A comparison between Sobol's indices and Shapley's effect for global sensitivity analysis of systems with independent input variables

Bruno Vuillod [a,b], Marco Montemurro [a,*], Enrico Panettieri [a], Ludovic Hallo [b]

[a] *Arts et Métiers Institute of Technology, Université de Bordeaux, CNRS, INRA, Bordeaux INP, HESAM Université, I2M UMR 5295, F-33405 Talence, France*
[b] *French Atomic Energy Commission, Route des Gargails, RP 2, Le Barp Cedex, France*

ABSTRACT

The model-based system engineering approach consists of assembling subsystems together to model a complete system. In this context, some functional blocks can have a considerable influence on the overall behaviour of the system. A preliminary identification of the influence of the subsystems on the output responses can help reducing the complexity of the overall system, with a negligible impact on the overall accuracy. Therefore, pertinent indicators must be introduced to achieve this goal. To this purpose, in this work, some well-established methods and algorithms for global sensitivity analysis (GSA) of linear and non-linear systems with independent input variables, i.e., approaches based on Sobol's indices (different algorithms are considered), and Shapley's effect, are compared on both benchmark functions and real-world engineering problems.

Specifically, in this paper, real-world engineering problems dealing with linear and non-linear systems are modelled through commercial finite element software and/or dedicated programming languages for solving complex non-linear dynamics models, like Modelica. Regarding Modelica models, an efficient strategy based on functional mock-up units is presented to speed up the simulation of highly non-linear dynamic systems. All numerical models are interfaced with the algorithms used for GSA through ad-hoc routines coded in Python environment. For each problem, a systematic comparison between the results provided by the different algorithms making use of Sobol's indices and Shapley's indices is performed, in terms of reliability, accuracy and computational costs.

1. Introduction

In the last decade, the topic of the sensitivity analysis experienced a renewed interest in both academy and industry. Indeed, a sensitivity analysis allows determining the parameters influencing the most the behaviour of a system or, alternatively, the ones having the least influence on it. Thanks to the sensitivity analysis, it is possible to simplify models and to dedicate computing resources only where (and when) it is necessary. Of course, for each mathematical model describing some physical phenomena, it is possible to identify a suitable sensitivity analysis method: this concept has been extensively discussed in a recent and interesting review article on this topic by Razavi et al. [1].

Initially, the sensitivity analysis focused on the study of a *nominal point sensitivity* in a model output space: this approach is often referred to as Local Sensitivity Analysis (LSA). However, LSA is neither suitable nor efficient for characterising the global behaviour of a system. To this purpose, the concept of Global Sensitivity Analysis (GSA) was introduced by Saltelli et al. [2], even if some methods were already available [3]. Generally speaking, GSA approaches can

be classified in four families [1]: (a) the *derivative-based* approaches, which are multi-local techniques firstly introduced by Morris [4]; (b) the *distribution-based* approaches, which consist in studying conditional model output variances, based on the Hoeffding decomposition [5] and introduced by Sobol [3] in 1993 (the so-called Sobol's indices); (c) the *variogram-based* approaches, linking *derivative-based* and *distribution-based* approaches [6,7]; (d) the *regression-based* approaches [8].

In the literature, the GSA method used as a reference is, very often, the distribution-based GSA approach making use of Sobol's indices [9–11]. It is noteworthy that the vast majority of real-world problems is characterised by dependent input variables, thus GSA approaches based on Sobol's indices in the case of independent input variables [12–14] must be properly modified. To overcome this issue, in the last decade, some distribution-based GSA approaches devoted to systems characterised by dependent input variables have been developed. For instance, Chastaing et al. [15,16] proposed a GSA approach, revisiting the works by Hooker [17] and Stone [18], which is based on classical Sobol's indices coupled with the covariance of two correlated variables.

---

* Corresponding author.

E-mail addresses: marco.montemurro@ensam.eu, marco.montemurro@u-bordeaux.fr (M. Montemurro).

Acronyms

ANOVA	ANalysis Of VAriance
APDL.	Ansys Parametric Design Language
CAD	Computer Aided Design
CI	Confidence Interval
DAE	Differential Algebraic Equation
FAST	Fast Amplitude Sensitivity Test
FE	Finite Element
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Units
GSA	Global Sensitivity Analysis
IPDV	Independent Pairs of Dependent Variables
LSA	Local Sensitivity Analysis
MC	Monte Carlo
MIMO	Multiple-Input-Multiple-Output
MISO	Multiple-Input-Single-Output
MSL	Modelica Standard Library
ODE	Ordinary Differential Equation
PDF	Probability Density Function
QMC	Quasi Monte Carlo
RBD	Random Balance Designs
XML	Extensible Mark-up Language

Nevertheless, this aspect constitutes also the limit of this method: it can be applied solely to problems characterised by an Independent Pairs of Dependent Variables (IPDV). To go beyond this restriction Broto et al. [19] presented a generalisation of this method.

Another strategy allowing the integration of dependent input variables in GSA is the Shapley effect [20]. The concept of Shapley effect comes from the game theory: by considering a team playing to a game to win a certain gain, the aim is to quantify the role of the single player of the team to win this gain. The mathematical theory behind the Shapley effect, in the context of GSA approaches, is presented in [14, 21–23]. Particularly, in [14], a comparison between the generalised Sobol's indices and the Shapley effect is discussed. According to the results presented in [14], the latter approach is more efficient than the former one, at least for the problems discussed in [14]. The GSA method based on Sobol's indices and the one based on the Shapley effect have also been compared in terms of complexity in [22]. It is noteworthy that in the literature one can find some works dealing with the problem of speeding up the algorithms based on the Shapley effect [24].

Further efficient approaches are available in the literature to perform the GSA of non-linear systems [25–30], but are not yet widespread enough to be available in the classical Python libraries (SALib, Open-Turn, Emukit) or MATLAB® environment (Global Sensitivity Analysis Toolbox). For example, in [31,32], the generalised Morris method is presented, possibly by considering uncertainty and correlated inputs, whilst in [33], the notion of Sobol tensor trains for GSA is introduced. In [34], low-rank tensor approximation is used to carry out the GSA, while in [35], a new moment-independent sensitivity index is developed for quantifying the effect of each input variable on the outputs in the case of systems characterised by dependent input variables. Among the most recent research works on GSA making use of alternative approaches and formulations, it is noteworthy to mention the methodology proposed in [11], based on the theory of active subspaces and Kriging surrogate metamodeling, and the methods based on the random forests algorithms discussed in [36]. Finally, all the methods mentioned above can be applied to both to Multiple-Inputs Multiple-Outputs (MIMO) [12,35] and Multiple-Input Single-Output (MISO) systems.

As one can infer from the above non-exhaustive literature survey, the comparison, in terms of both accuracy and reliability of results as well as computational costs, on the approaches based on Sobol's indices and Shapley's effect is not performed in a systematic way by considering both analytical benchmark problems and highly non-linear systems (including dynamical models) related to real-world engineering applications. To this end, in this paper, a comparison between different algorithms for GSA available in Python libraries based on Sobol's indices (different variants are considered) and the algorithm based on Shapley's effect proposed by Goda [37] (adapted to the Python environment in this study), is systematically performed on both analytical benchmark functions and engineering applications of industrial complexity. Only MISO systems with independent input variables are considered for the sake of simplicity. Specifically, the Sobol's indices will be computed through Saltelli's algorithms [38,39] and Fourier Amplitude Sensitivity Test (FAST) method [40].

As far as the real-world engineering applications are concerned, the related mathematical models are developed either in dedicated languages to solve highly non-linear dynamic systems (like Modelica) or coded in commercial Finite Element (FE) software (like ANSYS®). Regarding non-linear dynamic systems developed in Modelica environment, an efficient strategy that allows converting the Modelica model into Functional Mock-Up Units (FMUs) to reduce the computational costs is proposed. The interest of using FMUs relies on fast multiphysics simulations, with the possibility to easily change the sets of model parameters. Therefore, a statistical analysis (which requires a huge amount of simulation runs) can be carried out by modifying the model parameters through the use of FMUs in an efficient way.

The paper is organised as follows. In Section 2, the fundamental of Sobol's and Shapley's indices are recalled. Section 3 presents the comparison between the different algorithms to compute both Sobol's indices and Shapley's indices. Section 4 focuses on the GSA of highly non-linear dynamic models built in Modelica environment and converted into FMU format to reduce computational costs, while Section 5 describes the GSA performed on FE models. Lastly, Section 6 ends the paper with meaningful conclusions and prospects.

Notation. Upper-case bold letters and symbols are used to indicate matrices, while lower-case bold letters and symbols indicate column vectors. $\sharp S$ denotes the cardinality of the generic set S.

2. Global sensitivity analysis

A sensitivity analysis allows determining the input variables influencing the most the system outputs, in a qualitative or in a quantitative fashion, depending on the considered method. In the literature one can find several methodologies to carry out both LSA and GSA. This section is devoted solely to the methods for GSA used in the following sections, namely the Sobol's indices computed with the Saltelli's algorithm in the two variants proposed in 2002 in [38] and in 2010 in [39] and the FAST algorithm [41,42], as well as the Shapley's indices computed with the algorithm proposed by Goda [37], which has been adapted and optimised to the Python environment in this paper.

2.1. Sobol's indices and analysis of variance

Consider a MIMO system whose transfer function is $\mathcal{M} : \mathcal{A} \subseteq \mathbb{R}^n \longrightarrow \mathcal{B} \subseteq \mathbb{R}^m$, where n is the number of input variables, $\mathcal{A}$ is the definition domain (where input variables take values), m is the number of output responses of the system and $\mathcal{B}$ is the co-domain (where the system outputs take values). $\zeta^\mathrm{T} = (\zeta_1, \ldots, \zeta_n)$, $\zeta \in \mathcal{A}$ represents the vector of inputs, whilst $\mathbf{y}^\mathrm{T} = (y_1, \ldots, y_m)$, $\mathbf{y} \in \mathcal{B}$ represents the vector of outputs.

The variability of the inputs is modelled via random variables characterised by their Probability Density Functions (PDFs): $\zeta_i \sim d P_{\zeta_i}$, $i = 1, \ldots, n$ (which must be read: ζ_i follows a distribution of PDF $d P_{\zeta_i}$). The stochastic version of the MIMO system reads:

$$\mathbf{y} = \mathcal{M}(\zeta), \quad \zeta \in \mathcal{A} \subseteq \mathbb{R}^n, \quad \mathbf{y} \in \mathcal{B} \subseteq \mathbb{R}^m. \tag{1}$$

In the following of this work, only independent input variables are considered, i.e., the generic input variable cannot be expressed as a function of the remaining inputs (neither explicitly nor implicitly). Formally, this means that the PDF of ζ reads:

$$dP_\zeta = \prod_{i=1}^{n} dP_{\zeta_i}. \tag{2}$$

It is noteworthy that in this work, the model inputs are distributed according to a uniform distribution $\mathcal{U}$ in the definition domain $\mathcal{A}$. In this work, it is tacitly assumed that the results of the GSA, when considering a uniform distribution of the input variables (both continuous and discrete), are not influenced by the discretisation step. This means that the input variables considered in this paper are always independent, regardless of their nature, i.e., discrete (with a step defined by the user) or continuous (which are necessarily discretised when considering a uniform distribution with a discretisation step related to the precision of the machine). Moreover, for the sake of simplicity, in this section, the discussion is limited to the case of MISO systems, i.e., $m = 1$. Nevertheless, the proposed formulation can be easily extended to the most general case of MIMO systems.

Consider the generic subset $S \subseteq \{1, \dots, n\}$, with $S \neq \varnothing$, having cardinality $\sharp S = n_s \leq n$. If the scalar output function satisfies some basic hypotheses [43], the Hoeffding decomposition [5] of the output function y reads:

$$y = \mathcal{M}_0 + \sum_{i=1}^{n} \mathcal{M}_i(\zeta_i) + \sum_{1 \leq i < j \leq n} \mathcal{M}_{i,j}(\zeta_i, \zeta_j) + \cdots + \mathcal{M}_{1,\dots,n}(\zeta)$$
$$= \mathcal{M}_0 + \sum_{\substack{u \in S \\ u \neq 0}} \mathcal{M}_u(\zeta_u). \tag{3}$$

where ζ_u is a sub-array of ζ whose indices are defined through the components of the vector u whose cardinality is $\sharp u = n_u \leq n_s$ and whose components belong to the subset S, i.e., $u_k \in S$, $k = 1, \dots, n_u$ [34]. The uniqueness of the decomposition is ensured through the following conditions:

$$\mathcal{M}_0 = \mathbb{E}(y), \tag{4}$$

and

$$\mathbb{E}\left[\mathcal{M}_u(\zeta_u)\mathcal{M}_v(\zeta_v)\right] = 0, \ u_k, v_k \in S, \ u \neq v. \tag{5}$$

In Eqs. (4) and (5), the symbol $\mathbb{E}$ denotes the so-called *expected value*. Therefore, in Eq. (4), $\mathcal{M}_0$ is the mean value. It is noteworthy that the above formulæ imply that all terms $\mathcal{M}_u(\zeta_u)$ ($u \neq 0$) in Eq. (3) have zero mean values. The terms $\mathcal{M}_u(\zeta_u)$ can be obtained in a recursive way as follows:

$$\begin{cases} \mathcal{M}_i &= \mathbb{E}(y|\zeta_i) - \mathcal{M}_0, \\ \mathcal{M}_{i,j} &= \mathbb{E}(y|\zeta_i, \zeta_j) - \mathcal{M}_0 - \mathcal{M}_i - \mathcal{M}_j, \\ \dots & \end{cases} \tag{6}$$

The terms of increasing order in Eq. (6) are conditional expectations defined in a recursive way, constituting, thus, an orthogonal (and unique) decomposition of the output of the system [5,34].

The goal of the GSA is to determine the relative influence of each parameter ζ_i on the considered output y, in terms of variance. Following the same idea at the basis of Eq. (3), it is possible to introduce the ANalysis Of VAriance (ANOVA) decomposition [34,43,44] as:

$$\text{Var}(y) = \sum_{\substack{u \in S \\ u \neq 0}} \text{Var}\left(\mathcal{M}_u(\zeta_u)\right). \tag{7}$$

In the above formula, $\text{Var}\left(\mathcal{M}_u(\zeta_u)\right)$ represents the conditional variance for the sub-array ζ_u whose indices, collected in the vector u, belongs to the subset S [34]. The generic Sobol's index S_u is defined as:

$$S_u := \frac{\text{Var}\left(\mathcal{M}_u(\zeta_u)\right)}{\text{Var}(y)}, \tag{8}$$

which represents the ratio of the variance due to the interaction between the components of ζ_u (for $u \in S$) to the total variance of the output. Of course, the Sobol's indices satisfy the following relationship:

$$\sum_{\substack{u \in S \\ u \neq 0}} S_u = \sum_{i=1}^{n} S_i(\zeta_i) + \sum_{1 \leq i < j \leq n} S_{i,j}(\zeta_i, \zeta_j) + \cdots + S_{1,\dots,n}(\zeta) = 1. \tag{9}$$

According to Eq. (8), the first-order Sobol's index, also referred to as elementary Sobol's index, for a single variable ζ_i is defined as:

$$S_i := \frac{\text{Var}\left(\mathcal{M}_i(\zeta_i)\right)}{\text{Var}(y)}. \tag{10}$$

The first-order Sobol's index provides a measure of the influence of the single input variable ζ_i on the output y. However, the elementary Sobol's index does not provide any information about the influence of the variable ζ_i on the output y when interacting with other input variables ζ_k, $k \in S$, $k \neq i$.

To get this information, one has to consider the Sobol's indices from order 2 to n that, according to Eq. (8), can be expressed as:

$$S_{i,j} := \frac{\text{Var}\left(\mathcal{M}_{i,j}(\zeta_i, \zeta_j)\right)}{\text{Var}(y)}, \ S_{i,j,k} := \frac{\text{Var}\left(\mathcal{M}_{i,j,k}(\zeta_i, \zeta_j, \zeta_k)\right)}{\text{Var}(y)}, \dots, \tag{11}$$

where $S_{i,j}$ is the second-order Sobol's index translating the influence of the couple (ζ_i, ζ_j) on the output y, while $S_{i,j,k}$ is the third-order Sobol's index that provide a measure on the influence of the first three input variables $(\zeta_i, \zeta_j, \zeta_k)$ on the output y, etc.

The $2^n - 1$ Sobol's indices can provide precious informations for the GSA, but their computation can be prohibitive when a large number of variables is considered. To this end, a measure often referred to as the "total-effect index" or "total-order index" or "total Sobol's index", S_{T_i}, is used [34,38]. This index provides a measure of the contribution to the output variance of ζ_i, including all variance caused by its interactions, of any order, with the other input variables. The total Sobol's index related to the input variable ζ_i can be defined as [34,39]:

$$S_{T_i} := \sum_{\substack{u \in S \\ i \cup u \neq 0}} S_u. \tag{12}$$

The total Sobol's index S_{T_i} represents the total effect of the generic input variable ζ_i, including its direct effect on the output as well as all interactions with other input variables. According to Eq. (12), it is equal to the sum of all partial indices S_u involving the input variable ζ_i. Of course, unlike the elementary indices S_i, the sum of the total Sobol's indices can be greater than or equal to one, i.e.,

$$\sum_{i=1}^{n} S_{T_i} \geq 1. \tag{13}$$

This is due to the fact that the interaction between two variables, e.g., ζ_i and ζ_j, is counted in both the associated total indices, i.e., S_{T_i} and S_{T_j}. The sum of the total indices is equal to one only when the model is purely additive. It is noteworthy that in [38], Saltelli presented a numerical integration scheme requiring $N(n + 2)$ simulations, where N is the total number of samples allowing the computation of the n elementary indices $(S_i)_{i \in [1,n]}$ together with the n total indices. According to Eq. (12), if $S_{T_i} \approx 0$, one can state that the variable ζ_i does not influence at all the considered output.

In the following of this paper (and, in a more general sense, in almost all real-world engineering problems) only the elementary and total Sobol's indices are considered. The elementary Sobol's indices are indicated as S_i, whilst the total Sobol's indices are indicates as S_{T_i}. For a deeper insight in the matter the interested reader is addressed to [14,15,19,34,44].

2.1.1. Saltelli's algorithms

Saltelli's algorithms presented in [38,39] for evaluating Sobol's indices require the same number of simulations, i.e., $N(n+2)$, and both provide elementary and total indices, i.e., S_i and S_{T_i}, respectively. They

essentially differ in the way they compute these indices. Specifically, in [39], the Quasi-Monte-Carlo (QMC) technique is used to generate the samples. For a deeper insight in the matter the interested reader is addressed to [38,39].

2.1.2. Fourier amplitude sensitivity test method

FAST method uses periodic sampling and Fourier transformation to decompose the variance of a model output into partial variances attached to the different model parameters. By relating these partial variances to the overall variance, it is possible to compute Sobol's indices.

It is noteworthy that the FAST method is a relatively old technique [40], but it has been improved in the last decade by combining the random balance designs (RBD) technique with the FAST algorithm (RBD-FAST) [24,45–47]. However, most of these enhancements are either unavailable in classic Python or MATLAB® toolboxes, or incomplete. For example, RBD-FAST method, integrated in SALib,[1] only computes first order indices, although this method has been extended to total indices [48]. The interested reader is addressed to [41,42] for more details on the theoretical background of the FAST method.

One of the advantages of this method over Saltelli's algorithms is that it requires fewer simulations to compute all the indices. Indeed, Nn simulations are necessary against $N(n+2)$ for Saltelli's algorithms. Nevertheless, some precautions must be adopted when using this method. Indeed, Tissot and Prieur [46] have shown that for models with more than ten input variables, the values of the indices were biased and the algorithm become unstable.

Unlike Saltelli's algorithms that are based on the ANOVA decomposition, the FAST method makes use of the frequency decomposition of the outputs of the system. This leads to small discrepancies between the values of the Sobol's indices and those resulting from the FAST method. Nevertheless, these differences are negligible and the information provided by the FAST algorithm can be used to quantify and classify the influences of the different input variables in terms of percentage.

2.2. Shapley's indices

Recently, Owen [22] has proposed a GSA method for systems characterised by dependent input variables based on Shapley's effect, a concept taken from game theory [49]. Nevertheless, since in this work only non-linear dynamic problems characterised by independent input variables are considered, only the formulation of the Shapley's indices for non-correlated inputs is briefly recalled here below.

According to [14,50], and using the notation introduced in the above subsection for Sobol's indices, in the case of independent input variables, the Shapley's index related to the generic input ζ_i is defined as:

$$S_{H_i} := \sum_{\substack{\mathbf{u} \in S \\ i \subset \mathbf{u} \neq \emptyset}} \frac{S_{\mathbf{u}}}{n_{\mathbf{u}}}, \qquad (14)$$

where $n_{\mathbf{u}}$ is the cardinality of the array $\mathbf{u}$ collecting the indices $u_k \in S$ used to compute the elementary index $S_{\mathbf{u}}$, i.e., $n_{\mathbf{u}} = 1$ if $\mathbf{u}^T = (1)$, $n_{\mathbf{u}} = 2$ if $\mathbf{u}^T = (1,2)$, etc. The two main properties and advantages of the Shapley's indices are that they are positive semidefinite and their sum is equal to the unit [14].

It is noteworthy that, according to Eq. (14), the Shapley's index related to the generic input variable ζ_i is greater than or equal to the corresponding elementary Sobol's index and it is lower than or equal to the corresponding total Sobol's index, i.e. $S_i \leq S_{H_i} \leq S_{T_i}$. Of course, this property is due to the mathematical definition of the Shapley's index, but follows also a clear rationale: the first-order Sobol's index quantifies the influence of a single input variable on the output, the total Sobol's index takes into account the influence of the single

input on the output together with the interactions with the other inputs variables belonging to $\zeta_{\mathbf{u}}$ (and the interactions are evaluated multiple times), while in the definition of the Shapley's index the interaction effect is equally distributed to each input variable involved in the interaction.

In the following of this paper the Shapley's indices are indicated as S_{H_i}. The Python script used to assess the Shapley's index is provided in Appendix.

An overview of Sobol's indices and Shapley's indices main features together with the algorithms used in this study is reported in Table 1.

3. An analytical benchmark problem: the ishigami function

The Ishigami function [51,52] is widely used as a benchmark function in GSA for its strongly non-linear and non-monotonic behaviour. Moreover, its dependence on the third variable is quite particular as described in [53]. It reads:

$$f(\mathbf{x}) = \sin(x_1) + a\sin^2(x_2) + bx_3^4 x_1, \qquad (15)$$

with $x_i \in \mathcal{U}[-\pi;\pi], i = 1,2,3$, $a = 7$ and $b = 0.1$. The values of parameters a and b has been taken from [54].

To calculate the Sobol's indices according to the Saltelli 2002 algorithm [38], the OpenTURN[2] library is used, whilst for Saltelli 2010 algorithm [39] and FAST algorithm [42] the SALib library is employed. Both libraries are available in Python. For the Shapley indices, the code from [37] adapted to Python language is used (see Appendix). These algorithms are compared according to the following criteria:

- The convergence rate of the relative error between the Sobol's index (elementary or total) and the Shapley's index calculated through the considered algorithms and a reference value calculated via the same algorithms when considering a large number of samples, i.e., $N = 2^{22}$. The relative error on each index is considered as converged when it is lower than or equal to 0.01 during 10^4 successive samples. Of course, the lower the number of samples to achieve convergence the more efficient is the algorithm.
- The accuracy of the algorithm in assessing the Sobol's and Shapley's indices, when convergence is achieved, by comparing the values provided by the considered algorithm to analytical values available in the literature [54,55].
- The confidence interval (CI) on each index with a confidence level equal to 95%, which provides a measure of the reliability of the algorithm in assessing the considered indices.

The convergence rate of the relative error on the elementary index and on the Shapley's index for input variable x_1 is illustrated in Fig. 1 (results related to variables x_2 and x_3 are characterised by the same trend but are not reported for the sake of brevity). As it can be inferred from this figure, the FAST method requires a number of samples N lower than those required by Saltelli's algorithms to achieve convergence. Conversely, the convergence of the Shapley index is slower, but as it provides complementary information to the Sobol's indices, this algorithm will still be used in the following of this document.

The accuracy of the algorithms in computing Sobol's and Shapley's indices (for a number of samples $N = 2^{15}$) when compared to analytical values taken from [54,55] is illustrated in the histograms of Fig. 2. In this figure, the confidence interval related to the result of each algorithm for a number of samples $N = 2^{15}$ is also reported, by considering a confidence level equal to 95%. From the analysis of these results, one can infer that the results provided by Saltelli 2010 algorithm [39] are the most accurate ones, although the most reliable ones are those provided by FAST method [40,42]. It is noteworthy that

---

[1] https://salib.readthedocs.io/en/latest/

[2] https://openturns.github.io/www/index.html

Table 1
Main features of Sobol's indices, Shapley's indices and of the related algorithms used in this study.

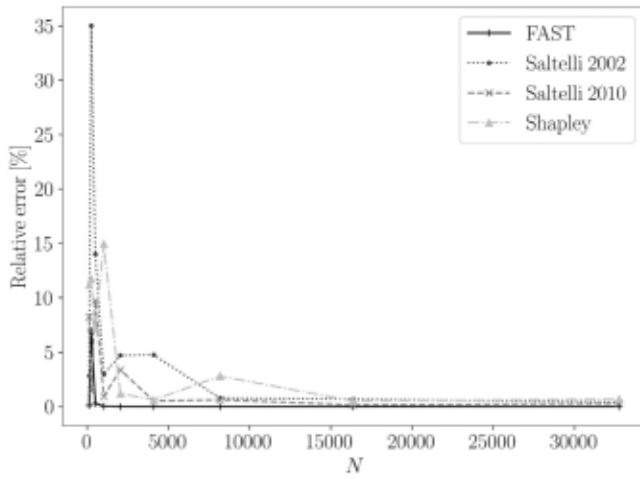|	Sobol		Shapley
	Elementary index	Total index	
Index	- Influence of one parameter - $0 \leq S_i \leq 1$	- Influence of one parameter and its interactions - Count the interactions several times - $\sum_{i=1}^{n} S_{T_i} \geq 1$	- Influence of a parameter and its contribution in each of its interactions - $\sum_{i=1}^{n} S_{H_i} = 1$ - $0 \leq S_{H_i} \leq 1$ - $S_i \leq S_{H_i} \leq S_{T_i}$
	Saltelli 2002 [38]	Saltelli 2010 [39]	FAST [40,42]	Goda 2021 [37]
Algorithm	- MC sampling - $N(n+2)$ simulations - $S_i, S_{i,j}, S_{T_i}$ - ANOVA decomposition	- QMC sampling according to the Sobol's sequence - $N(n+2)$ simulations - $S_i, S_{i,j}, S_{T_i}$ - ANOVA decomposition - Formulæ for determining indices in a more efficient way than in [38]	- QMC sampling according to the Sobol's sequence - Nn simulations - S_i, S_{T_i} - Frequency decomposition	- QMC sampling according to the Sobol's sequence - $N(n+1)$ simulations - S_{H_i}

Fig. 1. Relative error of the Sobol's elementary index and of the Shapley's index computed with the mentioned algorithms vs. the number of samples for variable x_1.
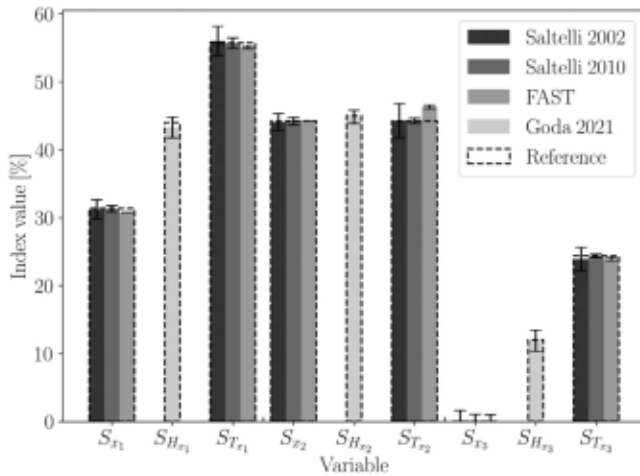

Fig. 2. Comparison between Sobol's elementary and total indices and the Shapley's indices with their analytical value [54,55] including the related confidence interval.

the FAST algorithm provides also a good level of accuracy, with the highest percentage error equal to 4.65% for the total index related to variable x_2. Moreover, the Shapley's index takes values between elementary and total index for each variable.

Therefore, for the other benchmark problems considered in this paper, the FAST algorithm is the one retained to assess the Sobol's indices, because it represents the best compromise between accuracy, reliability and computational costs (it requires only $N = 2^{12}$ samples to achieve convergence for the Ishigami function). Moreover, the Python version of the algorithm proposed by Goda [37] will be used to compute the Shapley's indices. Specifically, in the following, the FAST method will be used with a number of samples equal to 2^{12} and compared to the results provided by the Saltelli 2010 algorithm with a sample size of 2^{22} when no literature solutions are available. Regarding the convergence results, the Shapley's algorithm will be used with a greater sample size, i.e., with $N = 2^{14}$

4. Non-linear dynamics problems: the modelica programming language

4.1. Fundamentals of modelica language and the functional mock-up interface norm

Modelica[3] [56] is a freely accessible object-oriented formal calculation language for describing systems of equations. In physical modelling, it allows for an easy formal description of the dynamic behaviour of complex systems (particularly in the context of multi-field analyses), while reducing the issues associated with numerical resolution (e.g., strong non-linearity of the equations of the system) [57–61]. Modelica is an acausal object-oriented programming language. Therefore, to model a physical problem, it is sufficient to assemble physical subsystems (elementary blocks), each with an analytical description of its behaviour. When simulating the model, Modelica will assemble the different subsystems of equations into a global Differential Algebraic Equation (DAE) system before solving it by symbolic manipulation (acausal behaviour), translating it into C language, compiling it and executing it. More details about the Modelica language can be found in [56].

Among the interesting and useful features available in Modelica, this programming language inherited the ability to exchange data according to the Functional Mock-up Interface (FMI) standard and it is, thus, particularly efficient. Indeed, as a Modelica program is translated into C during simulation, it can easily be exported to the FMU format according to the FMI standard. This standard is not necessarily well-known and deserves a precise description. The development of the FMI standard was initiated in the framework of the European MODELISAR[4] project in 2008. The objective of this standard is to allow connections
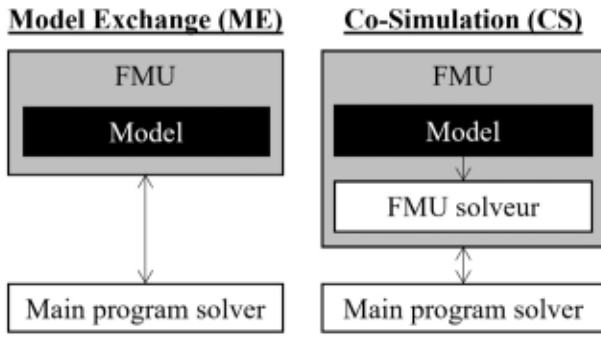

Fig. 3. Schematic representation of the model exchange and co-simulation operating modes.

Fig. 4. CAO model of the car shock absorber.

Fig. 5. Functional scheme of the car shock absorber.

between different simulation and modelling environments in order to benefit from their intrinsic functionalities within the same project without approximation. For example, a Modelica model can include models from Computer Aided Design (CAD) software such as CATIA®, which can be used to describe both the geometry of an object and its physical behaviour. It is interesting to note that there is an analogy between the exchange possibilities allowed by the FMI standard (design offices for 3D geometry and calculation offices for physical models) and the adaptation process in the German *V-and-V* model presented in [62,63] and applied in [64]. The latest usable version of this standard is presented in [65], and the last update was made in July 2021 [66].

From a practical point of view, the FMI standard defines the standalone FMU compressed file export format including: (a) the exported model code translated into C, (b) an Extensible Markup Language (XML) file describing the structure of the model at hand and used to interface the C file with the import program, (c) the resources needed to run the exported model such as point files, visualisation libraries, etc.

To be autonomous, an FMU file must be exported in *co-simulation* mode, i.e., it must include a solver defined by the user during the export procedure. Thus, when it is included in a more global model, it will be executed at the same time as the latter and will communicate with it via exchanges of scalar and flow variables according to a pre-defined time step. In the applications discussed in the following of this section, the GSA algorithm will call the FMU as many times as necessary via Python code. An FMU file can also be exported in the *model exchange* mode. In this second case, the FMU will be executed with the same solver as the global model and will be fully integrated. These export modes are illustrated in Fig. 3. Examples of the use of the FMI standard can be found in [67–71].

Due to the FMU format, even a complex model becomes cheap (in terms of computational costs) to be simulated with the possibility of efficient and non-intrusive coupling. This is interesting for statistical analyses requiring many simulations such as GSA.

4.2. The car shock absorber

4.2.1. Analytical description

The first mechanical system analysed in this work is a car shock absorber, illustrated in Fig. 4. Four main elements can be identified within the CAD model: the piston is a cylinder with a central hole, which allows for the oil flows during motion. This allows also absorbing the spring oscillations. The external piston radius is equal to the internal piston chamber radius, without gap. The passengers and the car masses are loaded on the whole assembly via the top mounting ring.

The scheme shown in Fig. 5 represents the CAD system in functional diagram form: a spring–damper parallel system fixed to a support structure represents the car shock absorber submitted to the weight of the car and of the passengers.
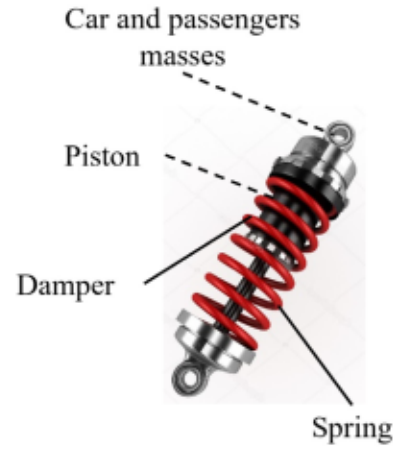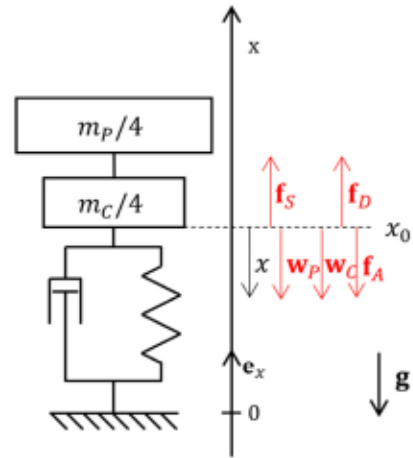
The dynamic response of such system is governed by the longitudinal displacement $x(t)$ of the top chamber piston. It can be determined by solving the equilibrium equation of the system:

$$M\mathbf{a} = \sum_{i=1}^{q} \mathbf{f}_{\text{ext},i}, \tag{16}$$

where $M = \frac{m_P}{4} + \frac{m_C}{4}$ is a quarter of the overall mass of the system (i.e., car + passengers masses), $\mathbf{a}$ is the acceleration, whose component along the x axis is equal to $\ddot{x}(t)$, $\mathbf{f}_{\text{ext},i}$ is the generic ith external force applied to the spring–damper (see Fig. 5) and q is the total number of applied forces.

The external forces applied to the system are listed below:

• Archimedes' thrust and spring force:

$$\mathbf{f}_A = -S_p h \rho_{\text{oil}} g \mathbf{e}_x, \quad \mathbf{f}_S = -k(x - L_0)\mathbf{e}_x, \tag{17}$$

where $S_p = \pi(r_{\text{ext}}^2 - r_{\text{int}}^2)$ is the cross-sectional area of the piston, r_{int} and r_{ext} are the inner radius and outer radius of the piston, respectively, h is the height of the piston, ρ_{oil} is the density of the oil, g is the gravitational acceleration and $\mathbf{e}_x$ is the unit vector of the x axis. Regarding the spring force, k is the spring constant and L_0 is its unstretched length. At $t = 0$ s, the spring is pre-stressed.

• Damper force:

$$\mathbf{f}_D = -\text{sign}(\dot{x})S_p \Delta P \mathbf{e}_x, \tag{18}$$

Table 2
Reference value of model parameters for the car shock absorber benchmark problem.

Parameter	Value
k [Nm^{-1}]	9000
L_0 [m]	0.45
r_{int} [m]	0.002
r_{ext} [m]	0.03 (constant)
m_P [kg]	150
h [m]	0.02
ρ_{oil}	884 (constant)
g [ms^{-2}]	9.81 (constant)
L_{bar} [m]	0.5 (constant)
L_{cham} [m]	0.25 (constant)
m_C [kg]	1000

where ΔP is the pressure drop, calculated considering a hollow piston sliding without clearance in its chamber, i.e.,

$$\Delta P = \rho_{oil} \frac{\omega_0^2}{2}(\zeta_{red} + \zeta_{exp}), \quad \omega_0 = \frac{\dot{x} S_p}{S_h}, \quad (19)$$

where $S_h = \pi r_{int}^2$ is the cross-sectional area of the piston hole. In addition, parameters ζ_{red} and ζ_{exp} are expressed as follows:

$$\zeta_{red} = \left(\frac{1}{C} - 1\right)^2, \zeta_{exp} = \left(1 - \frac{S_h}{S_p}\right)^2, \quad (20)$$

with $C = 0.63 + 0.37 \left(\frac{S_h}{S_p}\right)^2$.

- Action of the weight of the passengers and the weight of the car:

$$\mathbf{w}_P = -\frac{m_P}{4} g \mathbf{e}_x, \quad \mathbf{w}_C = -\frac{m_C}{4} g \mathbf{e}_x, \quad (21)$$

where m_P and m_C are the total mass of the passengers and of the car respectively.

4.2.2. Modelica model

The mechanical model of the car shock absorber is built with the Modelica [56] programming language via MapleSim [72] software and interfaced, though FMU format file, with the GSA algorithms.

The Modelica Standard Library (MSL) is used to generate the components of the mechanical system visible in Fig. 6. However, the mechanical behaviour of the damper was modelled via a custom force element to reproduce the constitutive law of Eq. (18). The element *mass with stop and friction* M_C was introduced to constrain the movement of the piston, limited by the chamber length. The car weight action is represented by the element w_C, while the passengers one by w_P. The whole system is described by means of eleven parameters of which five are set to predefined values.

When the car is loaded, the role of the damper is to limit the oscillations of the spring and to dissipate them in a short time at a given position. Consequently, the output considered for the sensitivity analysis is the position of the piston at a given time, i.e., $x(t)$. As a reference, the time constant for shock absorption has been set to $\tau_{abs} = 2$ s. The stable position of the piston is $x = 0.64$ m. The parameters of the reference motion are given in Table 2 and correspond, approximately, to a physically admissible situation. Note that the initial position of the piston is defined as $x(0) = L_{bar} + L_{cham}$, where L_{bar} is the bar length in contact with the ground (assimilated to the wheel) and L_{cham} is the length of the chamber of the piston, which limits its displacement. At $t = 0$ s, the spring is preloaded: $x(0) < L_0 + L_{bar}$.

Once the Modelica model is created and validated, the FMU file is generated in co-simulation mode. During the GSA, the FMU is considered as a black-box whose inputs variables are sampled according to a uniform law $\mathcal{U}$ in their respective intervals, as reported in Table 3. These intervals have been chosen according to the admissible mechanical conditions of the shock absorbers. Particularly, the interval
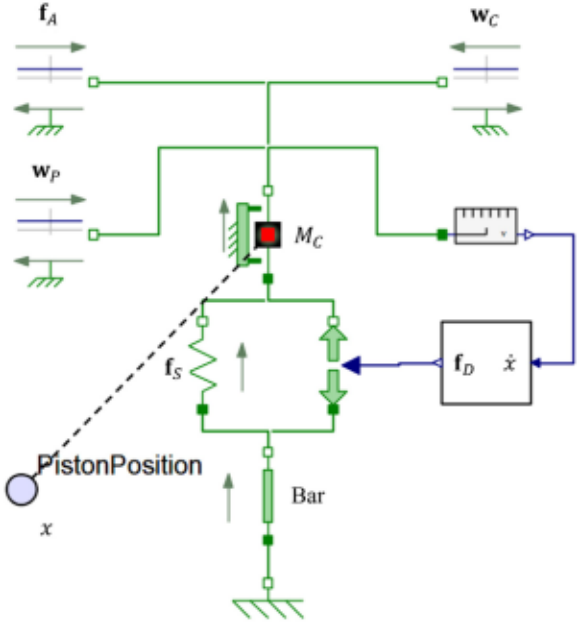

Fig. 6. Modelica model of the car shock absorber (notations similar to the model equations).

of definition of the parameter r_{int} is chosen to ensure a complete or semi-periodic absorbing behaviour.

When generating the FMU file in co-simulation mode, it is necessary to define the inputs, outputs and modifiable parameters of the model, but also the solver type integrated in the file. In this situation, a Runge–Kutta fourth-order algorithm was chosen, whose integration and communication steps with the global program are set to 10^{-5} s. Considering these characteristics, the simulation time of the reference model of the damper is reduced of factor approximately equal to 100 by switching from the initial MapleSim software to a call of the FMU file (C code) by Python.

Remark 4.1. Before proceeding to the GSA, the studied model can be simplified. When evaluating the orders of magnitude of the external forces, the Archimedes' force can be neglected in comparison to the spring force because the former is about three orders of magnitude smaller than the latter. Consequently, the parameter h can be removed from the input variables considered in the GSA.

4.2.3. Numerical results

Considering the interval of variation of each input variable, the motion of the piston can be of three types: motionless at its initial position (in this case the pre-load due to the spring is dominant), completely damped or semi-periodic. The output response being the final position of the piston, a GSA has been carried out by observing $x(t)$ for ten characteristic times, i.e., $t = i\tau_{abs}$, with $i \in [1, \dots, 10]$ and $\tau_{abs} = 2$ s, corresponding to an intermediate value of time for the motion when $i = 1$, and to a sufficient high time to reach the static position when $i = 10$.

The results of the GSA considering the FAST method to assess the Sobol's indices and the Python version of the algorithm proposed by Goda [37] to assess the Shapley's indices are shown in Fig. 7 for different values of t in the interval $[\tau_{abs}, 10\tau_{abs}]$. Moreover, when $t = 10\tau_{abs}$ the indices are compared to the reference results obtained with the Saltelli 2010 algorithm with $N = 2^{22}$ samples

As one can infer from Fig. 7, the Sobol's total indices of variables r_{int} and m_P are lower than the counterparts related to other input variables at $t = 10\tau_{abs}$. Nevertheless, the influence of variable r_{int} decreases with the time and goes to zero when $t > 3\tau_{abs}$, while that of m_P increases
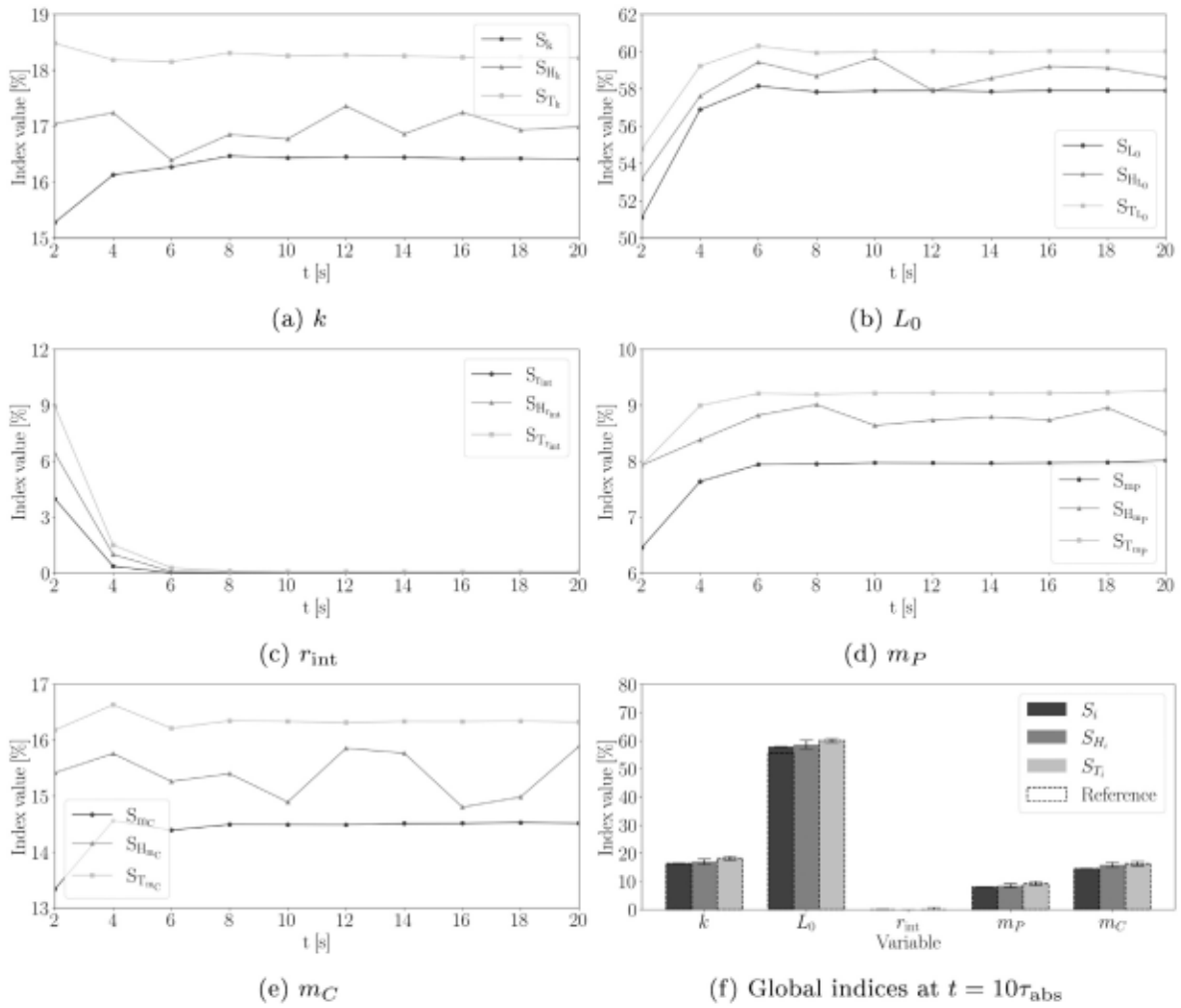
(a) k

(b) L_0

(c) r_{int}

(d) m_P

(e) m_C

(f) Global indices at $t = 10\tau_{abs}$

Fig. 7. Evolution of the elementary and global Sobol's and Shapley's indices over time for the variables (a) k, (b) L_0, (c) r_{int}, (d) m_P, (e) m_c. In (f) the comparison between the global indices and the reference solution is shown at $t = 10\tau_{abs}$.

Table 3

Input variables and corresponding intervals for the car shock absorber benchmark problem.

Variable	Interval
k [Nm^{-1}]	$\mathcal{U}([8000, 11000])$
L_0 [m]	$\mathcal{U}([0.4, 0.6])$
r_{int} [m]	$\mathcal{U}([0.001, 0.005])$
h [m]	$\mathcal{U}([0.01, 0.04])$
m_P [kg]	$\mathcal{U}([60, 360])$
m_C [kg]	$\mathcal{U}([900, 1300])$

with time. This is an expected result because r_{int} only intervenes in the expression of the damping force. This force depends upon the piston speed, thus, when the system reaches a stable state, the piston speed becomes zero and the influence of the r_{int} variable goes to zero too.

According to the definition of elementary and total indices of Eqs. (10) and (12), respectively, and by looking at the values illustrated in Fig. 7, one can conclude that, for each input variable, the influence of the interaction with the rest of the variables on the output is relatively small. It varies between 3.2% at $t = \tau_{abs}$ and 1.8% at $t = 10\tau_{abs}$ for variable k, 3.7% at $t = \tau_{abs}$ and 2.1% at $t = 10\tau_{abs}$ for variable L_0, 5.0% at $t = \tau_{abs}$ and 0.006% at $t = 10\tau_{abs}$ for variable r_{int}, 1.5% at $t = \tau_{abs}$ and 1.3% at $t = 10\tau_{abs}$ for variable m_P, 2.8% at $t = \tau_{abs}$ and 1.8% at $t = 10\tau_{abs}$ for variable m_C. Moreover, by classifying the Sobol's total indices in ascending order, one can conclude that the parameter L_0 is

the one influencing the most the piston motion for all values of the characteristic time.

As it can be inferred from the above results, r_{int} has a minor role on the behaviour of the system for $t > 3\tau_{abs}$. However, if one considers a larger variation range, with values of r_{int} higher than 0.005 m for example, the piston is characterised by an unstable motion in most cases. This highlights the importance of correctly defining the intervals of definition of the input variables for GSA. Accordingly, r_{int} is the input variable having the least influence on the piston motion.

The Shapley's index related to each input variable has been evaluated under the same conditions as those considered for the GSA based on Sobol's indices. They are also reported in Fig. 7. Their values always fall between S_i and S_{T_i}, for each time, and their sum is always equal to one. Overall, they provide the same information as S_{T_i}.

Regarding the reliability of the calculated indices, as shown in Fig. 7(f) at $t = 10\tau_{abs}$, the dispersion on the Shapley's indices is higher than the one on Sobol's indices, in agreement with the results obtained for the Ishigami function.

4.3. Dry friction oscillator

4.3.1. Analytical description

The second dynamical system studied in this work is a dry friction oscillator. It is modelled using a mass–spring system subjected to a frictional force from a conveyor belt, as shown in Fig. 8. This system
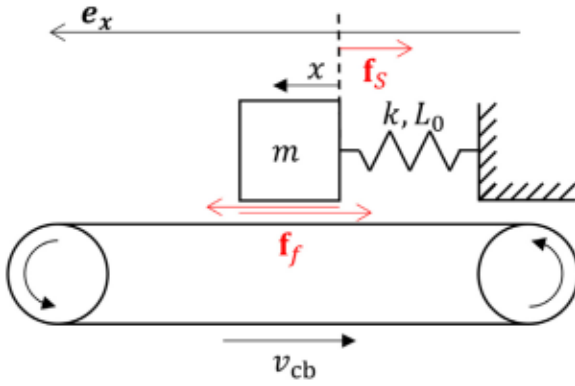
Fig. 8. Schematic representation of the dry friction oscillator.

has been studied extensively in [73,74] as a reference problem due to its strong non-linearity introduced by the friction force.

The equation of motion of this system reads:

$$m\mathbf{a} = \sum_{i=1}^{q} \mathbf{f}_{ext,i}, \tag{22}$$

where m is the mass, $\mathbf{a}$ is the acceleration of the mass (whose component along the x axis is denoted by $\ddot{x}(t)$), $\mathbf{f}_{ext,i}$ is the ith external force applied to the mass and q is the total number of external forces.

The external forces applied to the mass are listed below:

- Spring force:

$$\mathbf{f}_S = -k(x - L_0)\mathbf{e}_x, \tag{23}$$

where k is the spring constant and L_0 is its unstretched length.
- Friction force:

$$\mathbf{f}_F = -\frac{F_s \text{sign}(v_{rel})}{1 + \delta|v_{rel}|}\mathbf{e}_x, \tag{24}$$

where $F_s = 1$ N is the threshold value of the friction, $v_{rel} = \dot{x} - v_{cb}$ is the component along the x axis of the relative speed of the mass (with respect to the conveyor belt), where $\dot{x}$ is the mass speed and v_{cb} is the conveyor belt speed, which is constant, and δ is a constant. The model of the frictional force provided by Eq. (24) was taken from [74].

4.3.2. Modelica model

As explained in [74], this problem is characterised by some numerical issues when solving the motion equation due to the non-linear profile (hyperbola) of the frictional force $\mathbf{f}_F$ as a function of the relative speed of the mass. To overcome these issues, two numerical strategies are generally employed: the smooth method and the switch model.

The smooth method consists of introducing a term that makes the curve continuous for $1 + \delta|v_{rel}| = 0$. Conversely, the switch model is a succession of conditional tests allowing to determine the system behaviour mode (stuck, sliding or in transition) in order to apply the correct analytical description.

However, the specificities of Modelica programming language (i.e., the elementary blocks assembly strategy and the acausal nature) make it possible to get rid of these issues and to directly enter the friction force equation, without any approximation, in an adapted block. The resulting model is illustrated in Fig. 9 and the results obtained by considering the reference parameters listed in Table 4 coincide with those presented in [74].

These results are illustrated by the phase diagram of Fig. 10 that shows the evolution of the velocity of the mass vs. its position. Two phases can be observed: a first one starting from the origin and going towards the circular curve corresponding to the transient regime until the stable position is reached. The plateau occurs when the mass has a zero relative speed with respect to the conveyor belt.

For this example, the FMU file is exported in co-simulation mode, by incorporating a solver based on the Runge–Kutta fourth-order algorithm with integration and communication time steps both equal to 10^{-5} s.

4.3.3. Numerical results

Before starting the GSA, the first step is to define the observed output. In the case of the dry friction oscillator, the output response is the behaviour of the mass movement, which can be either stable or unstable. More precisely, the behaviour is considered stable when the relative speed of the mass is zero (sticking) during a time interval Δt_{stick} greater than or equal to $\tau_{stable} = 0.3$ s. Conversely, when the mass velocity does not have a constant part or is zero during a time interval $\Delta t_{stick} < \tau_{stable}$, the behaviour is considered unstable. A case of stable behaviour is illustrated in Fig. 10. Therefore, the output response is the time interval Δt_{stick} that must be greater than τ_{stable} when a stable behaviour is achieved.

The parameters used as input variables, with the corresponding definition ranges, are listed in Table 5. As in the case of the car shock absorber, the variation ranges have been selected to simulate the system characteristic behaviour.

The Sobol analysis results considering the FAST method are reported in Fig. 11 and compared to the reference results obtained with the Saltelli 2010 method for $N = 2^{22}$. From the analysis of these results, one can infer that the influence of the interaction among variables is significant due to the high difference between total and elementary indices. Indeed, the influence of the interaction is about 48%.

Moreover, from the histogram illustrated in Fig. 11, one can infer that the input variable with the strongest influence on the motion stability is v_{cb} with values of S_i and S_{T_i} equal to 32% and 75%, whilst the one having the weakest influence is k with elementary and total indices equal to 1% and 23%, respectively. However, although the elementary index for k is negligible when compared to the one of the other variables, the value of the total index cannot be neglected, highlighting that this variable influences the stability of the motion when copuled with the other inputs. Accordingly, particular attention must be paid to the choice of the frictional force model wherein the variable v_{cb} is involved. This conclusion is found in the literature as the same model is used with different frictional force expressions in order to highlight particular behaviours (the Hopf bifurcation or the slip–slip bifurcation in [73]). Unlike the values obtained for the Sobol's indices in the case of the car shock absorber, the accuracy in assessing the total indices of variables L_0 and F_S with the FAST algorithm is not enough,
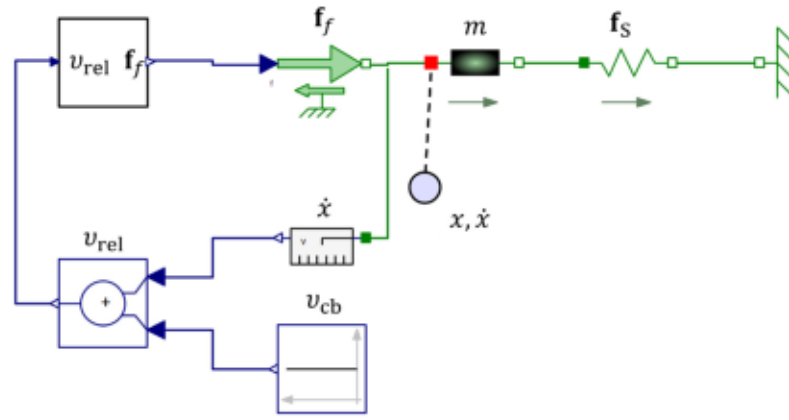
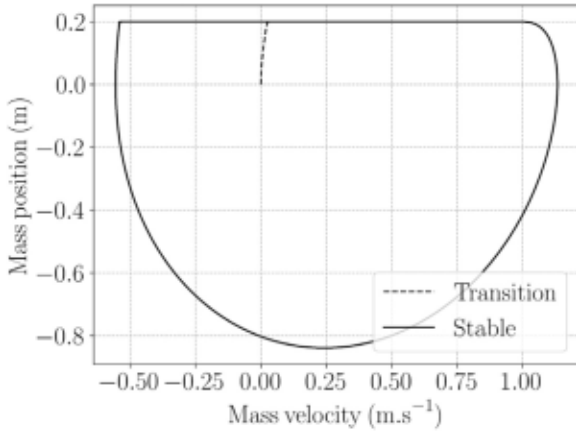Fig. 9. Modelica model of the dry friction oscillator system.

Fig. 10. Reference phase diagram plotting the evolution of the mass velocity (in m.s^{-1}) vs. its position (m), obtained with the Modelica model.

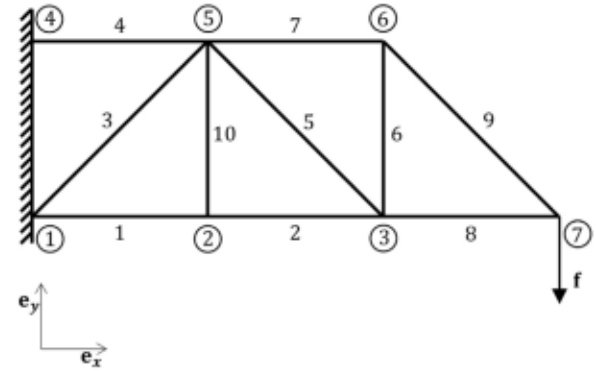Fig. 12. Truss structure model.

5. Global sensitivity analysis on finite element models for structural analysis

5.1. Problem 1: static analysis of a truss structure.

5.1.1. Problem formulation and numerical model

In the test cases discussed in sections 3 and 4, GSA is applied to an analytical benchmark and to the study of a mechanical system via a Modelica modelling approach, respectively. In this section, GSA is carried out on a model of a truss structure whose mechanical response is obtained via a dedicated Finite Element (FE) model. The goal is to evaluate the influence of the area of the cross section of the trusses on the vertical displacement of the structure computed at the node where the force is applied.

The geometry of the truss structure, shown in Fig. 12, has been taken from [54]. The relevant geometrical and material properties, used in the numerical analyses, are listed in Table 6. Particularly, the material properties (E and v in Table 6) are kept constant together with the lengths of the trusses constituting the structure (L_i in Table 6). A vertical force is applied at node 7 in Fig. 12 and the degrees of freedom are set to 0 at nodes 1 and 4. To reproduce exactly the problem formulation presented in [54], the cross sections areas, considered as input variables for the GSA, are those of the trusses from 1 to 9. Table 6 reports the interval of variation of the cross sections areas assumed as variables as well as the value of cross section area of truss n. 10, which is constant. The parametric FE model of the truss structure has been generated with the Ansys Parametric Design Language (APDL) within the ANSYS® FE commercial software. Each truss is modelled with LINK180 elements (truss elements with three degrees of freedom per node). Finally, linear static FE analyses have been carried out.
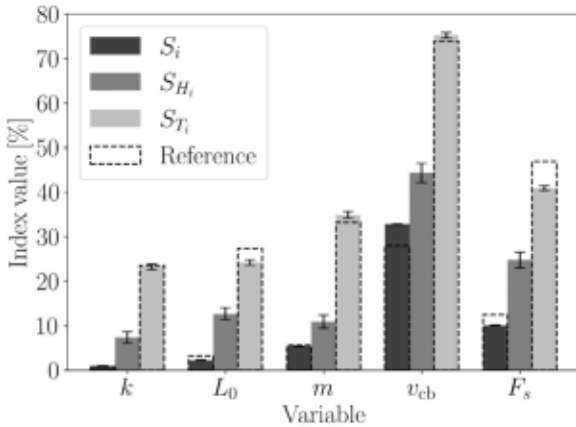

Fig. 11. GSA results on the dry friction oscillator model.

the percentage error with respect to the related reference values being 11% and 13%, respectively. This means that the number of samples N should be increased to obtain a better accuracy level. This conclusion is supported also by the dispersion on the total indices of these variables which is higher than that related to the rest of the input variables.

Regarding the Shapley's indices, the same remarks already done for the car shock absorber example can be repeated also in this case.

Table 6
Characteristic parameters and input variables of the truss problem.

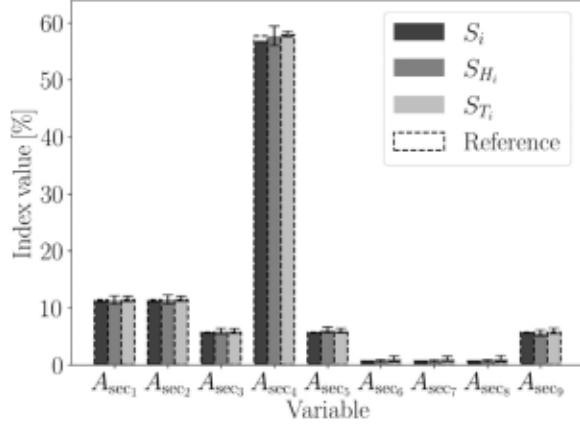| Parameter	Value
E [MPa]	2.1×10^5
v [-]	0.3
$L_{v6\{1,2,4,7,8,10\}}$ [mm]	1×10^3
$L_{v6\{3,5,6,9\}}$ [mm]	$\sqrt{2} \times 10^3$
$A_{sec_{10}}$ [mm^2]	1×10^2
$A_{sec_{9\{1,\ldots,9\}}}$ [mm^2]	$\mathcal{U}([0.5, 2.5]) \times 10^2$
F [N]	1000

Fig. 13. GSA results of the truss model.

Fig. 14. Composite stiffened panel scheme.

5.1.2. Numerical results

The results of the Sobol analysis, considering the FAST method, are reported in Fig. 13 and compared to the reference values of the elementary indices S_i taken from [54]. Regarding the results of the total indices, the corresponding reference values are obtained using the Saltelli 2010 algorithm with $N = 2^{22}$. From the analyses of these results, one can infer that the cross-influence among variables is negligible because the values of elementary and total indices are very close. Indeed, the sum of the correlated influence is approximately 2%.

Moreover, the results of Fig. 13 point out that the input variable with the strongest influence on the maximum vertical displacement is the cross-section of the truss n. 4, A_{sec_4}, with values of S_i and S_{T_i} equal to 56% and 58%, respectively. Conversely, the cross-sections with the weakest influence are those of bars n. 6, 7 and 8 with values of S_i and S_{T_i} smaller than 1.1%.

By comparing the relative error between the values of the computed indices and those used as references, it stands out that a number of samples $N = 2^{12}$ is enough to limit the average percentage error to a maximum of 2.5%. Regarding the Shapley's indices, the same remarks already discussed for the test cases presented in the above sections can be repeated here.

5.2. Problem 2: eigenvalue buckling analysis of a mono-stringer composite stiffened panel.

5.2.1. Problem formulation and numerical model

The usefulness of GSA is shown through the study of a realistic engineering problem dealing with a composite stiffened panel under uni-axial compression, a structural solution usually adopted in aircraft structures.

The geometry of the stiffened panel used to perform the GSA is taken from [75] and it is illustrated in Fig. 14 together with the relevant geometrical parameters used in the analysis. The search for trade offs between a lightweight solution and an adequate mechanical behaviour (especially when buckling is the main critical phenomenon
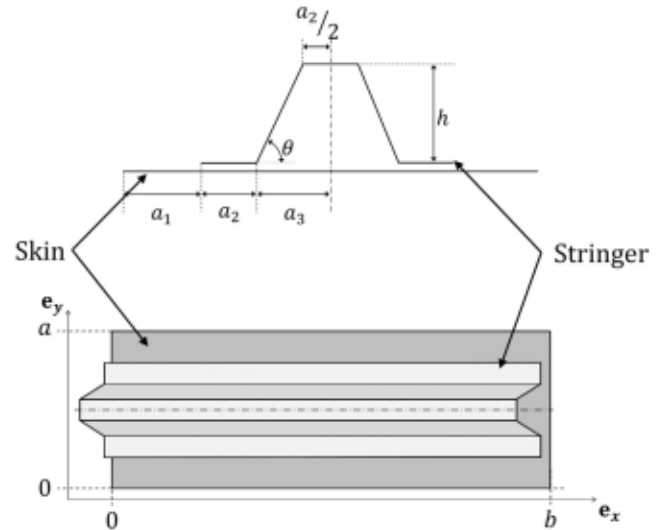
involved in the problem formulation) is not a trivial task if both the geometry of the stringer and the stacking sequences constituting skin and stringers are used as design variables. Even though design rules have been defined to limit the complexity related to the design of these structures, the great number of design variables (geometry and material) leads to a significant number of structural solutions to be assessed. Moreover, without the limitations of standard design rules, it is possible to obtain a remarkable improvement in the structural efficiency by addressing the problem via a multi-scale optimisation approach, as described in [75]. Nonetheless, GSA can be used as a preliminary design tool to determine whether it is possible to assess the sensitivity of the structural responses to the set of design variables and to determine, thus, the parameters that can be excluded from the set of design variables due to the negligible influence on the behaviour of the structure. Therefore, conducting a preliminary GSA will help reducing the computational effort of further numerical campaigns (including optimisation process).

In this context, the goal of the example presented in this section is to assess the influence of the geometrical variables, shown in Fig. 14, and of the polar parameters describing the anisotropic behaviour of the stiffness matrices of the laminate at the macroscopic scale [75,76] on the value of the first buckling load of the stiffened panel.

The stiffened panel is constituted of two regions, the skin and the stringer. The skin, characterised by a length a and a width b, consists of n_S carbon-epoxy pre-preg plies, whose material properties are given in [75]. The stringer whose geometry is completely defined by the geometrical parameters a_2, a_3 and h, is made of n_B plies. Normalised quantities are introduced to define the geometry of the stiffened panel as follows:

$$c_1 = 2\frac{a_2}{a}, \quad c_2 = 2\frac{a_3}{a_2}, \quad c_3 = \frac{h}{a_2}. \tag{25}$$

To avoid generating inconsistent geometries, two inequalities are introduced as follows:

$$a_1 = \frac{a}{2} - a_2 - a_3, \quad \theta = \text{atan}\left(\frac{h}{a_3 - \frac{a_2}{2}}\right), \quad \text{with } a_1 > 0, \text{ and } a_3 \geq \frac{a_2}{2}. \tag{26}$$

The first inequality is necessary to avoid overlap between neighbouring stiffeners and the second one prevents negative values of the angle θ. The mechanical behaviour of the laminates is defined via the polar formalism, which is an efficient approach to define the mechanical response of composite laminates [76], especially for design purposes. Specifically, as discussed in [76], only three polar parameters are

required to describe the macroscopic elastic behaviour (in terms of membrane, bending and membrane-bending coupling stiffness matrices) of a quasi-homogeneous orthotropic laminate: two anisotropic moduli, i.e., R_{0K}^{A*} and R_1^{A*}, and the polar angle Φ_1^{A*} (which represents the orientation of the main axis of orthotropy). Of course, the maximum value of the first buckling load can be achieved by aligning the main axis of orthotropy with the direction of the applied load. To satisfy this condition it is sufficient to set $\Phi_1^{A*} = 0$ for both stringer and skin, as discussed in [75]; thus, Φ_1^{A*} is not integrated among the design variables of the problem. As done for the geometrical parameters, normalised values are also used for defining the mechanical behaviour of the stiffened panel. Particularly, two coefficients are introduced as follows:

$$\rho_0 = \frac{R_{0K}^{A*}}{R_0}, \quad \rho_1 = \frac{R_1^{A*}}{R_1}, \tag{27}$$

where R_0 and R_1 are the anisotropic moduli of the ply reduced stiffness matrix [75]. Moreover, as suggested in [77], it is possible to avoid introducing the feasibility conditions on ρ_0 and ρ_1 (which are required to ensure the existence of a stack corresponding to the values of the anisotropic moduli and polar angles used at the macroscopic scale to describe the anisotropic behaviour of the laminate) by considering the following variable change:

$$(\alpha_0, \alpha_1) := \left(\frac{\rho_0 - 1}{2(\rho_0^2 - 1)}, \rho_1 \right). \tag{28}$$

Of course, different values of α_0 and α_1 are used for the skin and for the stringer, namely, α_{0_S} and α_{1_S} for the skin and α_{0_B} and α_{1_B} for the stringer. More details about the implementation of the polar formalism within FE analysis of composite structures can be found in [75,77]. The reference configuration of the mono-stringer stiffened panel is characterised by a value of the first buckling load equal to 445074 N [75]. The input variables of the GSA are listed in Table 7. A detailed description of the numerical strategy to recover the stacking sequences starting from the definition of the laminate polar parameters is proposed in [75].

The FE model of the stiffened panel, shown in Fig. 15, is created via APDL scripts and its mesh is made of SHELL181 elements (four-node elements with six degrees of freedom at each node). A particular care has been given to the definition of the mesh size to obtain at least three elements along the width of each horizontal skin and stringer partition (segments of length a_1, a_2 and a_3) and at least six elements for the segments of height h. For the reference stiffened panel, the FE model consists of a total of 4438 elements. To enforce periodic boundary conditions between the opposite edges of the skin of the stiffened panel suitable constraint equations have been used (see [75] for more details on this point). Moreover, rigid multi-point constraint elements (MPC184) have been created and linked with two reference nodes on both sides: at $x = 0$, the relevant reference node is clamped whereas, at $x = b$, all the degrees of freedom have been set to zero, except the displacement along x axis. Moreover, a unit force along the x axis is applied on this node. All details about the FE model of the stiffened panel are available in [75].

5.2.2. Numerical results

The results of the Sobol analysis obtained with the FAST algorithm with $N = 2^{12}$ are shown in Fig. 16 and compared to the reference results obtained with the Saltelli 2010 method for $N = 2^{22}$. By looking at these results, one can infer that the influence of the interactions among input variables is important with a value approximatively equal to 24%.

The results of Fig. 16 show that c_1 is the variable with the highest total index S_{T_i} (approximately equal to 61%). In fact, c_1 acts on the width of the flanges of the stiffener, which has a relevant impact on the flexural stiffness of the panel. Moreover, the same variable is also the one with the highest interaction with the other variables since the difference between the elementary and total Sobol's indices
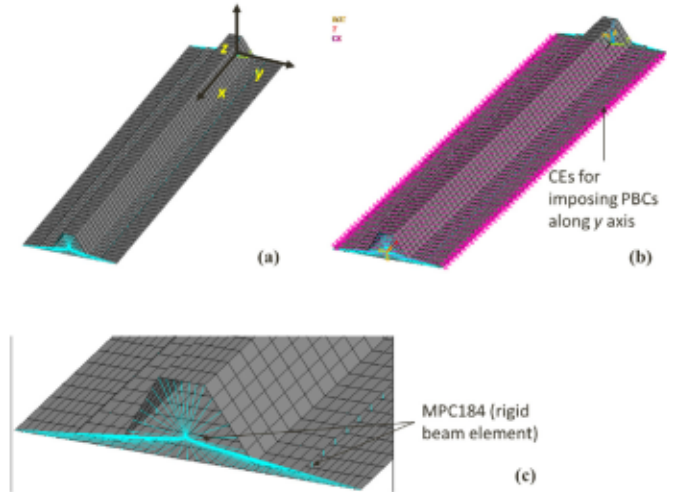

Fig. 15. (a) Finite element model of the repetitive unit and related reference frame, (b) details of constraint equation to impose periodic boundary conditions along y-axis and (c) details of MPC184 elements.

Table 7
Input variables of the composite stiffened panel problem.

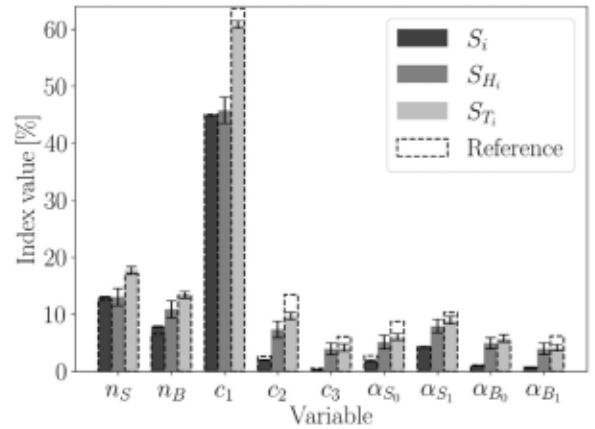| Variable	Value	Type
n_S	$\mathcal{U}([20, 32])$	Discrete
n_B	$\mathcal{U}([20, 32])$	Discrete
c_1	$\mathcal{U}([0.1, 0.45])$	Continuous
c_2	$\mathcal{U}([1, 3])$	Continuous
c_3	$\mathcal{U}([1, 3])$	Continuous
α_{0_S}	$\mathcal{U}([0, 1])$	Continuous
α_{1_S}	$\mathcal{U}([0, 1])$	Continuous
α_{0_B}	$\mathcal{U}([0, 1])$	Continuous
α_{1_B}	$\mathcal{U}([0, 1])$	Continuous

Fig. 16. GSA results on the stiffened panel model.

contributions for c_1 is about 15%. Conversely, the variables for which the GSA returns the smallest values of both total and elementary indices are c_3 and the α_{i_S}, α_{i_B} ($i = 0, 1$) quantities with values smaller than 10%. It is noteworthy that the buckling response can be very sensitive to both the geometric and polar parameters. For this reason, when considering the results of Fig. 16, it must be reminded that, while the output quantity used within the GSA is the eigenvalue, different set of input parameters can return different buckling mode shapes. As an example, the buckling mode shapes obtained with four sets of input variables are shown in Fig. 17. Particularly, the extreme values of the intervals of variation of the input variables c_1 and c_3 have been
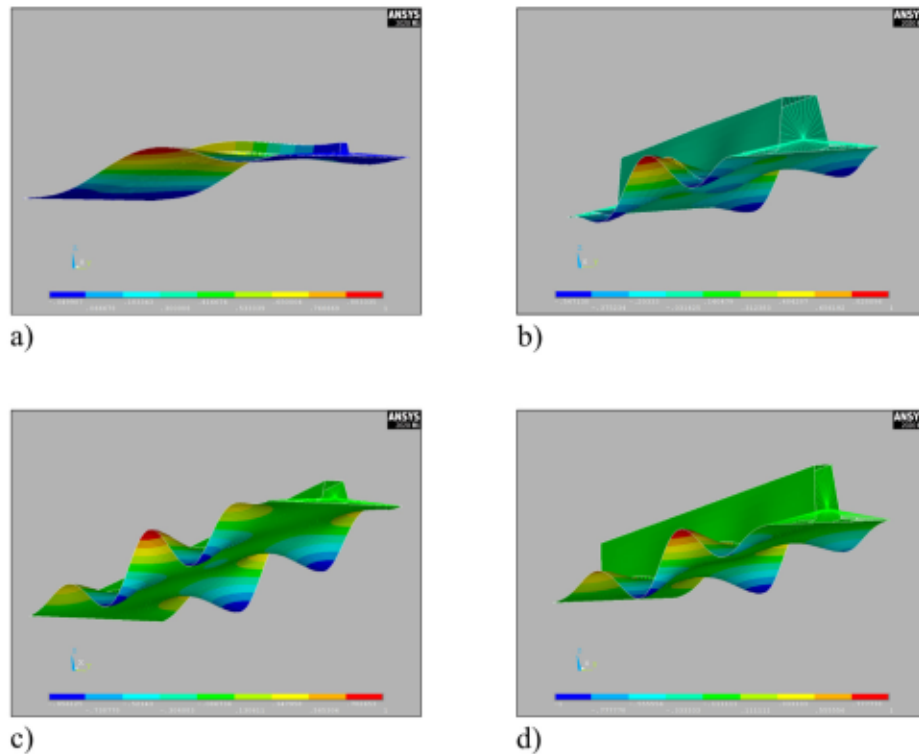
Fig. 17. Buckling mode shapes of the mono-stringer stiffened panel for (a) $c_1 = c_{1min}$, (b) $c_1 = c_{1max}$, (c) $c_3 = c_{3min}$ and (d) $c_3 = c_{3max}$.

considered, whilst the other parameters are set equal to those of the reference configuration, whose details are available in [75].

From the results of Fig. 16, it also stands out that for the variables c_2, c_3 and α_{iS}, α_{iB} ($i = 0, 1$) the percentage error of the Sobol's indices with respect to the reference values ranges between 28% and 32%. This result suggests that a greater value of N should be used. Regarding the Shapley's indices values in Fig. 16, the same remarks already done for other test cases hold also in this case.

6. Conclusions

In this work, the effectiveness of different algorithms for global sensitivity analysis has been evaluated on multiple-input–single-output problems of different nature. An analytical benchmark has been used to compare different algorithms available in the literature based both on Sobol's indices and Shapley's effect in terms of computational costs, accuracy, and reliability. As a result, regarding Sobol's indices, a suitable trade-off between the considered criteria was obtained with the Fast Amplitude Sensitivity Test algorithm. Moreover, Shapley's indices were also employed since they provide complementary information with respect to Sobol's indices. The global sensitivity analysis was then performed on two problems involving non-linear dynamical systems of industrial interest, i.e., a car shock absorber and a dry friction oscillator, modelled via the Modelica language. To further reduce the computational cost required by the considered algorithms, the Modelica models were exported via the Functional Mock-up Interface standard and coupled with the algorithms coded in Python environment.

Subsequently, the global sensitivity analysis has been conducted also on structural problems characterised by a greater number of variables if compared to the other studied problems, to be solved via finite element analyses, namely, the static analysis of a truss structure and the eigenvalue buckling analysis of a composite stiffened panel. Both numerical models have been created via parametric APDL scripts and coupled with the algorithms based on Sobol's indices and Shapley's effect.

The results of the global sensitivity analysis on the mechanical problems have pointed out, on the one hand, the input variables having the weakest influence on the observed quantity and, on the other hand, if interactions exist between the considered variables. These preliminary results would allow, if more detailed studies are needed, to reduce the number of sets of design variables to be evaluated.

This work represents a first step in the development of a thorough global sensitivity analysis methodology for multi-field problems, where the Modelica modelling strategy, used to gather the response of the different sub-systems constituting the global model, is enriched with meta-models describing the response of those sub-systems for which a high level of accuracy is required. With a smart use of both high-fidelity finite element models and experimental data to build and validate the meta-model, it will be possible to accurately simulate complex phenomena and drastically reducing the computational costs. Research is ongoing on this topic.

CRediT authorship contribution statement

Bruno Vuillod: Writing – original draft, Validation, Software, Methodology, Investigation. **Marco Montemurro:** Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Enrico Panettieri:** Writing – review & editing, Supervision. **Ludovic Hallo:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix. Python code for the calculation of the shapley's indices

```python
import numpy as np

#Inputs:
# n: number of samples
# d: number of independent input variables
# x_LB (numpy.array, size [d,1]): lower bound of
#     the array of input variables
# x_UB (numpy.array, size [d,1]): upper bound of
#     the array of input variables
# myfunc (function handle): the user-defined
#     function
#Outputs:
# phi1 (numpy.array, size [d,1]): the array
#     collecting the Shapley's indices
# phi2 (numpy.array, size [d,1]): the array
#     collecting the variance of Shapley's indices
# var_tot: the total variance of the output
def Shapley(n,d,x_LB,x_UB,myfunc):
  x = np.zeros((n,d)) #initialise array x
  y = np.zeros((n,d)) #initialise array y

  for i in range(d):
    # fill array x with n samples uniformly
    #   distributed between x_LB and x_UB along each
    #   dimension
    x[:,i] = np.random.uniform(x_LB[i], x_UB[i],
    (1,n))
    # fill array y with n samples uniformly
    #   distributed between x_LB and x_UB along each
    #   dimension
    y[:,i] = np.random.uniform(x_LB[i], x_UB[i],
    (1,n))

  shu = np.arange(0, d, 1)
  pm = np.zeros((n,d))

  for i in range(n):
      pm[i] = np.take(shu, np.random.permutation(
    shu.shape[0]), axis=0, out=shu)

  a = np.arange(0, d, 1)

  z = x.copy()
  fz1 = myfunc(z)
  fx = fz1

  phi1 = np.zeros(d) # initialization
  phi2 = np.zeros(d) # initialization

  ind = np.zeros((n,d)).astype(bool)

  for j in range(d):
      for k in range(n):
          ind[k] = (pm[k, j] == a)
      y_red = y*ind

      for i1 in range(n):
          for i2 in range(d):
              if(y_red[i1, i2] != 0):
                  z[i1, i2] = y_red[i1, i2]

      fz2 = myfunc(z)
      fmarg = ((fx-fz1/2-fz2/2)*(fz1-fz2))

      #Shapley's index for input variable j
      phi1 = phi1 + fmarg @ (ind/n)

      #Variance of Shapley's index
      phi2 = phi2 + fmarg**2 @ (ind/n)
      fz1 = fz2

  var_tot = sum(phi1) #total variance of the
    output

  return phi1, phi2, var_tot
```

References

[1] Razavi S, et al. The future of sensitivity analysis: An essential discipline for systems modeling and policy support. Environ Modell Softw 2021;137:104–954.

[2] Saltelli A, Tarantola S, Campolongo F. Sensitivity analysis as an ingredient of modeling. Statist Sci 2000;15(4):377–95.

[3] Sobol IM. Sensitivity estimates for nonlinear mathemitical models. MMCE 1993;1(4):407–14.

[4] Morris MD. Factorial sampling plans for preliminary computational experiments. Technometrics 1991;33(2):161–74.

[5] Hoeffding W. A class of statistics with asymptotically normal distribution. Ann Math Stat 1948;19(3):293–325.

[6] Razavi S, Gupta HV. A new framework for comprehensive, robust, and efficient global sensitivity analysis: 1. Theory. Water Resour Res 2016;52(1):423–39.

[7] Razavi S, Gupta HV. A new framework for comprehensive, robust, and efficient global sensitivity analysis: 2. Application. Water Resour Res 2016;52(1):440–55.

[8] Kleijnen JP. Sensitivity analysis and optimization of system dynamics models: Regression analysis and statistical design of experiments. Syst Dyn Rev 1995;11(4):275–88.

[9] Li L, Lu Z, Li W. State dependent parameter method for importance analysis in the presence of epistemic and aleatory uncertainties. Sci China Technol Sci 2012;55:1608–17.

[10] Wang P, Lu Z, Cheng L. Importance measures for imprecise probability distributions and their sparse grid solutions. Sci China Technol Sci 2013;56:1733–9.

[11] Zhou C, Shi Z, Kucherenko S, Zhao H. A unified approach for global sensitivity analysis based on active subspace and Kriging. Reliab Eng Syst Saf 2022;217:108080.

[12] Zhang K, Lu Z, Cheng K, Wang L, Guo Y. Global sensitivity analysis for multivariate output model and dynamic models. Reliab Eng Syst Saf 2020;204:107195.

[13] Thapa M, Missoum S. Uncertainty quantification and global sensitivity analysis of composite wind turbine blades. Reliab Eng Syst Saf 2022;222.

[14] Iooss B, Prieur C. Shapley effects for sensitivity analysis with correlated inputs: Comparisons with Sobol' indices, numerical estimation and applications. Int J Uncertain Quantif 2019;9(5):493–514.

[15] Chastaing G, Gamboa F, Prieur C. Generalized hoeffding-sobol decomposition for dependent variables - application to sensitivity analysis. Electron J Stat 2012;6:2420–48.

[16] Chastaing G, Gamboa F, Prieur C. Generalized sobol sensitivity indices for dependent variables: numerical methods. J Stat Comput Simul 2015;85(7):1306–33.

[17] Hooker G. Generalized functional ANOVA diagnostics for high-dimensional functions of dependent variables. J Comput Graph Statist 2007;16(3).

[18] Stone CJ. The use of polynomial splines and their tensor products in multivariate funciton estimation. Ann Statist 1994;6(4):701–26.

[19] Broto B, Bachoc F, Depecker M, Martinez JM. Sensitivity indices for independent groups of variables. Math Comput Simulation 2019;163.

[20] Song E, Barry LN, Staum J. Shapley effects for global sensitivity analysis: Theory and computation. SIAM/ASA J Uncertain Quantif 2016;4(1):1060–83.

[21] Moretti S, et al. Combining Shapley value and statistics to the analysis of gene expression data in children exposed to air pollution. BMC Bioinformatics 2008;9.

[22] Owen AB. Sobol' indices and shapley value. SIAM-ASA J Uncertain Quantif 2014;2(1).

[23] Xu C, Zdzislaw Gertner G. Uncertainty and sensitivity analysis for models with correlated parameters. Reliabil Eng Syst Safety 2008;93:1563–73.

[24] Plischke E. An effective algorithm for computing global sensitivity indices (EASI). Reliab Eng Syst Saf 2010;95(4):354–60.

[25] Sarazin G, Morio J, Lagnoux A, Balesdent M, c Brevault L. Reliability-oriented sensitivity analysis in presence of data-driven epistemic uncertainty. Reliab Eng Syst Saf 2021;215:107733.

[26] Zhu X, Sudret B. Global sensitivity analysis for stochastic simulators based on generalized lambda surrogate models. Reliab Eng Syst Saf 2021;214:107815.

[27] Tabandeh A, Sharma N, Gardoni P. Uncertainty propagation in risk and resilience analysis of hierarchical systems. Reliab Eng Syst Saf 2022;219:108–208.

[28] Azzini I, Rosati R. Sobol' main effect index: an innovative algorithm (IA) using dynamic adaptive variances. Reliab Eng Syst Saf 2021;213:107647.

[29] Damblin G, Ghione A. Adaptive use of replicated Latin Hypercube Designs for computing Sobol' sensitivity indices. Reliab Eng Syst Saf 2021;212:107507.

[30] Perrin TV, Roustant O, Rohmer J, Alata O, Naulin JP, Idier D, Pedreros R, Moncoulon D, Tinard P. Functional principal component analysis for global sensitivity analysis of model with spatial output. Reliab Eng Syst Saf 2021;211:107522.

[31] Lamboni M, Kucherenko S. Multivariate sensitivity analysis and derivative-based global sensitivity measures with dependent variables. Reliab Eng Syst Saf 2021;212.

[32] Ge Q, Menendez M. Extending Morris method for qualitative global sensitivity analysis of models with dependent inputs. Reliab Eng Syst Saf 2017;162:28–39.

[33] Ballester-Ripoll R, Paredes EG, Pajarola R. Sobol tensor trains for global sensitivity analysis. Reliab Eng Syst Saf 2019;183:311–22.

[34] Konakli K, Sudret B. Global sensitivity analysis using low-rank tensor approximations. Reliab Eng Syst Saf 2016;156:64–83.

[35] Liu F, Wei P, Tang C, Wang P, Yue Z. Global sensitivity analysis for multivariate outputs based on multiple response Gaussian process model. Reliab Eng Syst Saf 2019;189:287–98.

[36] Antoniadis A, Lambert-Lacroix S, Poggi JM. Random forests for global sensitivity analysis: A selective review. Reliab Eng Syst Saf 2021;206.

[37] Goda T. A simple algorithm for global sensitivity analysis with Shapley effects. Reliab Eng Syst Saf 2021;107702.

[38] Saltelli A. Making best use of model evaluation to compute sensitivity indices. Comput Phys Comm 2002;145(2):280–97.

[39] Saltelli A, Annoni P, Azzini I, Campolongo F, Ratto M, Tarantola S. Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. Comput Phys Comm 2010;181(2):259–70.

[40] Cukier RI, et al. Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. Theory. J Chem Phys 1973;59(8).

[41] Cukier RI, Levine HB, Shuler KE. Nonlinear sensitivity analysis of multiparameter model systems. J Comput Phys 1978;26(1):1–42.

[42] Saltelli A, Bolado R. An alternative way to compute Fourier amplitude sensitivity test (FAST). Comput Statist Data Anal 1998;26(4):445–60.

[43] Da Veiga S, Gamboa F, Iooss B, Prieur C. Basics and trends in sensitivity analysis theory and practice in R. Society for Industrial and Applied Mathematics,U.S.; 2021, p. 293.

[44] Iooss B, Lemaître P. A review on global sensitivity analysis methods. Uncertainty Management in Simulation-Optimization of Complex System. 2015;59:101–22.

[45] Tarantola S, Gatelli D, Mara TA. Random balance designs for the estimation of first order global sensitivity indices. Reliab Eng Syst Saf 2006;91(6):717–27.

[46] Tissot JY, Prieur C. Bias correction for the estimation of sensitivity indices based on random balance designs. Reliab Eng Syst Saf 2012;107:205–13.

[47] Goffart J, Woloszyn M. RBD-FAST : une méthode d'analyse de sensibilité rapide et rigoureuse pour la garantie de performance énergétique. In: Conférence francophone de L'international building performance simulation association. 2018, p. 1–8.

[48] Mara TA. Extension of the RBD-FAST method to the computation of global sensitivity indices. Reliab Eng Syst Saf 2009;94(8):1274–81.

[49] Shapley LS. A value for n-person games. Contribut Theo Games (AM-28) 1953;II:3–17.

[50] Heredia MB, Prieur C, Eckert N. Global sensitivity analysis with aggregated Shapley effects, application to avalanche hazard assessment. Reliab Eng Syst Saf 2022;222:245–51.

[51] Ishigami T, Homma T. An importance quantification technique in uncertainty analysis for computer models. In: Proceedings. First international symposium on uncertainty modeling and analysis, vol. 21. College Park, MD, USA; 1990, p. 99–104.

[52] Becker W. Metafunctions for benchmarking in sensitivity analysis. Reliab Eng Syst Saf 2020;204:107189.

[53] Sobol IM, Levitan YL. On the use of variance reducing multipliers in Monte Carlo computations of a global sensitivity index. Comput Phys Comm 1999;117:52–61.

[54] Wu Z, Wang W, Wang D, Zhao K, Zhang W. Global sensitivity analysis using orthogonal augmented radial basis function. Reliab Eng Syst Saf 2019;185:291–302.

[55] Benoumechiara N, Elie-Dit-Cosaque K. Shapley effects for sensitivity analysis with dependent inputs: bootstrap and kriging-based algorithms. ESAIM: Proc Surv 2019;65:266–93.

[56] Fritzson P. Principles of object oriented modeling and simulation with modelica 3.3: A cyber-physical approach. IEEE Press; 2014.

[57] Vuillod B, Hallo L, Panettieri E, Montemurro M. Sensitivity analysis of a car shock absorber through a functional mock-up units-based modelling strategy. In: Proceedings of 14th modelica conference 2021, LinköPing, Sweden, September 20-24, 2021. 2021, p. 307–14.

[58] Huimin Z, et al. Modelica modeling and simulation for a micro gas-cooled reactor. Proceedings of 14th modelica conference 2021, LinköPing, Sweden, September 20-24, 2021, vol. 181 2021;569–75.

[59] Prado-Velasco M. In-silico virtual prototyping multilevel modeling system for Cyborgs (CybSim) as a novel approach for current challenges in biosciencies. In: Proceedings of 14th modelica conference 2021, LinköPing, Sweden, September 20-24, 2021. 2021, p. 485–96.

[60] Pathak A, Schneider K, Norrefeldt V. Use of modelica to predict risk of Covid-19 infection in indoor environments. In: Proceedings of 14th modelica conference 2021, LinköPing, Sweden, September 20-24, 2021, vol. 181. 2021, p. 463–9.

[61] Campanini P, Ferretti G. Object-oriented models of parallel manipulators. In: Proceedings of 14th modelica conference 2021, LinköPing, Sweden, September 20-24, 2021, vol. 181. 2021, p. 241–8.

[62] Plogert K. The tailoring process in the German V-Model. J Syst Archit 1996;42:601–9.

[63] El Hefni B, Bouskela D. Modeling and simulation of thermal power plants with ThermoSysPro. Springer Cham; 2019, p. 99–152.

[64] Renier R, Chenouard R. De SysML a Modelica : aide à la formalisation de modèles de simulation en conception préliminaire.. In: 12ÈMe Colloque National AIP PRIMECA. Le Mont Dore; 2011, p. 1–9.

[65] Blochwitz T, et al. Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In: Proceedings of the 9th international MODELICA conference, September 3-5, 2012, Munich, Germany, vol. 76. 2012, p. 173–84.

[66] Gomes C, et al. The FMI 3.0 standard interface for clocked and scheduled simulations. In: Proceedings of 14th modelica conference 2021, LinköPing, Sweden, September 20-24, 2021, vol. 181. 2021, p. 27–36.

[67] Hasmukhbhai Shah N, Le Henaff P, Schiffer C, Krammer M, Benedikt M. Accurate robot simulation for industrial manufacturing processes using FMI and DCP standards. In: Proceedings of 14th modelica conference 2021, LinköPing, Sweden, September 20-24, 2021, vol. 181. 2021, p. 673–9.

[68] Eklund M, Savolainen J, Lukkari A, Karhela T. Optimizing life-cycle costs for pumps and powertrains using fmi co-simulation. In: Proceedings of 14th modelica conference 2021, LinköPing, Sweden, September 20-24, 2021, vol. 181. 2021, p. 681–9.

[69] Wiens M, Meyer T, Thomas P. The potential of FMI for the development of digital twins for large modular multi-domain systems. In: Proceedings of 14th modelica conference 2021, LinköPing, Sweden, September 20-24, 2021, vol. 181. 2021, p. 235–40.

[70] Sun Y, Vogel S, Steuer H. Combining advantages of specialized simulation tools and modelica models using functional mock-up interface (FMI). In: Proceedings from the 8th international modelica conference, technical univeristy, Dresden, Germany. 2011, p. 491–4.

[71] Schranz T, Møldrup Legaard C, Tola D, Schweiger G. Portable runtime environments for Python-based FMUs: Adding docker support to UniFMU. In: Proceedings of 14th modelica conference 2021, LinköPing, Sweden, September 20-24, 2021, vol. 181. 2021, p. 419–24.

[72] Waterloo Maple Inc. MapleSim user's guide. Maplesoft; 2022, p. 266.

[73] Sieber J, Bernd K. Control based bifurcation analysis for experiments. Nonlinear Dynam 2008;51:365–77.

[74] Leine RI, Van Campen DH, De Kraker A, Van Den Steen L. Stick-slip vibrations induced by alternate friction models. Nonlinear Dynam 1998;16:41–54.

[75] Montemurro M, Pagani A, Fiordilino GA, Pailhès J, Carrera E. A general multiscale two-level optimisation strategy for designing composite stiffened panels. Compos Struct 2018;201:968–79.

[76] Montemurro M. An extension of the polar method to the first-order shear deformation theory of laminates. Compos Struct 2015;127:328–39.

[77] Izzi MI, Catapano A, Montemurro M. Strength and mass optimisation of variable-stiffness composites in the polar parameters space. Struct Multidiscip Optim 2021;64(4):2045–73.