Design for sensitivity analysis, in Chapman and Hall "Handbook of Design of Experiments"

Andrea Saltelli, William Becker
Joint Research Centre,
Institute for the Protection and Security of the Citizen, Ispra

March 12, 2012

Abstract

The present review of design methods for sensitivity analysis is limited to a number of best practices selected by the authors as particularly suited to settings often encountered in sensitivity analysis. The case where a full quantitative analysis is needed will be distinguished from from that where a simple screening of the factors into influential and non-influential is sought – though methods shall be provided to move smoothly from the latter to the former.

For cases where the analyst has control over the design, e.g. where she can decide where to locate her points, two practices are recommended for computing respectively the first order sensitivity measure and the total sensitivity measure:

- The first practice in known as random balance design (RBD), and belongs to the class of Fourier analysis approaches [44]. RBD allows the first order sensitivity measure – also known as Pearson's correlation ratio η^2 to be computed.

- The second practice is richer in information but computationally more expensive. This involves the computation of the total sensitivity measures [15]. A quasi-random numbers based approach is recommended, in conjunction with best available estimators [37].

Both the first and total order sensitivity measures can also be computed using emulators, when few model samples are available (usually due to computational expense). The total sensitivity measure can also be adapted at low sample size and with a different estimator to work in a screening setting [6].

Finally, the case is discussed where points are "given" – e.g. where the analyst has to derive sensitivity measures with potentially arbitrarily-placed sample points (either from simulation or experiment). For this latter situation an alternative approach is shown which uses smoothing, spline, or kernel regression to estimate first order sensitivity measures [26].

Keywords: Sensitivity analysis, radial design, variance based measures, elementary effects method

1 Introduction

Sensitivity analysis is the study of how uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input [35]. Sometimes the term is also used to indicate simply the quantification of the uncertainty in model's prediction, although strictly speaking this is the closely-related discipline of *uncertainty analysis*. In general, sensitivity analysis is used to test the robustness of model-based inference, i.e. how much the results of the model depend on the assumptions made in its construction. In engineering and risk analysis, sensitivity analysis mostly involves an exploration of the multidimensional space of the input factors/assumptions. In econometrics, sensitivity analysis has been advocated more in the form of extreme bound analysis, e.g. using confounding factors in regression which either confirm the most or the least the inference [18, 19]. Very often, in chemistry, physics, biology and so on, one sees sensitivity analysis performed by changing one factor at a time, a practice which is not recommended [31]. Instead, current best practice involves designs based on a multidimensional exploration of the space of the input factors, as in classic experimental design. A succinct review of sensitivity analysis methods for use in impact assessment – e.g. in relation to models used for policy, is in [33].

In this chapter, the term "model" refers to a mathematical construct which attempts to model some physical, economic or other "real-world" process. Sensitivity analysis is applicable to any system that has quantifiable inputs and outputs. From this point of view, one can consider any model from the "black-box" perspective, such that it is an unknown function $f(\boldsymbol{x})$ of k inputs, where $\boldsymbol{x} = \{x_i\}_{i=1}^{k}$. The model will typically return a large number of output quantities, but for simplicity it shall be assumed that the output is a scalar y, such that $y = f(\boldsymbol{x})$. Note that although the $\boldsymbol{x}$ and y

will often appear as random variables, they will always be expressed in lower case.

Although the function (model) f is known, in the sense that it represents a computer program based on mathematical equations, it will generally be complex enough as to be only accessible via simulation. Therefore in practice, all sensitivity analysis approaches involve sampling the model a number of times according to some kind of experimental design, and estimating useful properties from the resulting data. Two main cases arise in this respect:

Case I The analyst can 'run' the model. A design can be specified in this case whereby e.g. n points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ are selected in the k-dimensional input space, to obtain corresponding outputs $\{y_1, y_2, ..., y_n\}$. In this case the sample of the input space is customarily generated without correlation among the input factors. Designs for correlated inputs are also available. y could represent some modeled property of a design such as an aeroplane wing or of a natural system such as a hydraulic transmissivity.

Case II The sample points are given and the analyst can neither control their positioning nor generate additional points. Such data might come either from measurements or experiments, or from a design that is not specifically intended for sensitivity analysis. The form of the model could be unknown, and the input variables in the sample could be correlated with one another in the sample. To give a simple example, y could be the Human Development Index computed over k countries and the x_i could be the indicators used in the construction of the index [26]. In this case one cannot generate additional points/countries.

In Case I (when the analysis can be designed) the best approach is determined by the cost of the analysis. In this context, "cost" refers to the total computational time required to evaluate the model at all the sample points, which is the product of the total number of model runs and the time required for each run. Since complex models can take minutes, hours or longer to evaluate for a single input point, it is not always feasible to sample a large number of input points. The strategies available for case I are as follows:

- For cheap models, a fully-fledged quantitative sensitivity analysis can be performed using Monte Carlo estimators, estimating all k first order indices and all k total order indices (see Section 3). This approach requires a large number of sample points (typically hundreds or thousands per input variable), but is preferred where possible since all sensitivity indices can be estimated with an accuracy related to the number of sample points.

- For expensive models a design based on Fourier analysis can be used to compute all first order indices at a cost which is weakly dependent from the number of factors. The cost is of the order of some hundreds model simulations. Alternatively, a space-filling design can be used in conjunction with an emulator (see Section 6). Although computationally cheaper, emulators introduce a data-modelling problem, which can be very difficult to deal with depending on the nature of the variation of the data.

- If one cannot afford more than a handful of points per factor, or one has many input factors and desires to proceed to a first screening of factors into influential and non-influential groups, the elementary effects methods can be applied (see Section 5).

For Case II, when data is given, two approaches are considered here:

- Generate additional points by emulation and then perform a Monte Carlo sensitivity analysis by the simultaneous computation of all k

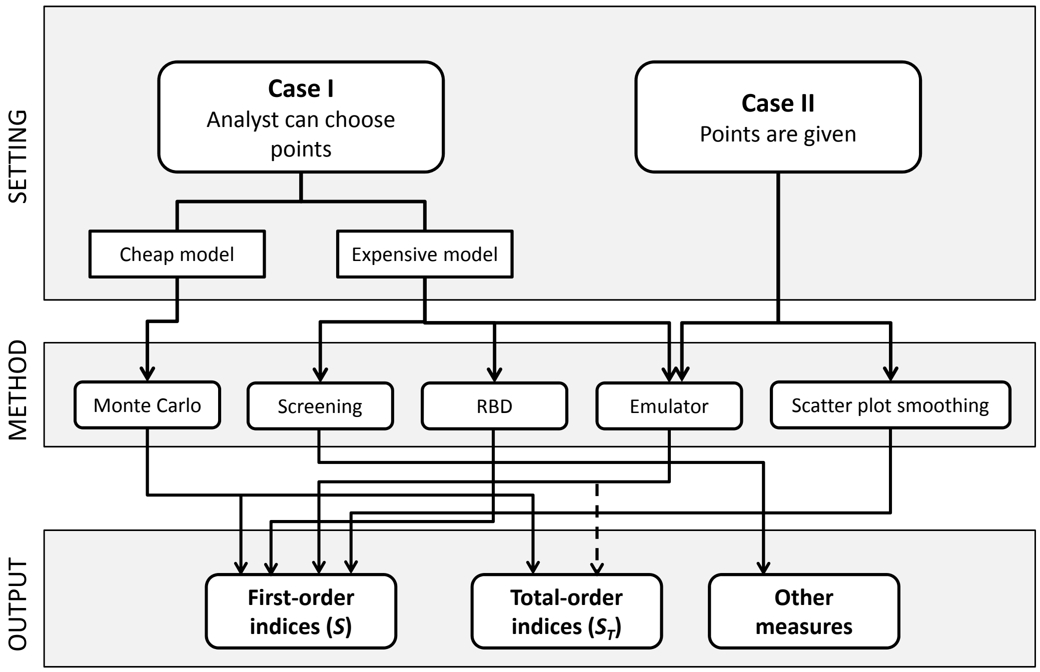

Figure 1: Various approaches to sensitivity analysis: when they can be used, and what they produce.

first order indices and all k total order indices (depending on method – see Section 6).

- Estimate directly the k first order indices by kernel regression (or equivalent regression approach) on the sorted model evaluations y_j's (see Section 6). In effect, this involves making one-dimensional scatter plots of y against each x_i, then fitting (nonlinear) trend lines. In simple problems, even a visual inspection of scatter plots may be useful.

The various approaches discussed here and the context in which they can be applied are summarised in Figure 1. Note that in the present chapter only three measures of sensitivity are proposed:

1. First order sensitivity index (see Section 2)
2. Total order sensitivity index (see Section 2)
3. Elementary effects (see Section 5)

The following section gives a brief description of variance-based sensitivity analysis, which underpins measures (1) and (2) above.

2 Variance-based sensitivity analysis

Many measures of sensitivity have been proposed in the literature. For example, a well-known measure is to regress the data against each x_i, and take the Pearson product moment correlation coefficient R_i^2 values to measure correlation. An obvious drawback of this is that linear regression can only meaningfully interpret linear data. While this approach can be extended by more sophisticated forms of regression, it is preferable not to rely on any modelling of a functional relationship between y and $\boldsymbol{x}$, since unwanted assumptions would thus be introduced.

Variance-based approaches have become very popular in recent years, since they allow for highly nonlinear model responses, and account for variations over the full input space.

A useful sensitivity measure for a given factor x_i is:

$$V_{x_i}\left(E_{\boldsymbol{x}_{\sim i}}\left(y \mid x_i\right)\right) \tag{1}$$

The meaning of the inner E operator is the expected value of y taken over all possible values of variables other than x_i (i.e. over $\boldsymbol{x}_{\sim i}$), while keeping x_i fixed (conditional mean). The outer V is the variance taken over all possible values of x_i.

The associated sensitivity measure (first order sensitivity coefficient) is stated as:

$$S_i = \frac{V_{x_i}\left(E_{\boldsymbol{x}_{\sim i}}\left(y \mid x_i\right)\right)}{V(y)} \tag{2}$$

Formula 2 has a long history and was first introduced by Karl Pearson under the name of correlation ratio, denoted as η^2. Note that η^2 is precisely the extension to nonlinear problems of the linear R_i^2 measure and the two measures coincide for linear problems.

In sensitivity analysis proper, S_i is the first term in a variance decomposition whereby the unconditional variance $V(y)$ is decomposed as the sum of a set of conditional variances of first, second, $\cdots$, up to the k^{th} order [39]. Such a decomposition holds only if the input factors x_i's are independent.

$$V(y) = \sum_i V_i + \sum_i \sum_{j>i} V_{i,j} + ... + V_{1,2,...,k} \tag{3}$$

where:

$$V_i = V(f_i(x_i)) = V_{x_i}[E_{\boldsymbol{x}_{\sim i}}(y|x_i)]$$

$$V_{i,j} = V(f_{i,j}(x_i, x_j)) = V_{x_i,x_j}(E_{\boldsymbol{x}_{\sim i,j}}(y|x_i, x_j))$$
$$-V_{x_i}(E_{\boldsymbol{x}_{\sim i}}(y|x_i)) - V_{x_j}(E_{\boldsymbol{x}_{\sim j}}(y|x_j))$$

and so on for the higher order terms. The reduced dimensionality terms f_i, $f_{i,j}$ are themselves linked by a functional decomposition analogous to 3:

$$f = f_0 + \sum_i f_i + \sum_i \sum_{j>i} f_{i,j} + ... + f_{1,2,...,k} \tag{4}$$

The first order terms $f_i = E_{\boldsymbol{x}_{\sim i}}(y|x_i) - E(y)$ shall be returned to in Section

6. Dividing all terms in 3 by $V(y)$:

$$\sum_i S_i + \sum_i \sum_{j>i} S_{i,j} + \ldots + S_{1,2,3,\ldots,k} = 1. \tag{5}$$

Computing all terms in 3 is impractical for larger k given that they number $2^k - 1$ in total. For this reason the total order sensitivity index S_T [15] is preferred, which measures the total effect of a factor, including its first order effect and interactions of any order:

$$S_{Ti} = \frac{E_{\boldsymbol{x}_{\sim i}}\left(V_{x_i}\left(y \mid \boldsymbol{x}_{\sim i}\right)\right)}{V(y)} = 1 - \frac{V_{\boldsymbol{x}_{\sim i}}\left(E_{x_i}\left(y \mid \boldsymbol{x}_{\sim i}\right)\right)}{V(y)} \tag{6}$$

where $\boldsymbol{x}_{\sim i}$ denotes the matrix of all variables but x_i. In $E_{\boldsymbol{x}_{\sim i}}\left(V_{x_i}\left(y \mid \boldsymbol{x}_{\sim i}\right)\right)$ the inner variance V of y, the scalar output of interest, is taken over all possible values of x_i while keeping $\boldsymbol{x}_{\sim i}$ fixed, while the output expectation E is taken over all possible values $\boldsymbol{x}_{\sim i}$ [14].

It is straightforward to see that $E_{\boldsymbol{x}_{\sim i}}\left(V_{x_i}\left(y \mid \boldsymbol{x}_{\sim i}\right)\right)$ is the main effect of $\boldsymbol{x}_{\sim i}$, or in other words, the sum of the main effects and interactions of all the variables in $\boldsymbol{x}_{\sim i}$. Since all sensitivity indices sum to 1 (see 5), the remainder must be equal to all terms *not* involving $\boldsymbol{x}_{\sim i}$, i.e. the main effect of x_i and any interactions involving it.

In the next section the design and estimation recipes for the the cases detailed in Section 1 are described.

3 Using design points - quantitative sensitivity analysis

First, the situation is described where the analyst has full control over the placement of input points (Case I) and the model is not expensive to estimate, i.e. where possibly thousands of model runs can be executed without difficulty. In this case the use of quasi-random numbers is suggested, specifically the LP_τ sequences of Sobol' [38, 41] (also known simply as Sobol' sequences) coupled with a Monte Carlo design described in [30, 37]. Note, however, that this approach is also valid with pseudo-random numbers and other low-discrepancy sequences - see [24] for a summary of many common approaches.

It is assumed that all random variables $x_1, x_2, \ldots, x_k$ are sampled in the k-dimensional unit hypercube $\mathcal{X}$;

$$\boldsymbol{x} \in \mathcal{X} : \; \mathcal{X} = [0,1]^k \tag{7}$$

Different distributions can easily be generated by mapping the points in (7) onto the desired distribution function (uniform, normal, log-normal, etc). This involves the inversion of the cumulative (target) distribution function, i.e. solving for x the equation $\zeta = \int_{-\infty}^{x} g(\eta) d\eta$, where ζ is the number in $[0,1]$ from (7) and g is the desired distribution function [34].

The use of quasi-random sequences is motivated by their good space filling properties; these sequences outperform both pseudo-random Monte Carlo sampling as well as Latin Hypercube Sampling in the estimation of multi-dimensional integrals [40]. Recent extensive testing with a large batch of functions spanning different degrees of dimensionality, linearity and additivity has demonstrated their suitability for sensitivity analysis [17]. An additional desirable property of LP_τ sequences when compared to LHS is that with the former, additional points can be added sequentially to the the analysis till a desired target accuracy is achieved. With the latter, the sample size cannot be extended once the analysis is performed, without starting again from the beginning.

An example of the first few rows of a three-dimensional sequence are given in Table 2. These sequences can be generated using freely available software both in FORTRAN, and MatLab (see [9]).

The steps needed to estimate a full set of first order and total order sensitivity indices via the Monte Carlo method are as follows (see Figure 2 for an illustration of the construction of the matrices):

1. Generate n points of a $2k$-dimensional LP_τ sequence. Take the first k columns and call these coordinates $\mathbf{A}$. Take the remaining k columns and call these $\mathbf{B}$. In this way, a total of N rows of the sequence have been used. The generic coordinates of $\mathbf{A}$ and $\mathbf{B}$ can be indicated respectively as $x_{ji}^{(a)}$ and $x_{ji}^{(b)}$. The index i runs from one to k, the number of factors, while the index j runs from one to n, the number of rows.

2. Generate an additional k matrices $\mathbf{A}_B^i$, such that the i^{th} matrix is entirely composed of coordinates from $\mathbf{A}$ but for its i^{th} column, which is the i^{th} column of $\mathbf{B}$. A total of $k+2$ sets of coordinates (matrices) have thus been generated. See figure 2 for an illustration.

3. To estimate S_i one needs to estimate first $V_{x_i}(E_{\boldsymbol{x}_{\sim i}}(y \mid x_i))$. Coordinates from $\mathbf{B}$ and $\mathbf{A}_B^i$ are used as follows:

$$V_{x_i}(E_{\boldsymbol{x}_{\sim i}}(y|x_i)) \approx \frac{1}{n}\sum_{j=1}^{n} f\left(\mathbf{B}\right)_j f\left(\mathbf{A}_B^i\right)_j - f_0^2, \tag{8}$$

   where $f\left(\mathbf{B}\right)_j$ indicates values of Y computed from a generic row j of matrix $\mathbf{B}$ and $f\left(\mathbf{A}_B^i\right)_j$ indicates values of y computed from a generic row j of matrix $\mathbf{A}_B^i$. f_0, the sample mean, and $V(y)$, the unconditional variance, are computed from matrix $\mathbf{A}$ alone.

4. For S_{Ti} one needs to estimate first $V_{\boldsymbol{x}_{\sim i}}\left(E_{x_i}\left(y \mid \boldsymbol{x}_{\sim i}\right)\right)$. This can be obtained using the couple $\mathbf{A}, \mathbf{A}_B^i$:

$$LP_\tau(4,3) = \left[\underbrace{\begin{matrix} 0.500 & 0.500 & 0.500 \\ 0.250 & 0.750 & 0.250 \\ 0.750 & 0.250 & 0.750 \\ 0.125 & 0.625 & 0.875 \end{matrix}}_{\mathbf{A}} \; \underbrace{\begin{matrix} 0.500 & 0.500 & 0.500 \\ 0.750 & 0.250 & 0.750 \\ 0.250 & 0.750 & 0.250 \\ 0.875 & 0.625 & 0.125 \end{matrix}}_{\mathbf{B}}\right]$$

$$\mathbf{A}_B^{(1)} = \begin{matrix} 0.500 & 0.500 & 0.500 \\ 0.750 & 0.750 & 0.250 \\ 0.250 & 0.250 & 0.750 \\ 0.875 & 0.625 & 0.875 \end{matrix}$$

$$\mathbf{A}_B^{(2)} = \begin{matrix} 0.500 & 0.500 & 0.500 \\ 0.250 & 0.250 & 0.250 \\ 0.750 & 0.750 & 0.750 \\ 0.125 & 0.625 & 0.875 \end{matrix}$$

$$\mathbf{A}_B^{(3)} = \begin{matrix} 0.500 & 0.500 & 0.500 \\ 0.250 & 0.750 & 0.750 \\ 0.750 & 0.250 & 0.250 \\ 0.125 & 0.625 & 0.125 \end{matrix}$$

Figure 2: Construction of the **A**, **B** and $\mathbf{A}_B^i$ matrices, using the LP$_\tau$ sequence with $k = 3$ and $N = 4$. Grey columns correspond to those taken from the matrix **B**.

$$V_{\boldsymbol{x}_{\sim i}}\left(E_{x_i}\left(y \mid \boldsymbol{x}_{\sim i}\right)\right) \approx \frac{1}{n}\sum_{j=1}^{n} f\left(\mathbf{A}\right)_j f\left(\mathbf{A}_B^i\right)_j - f_0^2, \tag{9}$$

with a similar meaning of symbols as above.

Note that each matrix $\mathbf{A}_B^i$ is used twice for each factor x_i, once to compute S_i and once to compute S_{Ti}. An explanation of estimators 8, 9 can be found in [34]. For the reader it will be straightforward to notice that the estimators make use of sums of products of function values, and that in each product the two function values being multiplied by one another have some symmetry. In the case of S_i the two functions have identical values for coordinate x_i. In the case of S_{Ti} the two functions have identical values for all coordinates but x_i. Take the case of S_i for illustration: if x_i is influential, then the two functions values being multiplied will resemble one another, high values being multiplied by high values and low values by low values. If x_i is non influential, high and low values will be randomly coupled, resulting in a lower value for the estimator for S_i.

For even better results with Sobol' sequences (due to the deterioration of uniformity in higher dimensions, as discussed in [37]), estimator (8) can be substituted with:

$$V_{x_i}(E_{\boldsymbol{x}_{\sim i}}(y|x_i)) \approx \frac{1}{n}\sum_{j=1}^{n} f\left(\mathbf{B}\right)_j \left(f\left(\mathbf{A}_B^i\right)_j - f\left(\mathbf{A}\right)_j\right) \tag{10}$$

and (9) with:

$$E_{\boldsymbol{x}_{\sim i}}\left(V_{x_i}\left(y \mid \boldsymbol{x}_{\sim i}\right)\right) \approx \frac{1}{2n}\sum_{j=1}^{n}\left(f\left(\mathbf{A}\right)_j - f\left(\mathbf{A}_B^i\right)_j\right)^2 \tag{11}$$

$\widehat{S_i}$ and $\widehat{S_{Ti}}$ are obtained by dividing equations 10 and 11 respectively by $V(y)$.

To show how the Monte Carlo estimators above perform at different n, consider a simple polynomial example,

$$y = 3x_1^2 + 2x_1x_2 - 2x_3; \tag{12}$$

where the coefficients have been chosen quite arbitrarily. Figure 3 shows the scatter plots of each variable. It is evident that x_1 has quite a strong, slightly nonlinear effect on y. x_2 has apparently quite a weak effect (there is little discernable trend), whereas x_2 has a slight negative effect. These trends are clearly reflected in the coefficients of equation (12) – of course, normally one would not have the coefficients of an analytical equation to examine.

To estimate the sensitivities of the variables, a Sobol' design is created

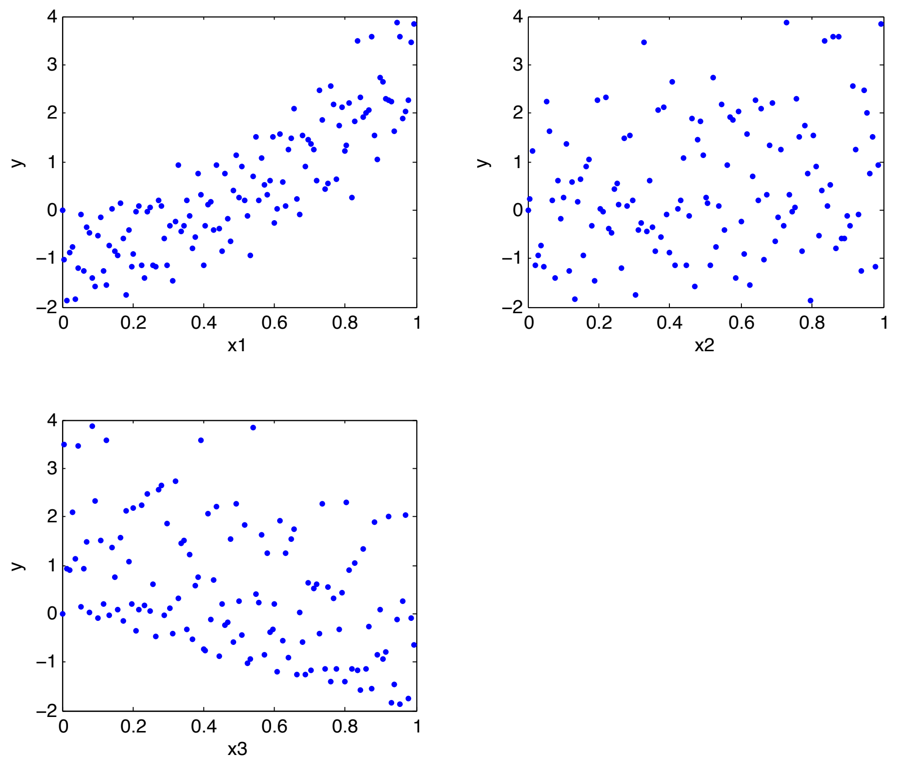

Figure 3: Scatter plots of the variables in the test equation (12).

Variable	S_i (MC)	S_i (analytic)	S_{Ti} (MC)	S_{Ti} (analytic)
x_1	0.7517	0.7568	0.7781	0.7720
x_2	0.0503	0.0456	0.0604	0.0608
x_3	0.1870	0.1824	0.1829	0.1824

Table 1: Comparison of S_i estimates from local-linear kernel regression of polynomial function against analytical values

in 3 dimensions, assuming a joint-uniform distribution for simplicity, and estimators (10) and (11) are used. The only choice is what value of n, the number of sample points, to use. Given that the Sobol' sequence allows sequential addition of new points, one can start with a small number of points, then gradually increase until convergence is observed. Figure 4 shows the convergence of these measures with n ranging from 8-4096. It is evident that the estimators converge quite quickly to an accurate estimate of the sensitivity indices; even at the lowest n, the variables are already correctly sorted, and at $n \geq 128$ the indices have converged to two decimal places. For most applications of sensitivity analysis, this would be sufficient accuracy. Table 3 shows the results at $n = 128$ compared to analytical values. Note that due to the weak interaction between x_1 and x_2, the S_T of these variables is slightly higher than their respective S values, which reflects a portion of the variance being attributed to $S_{1,2}$ (not estimated here, though it can be deduced from the table, noticing that x_3 does not interact).

Despite the flexibility of Monte Carlo estimators, one should remember that the cost is $N(k + 2)$ runs (see again figure 2). While this is fine for fast models, for large models it may be impractical. In the following sections some alternative approaches are discussed that have lower computational requirements.

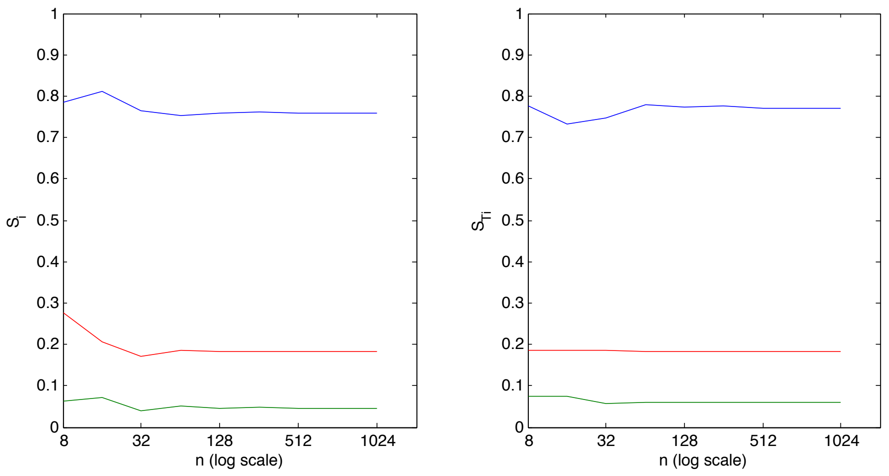

Figure 4: Convergence of the S_i and S_{Ti} of the polynomial equation (12) with increasing n. Lines represent, from top to bottom, x_1, x_3 and x_2 respectively.

4 Using design points - FAST and the Random Balance Design

Random Balance Designs (RBD) in sensitivity analysis make use of an approach known as the Fourier Amplitude Sensitivity Test (FAST) [10, 11]. FAST uses the Fourier series to represent a multivariate function in the frequency domain, using a single frequency variable s. Therefore, the integrals required to calculate sensitivity indices become univariate, resulting in computational savings. Basis (transformation) functions are used of the form,

$$x_i = G\left(\sin(\omega_i s)\right) \tag{13}$$

where s is a variable in $[-\pi, \pi]$, G is a specified transformation function, and ω_i is an integer, representing the fundamental frequency of x_i. Given appropriate choices of the constants and basis functions (mainly to avoid interferences between harmonics of variables), the variance is given as,

$$\begin{aligned} V(y) &= \tfrac{1}{2\pi}\int_{-\pi}^{\pi}\left(f^2(s)ds - E(y)^2\right)ds \\ &\approx 2\sum_{n=1}^{\infty}\left(a_\omega^2 + b_\omega^2\right) \end{aligned} \tag{14}$$

where

$$\begin{aligned} a_\omega &= \tfrac{1}{\pi}\int_{-\pi}^{\pi}\left(f(x)\cos\omega x\right)dx \\ b_\omega &= \tfrac{1}{\pi}\int_{-\pi}^{\pi}\left(f(x)\sin\omega x\right)dx \end{aligned} \tag{15}$$

The integrals in (15) can be evaluated numerically from the training points (e.g. Monte Carlo integration), which involves sampling from s; indirectly, this is sampling from $\boldsymbol{x}$, since each x_i will vary according to its basis function, therefore a "search curve" is created across the sample space. The oscillation of the x_i translates into an oscillation of the model output y at the fundamental frequency ω_i of each variable. If the amplitude of y at the fundamental frequency of an input variable is high, it indicates a high sensitivity to that variable – this provides the basis for sensitivity analysis. The sensitivity measure is written as,

$$\hat{V}_{\omega i} = 2\sum_{p=1}^{M}\left(a_{p\omega_i}^2 + b_{p\omega_i}^2\right) \tag{16}$$

where p is an integer, representing the pth harmonic of the fundamental frequency. When divided by the FAST estimate of variance, $\hat{V}_{\omega i}$ has been shown to be equivalent to the first order sensitivity index, S_i.

To complete the specification one must choose a basis function G. It is necessary to consider that the search curve produced by the basis function should sample as uniformly as possible over $\mathcal{X}$. In other words, uniformly sampling over s should produce a low-discrepancy sequence in $\boldsymbol{x}$. Several transformations have been proposed that have this property - see [36] for some examples. The resulting design gives an oscillating search curve over

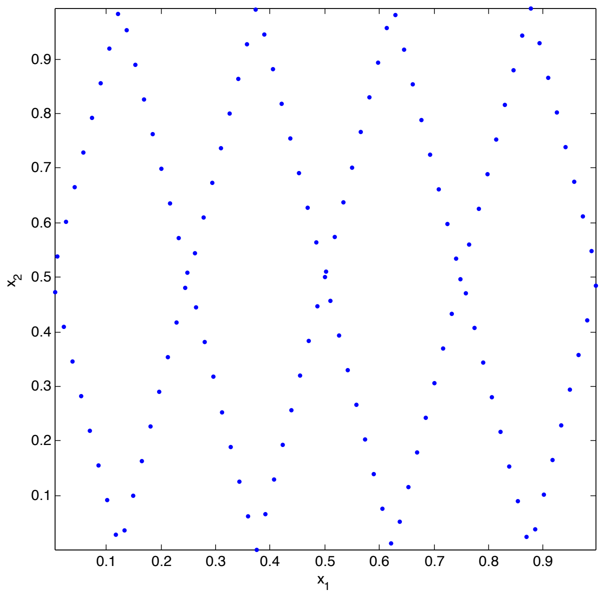

Figure 5: A 2D example of a FAST search curve using a triangular basis function [36] and with $\omega_1 = 1$ and $\omega_2 = 4$.

the design space. An example using a triangular basis function (possessing good uniformity) is shown in figure 5.

Since different frequencies must be used to investigate each variable, without interference, the computational cost of FAST quickly rises with k (albeit less than with the Monte Carlo approach), because higher frequencies are required, which require a greater density of points to represent. The RBD approach to FAST [44] circumvents this to some extent by using a single frequency ω for all inputs. At this point, the points will be very poorly distributed in the sample space. RBD takes random permutations of the coordinates of these points, generating a set of sample points that is roughly equivalent to a Latin hypercube design, which is known to have reasonably good space-filling properties. Figure 6 shows an illustration of this process before and after the scrambling of coordinates. After running the model, the

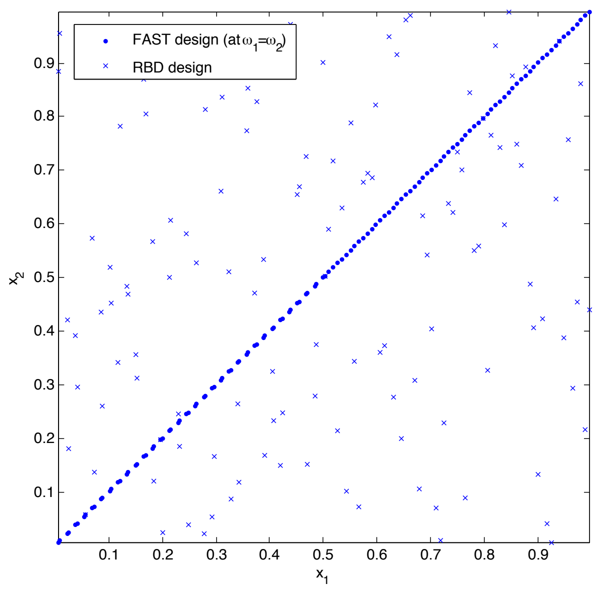

Figure 6: A 2D example of a RBD sample before and after scrambling of coordinates, with $\omega_1 = \omega_2 = 1$.

sample points can be reordered with respect to each input, and the FAST measures discussed above can be calculated.

5 Using design points - Screening

For cases where the model is expensive to run and one cannot use the emulators described in section 6, screening methods offer a computationally-efficient way of identifying influential and non-influential variables. Typically this will be used to weed out uninfluential variables before applying a more informative analysis to the remaining set.

The most common approach, suggested by Morris [22], is an extension of a basic sensitivity tests, the one-at-a-time (OAT) design. Like the OAT design, the method of Morris involves varying one input at a time, but the design follows a “winding stairs” pattern, such that an input is varied by a given amount Δ_i, then the next input is varied, keeping the new value of the first input, and so on for all inputs, giving a trajectory of $k + 1$ points. This is repeated R times at random starting points within a predefined grid. Figure 7 shows an example in three dimensions. Sensitivity is calculated for each input as the mean of R elementary effects,

$$EE_{i,r} = \frac{f(x_1, ..., x_{i-1}, x_i + \Delta_i, x_{i+1}, ..., x_k) - f(x_1, ..., x_k)}{\Delta_i} \tag{17}$$

where r is the index over trajectories. The screening measure is thus expressed as,

$$\mu_i = \frac{\sum_r^R EE_{i,r}}{r} \tag{18}$$

A further useful measure of nonlinearity and interaction is given by the variance σ_i of the elementary effects,

$$\sigma_i = \frac{\sum_r^R (EE_{i,r} - \mu_i)^2}{r} \tag{19}$$

The logic here is that if the response of the output to a given input were perfectly linear, the elementary effects would be identical anywhere in the input space; in the nonlinear case the opposite would be true.

A drawback with the sensitivity measure given in (18) is that if the main effect of an input is non-monotonic, the average of the elementary effects may be close to zero even though, individually, they are significant positive and negative values. The result is that the measure μ_i could potentially miss influential variables. A modified measure μ^*, proposed in [4] suggests the use of the modulus of the elementary effects, i.e.

$$\mu_i^* = \frac{\sum_r^R |EE_{i,r}|}{r} \tag{20}$$

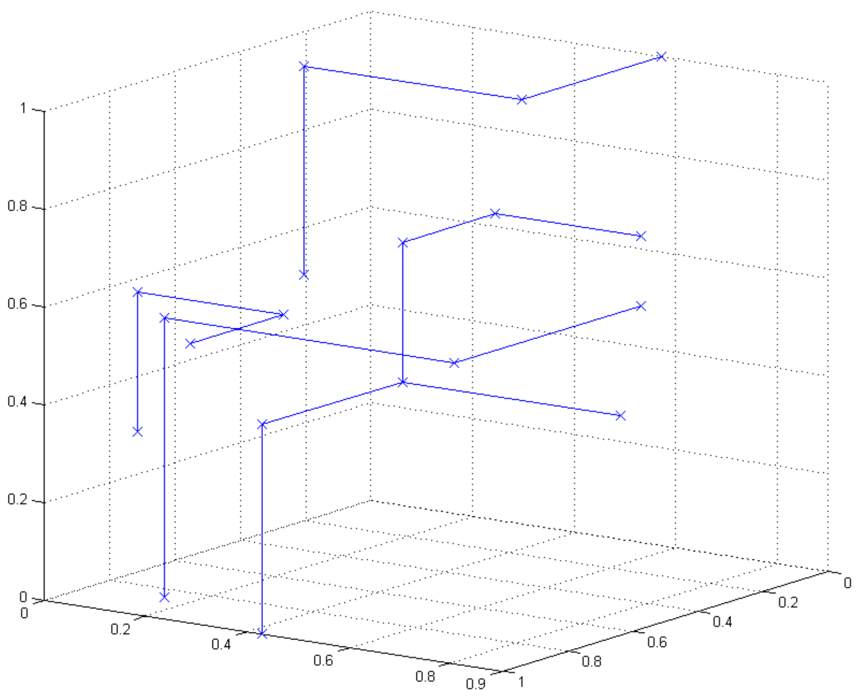

Figure 7: A trajectory screening design in three dimensions, with $R = 5$.

This measure has been shown to summarize both the μ_i and σ_i^2 measures, and has a close correlation with ST, the global variance-based measure discussed in Section 2 [5].

Improvements have also been made to the sampling strategy. The drawback of the winding stairs design is that there is no guarantee that the trajectories are well-spaced, and that the input space has been well-explored given the number of runs. A glance at the design in Figure 7 shows that points can sometimes be close to one another, therefore inefficiently exploring $\mathcal{X}$. An alternative implementation of this design that is suggested here as a best practice uses a so-called "radial" configuration based on Sobol's LP$_\tau$ sequence to achieve a screening design with well-spaced trajectories [7].

To construct the radial design, an LP$_\tau$ sequence in $2k$ dimensions is generated. Let the first k columns be called the "baseline points" $\{\boldsymbol{a}_j\}_{j=1}^k$, and the remaining k columns be the "auxiliary" points $\{\boldsymbol{b}_j\}_{j=1}^k$. For a given baseline and auxiliary point, a radial configuration of $k+1$ points is constructed as the following (discarding the row index for clarity),

$$\begin{array}{c} a_1, a_2, a_3, ..., a_k \\ b_1, a_2, a_3, ..., a_k \\ a_1, b_2, a_3, ..., a_k \\ a_1, a_2, b_3, ..., a_k \\ \vdots \\ a_1, a_2, a_3, ..., b_k \end{array}$$

where the a_i and b_i are the ith elements of the the $\boldsymbol{a}$ and $\boldsymbol{b}$ vectors respectively. Since in LP$_\tau$ designs the values of coordinates tend to repeat, it is recommended that the $\boldsymbol{a}$ and $\boldsymbol{b}$ points for a given radial configuration are not taken from the same row, otherwise there will be no perturbation in some dimensions and numerical errors. In practice it has been found that pairing

$\boldsymbol{a}_j$ and $\boldsymbol{b}_{j+4}$ gives good results [7]. This means that for a design of R radial configurations, one needs an LP$_\tau$ sequence of $R+4$ points in $2k$ dimensions, translating to a computational cost of $R(k+1)$ runs. An example of a radial screening design is given in Figure 9. To show the capability, this exact design is used to estimate μ_i^* measures of the polynomial example from equation 12. With $R=5$ and $d=3$, the cost is 20 model runs. The analysis returns,

$$\begin{aligned}\mu_1^* &= 3.2875\\ \mu_2^* &= 0.6500\\ \mu_3^* &= 2.0000\end{aligned}$$

Note that these values do not have an exact meaning with regard to the variance of the output (as compared to the S measures), but they allow the user to sort between influential and uninfluential variables. Much like the results in Table 3, one can see that variable 2 is relatively unimportant compared to variables 1 and 3. Consider that this example is trivial, since screening is generally for use with high-dimensional problems, but even with 20 runs the order and to some extent the magnitude of importance of each variable can be distinguished.

Note that an advantage of the radial design is that, if required (and if possible), the number of points can be increased until the points are dense enough to be used with the estimators for S and ST, discussed in Section 2, since the design is exactly equivalent. This fits naturally with the progression from a preliminary screening design to a more sophisticated global sensitivity analysis, making efficient use of all model runs.

6 Emulator Approaches and Smoothing

To estimate sensitivity measures when data points are given (the placement of points is arbitrary), two general methods are presented here, both of which

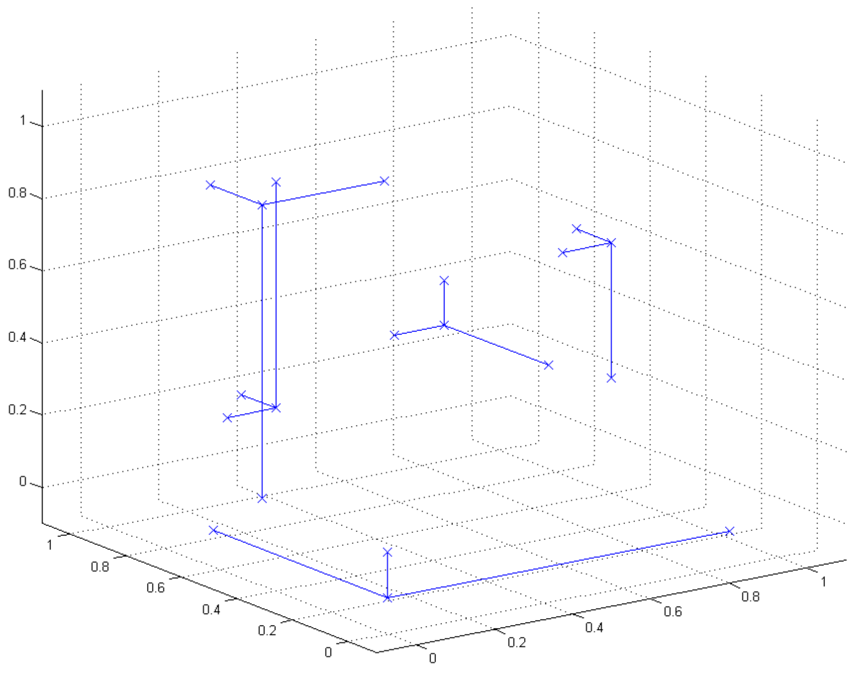

Figure 8: A radial screening design in three dimensions, with $R = 5$.

adopt a data modelling approach. The first is to try to fit an emulator (a relatively simple mathematical function, also known as a *metamodel*) to the data, which behaves in the same way as the model itself. The emulator can then be used to estimate new points (with the intention of performing a Monte Carlo estimation as in Section 3). An alternative approach is to project the data onto a single axis x_i (i.e. create k one-dimensional scatter plots) and attempt to infer the main effect $E(y|x_i)$ using a smoothing regression approach, for example kernel regression. Both methods have advantages and drawbacks, which will be summarised.

Emulators

The central idea of emulation is to find some relatively simple function η (the emulator) such that $\eta(\boldsymbol{x}) \approx f(\boldsymbol{x})$. If η is considerably cheaper to run than the original model, but produces very similar results for any $\boldsymbol{X} \in \Omega$, then it can be used to generate a very large number of points that can estimate sensitivity indices via Monte Carlo methods. Even better, if η is analytically tractable, it can be used to analytically evaluate sensitivity integrals, therefore bypassing the need for Monte Carlo completely.

The two issues associated with emulation are,

- First, an appropriate function (type of emulator) is needed to fit to the data.
- Second, parameters or hyperparameters associated with the model need to be estimated, i.e. the emulator must be *trained*.

A large number of competing methods are available to tackle both problems (see for example [2]). A comparison of some of these methods in the context of sensitivity analysis can be found in [42]. Once an appropriate emulation method is selected, the accurate estimation of parameters can be achieved provided that a sufficiently-large sample of training data (the given points) is available. How large this sample needs to be is somewhat dependent on the type of emulator, but strongly related to the number of input variables (the so-called curse of dimensionality). For this reason, emulator-based approaches are suited to situations with few input variables (perhaps less than thirty, depending on the emulator). Higher-dimensionality problems can sometimes be brought into the reach of emulators by a precursory screening analysis to reduce the number of factors.

While there are many emulators available in the literature, only a small subset is described here for illustration. Many of these methods rely on a technique known as High-Dimensional Model Representation (HDMR), which seeks to approximate the model by performing a functional decomposition into orthogonal terms, then truncating the series. This has already been given in equation 4, and is restated as:

$$\begin{aligned} f(\boldsymbol{x}) &= f_0 + \sum_i f_i(x_i) + \sum_i \sum_{j>i} f_{i,j}(\boldsymbol{x}_{i,j}) + ... + f_{12...k}(\boldsymbol{x}_{1,2,...,k}) \\ &\approx f_0 + \sum_i f_i(x_i) + \sum_i \sum_{j>i} f_{i,j}(\boldsymbol{x}_{i,j}) \end{aligned} \tag{21}$$

if the series is truncated after the second term. The task then remains to find suitable orthogonal functions to approximate the $f_i(x_i)$ and $f_{i,j}(\boldsymbol{x}_{i,j})$.

One approach that has been used with considerable success is the use of multivariate smoothing splines [28]. A smoothing spline model assumes that the underlying function to be emulated is continuous, and has a continuous first derivative, further that the second derivative is square-integrable. This corresponds to the function belonging to a second order Sobolev space. The smoothing spline estimate arises from considering the function g that minimises the following,

$$\frac{1}{n}\sum_{i=1}^{n}\{y_i - g(x_i)\}^2 + \lambda\int_0^1 \{g''(x)\}^2 dx \tag{22}$$

The first term in this “tradeoff” expression is simply the sum of squared error between the training data and the emulator g - if the function were to pass through every data point (exact interpolation), this term would be zero. The second term expresses the integral of the second derivative of g, which is a global measure of roughness. λ is a parameter that controls the weighting between the two terms. Overall therefore, the expression summarizes the tradeoff between interpolation and model simplicity. The tuning parameter can be set to give more weight to one or the other of these, and can be chosen via cross-validation. The solution to this minimisation problem can be shown to be a natural cubic spline, with knots at each of the data points. Natural cubic splines are simply local cubic polynomial functions between each data point and the next, with the constraints that the global function is continuous and the first derivative is continuous at knots (joins between the local cubic functions).

Splines can be extended to the multivariate case by the use of HDMR decompositions. The implication is however, that they cannot be used to estimate total effect indices S_T, since the HDMR series is truncated (usually after first-order interactions). An extension of multivariate splines, known as Adaptive Component Selection and Shrinkage Operator (ACOSSO), uses a modified version of (22) that uses norms rather than square-norms, and also includes the integral of the first derivative of g - see [20] for details. For Matlab scripts for performing sensitivity analyis with smoothing splines, see [9].

Another method that has attracted attention in recent years is the use of series of orthogonal polynomial functions for each term in the functional decomposition [43]. These take the form,

$$\begin{aligned} f_i(x_i) &= \sum_{r=1}^{\infty} \alpha_r^{(i)} \phi_r(x_i) \\ f_{i,j}(\boldsymbol{x}_{i,j}) &= \sum_{p=1}^{\infty}\sum_{q=1}^{\infty} \beta_{p,q}^{(i,j)} \phi_{p,q}(\boldsymbol{X}_{i,j}) \end{aligned} \tag{23}$$

where the ϕ are terms from a suitable series of orthogonal polynomials, and the α and β are their corresponding coefficients. Of the many series of orthogonal polynomials, it is typical to use either the Legendre or the Hermite types. Clearly, it is necessary to truncate the infinite series at a certain order R, which is usually determined by assessing convergence. Sensitivity indices can then be calculated analytically by the following,

$$\begin{aligned} \hat{S}_i &= \frac{\sum_{r=1}^{R} (\alpha_r^{(i)})^2}{\hat{V}(y)} \\ \hat{S}_{ij} &= \frac{\sum_{p=1}^{P} \sum_{q=1}^{Q} (\beta_{pq}^{(ij)})^2}{\hat{V}(y)} \end{aligned} \tag{24}$$

with S_{Ti} being approximated from the sum of S_i and its first order interactions. The accuracy of $\hat{S}_{Ti}$ of course depends on the accuracy of the HDMR truncation (i.e. if it is truncated after second order interactions, and third order interactions are significant, it will provide misleading estimates). The variance term $\hat{V}(y)$ can be calculated either from an analytical expression similar to those for the sensitivity indices, or from the original sample data. The latter has been shown to be preferable in some cases since it does not include bias from the HDMR truncation.

Another emulator that is widely used in sensitivity and uncertainty analysis is a *Gaussian process* (GP) . GPs are widely used in the machine learning community as a sophisticated form of nonlinear regression and classification [27]. In short, a GP is a *distribution over functions*, i.e. the random variable

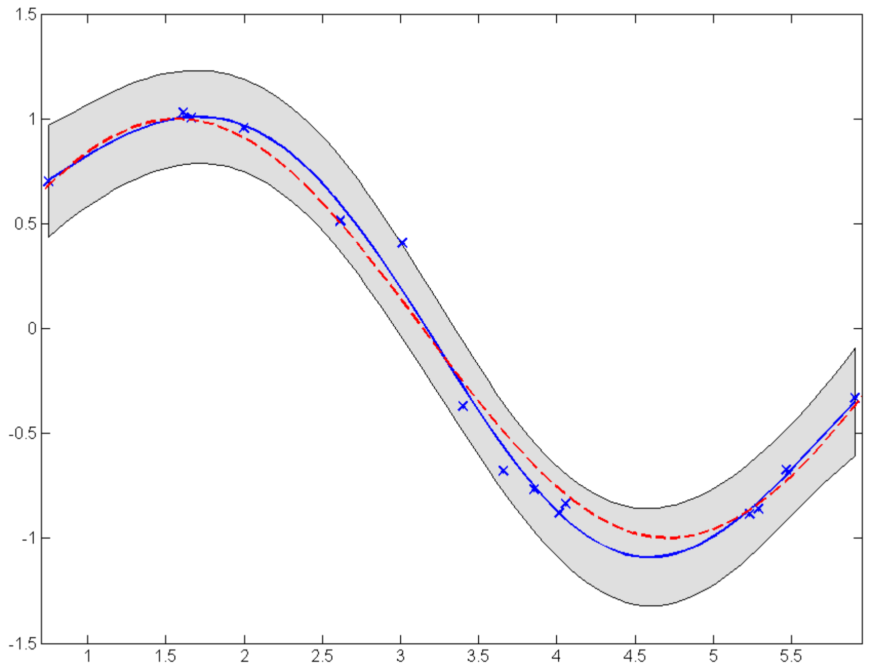

Figure 9: A Gaussian process fitted to sine wave data with added noise. Dotted line is the underlying sine function, solid line is the posterior mean of the GP and grey region represents 95% confidence bounds.

of the distribution is a function rather than a single number or fixed-length vector. Rather than returning a crisp output value y for any given input point $\boldsymbol{x}$ (as in a standard regression), the GP returns a specification for a Gaussian probability distribution. This means that for any given set of input points, the corresponding output values are distributed joint-normally. As such, a GP is completely defined by a mean function and covariance function, which specify the joint-normal distribution for any given set of input points,

$$f(\boldsymbol{x})|\sigma^2, \mathbf{w} \sim \mathcal{GP}\left(m(\boldsymbol{x}), \sigma^2 c(\boldsymbol{x}, \boldsymbol{x}')\right) \tag{25}$$

where $\boldsymbol{x}'$ is a point close to $\boldsymbol{x}$, and σ^2 and $\mathbf{w}$ are hyperparameters that are found from maximum likelihood estimation or Markov Chain Monte Carlo (see [2]). GPs have the advantage that they do not invoke the HDMR assumption, and as result can estimate sensitivity indices of all orders, including the S_{Ti}. However, the cost of training GPs scales poorly with model dimension, therefore users will encounter problems emulating models with $k > 30$ or so, depending on computational resources (this is however the case with all emulators, to some extent). A further useful property of the GP is that, given certain assumptions, sensitivity indices can be calculated analytically [25] (available as software from [16]). An extremely useful property of GPs is that, being a Bayesian approach, a full accounting of uncertainty in estimation is given, both from the data and from the estimation of parameters – this is passed in the form of confidence intervals to estimates of sensitivity indices. Some fairly recent additions in the field of GPs with respect to sensitivity analysis include a method of automatically screening out unimportant variables using the correlation parameter in the covariance function (see [21]), and the use of multiple GPs divided by decision trees to allow for bifurcating responses [13] (available as an R package).

As an example of the power of an emulator, consider again the simple

Variable	S_i (GP)	S_i (analytic)	S_{Ti} (GP)	S_{Ti} (analytic)
x_1	0.7566	0.7568	0.7715	0.7720
x_2	0.0456	0.0456	0.0605	0.0608
x_3	0.1829	0.1824	0.1830	0.1824

Table 2: Comparison of S_i and S_{Ti} estimates from a Gaussian process regression against analytical values

polynomial from equation 12. Using 128 points of the Sobol' sequence over the unit cube, a Gaussian process was trained, and sensitivity indices inferred analytically from the resulting posterior distribution. Table 6 shows the results. The GP is achieving accuracies of three or more decimal places on only 128 points – recall that the Monte Carlo estimator, for a similar level of accuracy, requires several thousands of runs per variable, therefore the GP is at least an order of magnitude more efficient. However, the GP and other emulators are only as good as their fit to the data: the polynomial function is a smooth, "well-behaved" function, which is an easy data modelling problem. For data that are heteroscedastic, bifurcating, or of varying smoothness, the emulators are likely to be much less reliable. Additionally, they scale poorly with dimensionality. However, for certain situations, emulators can offer a powerful solution.

Overall, there is no "best" emulator available. The approach will depend on computational resources, sample size and model dimension, amongst other things. Both [2] and [42] are recommended as background reading. Furthermore, it is essential to test the fit of any emulator by methods such as cross validation.

Custom Sampling for Emulators

Although the points were considered as "given" in the discussion above, it can happen that the analyst has the possibility to design her own set of training data for an emulator. This can be the case when, for example, the analyst only has a small number of input variables, but a very computationally-expensive model. In this scenario it makes sense to go directly to an emulator approach, since pure Monte Carlo would be too expensive and screening too inaccurate.

Experimental designs for emulators can be divided into two categories - space-filling designs, and model-based designs. In the former, the design is constructed to fill the sample space as evenly as possible, which is to say that points should be as far apart from each other as possible. The reasoning for this is that first, it is required to capture the behavior of the model over the whole input space with as few points as possible. Second, assuming that the output of the model is deterministic and smooth with respect to its inputs, little information can be gained by having points close to each other (since the outputs will be very similar). For this reason, purely random sampling is not an efficient design.

For a general-purpose emulator design, a space-filling design such as the Sobol' sequence discussed in section 3 is a good choice. Sobol' designs have a low-discrepancy property that avoids "clumping" of points, and allow the sequential addition of new points. For other space-filling designs the reader is referred to [24].

If the model itself is accessible, an even more sophisticated approach is to use a model-based design (also called optimal design). In this approach, the design is constructed so as to optimise some emulator-dependent criterion of interest. For example, a popular criterion, called "D-optimality", is to select points which minimise the variance of the estimators of the emulator parameters. Another way is to minimise the maximum variance of the emulator prediction at any given point (G-optimality). Clearly, a set of points which provide a good estimate of the parameters of one type of emulator might not be suitable for another type, which is why such designs are completely dependent on the emulator. Optimal designs can either be constructed in one pass, or step by step, perhaps starting from a rough initial design. The advantage of the sequential approach (known as adaptive design) is twofold – first, the output values will influence the optimum placement of new points, so knowledge of previous points will produce a more effective design than one that is constructed in one go. Second, by proceeding in small steps, one can generate exactly the required number of points to reach some level of accuracy of interest, perhaps measured by cross validation.

The theory of model-based designs is a large field of research that is beyond the remit of this chapter, therefore the reader is referred to [1] for a good general resource. There is also a strong interest in Bayesian approaches to optimal design; reviews can be found in [8, 12].

Scatter Plot Smoothing

A useful approach for handling "given" data is based on one-dimensional nonlinear smoothing regression. This method allows estimation of first-order sensitivity indices and, from a computational point of view, is less vulnerable to the curse of dimensionality.

A first visual indication of the effects of input variables can be gained by making k plots of x_i against y (see figure 3). If the data shows any kind of trend (or shape) with respect to x_i, this indicates that x_i is having some effect on the output. Indeed, the effect of x_i on the output is described by the curve $E(y|x_i)$ — in other words, the expected value of the model output if we were to fix x_i at some value. Over the range of x_i, this is equivalent to a moving average of the points in the x_i against y plot. If the $E(y|x_i)$ can be plotted, S_i can be estimated by taking the variance of this line (since $S_i = \text{var}\{E(y|x_i)\}$).

To plot such a moving average, it is simply a matter of using any of a number of smoothing regression approaches. Kernel regression has been already used for sensitivity analysis [26]. As with any smoothing regression, the data are modelled as,

$$y = m(x_i) + \epsilon \tag{26}$$

where $m(x)$ is the smoothed curve (ideally equivalent to $E(y|x_i)$), and ϵ is an independent error term with mean 0 and variance σ^2. In the kernel regression setting, $m(x)$ is typically chosen to be either a *local mean* or *local linear* kernel. The local mean estimator (first proposed by [23, 45]), for example, is expressed as,

$$\hat{m}(x) = \frac{\sum_{j=1}^{n} w(x_j - x; h) y_j}{\sum_{j=1}^{n} w(x_j - x; h)} \tag{27}$$

where w is a weighting function and h is a tuning parameter. The weighting function typically gives the strongest weight to points close to x, which reflects the belief that the closer two points are to each other in x, the more likely they are to have similar values in y. A commonly-used function that fulfils this requirement is a Gaussian density function with standard deviation h. The local linear estimator is expressed in a similar fashion (see [3] for details), and is generally regarded as preferable to the local mean, due to its improved properties near the edges of the data cloud. In all cases, the smoothing parameter h can be estimated by cross-validation.

Following the simple polynomial example, Figure 10 shows an illustration of local linear kernel regression applied to scatter plots of y against each x_i. The resulting estimates of sensitivity are given in Table 6.

In order to estimate sensitivity from the fitted kernel regression, the variance of the conditional expectation can be calculated using a standard variance identity (the domain of the expected value is explicitly stated as a subscript here for clarity),

$$var\{E_{\boldsymbol{x}_{\sim i}}(y|x_i)\} = E_{x_i}\{E_{\boldsymbol{x}_{\sim i}}(y|x_i)^2\} - E_{x_i}\{E_{\boldsymbol{x}_{\sim i}}(y|x_i)\}^2 \tag{28}$$

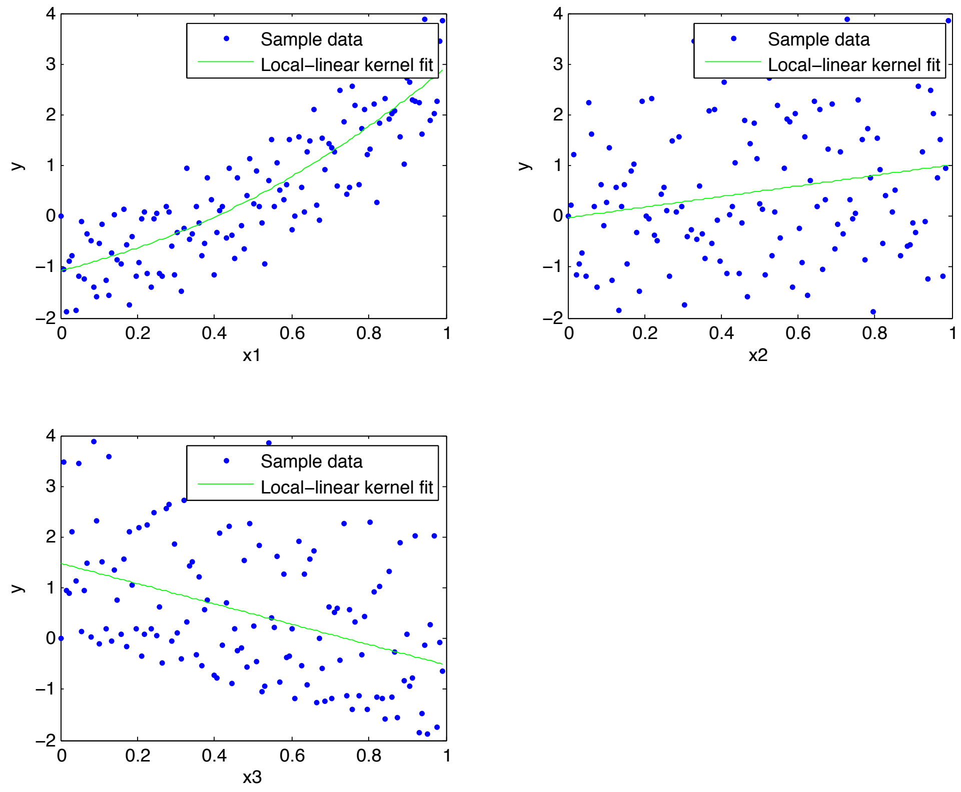

Figure 10: Local-linear kernel regression applied to the polynomial function.

Variable	S_i (kernel)	S_i (analytic)
x_1	0.735	0.758
x_2	0.049	0.046
x_3	0.182	0.182

Table 3: Comparison of S_i estimates from local-linear kernel regression of polynomial function against analytical values

Note here that since the expectation of a random variable A is defined as $\int A\, p(A)dA$, the expected values in (28) should be weighted by the distribution of points over x_i. The simplest way of doing this is to make kernel predictions at the same X_i values as the training data. A more sophisticated approach would be to estimate the underlying distribution with kernel density estimation or a similar technique. In both cases, the practitioner should ensure that the given data has been sampled with respect to underlying distributions – this may not necessarily be the case if the points have come from an optimization process, for example.

While the examples here have focused on kernel smoothing, this is by no means the only viable approach to obtaining $E(y|x_i)$. The problem is essentially an emulation/regression problem in one dimension, which can be tackled by any number of methods such as smoothing splines (see e.g. [29]) or Gaussian processes [27]. Even basic linear regression will provide a good estimate if the data is sufficiently linear. Good references on parametric and nonparametric regression can be found in [2] and [29].

Finally, it should be pointed out that while the idea of reducing a multidimensional problem to a series of one-dimensional problems is very appealing from a computational point of view, it is not a "silver bullet" solution. The estimation is dependent on a good approximation of $E(y|x_i)$, which can be difficult to obtain depending on the data and smoothing method used. Moreover, as the dimensionality of the problem increases, trends in scatterplots are increasingly confused by variation in other dimensions.

7 Conclusions

In this chapter, a number of "best practice" have been outlined that address many of the various situations that can confront an analyst. It is not claimed nor intended by the authors that an exhaustive review of all methods has been addressed here, but the reader should have found here enough material to apply or adapt to a practical case.

What did this chapter leave out?

- Monte Carlo filtering (MCF) is another possibly relevant setting for sensitivity analysis. In MCF the analyst is not interested in the distribution of y or in its variance, but only on a selected region therein, e.g. y values above or below a given threshold. MCF was not treated explicitly in this short introductory chapter, but the Monte Carlo quasi-random number approach described in section 3 is likely to be advisable also in this setting, all the more so since extremal values of y are likely to be associated to edges and corners in the hyperspace $\mathcal{X}$, which are more likely reached by quasi-random points than by ordinary random points [**?** **?**].

- The same applies to moment independent methods, such as those where the analysis is interested in how fixing a factor modifies the entire empirical probability distribution function of y. (reference to be added) The rational behind these methods is that variance is but one of several possible moments.

The present chapter has stressed some general concepts such as,

- The idea that a design should be explorative (against approaches such as one-factor-at-a-time so often seen in the literature and so poor at exploring nonlinear non-additive problems [32]).

- The benefit of having an iterative approach, so that one can start with a handful of points and then proceed in the analysis by adding more points. Likewise an iterative approach allows using a stopping rule, whereby the simulation stops as soon as a precision criterion has been met.
- The need to communicate the result of the analysis. Both first order and total order indices can be communicated in relation to both Pearson eta squared and the scatter plot based interpretation (for S_i) and to the theory of variance decomposition (for both S_i and S_{Ti}).

Given that every sensitivity analysis is a case apart (since every model has its idiosyncracies) these rules should hopefully be helpful to devise a suitable sensitivity setup which will fit to the characteristics of the model.

References

[1] A.C. Atkinson, A.N. Donev, and R. Tobias. *Optimum experimental designs, with SAS*, volume 34. Oxford University Press, USA, 2007.

[2] C.M. Bishop. *Pattern Recognition and Machine Learning.* Springer, New York, 2006.

[3] A.W. Bowman and A. Azzalini. *Applied smoothing techniques for data analysis: the kernel approach with S-Plus illustrations*, volume 18. Oxford University Press, USA, 1997.

[4] F. Campolongo, J. Cariboni, and A. Saltelli. An effective screening design for sensitivity analysis of large models. *Environmental Modelling & Software*, 22(10):1509–1518, 2007.

[5] F. Campolongo, J. Cariboni, and A. Saltelli. An effective screening design for sensitivity analysis of large models. *Environmental Modelling and Software*, 22:1509–1518, 2007.

[6] F. Campolongo, A. Saltelli, and J. Campolongo. From screening to quantitative sensitivity analysis. a unified approach. *Computer Physics Communication*, 182(4):978–988, 2011.

[7] F. Campolongo, A. Saltelli, and J. Cariboni. From screening to quantitative sensitivity analysis. a unified approach. *Computer Physics Communications*, 2011.

[8] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

[9] European Commission. Sensitivity analysis software - Joint Research Centre. `http://sensitivity-analysis.jrc.ec.europa.eu/software/index.htm`, 2012.

[10] R. I. Cukier, C.M. Fortuin, K. E. Schuler, A. G. Petschek, and J.H. Schaibly. Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. i theory. *The Journal of Chemical Physics*, 59:3873–3878, 1973.

[11] RI Cukier, HB Levine, and KE Shuler. Nonlinear sensitivity analysis of multiparameter model systems. *Journal of computational physics*, 26(1):1–42, 1978.

[12] A. DasGupta. 29 review of optimal bayes designs. *Handbook of Statistics*, 13:1099–1147, 1996.

[13] Robert B. Gramacy and Matthew Alan Taddy. Categorical inputs, sensitivity analysis, optimization and importance tempering with tgp version

2, an R package for treed Gaussian process models. *Journal of Statistical Software*, 33(6), 2010.

[14] T. Homma and A. Saltelli. Importance measures in global sensitivity analysis of model output. *Reliability Engineering and System Safety*, 52(1):1–17, 1996.

[15] T. Homma and A. Saltelli. Importance measures in global sensitivity analysis of nonlinear models. *Reliability Engineering & System Safety*, 52(1):1–17, 1996.

[16] Marc Kennedy. Gem-SA. `http://http://ctcd.group.shef.ac.uk/gem.html`, 2009.

[17] S Kucherenko, B Feil, N Shah, and W Mauntz. The identification of model effective dimensions using global sensitivity analysis. *Reliability Engineering and System Safety*, 96(4):440–449, 2011.

[18] E. Leamer. *Modelling Economic Series*, chapter Let's take the con out of econometrics, and Sensitivity analysis would help. Clarendon Press, Oxford, 1990.

[19] E. E. Leamer. Tantalus on the road to asymptopia. *Journal of Economic Perspectives*, 24(2):3146, 2010.

[20] Y. Lin and H.H. Zhang. Component selection and smoothing in smoothing spline analysis of variance models. *Annals of Statistics*, 34(5):2272–2297, 2006.

[21] C. Linkletter, D. Bingham, N. Hengartner, D. Higdon, and K.Q. Ye. Variable selection for gaussian process models in computer experiments. *Technometrics*, 48(4):478–490, 2006.

[22] M.D. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, pages 161–174, 1991.

[23] E.A. Nadaraya. On estimating regression. *Teoriya Veroyatnostei i ee Primeneniya*, 9(1):157–159, 1964.

[24] H. Niederreiter. *Quasi-Monte Carlo Methods.* Wiley Online Library, 1992.

[25] J.E. Oakley and A. O'Hagan. Probabilistic sensitivity analysis of complex models: a Bayesian approach. *Journal of the Royal Statistical Society B*, 66:751–769, 2004.

[26] P. Paruolo, A. Saltelli, and M. Saisana. Ratings and rankings: Voodoo or science? *Arxiv preprint arXiv:1104.3009*, 2011.

[27] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, Cambridge, Massachusetts, 2006.

[28] M. Ratto, A. Pagano, and P. Young. State dependent parameter metamodelling and sensitivity analysis. *Computer Physics Communications*, 177(11):863–876, 2007.

[29] D. Ruppert, M.P. Wand, and R.J. Carroll. *Semiparametric regression*, volume 12. Cambridge Univ Pr, 2003.

[30] A. Saltelli. Making best use of model evaluations to compute sensitivity indices. *Computer Physics Communications*, 145(2):280–297, 2002.

[31] A. Saltelli and P. Annoni. How to avoid a perfunctory sensitivity analysis. *Environmental Modelling & Software*, 25(12):1508–1517, 2010.

[32] A. Saltelli and P. Annoni. How to avoid a perfunctory sensitivity analysis. *Environ*, 2010.

[33] A. Saltelli and B. D'Hombres. Sensitivity analysis didn't help. a practitioner's critique of the stern review. *Global Environmental Change*, 20(2):298–302, 2010.

[34] A. Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global Sensitivity Analysis : The Primer*. John Wiley & Sons, Ltd., Chichester, 2008.

[35] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto. *Sensitivity Analysis in Practice. A Guide to Assessing Scientific Models*. John Wiley and Sons publishers, 2004.

[36] A. Saltelli, S. Tarantola, and K. Chan. Quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1):3956, 1999.

[37] Andrea Saltelli, Paola Annoni, Ivano Azzini, Francesca Campolongo, Marco Ratto, and Stefano Tarantola. Variance based sensitivity analysis of model output. design and estimator for the total sensitivity index. *Computer Physics Communications*, 181(2):259–270, 2010.

[38] I. M. Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

[39] I. M. Sobol'. Sensitivity estimates for nonlinear mathematical models. *Mathematical Modeling and Computational Experiment*, 1(4):407–414, 1993.

[40] I. M. Sobol' and S. S. Kucherenko. On global sensitivity analysis of quasi-monte carlo algorithms. *Monte Carlo Methods and Applications*, 11(1):83–92, 2005.

[41] I.M. Sobol'. Uniformly distributed sequences with an addition uniform property. *USSR Comput. Maths. Math. Phys.*, 16:236–242, 1976.

[42] C.B. Storlie and J.C. Helton. Multiple predictor smoothing methods for sensitivity analysis: Description of techniques. *Reliability Engineering & System Safety*, 93(1):28–54, 2008.

[43] B. Sudret. Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering & System Safety*, 93(7):964–979, 2008.

[44] S. Tarantola, D. Gatelli, and TA Mara. Random balance designs for the estimation of first order global sensitivity indices. *Reliability Engineering & System Safety*, 91(6):717–727, 2006.

[45] G.S. Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372, 1964.