Reliability Engineering and System Safety 94 (2009) 1149–1155

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

Screening important inputs in models with strong interaction properties

Andrea Saltelli, Francesca Campolongo *, Jessica Cariboni

European Commission, Joint Research Centre, 21020 Ispra, Varese, Italy

ARTICLE INFO

Article history:
Received 9 January 2008
Received in revised form
5 September 2008
Accepted 21 October 2008
Available online 5 November 2008

Keywords:
Screening design
Elementary effect method
Factorial design
Variance-based sensitivity measures

ABSTRACT

We introduce a new method for screening inputs in mathematical or computational models with large numbers of inputs. The method proposed here represents an improvement over the best available practice for this setting when dealing with models having strong interaction effects. When the sample size is sufficiently high the same design can also be used to obtain accurate quantitative estimates of the variance-based sensitivity measures: the same simulations can be used to obtain estimates of the variance-based measures according to the Sobol' and the Jansen formulas. Results demonstrate that Sobol' is more efficient for the computation of the first-order indices, while Jansen performs better for the computation of the total indices.

1. Introduction

We introduce a new method for screening inputs in mathematical or computational models with large numbers of inputs. In input screening one tries to identify active inputs in an input-rich model with a minimum number of computer simulations. The method proposed here competes with the best available practice for this setting, the method of elementary effects [1–4]. Reviews of screening methods for computer experiments are in [3,5], while review on sensitivity analysis in general are in [6–8] and [14]. Resources for sensitivity analysis are available in [9].

The idea of the new method is to explore the space of the model inputs by combining steps along the X_i axis, where X_i is one of the inputs, with steps along the $\mathbf{X}_{\sim i}$ axis, an axis where all inputs but X_i change their values. We anticipate that such an approach should be particularly efficient for inputs screening of non-additive models.

2. The method

A generic model (1) is considered:

$$Y = f(X_1, X_2, \ldots, X_k). \tag{1}$$

The new design is exemplified for a case where there are $k = 10$ model inputs and each input is explored over $l = 4$ different values or levels. The reason for these choices of k and l will be apparent in a moment. According to standard practice [2], the levels could correspond to equally spaced percentiles of the distribution of each input.

Table 1 shows how to arrange simulations so that steps in the $\mathbf{X}_{\sim i}$ direction are generated.

A step along the $\mathbf{X}_{\sim 1}$ axis is obtained comparing run B_1 with B_2, etc. as shown in Table 2. For example, a step along the $\mathbf{X}_{\sim 10}$ axis is obtained from B_1 and B_5 as all inputs but X_{10} change of levels between B_1 and B_5.

In the following r will indicate the number of rows in $\mathbf{B}$ and B_j a generic row of the same matrix. To obtain the complete design, the scheme of Table 2 is extended to include steps along the X_i axes as shown in matrix $\mathbf{M}$. The result is Table 3.

The first column in Table 3 indicates the simulation/run number. If we call N the total number of runs, we moved from $r = 5$ in $\mathbf{B}$ to $N = 25$ runs in $\mathbf{M}$. The original runs 1–5 have become in the order runs $1, 2, 5, 10, 17$. The additional runs have been generated as follows:

Run 3 is a step in the X_1 direction taken from run 1.
Run 4 is a step in the X_1 direction taken from run 2.
Run 5 is the base run 3 of matrix $\mathbf{B}$ while:
Run 6 is a step in the X_2 direction taken from run 2.
Run 7 is a step in the X_2 direction taken from run 5.
Run 8 is a step in the X_3 direction taken from run 1.
Run 9 is a step in the X_3 direction taken from run 5.
Run 10 the base run 4 of matrix $\mathbf{B}$.

Analogously:

Run 11 is a step in the X_4 direction taken from run 5.
Run 12 is a step in the X_4 direction taken from run 10.

* Corresponding author. Tel.: +390 332 78 5476; fax: +390 332 78 5733.
E-mail address: francesca.campolongo@jrc.it (F. Campolongo).

0951-8320/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ress.2008.10.007

Table 1
Base matrix **B**.

B_i	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
1	1	1	1	1	1	1	1	1	1	1
2	1	2	2	2	2	2	2	2	2	2
3	2	2	1	3	3	3	3	3	3	3
4	3	3	3	3	2	1	4	4	4	4
5	4	4	4	4	4	4	4	3	2	1

Run number and level selected for each input. The first column indicates the simulation (or 'run' as a simulation is termed in computer jargon) B_i from $i = 1$ to 5, X_j denotes input j and the tabled integer number indicates the level at which the input is set, e.g. input X_1 is set at level 3 in simulation B_4.

Table 2
Scheme for the computation of effects of the type $\mathbf{X}_{\sim i}$ from the runs of matrix **B**, Table 1.

$\mathbf{X}_{\sim i}$	Run	Run
1	2	1
2	3	2
3	3	1
4	4	3
5	4	2
6	4	1
7	5	4
8	5	3
9	5	2
10	5	1

Table 3
Matrix **M** and usage of runs to compute model inputs effect.

M_i	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	Type	For input
1	1	1	1	1	1	1	1	1	1	1	b	1, 3, 6, 10
2	1	2	2	2	2	2	2	2	2	2	b	1, 2, 5, 9
3	2	1	1	1	1	1	1	1	1	1	c1(1)	1
4	2	2	2	2	2	2	2	2	2	2	c2(1)	1
5	2	2	1	3	3	3	3	3	3	3	b	2, 3, 4, 8
6	1	3	2	2	2	2	2	2	2	2	c2(2)	2
7	2	3	1	3	3	3	3	3	3	3	c5(2)	2
8	1	1	2	1	1	1	1	1	1	1	c1(3)	3
9	2	2	2	3	3	3	3	3	3	3	c5(3)	3
10	3	3	3	3	2	1	4	4	4	4	b	4, 5, 6, 7
11	2	2	1	4	3	3	3	3	3	3	c5(4)	4
12	3	3	3	4	2	1	4	4	4	4	c10(4)	4
13	1	2	2	2	3	2	2	2	2	2	c2(5)	5
14	3	3	3	3	3	1	4	4	4	4	c10(5)	5
15	1	1	1	1	1	2	1	1	1	1	c1(6)	6
16	3	3	3	3	2	2	4	4	4	4	c10(6)	6
17	4	4	4	4	4	4	4	3	2	1	b	7, 8, 9, 10
18	3	3	3	3	2	1	3*	4	4	4	c10(7)	7
19	4	4	4	4	4	4	3*	3	2	1	c17(7)	7
20	2	2	1	3	3	3	3	4	3	3	c5(8)	8
21	4	4	4	4	4	4	4	4	2	1	c17(8)	8
22	1	2	2	2	2	2	2	2	3	2	c2(9)	9
23	4	4	4	4	4	4	4	3	3	1	c17(9)	9
24	1	1	1	1	1	1	1	1	1	2	c1(10)	10
25	4	4	4	4	4	4	4	3	2	2	c17(10)	10

Run type b, base run. Run type c$i(j)$, copy of run M_i with the level of input X_j increased of one step (with the exceptions highlighted by the * and explained below). There are $k = 10$ inputs and the number of base runs is $r = 5$ (see also formulas (2) and (3)). The total number of runs is five base runs plus $2k$ additional runs for a total of $r^2 = 25$ simulations. Each base run is cloned exactly five times. Each input is explored over four levels. The exception is marked with asterisk—the 3's in runs 18 and 19 should have been 5's. This type of exception occurs exactly twice regardless of number of model inputs k. Each base run is used for four different inputs. *Generalizing:* a number r of base runs can be used for $k = r(r-1)/2$ inputs explored over $l = (r-1)$ levels at a cost of $N = r^2$ simulations. Each base run is used for exactly $(r-1)$ inputs while each non-base (clone) run is used only for one input.

Run 13 is a step in the X_5 direction taken from run 2.
Run 14 is a step in the X_5 direction taken from run 10.
Run 15 is a step in the X_6 direction taken from run 1.
Run 16 is a step in the X_6 direction taken from run 10.
Run 17 the base run 5 of matrix **B**.

It should be clear how we have proceeded for the other runs.

Let us now discuss briefly the base matrix **B** and why we have chosen to exemplify the approach with $k = 10$ model inputs. In **B**, for a generic row or run number j we can estimate $(j-1)$ steps of the $\mathbf{X}_{\sim i}$ type, e.g. we compute the four steps $\mathbf{X}_{\sim 7}$ to $\mathbf{X}_{\sim 10}$ from row 5, three steps $\mathbf{X}_{\sim 4}$ to $\mathbf{X}_{\sim 6}$ from row 4 and so on. Thus the total number k of steps of the $\mathbf{X}_{\sim i}$ type which one can estimate with r simulations is simply

$$k = r - 1 + r - 2 + \cdots + 1 = \frac{r(r-1)}{2}. \tag{2}$$

Hence, for a total run number of $r = 3, 4, 5, 6, 7, 8, 9\ldots$ simulations we can compute steps of the $\mathbf{X}_{\sim i}$ type for $k = 3, 6, 10, 15, 21, 28, 36\ldots$ inputs, respectively, in matrix **B**. In other words, the number of matrix **B** simulations for a given number of inputs k is the solution of the quadratic equation (2) above, e.g.

$$r = \frac{1 + (1 + 8k)^{1/2}}{2}, \tag{3}$$

that has meaningful solutions for $k = 3, 6, 10, 15, 21, 28, 36\ldots$ inputs corresponding to $r = 3, 4, 5, 6, 7, 8, 9\ldots$ simulations in **B**, as just mentioned. In the matrix **B** given as example, $k = 10$ and $r = 5$, with four different levels to explore for each model input. It is easy to see that the number of levels explored is in general $l = r - 1$.

Note that the designs for $k = 3, 6$ are embedded into **B** or **M**. One just has to ignore the unneeded right-most columns. With $k = 6$, the dimension of matrix **B** would have been $r = 4$ and we would have explored conveniently three levels. Given that the present is a screening method, we anticipate that it will be applied to a k higher rather than lower than 10.

Note that in runs 18 and 19 we have moved levels from 4 to 3 instead than from 4 to 5, in order not to have isolated levels 5 explored. This technicality has to be kept in mind with different values of k as well to ensure that there will not be a level explored only twice in **M**. This exception will systematically occur twice in **M** irrespective of k.

To generalize the procedure for a number of inputs k, the first row of matrix **B** is obtained by setting all the inputs to their first level. Then the second row is obtained by leaving the first input as in the previous row (which is in level 1), and increasing all the others to level 2. The third row is obtained by leaving the second input as in the previous row (which is in level 2), setting the first input to the same level (level 2), decreasing the third input to level 1, and increasing all the others to level 3. The fourth row is obtained by leaving the fourth input as in the previous row (in level 3), setting the preceding inputs (first, second and third) to the same level (level 3), decreasing the fifth input to level 2, the sixth to level 1, and increasing all the others to level 4.

In general the n-th row is obtained by fixing the first input which in the previous row has taken the level $(n-1)$. All inputs preceding that one are set to the same level (i.e. $(n-1)$). The subsequent inputs decrease from $(n-2)$ to 1. The remainders are increased by one level with respect to the previous row.

The matrix **M** is obtained expanding **B** as it follows. We consider the first two rows of **B**, say B_1 and B_2, and select the input which is kept fixed among the two, input X_1. We create a row which is a copy of B_1 apart from input X_1 which is increased to the next level, and a row which is a copy of B_2 apart from input X_1 which is increased to the next level. We then consider the second

and third two rows of **B**, B_2 and B_3, and select the input which is kept fixed among the two, input X_2. We create a row which is a copy of B_2 apart from input X_2 which is increased to the next level, and a row which is a copy of B_3 apart from input X_2 which is increased to the next level. We proceed by comparing rows of **B** to find couples of rows, B_i and B_j, having a single input X_w which is kept fixed among the two (i.e. which is set to the same level). We create new rows that are copies, respectively, of B_i and B_j apart from input X_w which is increased by one level. This completes the matrix.

Given that our design in matrix **M** needs two additional simulations for each model input to compute steps along the X_i axis, the total numbers of simulations is the solution of Eq. (3) plus twice the number of model inputs k. Hence $N = r + 2r(r-1)/2 = r^2$. In the case of matrix **M** where $r = 5$ this makes $N = 25$.

In the simulation matrix **M** each integer number is meant to indicate the level at which a given model input is set. Thus, if using **M** in the present form, each input would start moving from its lowest level and would finish reaching its highest level. In order to randomize the inputs' path, we generate for each input X_i a vector $\mathbf{L}_i$ containing the l levels in randomized order, under the condition that the difference between two consecutive levels cannot be the maximum value of the range of variation of an input (this is simply to avoid a 'too big' step which covers the whole interval amplitude, moving from 0 to 1 in the space of percentiles). Then for each model input i, we assume that the integer number j in **M** identifies the level that occupies the j-th position in $\mathbf{L}_i$. If, for example, the value in **M** for X_1 is 3, then X_1 will be set to the level $\mathbf{L}_i(3)$.

Finally note that it is possible to repeat the (entire) design a number i of times by replicating the random choice of the levels to be assigned to each input. The cost of such an experiment is thus $C = Ni$. In this case it is also possible to randomly permute model inputs at each iteration, so that the order in which inputs are considered changes randomly from one iteration to the other. In the single iteration presented in Table 3 inputs are considered in the order $X_1, X_2, \ldots, X_k$.

Having illustrated the sample design, we now move to discuss how effects are estimated. Let us refer to input X_1 in matrix **M** to exemplify our approach.

Let us call Y_i the model value computed on the i-th row of **M**. We start with run M_3 which is simply a copy of run M_1 where input X_1 has been moved from level (1) to (2), and run M_4 which is a copy of run M_2 were again input X_1 has been moved from level (1) to (2). Thus, indicating with $\text{level}_z(j)$ the value that input X_z takes on level j both

$$EE_a(1) = \frac{|Y_1 - Y_3|}{|\text{level}_1(2) - \text{level}_1(1)|}$$

and

$$EE_b(1) = \frac{|Y_2 - Y_4|}{|\text{level}_1(2) - \text{level}_1(1)|}$$

can be taken as estimate of the so-called main effect of X_1 (see Fig. 1).

The elementary or first-order effect EE_1 of X_1 can be estimated via

$$EE_1 = \tfrac{1}{2}(EE_a(1) + EE_b(1)). \quad (4)$$

In the literature the above measure is often named μ^*. In the following we will adopt the common terminology and we will call it μ^*. It is also clear that both the couple of runs 1, 2 and 3, 4 provide an effect of moving all inputs but X_1. Following the formula used in the factorial design (see [3, p. 55]), we pose that

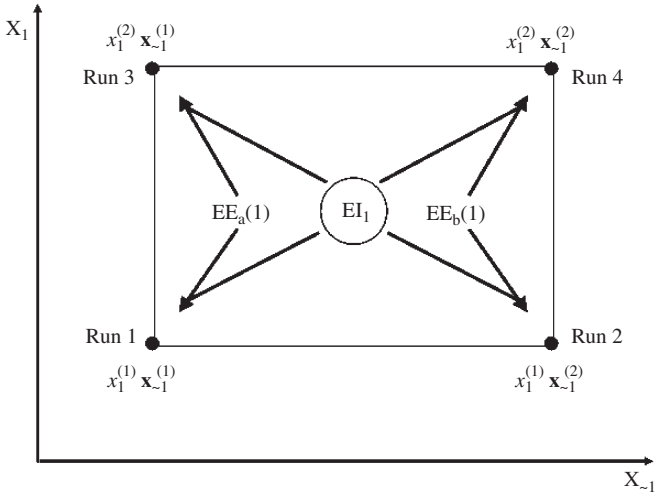

Fig. 1. Computational scheme for the effects of input X_1 in matrix **M**, Table 3. Note that effects can be obtained by comparing points both horizontally and vertically.

Table 4
Number of inputs k and total number of simulations N corresponding to a given column dimension of matrix **B**.

r	Number of inputs k	Number of simulations N
3	3	9
4	6	16
5	10	25
6	15	36
7	21	49
8	28	64
9	36	81
10	45	100
11	55	121
12	66	144
13	78	169
14	91	196
15	105	225
16	120	256
17	136	289
18	153	324
19	171	361
20	190	400

These simulations serve to estimate a complete set of main effects and interaction effects for all inputs.

an estimate of the interaction effect of input X_1 can be gauged by

$$EI_1 = \tfrac{1}{2}|Y_1 - Y_2 + Y_4 - Y_3|. \quad (5)$$

As mentioned the present approach favors number of inputs such as $k = 3, 6, 10, 15, 21, 28, 36 \ldots$ and the corresponding costs are in Table 4. These numbers have to be compared with the cost of the elementary effects methods [1,4].

When using the elementary effects method a number of trajectories t is launched into the space of the model inputs. In each trajectory all steps are taken along the X_i axes (no movements over the $\mathbf{X}_{\sim i}$ axes) and only elementary effects are computed using systematically adjacent steps of the trajectory. Within each trajectory, composed of $(k+1)$ lattice points, one model input at a time is changed till all k dimensions have been spanned. Each trajectory thus generates exactly k elementary effects, i.e. one elementary effect per input. The cost C of this procedure is thus the number of trajectories, t, times the number of inputs plus one, $C = t(k+1)$. Values of t are usually taken in the range of $t = 4, \ldots, 10$.

In the following we will show that our design can be convenient with respect to the elementary effects method.

The idea is that, with the same computational cost, and still maintaining the classic sensitivity measure based on the elementary effects, our method can provide the additional sensitivity measure presented in formula (5) which accounts for the interaction effects, and which can be particularly useful for models having pure interactions.

3. Experimental results

We test the new design against two benchmarks for sensitivity analysis, and we compare its performance with that of the elementary effects method. The two benchmarks are, respectively: (i) a typically non-additive model [5], and (ii) a 10-inputs version of the Morris' function [1], the standard benchmark for the elementary effects method.

The non-additive model, here taken with 10 model inputs, has the following form:

$$Y = \sum_{i=1}^{5} \Omega_i Z_i, \tag{6}$$

where

$$\begin{aligned} &Z_i \sim N(\bar{z}_i, \sigma_{Z_i}), \\ &\Omega_i \sim N(\bar{\Omega}_i, \sigma_{\Omega_i}), \end{aligned} \quad i = 1, 2, \ldots, 5, \tag{7}$$

with $\bar{z}_i = 0$, $i = 1, 2, \ldots, 5$, $\bar{\mathbf{\Omega}} = [0.5, 0.5, 0.5, 2, 2.5]$, $\boldsymbol{\sigma}_{\mathbf{Z}} = [1, 1, 0.5, 2, 2]$, and $\boldsymbol{\sigma}_{\mathbf{\Omega}} = [1, 1, 4, 5, 6]$.

The vector of model inputs for the analysis is thus

$$\mathbf{X} = (Z_1, Z_2, \ldots, Z_5, \Omega_1, \Omega_2, \ldots, \Omega_5). \tag{8}$$

Note that this model is appealing for sensitivity analysis testing as it has the peculiarity that some model inputs have effects of first order that are null, but at the same time a non-negligible overall importance because of their interaction effects.

Given that the input distributions are not uniform, in order to select the inputs' levels it is essential to decide how to cut the distribution tails (so to avoid exploration of 'extreme events'). The choice of the cut has a strong influence on the results as it defines a different region of the space to be explored. Here we have repeated the experiment twice: a first time by performing a 'light' cut, thus including more 'extreme events' (tails cut at 0.5 and 99.5), a second time by performing a more solid cut which explores most likely events (tails cut at 20 and 80).

The experiment is performed with a total computational cost of $C = 25$, and is confronted with the elementary effects design performed with two trajectories that correspond to $C = 22$ model evaluations. The comparison is carried out according to the following criterion. Each experiment is replicated 50 times and the number of 'failures' for each sensitivity measure are counted. As we are in a screening framework, we aim to avoid type II errors, i.e. to erroneously miss an important input. Thus we classify as a 'failure' for a given measure a replicate where one or more important inputs are not identified as important according to that measure. The threshold to identify a model input as important is the average value of that sensitivity measure taken over all inputs. A model input whose measure is above the average is important. A model input whose measure is below the average is not.

This criterion is subject to a certain level of arbitrariness and different rules to screen unimportant model inputs can be tested. For models where analytical values of the sensitivity indices are available, also ranks can be used to compare the performance of different methods. However, for most of the models where screening techniques are employed, analytical sensitivity measures are not available and thus it is necessary to define a threshold for discriminating influential inputs based only on the comparison of the sensitivity measures obtained in the experiment.

Table 5
Results for function (6) with tails cut at 20 and 80.

New design	$\mu*$ alone	14
	$\mu*$ + int	3
Elementary effects	$\mu*$ alone	16
	$\mu* + \sigma$	9

The most right column counts the number of failures.

Table 6
Results for function (6) with tails cut at 0.5 and 99.5.

New design	$\mu*$ alone	30
	$\mu*$ + int	15
Elementary effects	$\mu*$ alone	24
	$\mu* + \sigma$	24

The most right column counts the number of failures.

Tables 5 and 6 show the results. When looking at the sole sensitivity measure μ^*, we find that both the methods produce a non-negligible number of failures, especially for the more difficult case where extreme sample points are explored. However, when using the new design and introducing the measure of the interactions given in Eq. (5), the number of failures reduces, respectively, to 15 for the setting with a 'light' cut and to 3 for the setting with a 'strong' cut. Also the classic elementary effects method can make use of a sensitivity measure that estimate the interactions and that could reduce the number of failures of μ^*: the standard deviation of the elementary effects (see [1]). However, at very low sample size, this measure tends to become meaningless (here it represents the standard deviation of a sample of only two elementary effects!). In our first example (Table 5) its use allows to improve results, as it reduces the number of failures from 16 to 9; however, the improvement is much less pronounced than that obtained using the interaction measure given in Eq. (5). In the second example (Table 6) its added value is completely null.

For this test case, where the interactions play a fundamental role, it is clear that the new design is more convenient than the classic elementary effects one, as it offers the possibility to compute a sensitivity measure specifically designed for the interactions which can significantly reduce the occurrence of type II error made by μ^*. This is more evident when the test case also takes into account extreme events that enhance even more the role of the interaction terms.

To show that results are not depending on the number of inputs chosen, the experiment is repeated on the same test function but with a different number of inputs: 28. The new function has the form

$$Y = \sum_{i=1}^{14} \Omega_i Z_i, \tag{9}$$

where

$$\begin{aligned} &Z_i \sim N(\bar{z}_i, \sigma_{Z_i}), \\ &\Omega_i \sim N(\bar{\Omega}_i, \sigma_{\Omega_i}), \end{aligned} \quad i = 1, 2, \ldots, 14, \tag{10}$$

with $\bar{z}_i = 0$, $i = 1, 2, \ldots, 14$, $\bar{\mathbf{\Omega}} = [0.5, 0.5, 0.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7]$,

$\boldsymbol{\sigma}_{\mathbf{Z}} = [1,1,2,2,2,3,3,3,2,2,1,1,6,7]$ and
$\boldsymbol{\sigma}_{\boldsymbol{\Omega}} = [1,1,0.5,0.5,0.5,0.5,3,3,4,4,5,2,6,7]$.

Results, reported in Table 7, confirm our conclusions.

We now carry out the same comparison exercise on another benchmark for sensitivity analysis, the polynomial function introduced by Morris [1], to show the efficiency of the elementary effects method. Here we take a 10-input version of the Morris' function of the following form:

$$y = \beta_0 + \sum_{i=1}^{10} \beta_i w_i + \sum_{i<j}^{10} \beta_{i,j} w_i w_j + \sum_{i<j<l}^{10} \beta_{i,j,l} w_i w_j w_l, \tag{11}$$

where $w_i = 2(x_i - 0.5)$ for all i except for $i = 3$ where $w_i = 2(1.1x_i/(x_i + 0.1) - 0.5)$, $x_i \sim U[0,1]$, and

$\beta_i = -15, \quad i = 1,2,$
$\beta_{i,j} = 30, \quad i = 3,4,$
$\beta_{i,j,l} = 10, \quad i = 1,2,3,4.$

The remaining first- and second-order coefficients are independently generated from a standard normal distribution. The remaining third-order coefficients are set to zero. Note that the choice of $k = 10$ is for comparison reasons: the number $k = 10$ translates into a number of levels $l = 4$, which is the standard choice when applying the elementary effects method.

Table 8 shows the results for different computational costs, i.e. for different numbers t of trajectories and different numbers i of iterations of the new design. When the sample size is extremely low (i.e. $t = 2$, which corresponds to a single iteration, $i = 1$, of our design), both methods fail several times. This is due to the difficulty of this test case at a so low sample size. The previous example had a lower number of failures as the difference in importance among the inputs was more pronounced and hence the correct ranking for importance more detectable. When testing higher sample sizes, both methods converge at the correct numbers, but the relative performance of the two methods is unstable.

This test case shows that when the presence of interactions is not so relevant, the benefit of using the new design disappears. In general we believe that, in a screening setting, the method proposed here has the potential for an added value with respect to the standard 'best practice' based on the classic elementary effects method. The test cases presented here show that the method is convenient when the model presents pure interaction terms but it may be less performing on other test cases.

Table 7
Results for function (9) and with tails cut at 20 and 80.

New design	$\mu*$ alone	11
	$\mu*$ + int	5
Elementary effects	$\mu*$ alone	11
	$\mu* + \sigma$	7

The most right column counts the number of failures.

Table 8
Results for function (11) at different sample sizes.

	$i = 1$ $t = 2$	$i = 2$ $t = 4$	$i = 2$ $t = 5$
New design			
$\mu*$ alone	37	12	12
$\mu*$ + int	21	2	2
Elementary effects			
$\mu*$ alone	23	5	4
$\mu* + \sigma$	17	4	1

Numbers indicate the number of failures.

4. Comparison with a quantitative method

There are points of similarities between the approach presented in this paper and that of Saltelli [10], which is an improvement of the Sobol' method [11] for the computation of the so-called variance based measures.

A variance based first-order effect for a given model input X_i can be written as

$$V_{X_i}(E_{\mathbf{X}_{\sim i}}(Y|X_i)). \tag{12}$$

The meaning of the inner expectation operator is that the mean of Y is taken over all possible values non-X_i, i.e. over all possible values of $\mathbf{X}_{\sim i}$, while keeping X_i fixed. The outer variance is taken over all possible values of X_i.

The associated sensitivity measure (first-order sensitivity coefficient) is written as

$$S_i = \frac{V_{X_i}(E_{\mathbf{X}_{\sim i}}(Y|X_i))}{V(Y)}. \tag{13}$$

Because of relation

$$E_{X_i}(V_{\mathbf{X}_{\sim i}}(Y|X_i)) + V_{X_i}(E_{\mathbf{X}_{\sim i}}(Y|X_i)) = V(Y) \tag{14}$$

S_i will always be between zero and unit. It measures the first-order (e.g. additive) effect of X_i on the model.

Another popular variance based measure is the total effect term or index [10,12]:

$$S_{Ti} = \frac{E_{\mathbf{X}_{\sim i}}(V_{X_i}(Y|\mathbf{X}_{\sim i}))}{V(Y)}. \tag{15}$$

S_{Ti} measures the total effect (both first order and higher, e.g. interactions) of input X_i.

Variance based methods are quantitative so that these measures are not simply estimated over levels of the inputs but over the entire inputs distribution using customarily Monte Carlo methods of various sophistication. Sampling matrices will be designed to allow the computation of measures (13) and (15).

Following Saltelli [10] we imagine to have two independent sampling matrices $\mathbf{A}$ and $\mathbf{B}$. We can write $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{X}_1^{(a)}, \ldots, \mathbf{X}_i^{(a)}, \mathbf{X}_{i+1}^{(a)}, \ldots, \mathbf{X}_k^{(a)}, \tag{16}$$

where each term is a column vector of length H (the number of simulations in the matrix). For simplicity of notation this equation can be written as

$$\mathbf{A} = \mathbf{X}_i^{(a)}, \mathbf{X}_{\sim i}^{(a)}, \tag{17}$$

where $\mathbf{X}_i^{(a)}$ will be a column vector of length H and $\mathbf{X}_{\sim i}^{(a)}$ will be a matrix of dimension H times $(k-1)$.

Analogously the second matrix $\mathbf{B}$ will be

$$\mathbf{B} = \mathbf{X}_i^{(b)}, \mathbf{X}_{\sim i}^{(b)}, \tag{18}$$

and two more matrices can be constructed as

$\mathbf{A_b} = \mathbf{X}_i^{(a)}, \mathbf{X}_{\sim i}^{(b)},$
$\mathbf{B_a} = \mathbf{X}_i^{(b)}, \mathbf{X}_{\sim i}^{(a)}.$

The theory of variance based measures is based on computing S_i from either the couple of matrices $\mathbf{A}, \mathbf{A_b}$ or $\mathbf{B}, \mathbf{B_a}$ [11]. Analogously S_{Ti} can be computed either using the couple $\mathbf{A}, \mathbf{B_a}$ or matrices $\mathbf{B}, \mathbf{A_b}$ (see Table 9).

Saltelli [10] noted that all what is needed to compute both the set of all S_i and S_{Ti} for the k inputs is the triplet of matrices $\mathbf{A}, \mathbf{B}, \mathbf{B_a}$ or alternatively the triplet matrices $\mathbf{A}, \mathbf{B}, \mathbf{A_b}$. If we use e.g. the former triplet we can compute all S_i and S_{Ti} using the scheme in Table 10.

Table 9
Possible ways of computing S_i and S_{Ti}.

Matrix	Matrix	Effect
$\mathbf{X}_i^{(a)}, \mathbf{X}_{\sim i}^{(a)}$	$\mathbf{X}_i^{(a)}, \mathbf{X}_{\sim i}^{(b)}$	S_i
$\mathbf{X}_i^{(a)}, \mathbf{X}_{\sim i}^{(a)}$	$\mathbf{X}_i^{(b)}, \mathbf{X}_{\sim i}^{(a)}$	S_{Ti}
$\mathbf{X}_i^{(b)}, \mathbf{X}_{\sim i}^{(b)}$	$\mathbf{X}_i^{(b)}, \mathbf{X}_{\sim i}^{(a)}$	S_i
$\mathbf{X}_i^{(b)}, \mathbf{X}_{\sim i}^{(b)}$	$\mathbf{X}_i^{(a)}, \mathbf{X}_{\sim i}^{(b)}$	S_{Ti}

Table 10
How to compute S_i and S_{Ti} according to [10].

Matrix	Matrix	Effect
$\mathbf{X}_i^{(a)}, \mathbf{X}_{\sim i}^{(a)}$	$\mathbf{X}_i^{(a)}, \mathbf{X}_{\sim i}^{(b)}$	S_i
$\mathbf{X}_i^{(b)}, \mathbf{X}_{\sim i}^{(b)}$	$\mathbf{X}_i^{(a)}, \mathbf{X}_{\sim i}^{(b)}$	S_{Ti}

Table 11
Generic bi-dimensional square in the hyper-lattice.

Point	Point	Effect
$x_i^{(j)}, \mathbf{x}_{\sim i}^{(j)}$	$x_i^{(j)}, \mathbf{x}_{\sim i}^{(j+1)}$	S_i
$x_i^{(j)}, \mathbf{x}_{\sim i}^{(j)}$	$x_i^{(j+1)}, \mathbf{x}_{\sim i}^{(j)}$	S_{Ti}
$x_i^{(j+1)}, \mathbf{x}_{\sim i}^{(j)}$	$x_i^{(j+1)}, \mathbf{x}_{\sim i}^{(j+1)}$	S_i
$x_i^{(j)}, \mathbf{x}_{\sim i}^{(j+1)}$	$x_i^{(j+1)}, \mathbf{x}_{\sim i}^{(j+1)}$	S_{Ti}

$(x_i^{(j)}, \mathbf{x}_{\sim i}^{(j)})$ and $(x_i^{(j+1)}, \mathbf{x}_{\sim i}^{(j+1)})$ represent the first and last point of the square built for input X_i; $(x_i^{(j+1)}, \mathbf{x}_{\sim i}^{(j)})$ is the point where input i moves respect to the first point (step along direction i); $(x_i^{(j)}, \mathbf{x}_{\sim i}^{(j+1)})$ is the point where all but input i change their values respect to the first point (step along direction $\sim i$).

Note that $2H$ simulations are needed for computing Y corresponding to matrices $\mathbf{A}, \mathbf{B}$ while kH simulations are needed to compute Y for matrix $\mathbf{B_a}$ for all inputs. As a result the cost of this quantitative method is $C = H(k+2)$ with H usually a large number (1000 or higher).

This has similarities with the approach taken in the present paper, where the computation of the sensitivity effects are based on 'squares' in the hyper-lattice such as Table 11. For example, the square for input X_1 in Table 3 is in Fig. 1.

Following this idea, we tried to explore the properties of our design when moving from a screening setting to a quantitative setting, where the aim is to produce accurate sensitivity measures that converge to analytical values.

We stick to the same sampling design, i.e. we perform the same simulations, but we use them differently: we arrange the simulation outcomes so to estimate the variance-based sensitivity measures according to the classic formulas proposed, respectively, by Sobol' and by Jansen (see [13]). Moreover, instead of mapping matrix **M** onto levels, we map them onto a sequence of Sobol' points.

Thus, we estimate S_i and S_{T_i} either as

$$S_i = \frac{E[f(X_i, \mathbf{X}_{\sim \mathbf{i}}) f(X_i, \mathbf{X}'_{\sim \mathbf{i}})] - E^2[Y]}{V}, \tag{19}$$

$$ST_i = 1 - \frac{E[f(X_i, \mathbf{X}_{\sim \mathbf{i}}) f(X'_i, \mathbf{X}_{\sim \mathbf{i}})] - E^2[Y]}{V} \tag{20}$$

according to the formula of Sobol', or as

$$S_i = \frac{V - \frac{1}{2} E[f(X_i, \mathbf{X}_{\sim \mathbf{i}}) - f(X_i, \mathbf{X}'_{\sim \mathbf{i}})]^2}{V}, \tag{21}$$

$$ST_i = \frac{\frac{1}{2} E[f(X_i, \mathbf{X}_{\sim \mathbf{i}}) - f(X'_i, \mathbf{X}_{\sim \mathbf{i}})]^2}{V} \tag{22}$$

according to the formula of Jansen.

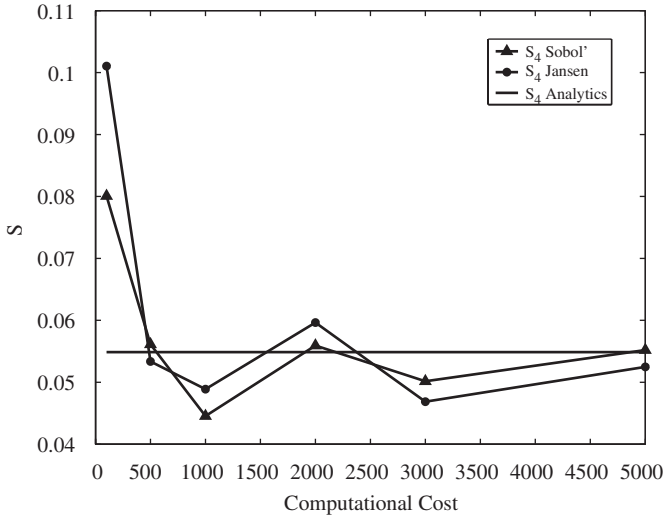

Fig. 2. S_i values for X_4 as computed via the Sobol' and the Jansen formulas at different computational costs. The estimates are confronted with the analytical value.

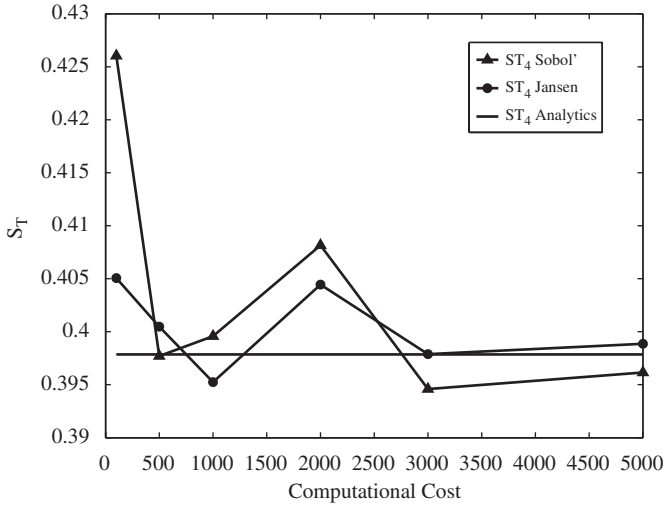

Fig. 3. ST_i values for X_4 as computed via the Sobol' and the Jansen formulas at different computational costs (number of simulations). The estimates are confronted with the analytical value.

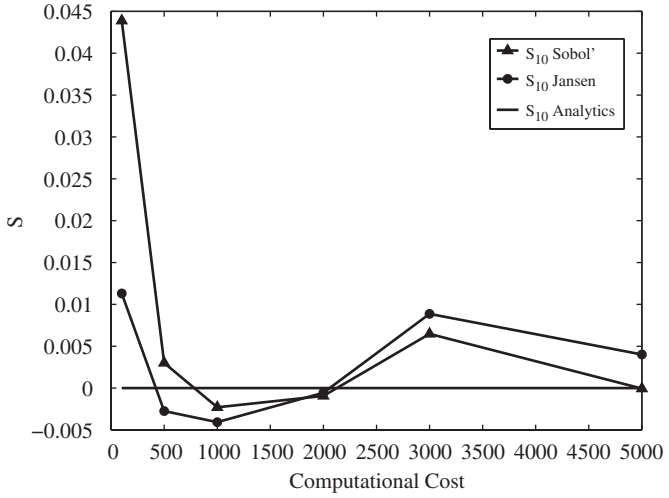

Fig. 4. S_i values for X_{10} as computed via the Sobol' and the Jansen formulas at different computational costs (number of simulations). The estimates are confronted with the analytical value.

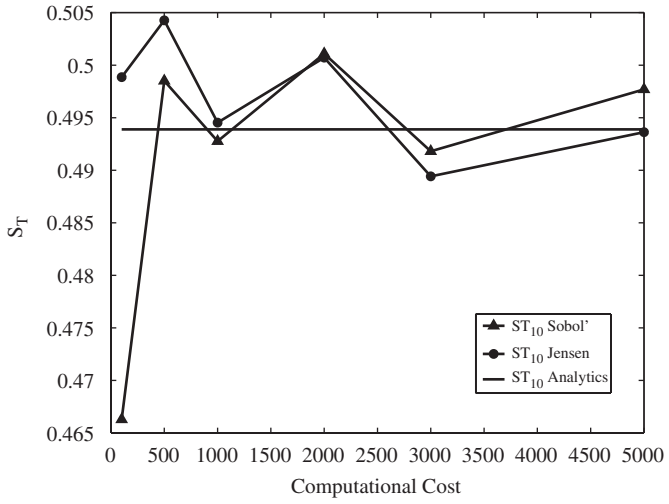

Fig. 5. ST_i values for X_{10} as computed via the Sobol' and the Jansen formulas at different computational costs (number of simulations). The estimates are confronted with the analytical value.

Results for function (6) are shown in Figs. 2–5 that show the convergence of the sensitivity indices, both first order and total, as estimated, respectively, using the Sobol' and the Jansen formulas. The graphs show that all the estimates converge to the analytical values, confirming that the proposed design has the advantage of being suitable also for quantitative estimates, provided that the sampling size is sufficiently large. Moreover, it is worth noting that, as conjectured by Chan et al. [13], the Sobol' formula is more effective when computing the first-order indices (as it converges quickly to the analytical values), while the Jansen formula is more efficient for the computation of the total indices.

5. Conclusions

In this paper we have proposed an alternative method to screen model inputs in input-rich models. With respect to the elementary effects method, so far a 'best practice' for screening settings, the new method has the advantage of providing an additional sensitivity measure, at no extra computational cost, which estimates the interaction effects and that can be particularly useful for models characterized by pure interactions terms. When modified to perform in quantitative settings, the design provides sensitivity estimates of the variance-based indices that converge to the analytical values.

References

[1] Morris MD. Factorial sampling plans for preliminary computational experiments. Technometrics 1991;33(2):161–74.
[2] Campolongo F, Tarantola S, Saltelli A. Tackling quantitatively large dimensionality problems. Computer Physics Communication 1999;117:75–85.
[3] Campolongo F, Kleijnen J, Andres T. Screening methods. In: Saltelli A, Chan K, Scott M, editors. Sensitivity analysis. New York: Wiley; 2000. p. 65–80.
[4] Campolongo F, Cariboni J, Saltelli A. Simplifying a large chemical reaction model via an effective screening design. Environmental Modelling & Software 2007;22:1509–18.
[5] Saltelli A, Ratto M, Andres T, Campolongo F, Cariboni J, Gatelli D, et al. Global sensitivity analysis. The primer. England: Wiley; 2008.
[6] Saltelli A, Tarantola S, Campolongo F, Ratto M. Sensitivity analysis in practice. In: A guide to assessing scientific models. New York: Wiley; 2004.
[7] Saltelli A, Ratto M, Tarantola S, Campolongo F. Sensitivity analysis for chemical models. Chemical Reviews 2005;105(7):2811–28.
[8] Helton JC, Johnson JD, Sallaberry CJ, Storlie CB. Survey of sampling based methods for uncertainty and sensitivity analysis. SANDIA Report SAND2006-2901; 2006.
[9] Sensitivity analysis web at JRC ⟨http://sensitivity-analysis.jrc.ec.europa.eu/⟩.
[10] Saltelli A. Making best use of model valuations to compute sensitivity indices. Computer Physics Communications 2002;145:280–97.
[11] Sobol' IM. Sensitivity estimates for nonlinear mathematical models. Matematicheskoe Modelirovanie 1990;2:112–8 [in Russian. translated in English as Sobol' IM, Sensitivity analysis for non-linear mathematical models. Mathematical Modelling and Computational Experiment 1993;1:407–14].
[12] Homma T, Saltelli A. Importance measures in global sensitivity analysis of model output. Reliability Engineering and System Safety 1996;52(1):1–17.
[13] Chan K, Saltelli A, Tarantola S. Winding stairs: a sampling tool to compute sensitivity indices. Statistics and Computing 2000;10:187–96.
[14] Saltelli A, Chan K, Scott M, editors. Sensitivity analysis. Probability and statistics series. New York: Wiley; 2000.