irrelevant importance set. The look of the total-effect sensitivity indices confirms that the impact of the spatially distributed input variables decays gradually as a function of distance. Thus, one may not anticipate the design function to be dependent on a 'few most significant inputs', which is often a basic assumption underlying the possible use of the PC expansion. Thus, the third-order PC expansion for such models looks like a limit.

Regarding UQ, the results show that the second-order PC-expansion gives a noticeable improvement over the first-order PC-expansion in approximating the pdf of the design function. In particular, it allows its asymmetry and the position of the mode to be revealed. Let us note that, with our method, computing the second-order PC expansion is feasible for the full input of almost any size. Next, given the second-order PC coefficients, one can compute the improved sensitivity measure, which can be used for ranking and defining the importance set. The third-order PC coefficients can be computed for the importance set only. The third-order PC expansion is far more expensive than the second-order PC expansion, however its contribution to the estimated pdf had been important only in one case ($x^* = 0.48$). The main conclusion here is that the second-order analysis is, probably, the sought-for tradeoff: it is feasible for the existing models in mind and efficient to reveal some important non-gaussian features of the design function. In this respect, it is important to develop a criterion which may predict the usefulness of computing the next-order PC coefficients beforehand. Another important issue is the upgrade of the third-order filtering procedure. This requires the Lanczos algorithm to be generalized for the cube (at least).

The paper shows how the Hessian information can be utilized, however the conducted numerical tests have not provided a definite answer on its usefulness in practical terms. In one hand, the Hessian could be a far more expensive object than the gradient. On the other hand, while one gradient provides N constraints, one Hessian provides $N(N+1)/2$ constraints. Thus, computing the Hessian is practical if the tangent linear and adjoint models are much less expensive than the forward model. This could be the case when the implicit time integration scheme is used and one must perform a significant number of nonlinear iterations (fixed point or Newton) at each time step.

Acknowledgements

This work was supported by the French national programme LEFE/INSU. The work of V. Shutyaev was carried out within Russian Science Foundation project 17-77-30001 (studies in Sections 4–5), and the project 18-01-00267 of the Russian Foundation for the Basic Research. The authors also acknowledge the useful contributions of two anonymous reviewers.

Appendix 1. Proof of Theorem 1

We will use the equivalent representation of (7) in the form [45]:

$$\varepsilon = \sum_{|\alpha|\le 3} a_\alpha \psi_\alpha(\xi), \tag{72}$$

where $\alpha = (\alpha_1, \dots, \alpha_N)$ is an integer sequence (tuple, or multi-index), $|\alpha| = \alpha_1 + \dots + \alpha_N$, $\xi = (\xi_1, \dots, \xi_N)$, $\psi_\alpha(\xi) = \prod_{l=1}^{N} K_{\alpha_l}(\xi_l)$, and $K_{\alpha_l}(\xi_l)$ is the univariate Hermite polynomial of degree α_l dependent on the variable ξ_l.

The correspondence between two equivalent representations of ε is given by the equalities:

$$b_0 = a_{\alpha_0},\ \text{where}\ \alpha_0 = (0, \dots, 0),\ \psi_{\alpha_0}(\xi) = H_0 = 1; \tag{73}$$

$$b_i = a_\alpha,\ \text{where}\ \alpha = (\alpha_1, \dots, \alpha_N),\ \alpha_l = \delta_{il},\ l = \overline{1, N}, \tag{74}$$

$$\psi_\alpha(\xi) = \prod_{l=1}^{N} K_{\alpha_l}(\xi_l) = K_{\alpha_i}(\xi_i) = K_1(\xi_i) = H_1(\xi_i);$$

$$b_{ij} = a_\alpha,\ \text{where}\ \alpha = (\alpha_1, \dots, \alpha_N), \tag{75}$$

$$\alpha_l = \begin{cases} 2,\ l = i = j, \\ 1,\ l = i,\ l = j,\ i \ne j,\ j < i, \\ 0,\ \text{else}, \end{cases} \tag{76}$$

$$\psi_\alpha(\xi) = \prod_{l=1}^{N} K_{\alpha_l}(\xi_l) = \begin{cases} K_{\alpha_i}(\xi_i), & i = j \\ K_{\alpha_i}(\xi_i) K_{\alpha_j}(\xi_j), & i \ne j \end{cases} = \begin{cases} K_2(\xi_i), & i = j \\ K_1(\xi_i) K_1(\xi_j), & i \ne j \end{cases} = H_2(\xi_i, \xi_j);$$

$$b_{ijk} = a_\alpha,\ \text{where}\ \alpha = (\alpha_1, \dots, \alpha_N), \tag{77}$$

$$\alpha_l = \begin{cases} 3,\ l = i = j = k \\ 2,\ l = i = j,\ l = j = k,\ i > k \\ 1,\ l = i,\ l = j,\ l = k,\ i > j > k,\ l = \overline{1, N}, \\ 0,\ \text{else}, \end{cases} \tag{78}$$

$$\psi_\alpha(\xi) = H_3(\xi_i, \xi_j, \xi_k).$$

Using the properties of the polynomials ψ_α [45], from (72) we get

$$E(\psi_\alpha\, \varepsilon) = \sum_{|\beta|\le 3} a_\beta E(\psi_\alpha \psi_\beta) = \sum_{|\beta|\le 3} a_\beta E(\psi_\alpha^2)\delta_{\alpha\beta} = a_\alpha E(\psi_\alpha^2),$$

$$\psi_\alpha^2 = \Big(\prod_{l=1}^{N} K_{\alpha_l}(\xi_l)\Big)^2 = \prod_{l=1}^{N} K_{\alpha_l}^2(\xi_l),$$

$$E(\psi_\alpha^2) = \prod_{l=1}^{N} E\Big(K_{\alpha_l}^2(\xi_l)\Big) = \prod_{l=1}^{N} \alpha_l!.$$

Then

$$E(\psi_\alpha\, \varepsilon) = a_l \prod_{l=1}^{N} \alpha_l!. \tag{79}$$

For $\alpha = 0$ we have $\psi_\alpha = H_0 = 1$ and $a_\alpha = b_0$, then

$$E(\varepsilon) = b_0. \tag{80}$$

For $|\alpha| = 1$ we have $\psi_\alpha = H_1(\xi_i)$ and $\alpha_l = \delta_{li}$, $a_\alpha = b_i$ (according to (74)), then

$$E(H_1(\psi_i)\,\varepsilon) = b_i = \beta_i b_i, \tag{81}$$

where $\beta_i = 1,\ i = \overline{1, N}$.

For $|\alpha| = 2$ we have $\psi_\alpha = H_2(\xi_i, \xi_j)$, α_l is defined in (76), $a_\alpha = b_{ij}$, then

$$E(H_2(\psi_i, \xi_j)\,\varepsilon) = \beta_{ij} b_{ij}, \tag{82}$$

where

$$\beta_{ij} = \begin{cases} 2!, & i = j \\ 1, & i \neq j,\ i > j. \end{cases}$$

For $|\alpha| = 3$ we have $\psi_\alpha = H_3(\xi_i, \xi_j, \xi_k)$, α_l is defined by (78), $a_\alpha = b_{ijk}$, then from (79) we get

$$E(H_3(\xi_i, \xi_j, \xi_k)\,\varepsilon) = \beta_{ijk} b_{ijk}, \tag{83}$$

where

$$\beta_{ijk} = \begin{cases} 3!, & i = j = k \\ 2!, & i = j \neq k,\ i \neq j = k \\ 1!, & i \neq j \neq k. \end{cases}$$

The formulas (80)–(83) give us the first equalities in (19)–(22).

Consider $E\left(\psi_\alpha \frac{\partial \varepsilon}{\partial \xi_k}\right)$, $|\alpha| \leq 3$. For $\frac{\partial \varepsilon}{\partial \xi_k}$ we have:

$$\frac{\partial \varepsilon}{\partial \xi_k} = \sum_{|\beta| \leq 3} a_\beta \frac{\partial \psi_\beta}{\partial \xi_k} = \sum_{|\beta| \leq 3} a_\beta \prod_{l=1, l \neq k}^{N} K_{\beta_l}(\xi_l) \beta_k K_{\beta_k - 1}(\xi_k).$$

Then

$$\psi_\alpha \frac{\partial \varepsilon}{\partial \xi_k} = \sum_{|\beta| \leq 3} a_\beta \prod_{l=1}^{N} K_{\alpha_l}(\xi_l) \cdot \prod_{l=1, l \neq k}^{N} K_{\beta_l}(\xi_l) \beta_k K_{\beta_k - 1}(\xi_k).$$

Due to properties of $K_{\alpha_l}(\xi_l)$ [45], we get

$$E\left(\psi_\alpha \frac{\partial \varepsilon}{\partial \xi_k}\right) = \sum_{|\beta| \leq 3} a_\beta \prod_{l=1, l \neq k}^{N} E\left(K_{\alpha_l}(\xi_l) K_{\beta_l}(\xi_l)\right) \cdot \beta_k \cdot E\left(K_{\alpha_k}(\xi_k) K_{\beta_k - 1}(\xi_k)\right) =$$

$$= \sum_{|\beta| \leq 3} a_\beta \prod_{l=1, l \neq k}^{N} \alpha_l! \delta_{\alpha_l \beta_l} \cdot \beta_k \cdot \alpha_k! \delta_{\alpha_k \beta_k - 1}.$$

The expression under the summation sign is nonzero for $\beta_l = \alpha_l,\ l \neq k$ and $\beta_k = (1 + \alpha_k)$, then we get

$$E\left(\psi_\alpha \frac{\partial \varepsilon}{\partial \xi_k}\right) = \sum_{\substack{|\beta| \leq 3 \\ \beta_l = \alpha_l,\, l \neq k \\ \beta_k = \alpha_k + 1}} a_\beta \prod_{l=1, l \neq k}^{N} \alpha_l! (\alpha_k + 1)! \tag{84}$$

For $\alpha = 0$ we have $\psi_\alpha = H_0 = 1,\ \alpha_l = 0,\ l = \overline{1, N},\ \beta_l = \delta_{lk}$, then according to (74), $a_\beta = b_k$ and

$$E\left(H_0 \frac{\partial \varepsilon}{\partial \xi_k}\right) = a_\beta = b_k = \beta_k b_k. \tag{85}$$

For $|\alpha| = 1$ we have $\psi_\alpha = H_1(\xi_i),\ \alpha_l = \delta_{il}\ i = \overline{1, N}$. Then

$$\beta_l = \begin{cases} 2, & l = i = k \\ 1, & l = i \neq k,\ l = k \neq i, \\ 0, & \text{else.} \end{cases}$$

Hence, according to (75), $a_\beta = b_{ik}$ and from (84) we get

$$E\Big(H_1(\xi_i)\frac{\partial \varepsilon}{\partial \xi_k}\Big) = a_\beta \cdot \begin{cases} 2!, & i=k \\ 1, & i \neq k, \end{cases} = \beta_{ik} b_{ik}. \tag{86}$$

For $|\alpha| = 2$ we have $\psi_\alpha = H_2(\xi_i, \xi_j)$, α_l is defined by (76). Then in (84)

$$\beta_l = \begin{cases} 3, & l = i = j = k \\ 2, & l = i = j \neq k,\ l = j = k \neq i,\ i > k \\ 1, & l = i,\ l = j,\ l = k \\ 0, & \text{else.} \end{cases}$$

Hence, according to (77), $a_\beta = b_{ijk}$ and from (84) we get

$$E\Big(H_2(\xi_i, \xi_j)\frac{\partial \varepsilon}{\partial \xi_k}\Big) = a_\beta \cdot \begin{cases} 3!, & i=j=k \\ 2!, & i=j\neq k \\ 2!, & i\neq j=k \\ 1, & i\neq j\neq k \end{cases} = \beta_{ijk} b_{ijk}. \tag{87}$$

Formulas (85)–(87) give us the second equalities in (20)–(22).

Consider $E\Big(\psi_\alpha \frac{\partial^2 \varepsilon}{\partial \xi_j \partial \xi_k}\Big)$, $|\alpha| \le 3$. For $\frac{\partial^2 \varepsilon}{\partial \xi_j \partial \xi_k}$ we have

$$\frac{\partial^2 \varepsilon}{\partial \xi_j \partial \xi_k} = \sum_{|\beta| \le 3} a_\beta \frac{\partial^2 \psi_\beta}{\partial \xi_j \partial \xi_k},$$

where

$$\frac{\partial^2 \psi_\beta}{\partial \xi_j \partial \xi_k} = \begin{cases} \beta_k(\beta_k - 1) K_{\beta_k - 2}(\xi_k) \cdot \prod\limits_{l=1, l\neq k}^{N} K_{\beta_l}(\xi_l), & j = k \\ \beta_j K_{\beta_j - 1}(\xi_j) \beta_k K_{\beta_k - 1}(\xi_k) \cdot \prod\limits_{l=1, l\neq k, j}^{N} K_{\beta_l}(\xi_l), & j \neq k. \end{cases}$$

Then

$$\psi_\alpha \frac{\partial^2 \varepsilon}{\partial \xi_j \partial \xi_k} = \sum_{|\beta| \le 3} a_\beta \psi_\alpha \frac{\partial^2 \psi_\beta}{\partial \xi_j \partial \xi_k} =$$

$$= \sum_{|\beta| \le 3} a_\beta \cdot \begin{cases} \prod\limits_{l=1, l\neq k}^{N} K_{\alpha_l}(\xi_l) K_{\beta_l}(\xi_l) \beta_k(\beta_k - 1) K_{\beta_k - 2}(\xi_k) K_{\alpha_k}(\xi_k), & j = k \\ \prod\limits_{l=1, l\neq k, j}^{N} K_{\alpha_l}(\xi_l) K_{\beta_l}(\xi_l) \beta_j \cdot \beta_k \cdot K_{\alpha_j}(\xi_j) K_{\beta_j - 1}(\xi_j) \cdot K_{\alpha_k}(\xi_k) K_{\beta_k - 1}(\xi_k), & j \neq k. \end{cases}$$

Hence

$$E\Big(\psi_\alpha \frac{\partial^2 \varepsilon}{\partial \xi_j \partial \xi_k}\Big) = \sum_{|\beta| \le 3} a_\beta \times$$

$$\times \begin{cases} \prod\limits_{l=1, l\neq k}^{N} E\Big(K_{\alpha_l}(\xi_l) K_{\beta_l}(\xi_l)\Big) \cdot \beta_k(\beta_k - 1) E\Big(K_{\alpha_k}(\xi_k) K_{\beta_k - 2}(\xi_k)\Big), & j = k \\ \prod\limits_{l=1, l\neq k, j}^{N} E\Big(K_{\alpha_l}(\xi_l) K_{\beta_l}(\xi_l)\Big) \cdot \beta_j \beta_k E\Big(K_{\alpha_j}(\xi_j) K_{\beta_j - 1}(\xi_j)\Big) \cdot E\Big(K_{\alpha_k}(\xi_k) \cdot K_{\beta_k - 1}(\xi_k)\Big), & j \neq k. \end{cases}$$

Due to $E\Big(K_{\alpha_l}(\xi_l) K_{\beta_l}(\xi_l)\Big) = \delta_{\alpha_l \beta_l}$, $E\Big(K^2_{\alpha_l}(\xi_l)\Big) = \alpha_l!$ $\forall \alpha_l, \beta_l$, we get

$$E\Big(\psi_\alpha \frac{\partial^2 \varepsilon}{\partial \xi_j \partial \xi_k}\Big) = \sum_{|\beta| \le 3} a_\beta \cdot \begin{cases} \prod\limits_{l=1, l\neq k}^{N} \delta_{\alpha_l \beta_l} \alpha_l! \beta_k(\beta_k - 1) \alpha_k! \delta_{\alpha_k \beta_k - 2}, & j = k \\ \prod\limits_{l=1, l\neq k, j}^{N} \delta_{\alpha_l \beta_l} \alpha_l! \beta_j \beta_k \alpha_j! \alpha_k! \delta_{\alpha_j \beta_j - 1} \delta_{\alpha_k \beta_k - 1}, & j \neq k. \end{cases}$$

Under the summation sign, the expressions are nonzero only for $\beta_l=\alpha_l,\ l\neq k\ \beta_k=2+\alpha_k$ (if $j=k$) and for $\beta_l=\alpha_l,\ l\neq k,\ j$, $\beta_j=1+\alpha_j,\ \beta_k=1+\alpha_k$ (if $j\neq k$).

Then

$$E\Big(\psi_\alpha\frac{\partial^2\varepsilon}{\partial\xi_j\partial\xi_k}\Big)=\begin{cases}\sum\limits_{\substack{|\beta|\le 3\\ \beta_l=\alpha_l,\ l\neq k,\\ \beta_k=2+\alpha_k}}\ \prod\limits_{l=1,\ l\neq k}^{N}\alpha_l!(2+\alpha_k)!,\ \ j=k\\ \sum\limits_{\substack{|\beta|\le 3\\ \beta_l=\alpha_l,\ l\neq k,j\\ \beta_j=1+\alpha_j\ \beta_k=1+\alpha_k}}\ \prod\limits_{l=1,\ l\neq k,\ j}^{N}\alpha_l!(1+\alpha_j)!(\alpha_k+1)!,\ \ j\neq k.\end{cases}\tag{88}$$

For $\alpha=0$ we have $\psi_\alpha=H_0=1$, $\alpha_l=0,\ l=\overline{1,N}$, then $\beta_l=\begin{cases}2,\ l=j=k\\ 1,\ l=j\neq k,\ l=k\neq j\\ 0,\ \text{else.}\end{cases}$ Hence, according to (75), $a_\beta=b_{jk}$, and we get from (88):

$$E\Big(H_0\frac{\partial^2\varepsilon}{\partial\xi_j\partial\xi_k}\Big)=\begin{cases}2!\,b_{jj},\ \ j=k\\ 1!\,b_{jk},\ \ j\neq k\end{cases}=\beta_{jk}b_{jk}.\tag{89}$$

For $|\alpha|=1$ we have $\psi_\alpha=H_1(\xi_i)$, $\alpha_l=\delta_{il},\ l=\overline{1,N}$. Then, according to (88),

$$\beta_l=\begin{cases}3,\ l=i=j=k\\ 2,\ l=j=k\neq i\\ 2,\ l=i=j\neq k\\ 1,\ l=i,\ l=j,\ l=k\\ 0,\ \text{else.}\end{cases}$$

Then, due to the equivalence (77), $a_\beta=b_{ijk}$, and from (88) we get

$$E\Big(H_1\frac{\partial^2\varepsilon}{\partial\xi_j\partial\xi_k}\Big)=\begin{cases}3!\,b_{iii},\ \ i=j=k\\ 2!\,b_{ijj},\ \ i\neq j=k\\ 2!\,b_{iik},\ \ i=j\neq k\\ 1!\,b_{ijk},\ \ i\neq j\neq k\end{cases}=\beta_{ijk}b_{ijk}.\tag{90}$$

The formulas (89)–(90) prove the third equalities in (21)–(22). The last equality in (22) is proved similarly. Thus, the Theorem 1 is proved.

Appendix 2. Proof of Theorem 2

Since $\varepsilon=\sum\limits_{|\alpha|\le 3}a_\alpha\psi_\alpha(\xi)$, $\psi_\alpha=\prod\limits_{l=1}^{N}K_{\alpha_l}(\xi_l)$, then

$$\Big(\frac{\partial\varepsilon}{\partial\xi_i}\Big)^2=\sum_{|\alpha|\le 3}\sum_{|\beta|\le 3}a_\alpha a_\beta\frac{\partial\psi_\alpha}{\partial\xi_i}\frac{\partial\psi_\beta}{\partial\xi_i},$$

where

$$\frac{\partial\psi_\alpha}{\partial\xi_i}\frac{\partial\psi_\beta}{\partial\xi_i}=\prod_{l=1,l\neq i}^{N}K_{\alpha_l}(\xi_l)K_{\beta_l}(\xi_l)\alpha_i\beta_iK_{\alpha_i-1}(\xi_i)K_{\beta_i-1}(\xi_i).$$

Taking into account the properties of $K_{\alpha_l}(\xi_l)$, we have

$$E\left(\frac{\partial\psi_\alpha}{\partial\xi_i}\frac{\partial\psi_\beta}{\partial\xi_i}\right)=\prod_{l=1,l\neq i}^{N}E\Big(K_{\alpha_l}(\xi_l)K_{\beta_l}(\xi_l)\Big)\alpha_i\beta_iE\Big(K_{\alpha_i-1}(\xi_i)K_{\beta_i-1}(\xi_i)\Big)=$$

$$=\prod_{l=1,l\neq i}^{N}\alpha_l!\delta_{\alpha_l\beta_l}\cdot\alpha_i\beta_i(\alpha_i-1)!\delta_{\alpha_i-1,\beta_i-1}=\delta_{\alpha\beta}\prod_{l=1,l\neq i}^{N}\alpha_l!\alpha_i\cdot\alpha_i!=\delta_{\alpha\beta}\alpha_i\prod_{l=1}^{N}\alpha_l!.$$

Then

$$E\left[\left(\frac{\partial\varepsilon}{\partial\xi_i}\right)^2\right]=\sum_{|\alpha|\le 3}a_\alpha^2\alpha_i\prod_{l=1}^{N}\alpha_l!. \tag{91}$$

Using one-to-one correspondence between a_α and b, we get

$$\sum_{|\alpha|=1}a_\alpha^2\alpha_i\prod_{l=1}^{N}\alpha_l!=b_i^2, \tag{92}$$

$$\sum_{|\alpha|=2}a_\alpha^2\alpha_i\prod_{l=1}^{N}\alpha_l!=2\cdot 2!b_{ii}^2+\sum_{j=1}^{i-1}b_{ij}^2=\sum_{j=1}^{i}\widetilde{\beta}_{ij}b_{ij}^2, \tag{93}$$

$$\sum_{|\alpha|=3}a_\alpha^2\alpha_i\prod_{l=1}^{N}\alpha_l!=3\cdot 3!b_{iii}^2+2\cdot 2!\sum_{k=1}^{i-1}b_{iik}^2+2!\sum_{j=1}^{i-1}b_{ijj}^2+\sum_{j=1}^{i-1}\sum_{k=1}^{j-1}b_{ijk}^2= \tag{94}$$

$$=\sum_{j=1}^{i}\sum_{k=1}^{j}b_{ijk}^2\widetilde{\beta}_{ijk},$$

where

$$\widetilde{\beta}_{ij}=\begin{cases}2\cdot 2!, & i=j\\ 1, & i\ne j\end{cases};\qquad \widetilde{\beta}_{ijk}=\begin{cases}3\cdot 3!, & i=j=k\\ 2\cdot 2!, & i=j\ne k\\ 2!, & j=k\ne i\\ 1, & i\ne j\ne k\end{cases}.$$

Summarizing (92)–(94) and substituting the result in (91), we obtain the assertion of Theorem 2.

Appendix 3. Sobol indices

By definition [7],

$$S_i=\frac{1}{D}\sum_{\alpha\in\mathcal{I}_i}a_\alpha^2E[\psi_\alpha^2], \tag{95}$$

where $D=\sum_{|\alpha|\le 3}a_\alpha^2E[\psi_\alpha^2]$ is the total variance, and

$$\mathcal{I}_i=\left\{\alpha=(\alpha_1,\dots,\alpha_N):\alpha_i\ne 0,\ \alpha_j=0,\ j=\overline{1,N},\ j\ne i\right\}.$$

For $|\alpha|=1,\ \alpha\in\mathcal{I}_i$ we have $\alpha_l=\delta_{il},\ l=\overline{1,N},\ \psi_\alpha=H_1(\xi_i)$ and according to (74), $a_\alpha=b_i$. Then

$$a_\alpha^2E[\psi_\alpha^2]=b_i^2E\left(H_1^2(\xi_i)\right)=1!b_i^2. \tag{96}$$

For $|\alpha|=2,\ \alpha\in\mathcal{I}_i$ we have $\alpha_l=2\delta_{il},\ l=1,\overline{1,N},\ \psi_\alpha=K_2(\xi_i)$ and according to (75), $a_\alpha=b_{ii}$. Then

$$a_\alpha^2E[\psi_\alpha^2]=b_{ii}^2E\left(K_2^2(\xi_i)\right)=2!b_{ii}^2. \tag{97}$$

For $|\alpha|=3,\ \alpha\in\mathcal{I}_i$ we have $\alpha_l=3\delta_{il},\ l=\overline{1,N},\ \psi_\alpha=K_3(\xi_i)$ and according to (77), $a_\alpha=b_{iii}$, then

$$a_\alpha^2E[\psi_\alpha^2]=b_{iii}^2E\left(K_3^2(\xi_i)\right)=3!b_{iii}^2. \tag{98}$$

Summarizing (96)–(98) and substituting in (95), we get

$$S_i=\frac{1}{D}(\beta_ib_i^2+\beta_{ii}b_{ii}^2+\beta_{iii}b_{iii}^2). \tag{99}$$

By definition [7],

$$S_i^T=\frac{1}{D}\sum_{\alpha\in\mathcal{I}_i^+}a_\alpha^2E[\psi_\alpha^2], \tag{100}$$

where

$$\mathcal{I}_i^+ = \{\alpha = (\alpha_1, \ldots, \alpha_N) : \ 0 \le |\alpha| \le 3, \ \alpha_i \ne 0\}.$$

For $|\alpha| = 1, \ \alpha \in \mathcal{I}_i^+$ we have $\alpha = (\alpha_1, \ldots, \alpha_N), \ \alpha_l = \delta_{il}$ and according to (74), $a_\alpha = b_i$, then

$$\sum_{\alpha \in \mathcal{I}_i^+, \ |\alpha|=1} a_\alpha^2 E[\psi_\alpha^2] = b_i^2 E\left(H_1^2(\xi_i)\right) = 1! b_i^2. \tag{101}$$

For $|\alpha| = 2, \ \alpha \in \mathcal{I}_i^+$ we have $\alpha = (\alpha_1, \ldots, \alpha_N)$, where α_l is given by (76), $\psi_\alpha = H_2(\xi_i, \xi_j)$, then $a_\alpha = b_{ij}$ and

$$\sum_{\alpha \in \mathcal{I}_i^+, \ |\alpha|=2} a_\alpha^2 E[\psi_\alpha^2] = \sum_{j=1}^{i} b_{ij}^2 E\left(H_2^2(\xi_i, \xi_j)\right) = 2! b_{ii}^2 + \sum_{j=1}^{i-1} b_{ij}^2 = \sum_{j=1}^{i} \beta_{ij} b_{ij}^2. \tag{102}$$

For $|\alpha| = 3, \ \alpha \in \mathcal{I}_i^+$ we have $\alpha = (\alpha_1, \ldots, \alpha_N)$, where α_l is given by (78), $\psi_\alpha = H_3(\xi_i, \xi_j, \xi_k)$. Then $a_\alpha = b_{ijk}$ and

$$\sum_{\alpha \in \mathcal{I}_i^+, \ |\alpha|=3} a_\alpha^2 E[\psi_\alpha^2] = \sum_{j=1}^{i} \sum_{k=1}^{j} b_{ijk}^2 E\left(H_3^2(\xi_i, \xi_j, \xi_k)\right) = \sum_{j=1}^{i} \sum_{k=1}^{j} \beta_{ijk} b_{ijk}^2. \tag{103}$$

Summarizing (101)–(103) and substituting the result in (100), we get

$$S_i^T = \frac{1}{D}\left(\beta_i b_i^2 + \sum_{j=1}^{i} \beta_{ij} b_{ij}^2 + \sum_{j=1}^{i} \sum_{k=1}^{j} \beta_{ijk} b_{ijk}^2\right).$$

References

[1] S. Abraham, M. Raisee, G. Ghorbaniasl, F. Contino, C. Lacor, A robust and efficient stepwise regression method for building sparse polynomial chaos expansions, J. Comput. Phys. 332 (2017) 461–474.
[2] A.K. Alekseev, I.M. Navon, M.E. Zelentsov, The estimation of functional uncertainty using polynomial chaos and adjoint equations, Int. J. Numer. Methods Fluids 67 (3) (2011) 328–341.
[3] A. Alexanderian, J. Winokur, I. Sraj, A. Srinivasan, M. Iskandarani, W.C. Thacker, O.M. Knio, Global sensitivity analysis in an ocean general circulation model: a sparse spectral projection approach, Comput. Geosci. 16 (3) (2012) 757–778.
[4] J. An, A. Owen, Quasi-regression, J. Complex. 17 (4) (2001) 588–607.
[5] M. Berveiller, B. Sudret, M. Lemaire, Stochastic finite element: a non intrusive approach by regression, Eur. J. Comput. Mech. 15 (2006) 81–92.
[6] G. Blatman, B. Sudret, Efficient computation of global sensitivity indices using sparse polynomial chaos expansion, Reliab. Eng. Syst. Saf. 95 (2010) 1216–1229.
[7] G. Blatman, B. Sudret, Adaptive sparse polynomial chaos expansion based on least angle regression, J. Comput. Phys. 230 (2011) 2345–2367.
[8] D.G. Cacuci, M. Ionescu-Bujor, I.M. Navon, Sensitivity and Uncertainty Analysis, Volume II. Applications to Large-Scale Systems, CRC Press, 2005, 368 pp.
[9] R.E. Caflisch, Monte Carlo and quasi-Monte Carlo methods, Acta Numer. 7 (1998) 1–49.
[10] R.H. Cameron, W.T. Martin, The orthogonal development of non-linear functionals in series of Fourier–Hermite functionals, Ann. Math. 48 (2) (1947) 385–392.
[11] H. Cheng, A. Sandu, Collocation least-squares polynomial chaos method, in: Proceedings of the 2010 Spring Simulation Multiconference, 11–15 April 2010, Orlando, FL, USA, 2010, pp. 1–6.
[12] A.M. Clayton, A.C. Lorenc, D.M. Barker, Operational implementation of a hybrid ensemble/4D-Var global data assimilation system at the Met Office, Q. J. R. Meteorol. Soc. 139 (2013) 1445–1461.
[13] P. Courtier, J.-N. Thepaut, A. Hollingsworth, A strategy for operational implementation of 4D-Var, using an incremental approach, Q. J. R. Meteorol. Soc. 120 (1994) 1367–1387.
[14] M.S. Eldred, J. Burkardt, Comparison of Non-intrusive Polynomial Chaos and Stochastic Collocation Methods for Uncertainty Quantification, AIAA Paper 0976, 2009, pp. 1–20.
[15] I.Yu. Gejadze, P.-O. Malaterre, Design of the control set in the framework of variational data assimilation, J. Comput. Phys. 325 (2016) 358–379.
[16] G.J.M. Copeland, I.Yu. Gejadze, F.-X. Le Dimet, V.P. Shutyaev, Computation of the analysis error covariance in variational data assimilation problems with nonlinear dynamics, J. Comput. Phys. 230 (22) (2011) 7923–7943.
[17] D.M. Ghiocel, R.G. Ghanem, Stochastic finite element analysis of seismic soil-structure interaction, J. Eng. Mech. 128 (2002) 66–77.
[18] O.P. Le Maitre, M. Reagan, H.N. Najm, R.G. Ghanem, O.M. Knio, A stochastic projection method for fluid flow – II. Random processes, J. Comput. Phys. 181 (2002) 9–44.
[19] P.C. Hansen, D.P. O'Leary, The use of the L-curve in the regularization of discrete ill-posed problems, SIAM J. Sci. Comput. 14 (1993) 1487–1503.
[20] L. Hascoët, V. Pascual, TAPENADE 2.1 User's Guide, INRIA Technical Report, No. 0300 2004, 78 pp.
[21] S. Hosder, R.W. Walters, R. Perez, A non-intrusive polynomial chaos method for uncertainty propagation in CFD simulations, in: AIAA paper 2007-1939 (2007) 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, 23–26 April 2007, Honolulu, HI, 2007, pp. 23–26.
[22] A.G. Kalmikov, P. Heimbach, A Hessian-based method for uncertainty quantification in global ocean state estimation, SIAM J. Sci. Comput. 36 (5) (2014) S267–S295.
[23] M.G. Kendall, A. Stuart, The Advanced Theory of Statistics, Volume 1: Distribution Theory, 3rd edition, Griffin, 1969.
[24] S. Kucherenko, S.F. Song, Derivative-based global sensitivity measures and their link with Sobol' sensitivity indices, in: R. Cools, D. Nuyens (Eds.), 11th International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing and Statistics, 6–11 April 2014.
[25] F.-X. Le Dimet, O. Talagrand, Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects, Tellus 38A (1986) 97–110.
[26] C. Lacor, É. Savin, General introduction to polynomial chaos and collocation methods, in: Uncertainty Management for Robust Industrial Design in Aeronautics, in: Notes on Numerical Fluid Mechanics and Multidisciplinary Design, vol. 140, Springer, 2019, pp. 109–122.

[27] M. Leutbecher, T.N. Palmer, Ensemble forecasting, J. Comput. Phys. 27 (2008) 3515–3539.
[28] E. Liberge, A. Hamdouni, Reduced order modelling method via proper orthogonal decomposition (POD) for flow around an oscillating cylinder, J. Fluids Struct. 26 (2010) 292–311.
[29] J.L. Lions, Contrôle optimal des systèmes gouvernés par des équations aux dérivées partielles, Dunod, Paris, 1968.
[30] H.N. Najm, Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics, Annu. Rev. Fluid Mech. 41 (2009) 35–52.
[31] S.V. Patankar, Numerical Heat Transfer and Fluid Flow, Hemisphere Publishing Corporation, New York, 1980.
[32] J. Peng, J. Hampton, A. Doostan, A weighted l_1-minimization approach for sparse polynomial chaos expansion, J. Comput. Phys. 230 (2011) 3015–3034.
[33] J. Peng, J. Hampton, A. Doostan, On polynomial chaos expansion via gradient-enhanced l_1-minimization, J. Comput. Phys. 310 (2016) 440–458.
[34] O. Roderick, M. Anitescu, P. Fischer, Polynomial regression approaches using derivative information for uncertainty quantification, Nucl. Sci. Eng. 162 (2) (2010) 122–139.
[35] C.W. Rowley, I. Mezic, S. Bagheri, P. Schlatter, D.S. Henningson, Spectral analysis of nonlinear flows, J. Fluid Mech. 641 (2009) 115–127.
[36] O. Roustant, J. Fruth, B. Ioos, S. Kuhnt, Crossed-derivative based sensitivity measures for interaction screening, Math. Comput. Simul. 105 (2014) 105–118.
[37] A. Saltelli, Making best use of model evaluations to compute sensitivity indices, Comput. Phys. Commun. 145 (2002) 280–297.
[38] J.N. Shadid, T.M. Smith, E.C. Cyrc, T.M. Wildey, R.P. Pawlowski, Stabilized FE simulation of prototype thermal-hydraulics problems with integrated adjoint-based capabilities, J. Comput. Phys. 321 (2016) 321–341.
[39] P.J. Schmid, Dynamic mode decomposition of numerical and experimental data, J. Fluid Mech. 656 (1) (2010) 5–28.
[40] S. Smolyak, Quadrature and interpolation formulas for tensor products of certain classes of functions, Sov. Math. Dokl. 4 (1963) 240–243.
[41] I.M. Sobol', Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates, Math. Comput. Simul. 55 (2001) 271–280.
[42] I.M. Sobol', S. Kucherenko, Derivative based global sensitivity measures and their link with global sensitivity indices, Math. Comput. Simul. 79 (10) (2009) 3009–3017.
[43] I.M. Sobol', B.V. Shukman, Random and quasirandom sequences: numerical estimates of uniformity of distribution, Math. Comput. Model. 18 (8) (1993) 39–45.
[44] B. Sudret, C.V. Mai, Computing derivative-based global sensitivity measures using polynomial chaos expansions, Reliab. Eng. Syst. Saf. 134 (2015) 241–250.
[45] B. Sudret, Global sensitivity analysis using polynomial chaos expansions, Reliab. Eng. Syst. Saf. 93 (2008) 964–979.
[46] S.T. Tokdar, R.E. Kass, Importance sampling: a review, WIREs: Comput. Stat. 2 (2010) 54–60.
[47] Z. Wang, I.M. Navon, F.-X. Le Dimet, X. Zou, The second order adjoint analysis: theory and applications, Meteorol. Atmos. Phys. 50 (1992) 3–20.
[48] D. Xiao, F. Fang, C.C. Pain, I.M. Navon, A parameterized non-intrusive reduced order model and error analysis for general time-dependent nonlinear partial differential equations and its applications, Comput. Methods Appl. Mech. Eng. 317 (2017) 868–889.
[49] D. Xiu, G.E. Karniadakis, The Wiener–Askey polynomial chaos for stochastic differential equations, SIAM J. Sci. Comput. 24 (2002) 619–644.