Journal of Computations & Modelling, vol.3, no.3, 2013, 17-56
ISSN: 1792-7625 (print), 1792-8850 (online)
Scienpress Ltd, 2013

Matlab automation algorithm for performing global sensitivity analysis of complex system models with a derived FAST method

Hery Tiana Rakotondramiarana[1] and Ando Ludovic Andriamamonjy[2]

Abstract

This study presents a Global Sensitivity Analysis Tool, named GoSAT, which is a ready to use Matlab automation algorithm of a derived Fourier Amplitude Sensitivity Test (FAST) method that automatically ranks, by decreasing order in a bar diagram, both the first order effects and the order one interaction effects of a model factors. To perform a global sensitivity analysis with the proposed algorithm, only the Matlab code of the surveyed model, all factors' names, their respective nominal values and ranges of variation, and their distinct attributed frequencies are necessary to be known as inputs. To test the efficiency of the proposed procedure, it was applied on a ten factors Sobol function and the expected outcome was obtained, that is, the factors related to the least valuable Sobol parameter are the most influent ones. With the proposed algorithm,

[1] Institute for the Management of Energy (IME), University of Antananarivo, Madagascar.
[2] Institute for the Management of Energy (IME), University of Antananarivo, Madagascar.

Article Info: *Received* : May 29, 2013. *Revised* : July 10, 2013
Published online : September 15, 2013

modelers do not have to wait for other researchers' subsequent work completion to have feedbacks on data input and factors of their own models but can directly perform global sensitivity analysis right after developing the computing codes related to these models.

Mathematics Subject Classification: 49Q12; 65C20; 65T50; 62M15
Keywords: Global sensitivity analysis; FAST method; Automation algorithm; Complex system modeling; Sobol function

1 Introduction

Sensitivity analysis is recognized as a key component of model development [1]. Not only can sensitivity analysis form a part of model calibration, informing the modeler of which parameters will have considerable impact on the model when altered, but it can also allow identification of ways that the model can be simplified, thus helping to avoid overparameterisation [2]. Further, it can help modelers identify any unexpectedly strong dependencies on parameters which should not be highly influential, allowing correction of the model [3].

The sensitivity of a model output to a given input factor has been traditionally expressed mathematically as the derivative of the model output with respect to the input variation, sometimes normalized by either the central values where the derivative is calculated or by the standard deviations of the input and output values [4]. These sensitivity measurements are "local" because they are fixed to a point (base value) or narrow range where the derivative is taken. These local sensitivity indexes are classified as "one-parameter-at-a-time" (OAT) methods, as they quantify the effect of a single parameter by assuming all others are fixed [5]. Local OAT sensitivity indices are only efficient if all factors in a

model produce linear output responses, or if some type of average can be used over the parametric space. Often, the model outputs' responses to changes in the input factors are non-linear, and an alternative global sensitivity approach, where the entire parametric space of the model is simultaneously explored for all input factors, is needed [6].

The global sensitivity analysis investigates the relationship between uncertainty in the inputs of a computational model and the uncertainty in the output. So-called variance-based techniques are based on a decomposition of the variance in the model output into components each depending on just one input variable, components each depending on two variables and so forth. Correspondingly, the output variance can be decomposed into contributions each coming from only one input variable ("first order effects"), from just two variables ("second order"), etc [7]. Hence, almost the available global sensitivity method like Fourier Amplitude Sensitivity Test (FAST) [8], extended Fourier amplitude sensitivity test (EFAST) [9], random balance design (RBD) [10], Ishigami-Saltelli-Homma-method (HIS) [11] or the Sobol_ algorithm [12] are focused on computing the sensitivity index of the factors, and are more and less tricky to implement.

Moreover, today’s simulation codes are even getting much more complex and unavoidably more time expensive. The multidisciplinary nature of design and the need for incorporating uncertainty in design optimization have posed additional challenges. A widely used strategy is to utilize approximation models which are often referred to as metamodels as they provide a model of the model [13], replacing the expensive simulation model during the process [14].

Our work aimed to propose a Matlab [15] ready to use automatic algorithm which combines the computing of factors’ effect and the determination of the metamodel or the High Dimensional Model Representation (HDMR) [16] of the surveyed model. Indeed, for performing global sensitivity analysis of complex system models, Mara [17] has suggested a derived FAST method that has been

improved by Rakotondramiarana [18]. Such method investigates the relationship between the factors' effect and the metamodel polynomial regression coefficients that are known to be proportional to Fourier amplitudes.

Belonging to the same class of algorithm as FAST and RBD, the proposed algorithm is using a frequency based approach i.e., signals of known frequencies are assigned to the input factors, and a frequency analysis is carried out on the output that computes the influence of each input factor on the output [7].

By using the Matlab code of the model and each input factor's nominal value, range of variation, assigned frequency and name, the proposed algorithm plots automatically in a bar diagram the first order effect of the factors and in another one the higher-order effect, particularly those related to the order one interactions of factors. Unlike the other global sensitivity analysis algorithms, this one also automatically computes the metamodel of the surveyed model by using a least square algorithm such as the Levenberg-Marquardt algorithm [19]. To test the efficiency of the proposed procedure, it was applied on the Sobol function [20]. More often, the researcher who first models a complex system is different from the one who subsequently performs global sensitivity analysis of the suggested model. Then, the modeler must wait for the completion of the second work to have feedbacks on data input and to do the appropriate corrections based on her/his model sensitivity analysis results. Accordingly, the proposed algorithm should also be a tool helping modelers perform themselves global sensitivity analysis of their own models. Being a Global Sensitivity Analysis Tool, this algorithm was named GoSAT.

2 Methodology

2.1 Theoretical foundation

The automation procedure is based on a derived FAST method. Interested

readers are referred to [17] and [18] for broader theoretical investigations.

p being an integer; let us consider a p factors model f one of which outputs is y.

$$y = f\left(x_1, x_2, ..., x_h, ..., x_p\right) \tag{1}$$

where each factor x_h has a variation range R_h:

$$R_h = \left[L_h, U_h\right] \text{ and } \qquad L_h \le x_h \le U_h \qquad (h = 1 \text{ to } p) \tag{2}$$

The nominal value of each factor x_h being designated by x_h^0; the nominal model outcome y_0 is defined as

$$y_0 = f\left(x_1^0, x_2^0, ..., x_h^0, ..., x_p^0\right) \tag{3}$$

In order to know the evolution of y around the nominal model outcome y_0, one defines the factor's restricted range of variation I_h^* ($h = 1$ to p) as follows [18]:

$$I_h^* = \left[x_h^0 - \delta_h, x_h^0 + \delta_h\right] \tag{4}$$

where the half width δ_h is defined as:

$$L_h + x_h^0 < \delta_h < U_h - x_h^0$$

$$\text{with} \qquad \delta_h = \frac{\alpha_p\left(U_h - L_h\right)}{100} \quad (h = 1 \text{ to } p) \tag{5}$$

in which α_p (%) is the variation rate around the nominal values of factors.

Indeed, by carrying out simulation(s) with the model, such that the model factors x_h ($h = 1$ to p) simultaneously oscillate in their restricted ranges of variation, one should expected that the output y^* value(s) oscillates around y_0 as follows [18]:

$$y^* = f\left(x_1^*, x_2^*, ..., x_h^*, ..., x_p^*\right)$$

which can be rewritten as:

$$y^* = y_0 + g\left(x_1^*, x_2^*, ..., x_h^*, ..., x_p^*\right)$$

where, all nomenclatures given for Equations (1) to (5) being still available,

g: *p* factors "virtual model" which computes the difference between y^* and y_0

$$g = y^* - y_0 \tag{6}$$

x_h^* (h = 1 to p): value taken by every factor x_h while oscillating in its restricted range of variation I_h^* and calculated by

$$x_h^* = x_h^0 + \delta_h \sin(2\pi w_h) \tag{7}$$

w_h (h = 1 to p): frequency attributed to every factor x_h.

Hence, the sensitivity analysis of the surveyed output of the model consist in the estimation of the sound effect generated by the p factors to corrupt the nominal outcome y_0, and determine which of them cause the most important effect on y_0 [18]. So one has to find among the factors those, whether by their own effects or by their effects due to interactions with other factors, that maximize the absolute value of the virtual model g (Equation (6)).

According to Mara [17], the polynomial approximation of virtual model g is of the form

$$g = \sum_{h=1}^{p} \beta_h \langle x_h \rangle + \sum_{h=1}^{p} \sum_{\substack{h'=1 \\ h' \le h}}^{p} \beta_{hh'} \langle x_h \rangle \langle x_{h'} \rangle + \sum_{h=1}^{p} \sum_{\substack{h'=1 \\ h' \le h}}^{p} \sum_{\substack{h''=1 \\ h'' \le h'}}^{p} \beta_{hh'h''} \langle x_h \rangle \langle x_{h'} \rangle \langle x_{h''} \rangle + \ldots$$

where the standardized value $\langle x_h \rangle$ of every factor x_h (h = 1 to p) is calculated by

$$\langle x_h \rangle = \sin(w_h s)$$

and s is a common variable to all standardized factors $\langle x_h \rangle$ as frequency w_h is attributed to every factor x_h (h = 1 to p)

Hence, virtual model g permits to find the factors effects on the model such as the linear effect (β_h) and the interaction effect ($\beta_{hh'}, \beta_{hh'h''}, \ldots$). Moreover, the metamodel $\tilde{y}$ of the model is given by:

$$\tilde{y} = y_0 + \sum_{h=1}^{p} \beta_h \langle x_h \rangle + \sum_{h=1}^{p} \sum_{\substack{h'=1 \\ h' \le h}}^{p} \beta_{hh'} \langle x_h \rangle \langle x_{h'} \rangle + \ldots$$

By using the Fast Fourier Transform (FFT) [21], a spectral analysis of g

enables to detect the peaks on the spectrum corresponding to the dominating factors frequencies, thus the significant terms of the metamodel are identified. A least square algorithm such as the Levenberg-Marquardt algorithm is used to minimize $(y^* - \tilde{y})$ and get the regression coefficients (β_h , $\beta_{hh'}$, and $\beta_{hh'h''}$).

2.2 Automation procedures of GoSAT

The following twenty one steps show how to automatically perform a global sensitivity analysis with the proposed algorithm using FAST method:

1) Assign distinct frequencies w_h to all the surveyed model factors x_h (h = 1 to p). These frequencies should be free of interferences.
2) Give the nominal values x_h^0 (h = 1 to p) of the model factors, the range of variation R_h of each factor, the proportionality coefficient α_p $(0 < \alpha_p < 100)$ and a threshold number N_t which limits the number of frequencies to identify on each iterative spectral representation.
3) By using the nominal values x_h^0 outlined in step 2), do a simulation in order to obtain the nominal value y_0 of the surveyed model output.
4) Give the number N_s of simulations to do for the analysis while respecting the Shannon theorem $N_s \geq 2\max(w_j)\big|_{j=1 \text{ to } q}$ ($q > p$ as induced frequencies are also included).
5) Compute the half width δ_h of the restricted range of variation I_h^* of each factor as defined in Equation (5).
6) Do the N_s simulations with the surveyed model, where every factor x_h describes a sinusoidal function around its nominal value x_h^0 as defined in Equation (7). Hence, for N_s simulations, we have:

$$y_k^* = f\left(x_1^{*k}, x_2^{*k}, ..., x_h^{*k}, ..., x_p^{*k}\right)$$

where $x_h^{*k} = x_h^0 + \delta_h \sin(w_h s_k)$, $s_k = \frac{2\pi k}{N_s}$, and $k = 0$ to N_s

7) Initialize as empty
   a. A vector H_q which stores factor numberings corresponding to fundamental frequencies associated to influent factors.
   b. A 2-columns matrix H_m which stores twins of factor numberings corresponding to the order-1 interaction between 2 factors.
   c. A 3-columns matrix H_n which stores triplets of factor numbering corresponding to the order-2 interaction between 3 factors.
8) Initialize the metamodel $\tilde{y}$ as: $\tilde{y}_k = y_0$ where $k = 0$ to N_s;
9) Compute the "virtual model" g which generates the difference between y_0 and each value of y^*:

$$g_k = y_k^* - \tilde{y}_k \quad \text{for} \quad k = 0 \text{ to } N_s$$

10) Compute the Fourier coefficients of the N_s values of the virtual model g by means of the FFT (Fast Fourier Transform) that is an algorithm proposed by Cooley-Tukey [21]. Indeed, the Fourier amplitude related to whether fundamental frequencies or induced ones are proportional to the regression coefficients of the metamodel $\tilde{y}$. Therefore, the following quantity is to be assessed:

$$TFG = \frac{|FFT(g)|}{N_s} \tag{8}$$

11) Among the values of *TFG* calculated by Equation (8), nullify those which do not represent peaks; such that for $k = 0$ to N_s, TFG_k values are included in the following ranges:
   - $TFG_{k-1} < TFG_k < TFG_{k+1}$ (Figure 1.a)

- $TFG_{k+1} < TFG_k < TFG_{k-1}$ (Figure 1.b)
- $TFG_k < TFG_{k-1}$ and $TFG_k < TFG_{k+1}$ (Figure 1.c)

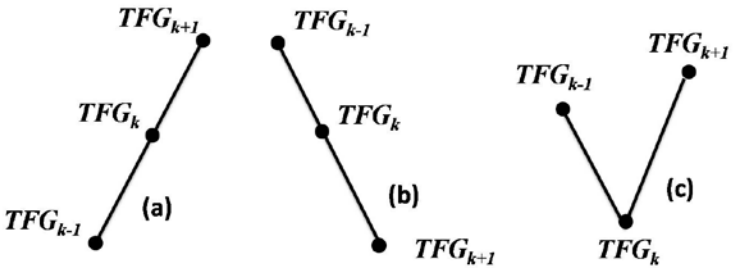

Figure 1: values of TFG_k to nullify

12) Taking into account the symmetric behavior of the spectral representation, $E\left(\frac{N_s}{2}\right)$ number of simulations are enough to do the spectral analysis (operator E calculating the fix part of any real number)
13) Only frequencies related to the N_t (or inferior to N_t) first peaks of the Fourier amplitude will be considered for each spectral representation.
14) Let us assign to a vector h ($h = 1,2,..., p$) the numbering of the studied model factors. Among the surveyed frequencies, find those which are eventually equal to:
    a) The fundamental frequencies w_h ($h = 1$ to p as given in step 1), and add to the elements of the vector H_q (initialized in step 7.a) the factors numberings h associated to the so detected frequencies.
    b) Order 2 induced frequencies: both ($w_{h'} + w_{h''}$) and $\left|w_{h'} - w_{h''}\right|$; and add, as next row(s), to the elements of the matrix H_m (initialized in step 7.b) the so detected twin(s) of factor numberings (h', h'').

c) Order 3 induced frequencies: all the following four frequencies should be detected: $\left(w_h''' + w_h^{iv} + w_h^{v}\right)$, $\left|w_h''' - w_h^{iv} + w_h^{v}\right|$, $\left|w_h''' + w_h^{iv} - w_h^{v}\right|$, and $\left|w_h''' - w_h^{iv} - w_h^{v}\right|$. Add, as next row(s), to the elements of the matrix H_n (initialized in step 7.c) the so detected triplet(s) of factor numberings (h''', h^{iv}, h^{v}).

15) By means of y^* values obtained in step 6), determine the regression coefficients α, β, and γ, by minimizing (y^* - $\tilde{y}$) with the Levenberg-Marquardt least square algorithm in order to have the following metamodel $\tilde{y}$:

$$\tilde{y}_k = y_0 + \sum_{i=1}^{N_{Hq}} \alpha_i . \left\langle x\left(H_q(i)\right)\right\rangle_k + \sum_{i=1}^{N_{Hm}} \beta_i . \left\langle x\left(H_m(i,1)\right)\right\rangle_k . \left\langle x\left(H_m(i,2)\right)\right\rangle_k + \sum_{i=1}^{N_{Hn}} \gamma_i . \left\langle x\left(H_n(i,1)\right)\right\rangle_k . \left\langle x\left(H_n(i,2)\right)\right\rangle_k . \left\langle x\left(H_n(i,3)\right)\right\rangle_k$$

in which, N_{Hq} is the number of elements of the vector H_q, N_{Hm} the number of rows of the matrix H_m and N_{Hn} is the number of rows of the matrix H_n. And

$$\left\langle x\left(H_q(i)\right)\right\rangle_k = \sin\left(w\left(H_q(i)\right) * s_k\right)$$

$$\left\langle x\left(H_m(i,1)\right)\right\rangle_k = \sin\left(w\left(H_m(i,1)\right) * s_k\right)$$

$$\left\langle x\left(H_m(i,2)\right)\right\rangle_k = \sin\left(w\left(H_m(i,2)\right) * s_k\right)$$

$$\left\langle x\left(H_n(i,1)\right)\right\rangle_k = \sin\left(w\left(H_n(i,1)\right) * s_k\right)$$

$$\left\langle x\left(H_n(i,2)\right)\right\rangle_k = \sin\left(w\left(H_n(i,2)\right) * s_k\right)$$

$$\left\langle x\left(H_n(i,3)\right)\right\rangle_k = \sin\left(w\left(H_n(i,3)\right) * s_k\right)$$

where $w\left(H_q(i)\right)$, $w\left(H_m(i,1)\right)$, $w\left(H_m(i,2)\right)$, $w\left(H_n(i,1)\right)$, $w\left(H_n(i,2)\right)$, $w\left(H_n(i,3)\right)$ are frequencies associated to the factor no. $H_q(i)$, $H_m(i,1)$, $H_m(i,2)$, $H_n(i,1)$, $H_n(i,2)$, $H_n(i,3)$ respectively; and

$$s_k = \frac{2\pi k}{N_s}, \qquad k = 0 \text{ to } N_s$$

On Matlab, the non-linear least square built-in function “lsqnonlin.m” can be used to obtain these regression coefficients.

16) On Matlab, plot Fourier Amplitude (*TFG*) versus frequency w_h $\left(w_h = 0 \; to \; E\left(\frac{N_s}{2}\right)\right)$ and show on spectral representation, by means of textarrow properties of the annotation function, the identified frequencies at step 14).

17) Delete the newly identified frequencies at step 14) by computing the new value of the “virtual model” g as defined in step 9);

18) This iterative spectral identification procedure is ended with one of the following criterions:

- All the remaining Fourier amplitudes are less than a given tolerance ε
- There is no identified frequency at step 14)
- When frequencies previously identified are detected at the next iterative spectral representation,

otherwise, the procedure is to be resumed from step 10).

19) Write the final metamodel expression

20) Validate the metamodel expression by plotting (y^* versus $\tilde{y}$). A thin dot snow bisector should be obtained if the metamodel represents a good accuracy.

21) By plotting a bar diagram:

- compare α_i values and rank them according to their absolute values
- compare β_i values and rank them according to their absolute values

3 Test and Results

3.1 Test function and assigned frequencies

In order to test the proposed algorithm, an analytical function attributed to Sobol [22] [23] was used:

$$y(x)=\prod_{i=1}^{p}\frac{\left|4x_i-2\right|+a_i}{1+a_i} \tag{9}$$

where $a_i \geq 0$ are parameters. This function is widely used as a test function in global sensitivity analysis algorithms because of its complexity, that is, it is strongly nonlinear and non-monotonic, and all its interaction terms are non zero by definition. The values of a_i determine the relative importance of the factor x_i such that the higher is a_i , the lower is the importance of x_i [16].

For this example, a 10 parameters Sobol function with $a_i=\{99,0,1,4,0,4,99,0,99\}$ was used. With this set of a_i parameters, it should be expected that factors x_2, x_5 , and x_9 are the most dominating ones, x_3 is an important one while x_4 and x_6 are less influent, and the others have a negligible contribution.

Thus, these 10 factors are numbered and symbolized as x_i ($i=1$ to 10). Their common nominal value is 0 while their common range of variation is between $[-1;1]$. To each of the above mentioned 10 factors x_i ($i=1$ to 10), the following distinct numbers were respectively attributed as frequency: 3, 7, 15, 31, 63, 127, 255, 511, 677, and 259.

3.2 Results and discussion

By using the proposed Matlab automation algorithm of the FAST method on the test function (Equation (9)), with $\alpha_p=5\,(\%)$ and $N_t=8$ that are defined

in Equation (5) and in step 2 of §2.2- respectively, successive spectra schemes are presented on Figures 2 to 10.

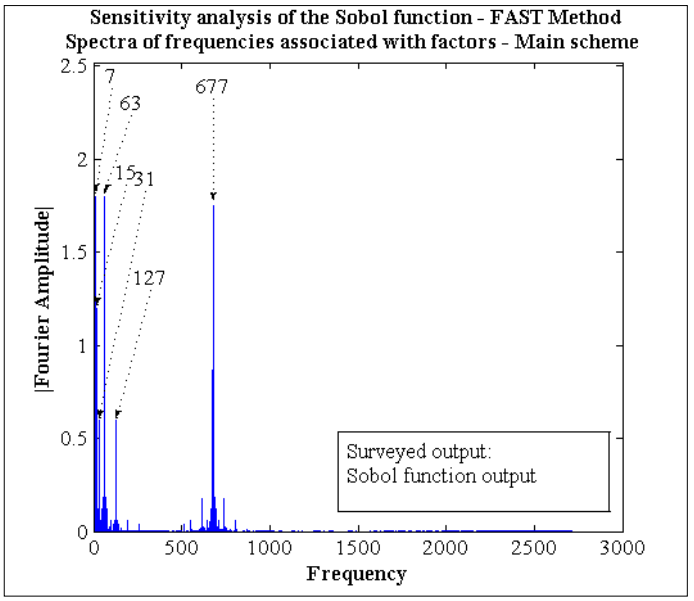

Figure 2: Main spectra scheme for identifying the most dominant factors

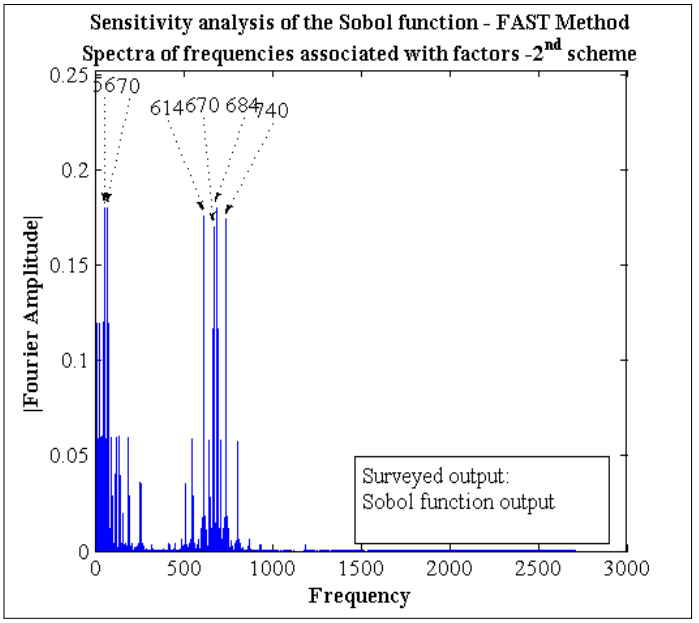

Figure 3: 2nd spectra scheme obtained after elimination of the main scheme identified spectra

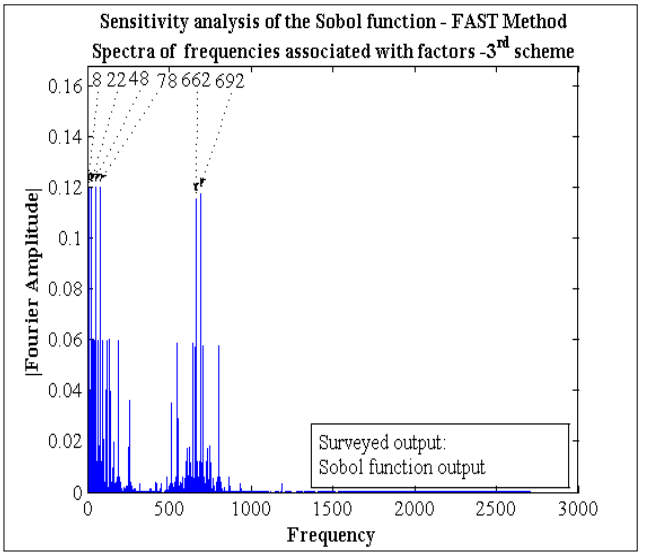

Figure 4: 3rd spectra scheme obtained after elimination of the 2nd scheme identified spectra

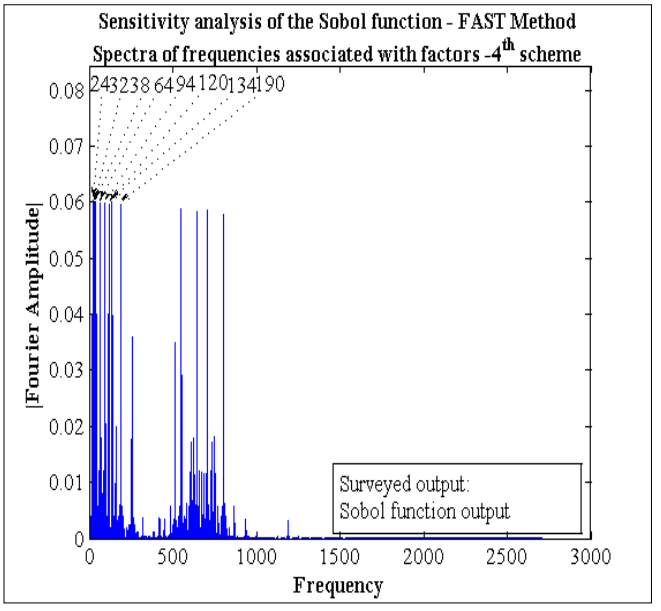

Figure 5: 4th spectra scheme obtained after elimination of the 3rd scheme identified spectra

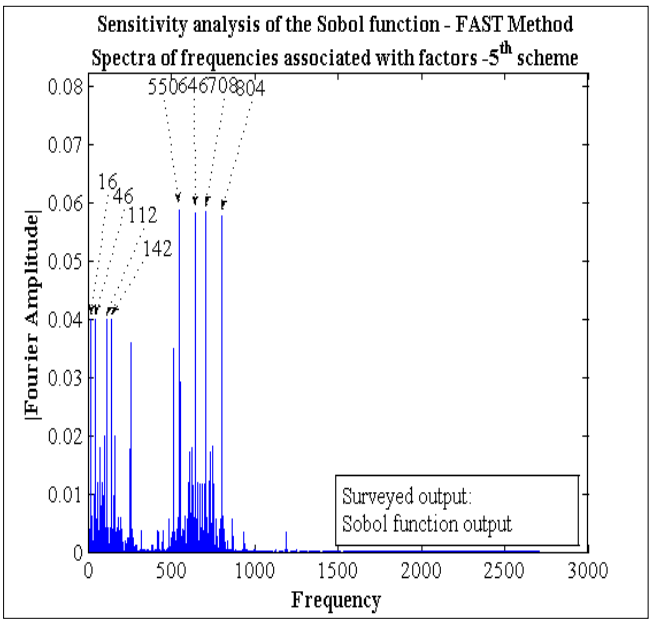

Figure 6: 5th spectra scheme obtained after elimination of the 4th scheme identified spectra

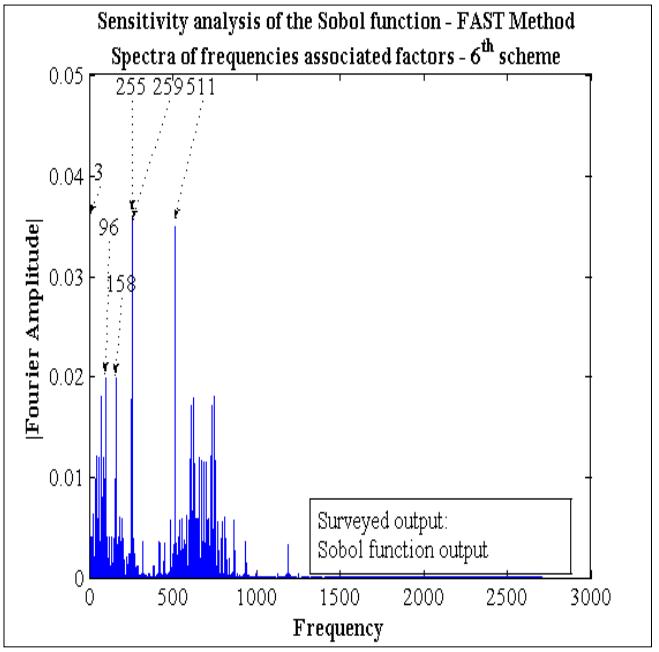

Figure 7: 6th spectra scheme obtained after elimination of the 5th scheme identified spectra

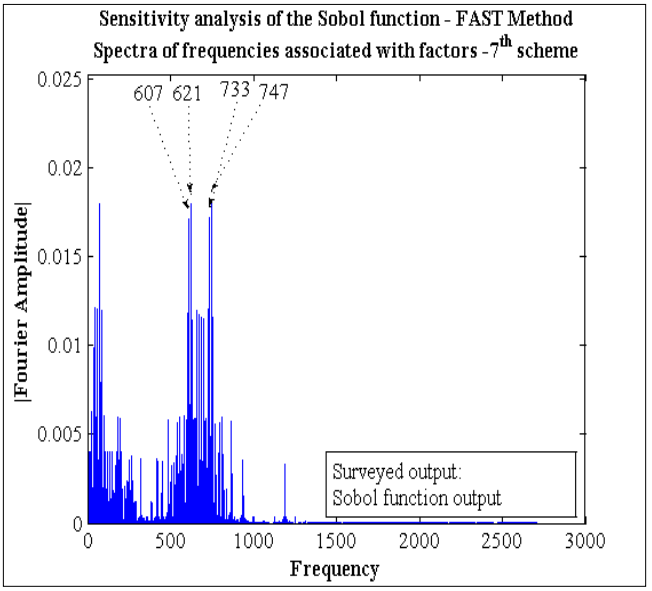

Figure 8: 7th spectra scheme obtained after elimination of the 6th scheme identified spectra

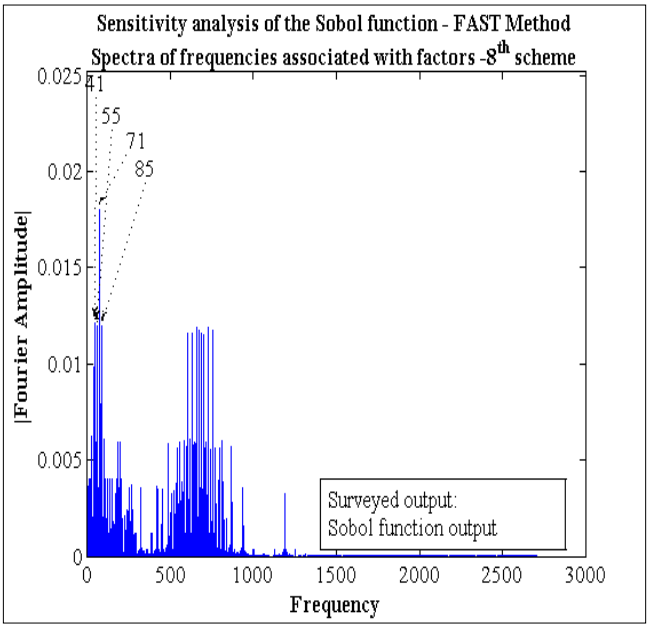

Figure 9: 8th spectra scheme obtained after elimination of the 7th scheme identified spectra

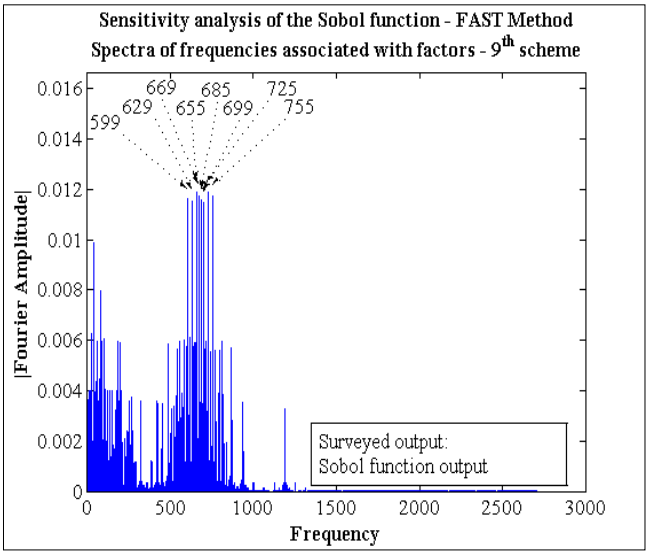

Figure 10: 9th and last spectra scheme obtained after elimination of the 8th scheme identified spectra

The automation algorithm also provides the metamodel related to the studied Sobol function output as given in Equation (10).

$$\begin{aligned}\tilde{y} = {} & 17.9816 - 3.5963\langle x_2\rangle - 2.3976\langle x_3\rangle - 1.1988\langle x_4\rangle - 3.5963\langle x_5\rangle - 1.1988\langle x_6\rangle - 3.5963\langle x_9\rangle \\ & - 0.07123\langle x_1\rangle - 0.07122\langle x_7\rangle - 0.07124\langle x_{10}\rangle - 0.0711\langle x_8\rangle \\ & + 0.7200\langle x_2\rangle\langle x_5\rangle + 0.71926\langle x_5\rangle\langle x_9\rangle + 0.71925\langle x_2\rangle\langle x_9\rangle + 0.47953\langle x_2\rangle\langle x_3\rangle \\ & + 0.47978\langle x_3\rangle\langle x_5\rangle + 0.4795\langle x_3\rangle\langle x_9\rangle + 0.24057\langle x_2\rangle\langle x_4\rangle + 0.23974\langle x_4\rangle\langle x_5\rangle \\ & + 0.23973\langle x_5\rangle\langle x_6\rangle + 0.23973\langle x_2\rangle\langle x_6\rangle + 0.15983\langle x_3\rangle\langle x_4\rangle + 0.15983\langle x_3\rangle\langle x_6\rangle \\ & + 0.23972\langle x_6\rangle\langle x_9\rangle + 0.23972\langle x_4\rangle\langle x_9\rangle + 0.07983\langle x_4\rangle\langle x_6\rangle - 0.1439\langle x_2\rangle\langle x_5\rangle\langle x_9\rangle \\ & - 0.1082\langle x_2\rangle\langle x_3\rangle\langle x_5\rangle - 0.0959\langle x_3\rangle\langle x_5\rangle\langle x_9\rangle - 0.0958\langle x_2\rangle\langle x_3\rangle\langle x_9\rangle + \ldots\end{aligned} \tag{10}$$

Figures 11 and 12 highlight the validation of the metamodel. Indeed, a thin dot snow bisector is obtained while plotting the prediction of the metamodel versus the prediction of the original Sobol function (Figure 11). As for Figure 12, it shows a good accuracy of the metamodel compared to the original Sobol function (Equation (9)).

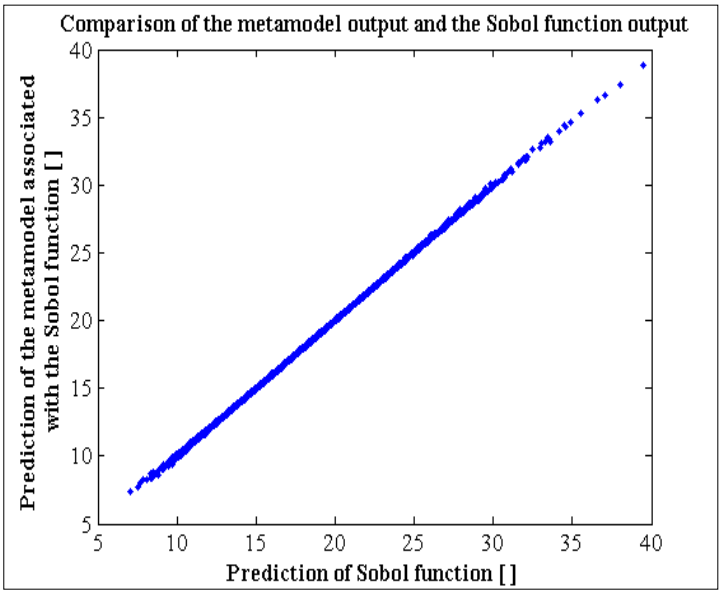

Figure 11: Comparison of the metamodel output and the Sobol function output

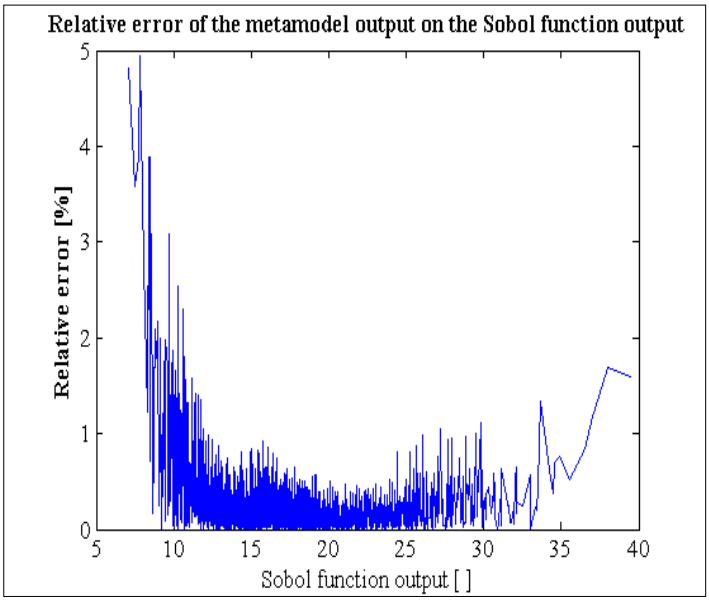

Figure 12: Accuracy of the metamodel compared to the Sobol function output

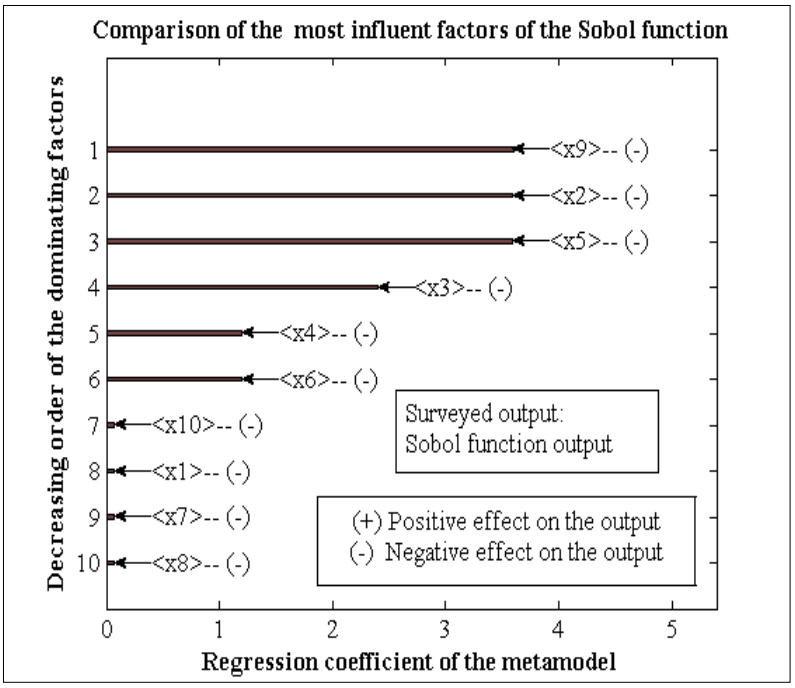

Figure 13: Ranking of the most dominating model factors

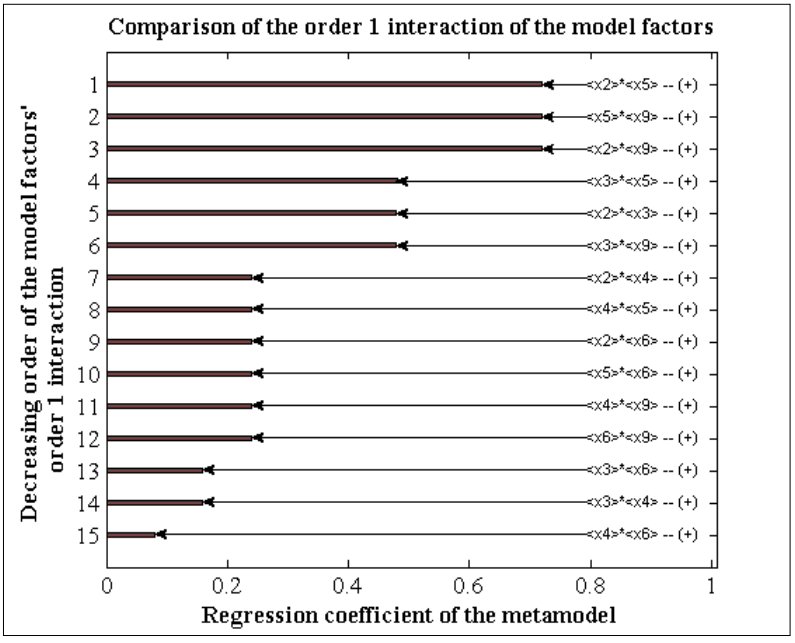

Figure 14: Ranking of the most dominating order 1 interaction between the model factors

Figure 13 illustrates that the factors related to a low value Sobol parameter a_i are the dominating factors. Indeed, x_2, x_5, x_9 which are the most influent factors are associated with $a_i = 0 \quad (i = 2, 5, 9)$; x_3 which is an important factor is associated with $a_3 = 1$; whereas x_4 and x_6 are related to 4. While the negligible factors such as x_1, x_7, x_8, x_{10} are associated with 99. According to [16], these results are coherent with the expected ones.

Figure 14, compares the order one interaction effects of factors. It can be observed that the interaction effects between the most influent factors, such as x_2, x_5, x_9, are the most dominating.

4 Conclusion

This study aimed to develop a Matlab automation algorithm for performing global sensitivity analysis of complex system models. In order to give the factors order of influence and the metamodel of the surveyed model, the proposed automation procedures are based on a method that was derived from FAST method. By applying the proposed automation algorithm on a ten parameters Sobol function, the expected results were obtained, such that the factor associated with the least valuable Sobol function parameter a_i is the most dominating one. Validation of the proposed algorithm being satisfactory, it was named GoSAT and can be used as a Global Sensitivity Analysis Tool for any models of complex systems that are coded on Matlab. One of extension works is to adapt and implement GoSAT on other programming languages such as C++ and FORTRAN.

References

[1] A. Saltelli, K. Chan and E. M. Scott, *Sensitivity Analysis: Gauging the Worth of Scientific Models*, John Wiley & Sons Ltd, 2000.

[2] S. Tarantola, N. Giglioli, J. Jesinghaus and A. Saltelli, Can global sensitivity analysis steer the implementation of models for environmental assessments and decisionmaking?, *Stochastic Environmental Research and Risk Assessment (SERRA)*, **16**(1), (2002), 63-76.

[3] A. Saltelli and M. Scott, Guest editorial: The role of sensitivity analysis in the corroboration of models and its link to model structural and parametric uncertainty, *Reliability Engineering & System Safety*, **57**(1), (1997), 1-4.

[4] C. T. Haan, B. Allred, D. E. Storm, G. J. Sabbagh and S. Prabhu, Statistical procedure for evaluating hydrologic/water quality models, *ASAE Transactions*, **38**(3), (1995), 725-733.

[5] A. Saltelli, M. Ratto, S. Tarantola and F. Campolongo, Sensitivity analysis for chemical models, *Chemical Reviews*, **105**(7), (2005), 2811 – 2828.

[6] R. Muñoz-Carpena, Z. Zajac and Y. M. Kuo, Global Sensitivity and Uncertainty Analyses of the Water Quality Model VFSMOD-W, *ASABE Transactions*, **50**(5), (2007), 1719-1732.

[7] E. Plischke, An Effective Algorithm for computing global Sensitivity Indices (EASI), *Reliability Engineering & System Safety*, **95**(4), (2010), 354-360.

[8] R. I. Cukier, H. B. Levine and K. E. Shuler, Nonlinear Sensitivity Analysis of Multiparameter Model Systems, *Journal of Computational Physics*, **26**(1), (1978), 1-42.

[9] A. Saltelli, S. Tarantola and K. Chan, A quantitative model-independent method for global sensitivity analysis of model output, *Technometrics*, **41**(1), (1999), 39-56.

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

[11] A. Saltelli, Making best use of model evaluations to compute sensitivity indices, *Computer Physics Communication*, **145**(2), (2002), 280-297.

[12] I.M. Sobol', S. Tarantola, D. Gatelli, S.S. Kucherenko and W. Mauntz, Estimating the approximation error when fixing unessential factors in global sensitivity analysis, *Reliability Engineering & System Safety*, **92**(7), (2007), 957-960.

[13] J. P. C. Kleijnen, *Statistical Tools for Simulation Practitioners*, Marcel Dekker Inc, 1987.

[14] R. Jin, W. Chen and A. Sudjianto, On Sequential Sampling for Global Metamodeling in Engineering Design, *Proceedings ASME Design Engineering Technical Conferences DETC and Computers and Information in Engineering Conference CIE*, **2**, (2002), 539-548.

[15] Matlab R2010a, *High-performance numerical computation and visualization software*, The Mathworks Inc., 2010.

[16] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana and S. Tarantola, *Global Sensitivity Analysis - The Primer*, John Wiley & Sons Ltd, 2008.

[17] T. A. Mara, Contribution à la validation d'un logiciel de simulation du comportement thermo-aéraulique du bâtiment : Proposition de nouveaux outils d'aide à la validation , *PhD Dissertation*, University of Reunion, France, (2000). Available from: http://tel.archives-ouvertes.fr/tel-00475440

[18] H. T. Rakotondramiarana, Etude théorique du séchage thermique et de la digestion anaérobie des boues des stations d'épuration - Mise au point des dispositifs pilotes de laboratoire pour la caractérisation expérimentale liée au séchage et à la méthanisation des boues , *PhD Dissertation*, University of Antananarivo, Madagascar, (2004). Available from: http://theses.recherches.gov.mg/pdfs/rakotondramiaranaherytiana_pc_doc3_04.pdf

[19] D. W. Marquardt, An Algorithm for Least-Squares Estimation of Nonlinear

Parameters, *Journal of the Society for Industrial and Applied Mathematics*, **11**(2), (1963), 431-441.

[20] D. Gatelli, S. Kucherenko, M. Ratto and S. Tarantola, Calculating first-order sensitivity measures: A benchmark of some recent methodologies, *Reliability Engineering & System Safety*, **94**(7), (2009), 1212-1219.

[21] J. W. Cooley and J. W. Tukey, An algorithm for the machine calculation of complex Fourier series, *Mathematics of Computation*, **19**(90), (1965), 297-301.

[22] I.M. Sobol', Sensitivity estimates for nonlinear mathematical models, *Matematicheskoe Modelirovanie*, **2**, (1990), 112–118 (in Russian), translated in English in Sobol' [23].

[23] I.M. Sobol', Sensitivity estimates for nonlinear mathematical models, *Mathematical Modeling & Computational Experiment* (English translation of Russian original paper [22]), **1**(4), (1993), 407–414.

Appendix: Significant algorithms in GoSAT (Global Sensitivity Analysis Tool)

In the following algorithms:

- Functions are written in capital letter
- Comments are between brackets {}
- Matlab codes are framed

GLOBAL_SENSITIVITY_ANALYSIS_ALGORITHM (MODEL_OUTPUT,NominalValue,Prange ,freq,alpha,error,n)

----------Input----------

MODEL_OUTPUT : Output of the studied model

NominalValue: contain the nominal value of each factor

Prange: contain the factors range of variation

freq: contain the set of frequency assigned to the factors

alpha: proportionality coefficient α_p (]0,100[)

error: Stop the operation when the Fourier amplitude is less than error

Nt: Define the number of detected frequencies by spectral representation

name: Store the name of the factors

nammod : Store the name of the surveyed model

I,j,k,m,cnt1,cnt2,cnt3,cnt4,cnt5,cnt6,cnt7,Ns,sz,szb1,szb2,szb3,SurvFreq,Nt: integer

y0,nt,alpha,cstop:Real number

ystar,ytild,y2,TFG,TFG1,G:Ns length Vector of real number

Prange:two columns and p rows matrix of real number{the first column store the lower bounds of the parameter and the second the upper}

Freq: p length vector of integer {contain the frequencies of the factors}

Delta: p length vector of real number {contain the frequencies of the factors}

TFG2: E(Ns/2)length vector of real number

EvalFreq,DetectFreq,FinalDetectFreq:Variable length vector of integer

Coefficient,h1i,h1,Hq,ind2:variable length vector of real number

x : Ns rows, p columns matrix of real number

xp,r: variable number of rows, variable number of columns matrix of real number

h2ii,h2ij,h2;Hm:two columns, variable number of rows matrix of real number

h3iii,h3ijj,h3ijk,h3,Hn:three columns ,variable number of rows matrix of real number

---

Start

---------- Step (3) ----------

y0 ← MODEL_OUTPUT (NominalValue)

---------- **Step (4)** ----------

Ns ← 2*4*MAX(Freq) + 1

{MAX(vector):Find the maximum value of the component of the vector }

---------- **Step (5)** ----------

For h ← 1 to LENGTH(Freq)

{LENGTH(vector or matrix) : evaluate the length of the vector or the number of row of a matrix}

Do Delta[h] ← Alpha*(Prange[h , 2] – Prange[h , 1])/100

End For

---------- **Step (6)** ----------

s[1] ← 0

For i ← 1 to Ns-1

Do s[i +1] ← s[i] + 2*pi/(Ns-1)

End For

For j ← 1 to LENGTH(Freq)

Do **For** i = 1 to Ns

Do x[i,j] ← NominalValue[j] + delta[j]*sin(Freq[j]*s[i])

End For

End For

For i ← 1 to Ns

Do ystar[i] ← MODEL_OUTPUT (x[i,1 to LENGTH(Freq)])

{ by using each row of x, evaluate the Ns values of y^* }

End For

---------- **Step (7)** ----------

{Initialize matrices**:** h1i, h1, h2ii, h2ij, h2, h3iii, h3ijj, h3ijk, h3**}**

---------- **Step (8)** ----------

For i←1 to Ns

Do ytild[i] ← y0

End For

Do

m←1

---------- **Step (9)** ----------

For i←1 to Ns

Do G[i]← ystar - ytild

End For

---------- **Step (10)** ----------

TFG←ABS (FFT(G))/Ns

---------- **Step (11)** ----------

ind ←0 {initialize all the components of end as zero}

For i ← 2 to LENGTH(TFG) – 1

Do **If** (TFG[i] > TFG[i +1]) AND (TFG[i] < TFG[i-1]) OR (TFG[i] < TFG[i +1])

AND (TFG[i] > TFG[i-1]) OR (TFG[i] < TFG[i +1]) AND (TFG[i] < TFG[i-1])

Do ind[i] ← i

End If

End For

ind2←FIND1(ind~=0)

{Find among the values of ind those which are not zero and store the index in the vector ind2}

For i←1 to length(ind2)

Do TFG[ind2[i]]←0

End For

```
%matlab code of the step 11
in = zeros(1,length(TFG));
in(1)=1;
in(length(TFG))=1;
for i=2:length(TFG)-1
   if(((TFG(i)>TFG(i+1))&&(TFG(i)<TFG(i-1)))||((TFG(i)<TFG(i+1))&&...
         (TFG(i)>TFG(i-1)))||((TFG(i)<TFG(i+1))&&(TFG(i)<TFG(i-1))))
      in(i)=1;
   end %end if
end %end for
TFG(find(in))=0;
```

---------- **Step (12)** ----------

For i ← 1 to E(Ns/2)

{E(real) : find the fix part of the real}

Do TFG1[i] ← TFG[i]

End For

---------- **Step (13)** ----------

TFG2← DECREASING_ORDER(TFG1)

{DECREASING_ORDER(vector) : put in decreasing order the components of the vector }

nt← TFG2[Nt]

EvalFreq = FILTER(TFG1,nt)

{FILTER(vector,real) :Eliminate the value of the vector which are inferior to the real }

```
%Matlab code of the step 13
TFG2 =unique(sort(TFG1(find(TFG1))));
nt = TFG2(end - Nt);
EvalFreq= find(tfg1 > nt);
EvalFreq = EvalFreq-1;
```

---------- **Step (14)** ----------

cnt1←1,cnt2←1,cnt3←1,cnt4←1,cnt5←1,cnt6←1,cnt7←1

For cnt ← 1 to LENGTH(EvalFreq)

Do SurvFreq ←EvalFreq[cnt]

For i ← 1 to LENGTH(Freq)

Do

```
---------- Step (14-a) ----------
If SurvFreq = Freq[i]
  Do  h1i[cnt1]← i
      cnt1←cnt1+1
      DetectFreq[cnt7]←SurvFreq
      cnt7←cnt7+1
End If
---------- Step (14-b) ----------
If SurvFreq=2*Freq[i]
  Do  h2ii[cnt2,1]←i
      h2ii[cnt2,2]←i
      cnt2←cnt2+1
      DetectFreq[cnt7]←SurvFreq
      cnt7←cnt7+1
End If
For j←i+1 to LENGTH(Freq)
  Do  If  SurvFreq = ABS(Freq[i]-Freq[j])AND   FIND(EvalFreq, Freq[i]+Freq[j])
          {FIND (vector,real):find if the real number is among the conponent of the vector;if it
          is the case,return 0 else 1 }
          Do    h2ij[cnt3,1]←i
                h2ij[cnt3,2]←j
                cnt3←cnt3+1
                DetectFreq[cnt7]← ABS(Freq[i]- Freq[j])
                DetectFreq[cnt7+1]← Freq[i]+Freq[j]
                cnt7←cnt7+2
      End If
End For
---------- Step (14-c) ----------
If SurvFreq = 3*Freq[i]
  Do      h3iii[cnt4,1]←i
          h3iii[cnt4,2]←i
          h3iii[cnt4,3]←i
          cnt4←cnt4+1
          DetectFreq[cnt7]←SurvFreq
          cnt7←cnt7+1
End If
For j←1 to LENGTH(Freq)
  Do      If j=LENGTH(Freq)
            Do BREAK
                {BREAK: Terminate the execution of the current " for"   loop}
          End If
          If j>=i
            Do If   SurvFreq = MIN(ABS(Freq[i] -2*Freq[j+1]), Freq[i]+2*Freq[j+1])
```

```
                AND FIND(EvalFreq, MAX(ABS(Freq[i]-
                2*Freq[j+1]),Freq[i]+2*Freq[j+1])
                Do    h3ijj[cnt5,1]←i
                      h3ijj[cnt5,2]←j+1
                      h3ijj[cnt5,3]←j+1
                      cnt5←cnt5+1
                      DetectFreq[cnt7]← ABS(Freq[i]-2*Freq[j+1])
                      DetectFreq[cnt7+1]← Freq[i]+2*Freq[j+1]
                      cnt7←cnt7+2
            End If
        End If
        If j < i
          Do If SurvFreq = MIN(ABS(Freq[i]-2*Freq[j]),Freq[i]+2*Freq[j])
                AND FIND( Evalfreq,MAX(ABS(Freq[i]-2*Freq[j]))
                AND FIND( Evalfreq,Freq[i]+2*Freq[j])
                Do    h3ijj[cnt5,1]←i
                      h3ijj[cnt5,2]←j
                      h3ijj[cnt5,3]←j
                      cnt5←cnt5+1
                      DetectFreq[cnt7]← ABS(Freq[i]-  2*Freq[j])
                      DetectFreq[cnt7+1]← Freq[i]+2*Freq[j]
                      cnt7←cnt7+2
            End If
        End If
End For
For j←i+1 to LENGTH(Freq)
  Do For k←j+1 to LENGTH(Freq)


          Do If   SurvFreq= MIN(Freq[i]+Freq[j]+Frequecne[k],ABS(Freq[i]
                -Freq[j]+Frequecne[k]), ABS(Freq[i]+Freq[j]-Freq[k]),
                ABS(Freq[i]-Freq[j]-Freq[k]))
                AND   FIND(Evalfreq,Freq[i]+Freq[j]+Freq[k])
                AND FIND(Evalfreq,ABS(Freq[i]-Freq[j]+Freq[k]))
                AND FIND(Evalfreq,ABS(Freq[i]+Freq[j]-Freq[k]))
                AND FIND(Evalfreq,ABS(Freq[i]-Freq[j]-Freq[k]))
                Do  h3ijk[cnt5,1]←i
                    h3ijk[cnt5,2]←j
                    h3ijk[cnt5,3]←k
                    cnt5←cnt5+1
                    DetectFreq[cnt7]←Freq[i]+Freq[j]+Freq[k]
                    DetectFreq[cnt7+1]← ABS(Freq[i]- Freq[j]+Freq[k])
                    DetectFreq[cnt7+2]← ABS(Freq[i]+Freq[j] Freq[k])
                    DetectFreq[cnt7+3]← ABS(Freq[i]-Freq[j]-  Freq[k])
```

cnt7←cnt7+4

End If

End For

End For

End For

End For

FinalDetectFreq←NO _DOUBLE(DetectFreq)

{NO_DOUBLE(vector) : Eliminate the redundant elemnts of the vector }

```
%Matlab code of the NO_DOUBLE function
function fs = NO_DOUBLE(freqs)
for i = 1:length(freqs)
   for j = i+1 :length(freqs)
      if freqs(i) ==freqs(j)
         freqs(j)=0;
      end
   end
end
fs = freqs(find(freqs));
```

h1 ←hi

h2←CONCATENATE(h2ii,h2ij)

{CONCATENATE(matrix1,matrix2) : Put the matrix2 at the end of the matrix1, both have the same number of columns }

```
%On Matlab, to concatenate two( or three,…) matrix which have the same number
of column, we use the next syntax
h2 = [h2ii;h2ij];
```

h3←CONCATENATE(h3iii,h3ijj,h3ijk)

If i=1

Do For j←1 to LENGTH(h1)

Do Hq[j]←h1[j]

End For

For j←1 to LENGTH(h2)

Do Hm[j]←h2[j]

End For

For j←1 to LENGTH(h3)

Do Hn[j]←h3[j]

End For

Else

Do Hq←CONCATENATE(h1p,h1)

Hm←CONCATENATE(h2p,h2)

Hn←CONCATENATE(h3p,h3)

End If

For j←1 to LENGTH(Hq)

```
    Do   h1p[j]←Hq[j]
End For
For j←1 to LENGTH(Hm)
    Do h2p[j]←Hm[j]
End For
For j←1 to LENGTH(Hn)
    Do h3p[j]←Hn[j]
End For
---------- Step (15) ----------
szb1←LENGTH(Hq)
szb2←LENGTH(Hm)
szb3←LENGTH(Hn)
cnt1←1
cnt2←1
cnt3←1
sz ←szb1+szb2+szb3
For   j ←1 to LENGTH(Freq)
    Do For i←1 to N
            Do   x[i,j]←sin(Freq[j]*s[i])
        End For
End For
For i←1 to sz
    Do If i<= szb1
            Do  For j←1 to Ns
                    Do xp[j,i]←x[j,Hq[cnt1]]
                End For
                cnt1←cnt1 + 1
        End If
    Do If (i > szb1) AND (i<= szb1 + szb2)
            Do  For j←1 to Ns
                    Do xp[j,i]←x[j,Hm[cnt2,1]]* x[j,Hm[cnt2,2]]
                End For
                cnt2←cnt2+1
        End If
    Do If (i > szb1 + szb2) AND ( i <= sz)
            Do  For j←1 to Ns
                    Do xp[j,i]←x[j,Hn[cnt3,1]]* x[j,Hn[cnt3,2]]* x[j,Hn[cnt3,3]]
                End For
                cnt3←cnt3+1
        End If
End For
x0←MATRIX_DIVIDE(ystar,xp)
{Divide the matrix ystar by the matrix xp}
Coefficient ← LEVENBERG-MARQUARDT_ALGORITHM(y0,ystar,xp,x0)
```

```
%Matlab code of the step 15
a = size(Hq);
b = size(Hm);
c = size(Hn);
szb1 = a(1);
szb2 = b(1);
szb3 = c(1);
if isempty(Hq)
   szb1 =0;
end%end if
if isempty(Hm)
   szb2=0;
end%end if
if isempty(Hn)
   szb3 = 0;
end%end if
sz = szb1 + szb2 + szb3;
x = zeros(length(s),4);
ct1 =1;
ct2=1;
ct3=1;
xp = zeros(length(s),sz);
for i=1 : length(f)
   x(:,i) = sin(f(i)*s);
end%end for
for i = 1 : sz
      if i <= szb1
         xp(:,i) = x(:,Hq(ct1));
         ct1=ct1+1;
      end %end if
      if (i > szb1)&&(i <= szb1+ szb2)
         xp(:,i) = x(:,Hm(ct2,1)).*x(:,Hm(ct2,2));
         ct2 = ct2 + 1;
      end  %end if
      if (i > szb1+ szb2)&&(i<=sz)
         xp(:,i) = x(:,Hn(ct3,1)).*x(:,Hn(ct3,2)).*x(:,Hn(ct3,3));
         ct3 = ct3 + 1 ;
      end % ind if
end %end for
x12 = xp\ystar;
```

```
x0=x12';
options = optimset('LevenbergMarquardt','on','MaxFunEvals',...
5000,'TolX',1e-7,...
                'LargeScale','on','Display','on');
 [Coefficient]=lsqnonlin(@newfcn1,x0,[],[],options,y0,ystar,xp);
%with newfcn1 (saved in another Matlab file)
function dy = newfcn1(a,y0,y,xvar)
szxvtmp = size(xvar);
szxv = szxvtmp(2);
r = 0;
   for i = 1:szxv
   r = a(i)*xvar(:,i) + r;
   end
dy = y -(r+y0);
```

---------- **Step (16)** ----------

DRAW_SPECTRUM (FinalDetectFreq,TFG1,nammod)

```
%Matlab code of the step 16
figure(m);
   set(gcf,'Name','Sensivity Analysis Tool','NumberTitle','off');
   set(gcf, 'Color',[0.50 0.50 0.50]);
   axes1 = axes('Parent',figure(m),'FontSize',8,'FontName','Times');
   ylim(axes1,[0 max(TFG1)+0.4*max(TFG1)]);
   box(axes1,'on');
   hold(axes1,'all');
      plot(1:ceil(length(s)/2),TFG1);
   lx = xlabel('Frequency','FontSize',9,...
     'FontName','Times','fontweight','b');
   set(lx);
   ly = ylabel('|Fourier Amplitude|',...
   'FontSize',9,...
   'FontName','Times','fontweight','b');
   set(ly);
   if (m==1)
   lt = title({['Sensitivity analysis of the ' nammod ...
      ' - FAST Method'],...
   ['Spectrum of the frequencies associated to'...
   ' the dominating parameters - Main figure ']},...
   'FontSize',9,...
   'FontName','Times','fontweight','b');
```

```
    set(lt);
    else
    lt = title({['Sensitivity analysis of the ' nammod ...
        ' - FAST Method'],...
    ['Spectrum of the frequencies associated '...
    'to the dominating parameters -'...
    num2str(m) '^{th} figure']},...
    'FontSize',9,...
    'FontName','Times','fontweight','b');
    set(lt);
    end%end if
    %show detected frequencies at step 14 by using textarrow properties
    for i =1: length(FinalDetectFreq)
           [figx,figy]=dsxy2figxy(gca,...
              [ FinalDetectFreq(i)+1 FinalDetectFreq(i)+1],...
              [tfg1(FinalDetectFreq(i)+1)+0.3*max(tfg1)...
              tfg1(FinalDetectFreq(i)+1)]);
           annotation(figure(m),'textarrow',figx,...
            figy,'TextEdgeColor','none','String',...
            num2str(FinalDetectFreq(i)),...
             'HeadLength',5,'HeadWidth',4,...
            'HeadStyle','vback1','fontsize',...
            10,'fontname','times','LineStyle',':');
    end%end for
```

---------- **Step (17)** ----------

r←0

For i←1 to sz

Do For j←1 to Ns

Do r[j] ← Coefficient[i]*xp[j,i]+ r[j]

End For

End For

For j←1 to Ns

Do ytild[j] ← r[j] + y0

End For

---------- **Step (18)** ----------

TFG3←FIND_REMAINING_FA(TFG1,nt)

{find the remaining Fourier amplitude which are under the number nt}

```
%Matlab code which computes TFG3
TFG3 = TFG11(find(TFG1<nt));
```

If m= 1

Do cstop=0;
Else cstop=0;
Do For i = 1:length(oldfreq)
Do For j = 1 :length(FinalDetectFreq)
If oldfreq[i] =[j]
Do cstop=1;
End If
End For
End For
End If
oldfreq ← FinalDetectFreq
m←m+1

While (MAX(TFG3) > error) AND (cstop =0) AND (0 = ISEMPTY(FinalDetectFreq))

{ISEMPTY (vector) : return 1 if the vector is void, otherwise return 0}

---------- **Step (19)** ----------

WRITE_THE_METAMODEL(Hq,Hm,Hn,Coefficient)

```
%Matlab code of the step 18
ct1 =1;
ct2 = 1;
ct3 = 1;
nm1 ='';
nm2 = '';
nm3 = '';
%write the metamodel name on the Matlab command window
    for i= 1 : sz
        if i<= szb1
            nm1{ct1} = [ num2str(Coefficient(i)) ...
                ' * <' name{Hq(ct1)} '> + '];
            ct1 = ct1 + 1;
        end
        if (i > szb1)&&(i <= szb1 + szb2)
            nm2{ct2} = [num2str(coeffficient(i)) ...
                ' * <' name{Hm(ct2,1)} ...
                '> * <' name{Hm(ct2,2)} '> + ' ];
            ct2 = ct2 + 1;
        end
        if i > szb1 + szb2
            nm3{ct3} = [num2str(coefficient(i)) ...
                ' * < ' name{Hn(ct3,1)} ...
                '> * <' name{Hn(ct3,2)} ...
```

```
                '> * <' name{Hn(ct3,3)} '> + '];
            ct3 = ct3+1;
        end
    end
    namemetamod = {num2str(y0),'+',nm1,nm2,nm3,'...'};
```

```
% Write the metamodel name in a .txt file named metamodel.txt
    fid = fopen('metamodel.txt', 'w');
    fprintf(fid, ' %s', [num2str(y0) ' + ']);
    for i =1 :szb1
    fprintf(fid, ' %s', nm1{i});
    end%end for
    fprintf(fid,'\n');
    if (0==isempty(nm2))
        for j = 1 : szb2
           fprintf(fid, ' %s', nm2{j});
        end%end for
    end%end if
     fprintf(fid,'\n');
    if (0==isempty(nm3))
        for k = 1 : szb3
           fprintf(fid, ' %s', nm3{k});
        end%end for
    end%end if
    fprintf(fid,' ...');
    fclose(fid);
    % view the contents of the file
    type metamodel.txt
```

---------- **Step (20)** ----------

VALIDATION(ystar,ytild)

```
m = [ystar,ytild];
r2 = regress(ystar,ytild);
dm = abs(diff(m,1,2))./ m(:,1);
me = mean(dm) ;
[sm,I]= sort(m(:,1)) ;
figure ;
set(gcf,'Name','Sensivity Analysis Tool','NumberTitle','off');
set(gcf, 'Color',[0.50 0.50 0.50]);
set(gca,'FontSize',8,'FontName','Times');
```

```
%Plot the relative error of the metamodel compared to the original model
plot(sm,dm(I)*100);
xlabel([modnam 'output []'],'FontSize',9,...
    'FontName','Times','fontweight','b');
   % Create ylabel
ylabel('Relative error [%]',...
    'FontSize',9,...
    'FontName','Times','fontweight','b');
   % Create title
title({['Relative error of the metamodel '...
    'output on the' nammod ' output']},...
    'FontSize',9,...
    'FontName','Times','fontweight','b');
figure1 = figure;
set(gca,'FontSize',8,'FontName','Times');
set(gcf,'Name','Sensivity Analysis Tool','NumberTitle','off');
set(gcf, 'Color',[0.50 0.50 0.50]);
%compare the metamodel output and the original output
plot(ystar,ytild,'.');
   %create xlabel
xlabel(['Prediction of ' nammod ' []'],'FontSize',9,...
    'FontName','Times','fontweight','b');
   % Create ylabel
ylabel(['Prediction of the metamodel associated to the '...
    nammod ' []'],...
    'FontSize',9,...
    'FontName','Times','fontweight','b');
   % Create title
title({['Comparison of the metamodel output and the '...
    nammod ' output']},...
    'FontSize',9,...
    'FontName','Times','fontweight','b');
```

---------- **Step (21)** ----------

CONPARING_THE_REGRESSION_COEFFICIENT (Coefficient,Hq,Hm)

End

```
%Matlab code of the step 21
ydat = abs(Coefficient);
long1 = length(Hq);
szb2 = size(Hm);
long2 = szb2(1);
ydat1 = ydat(1:long1);
ydat2 = ydat(long1+1 :long1+long2);
nxticklab1 = '';
nxticklab2 = '';
%first order effects
if (long1>1)
    [y1,ind] = sort(ydat1);
    figure1 = figure;
    set(gcf,'Name','Sensivity Analysis Tool','NumberTitle','off');
    set(gcf, 'Color',[0.50 0.50 0.50]);
    for j = 1 :length(y1)
       nxticklab2{j} = num2str(length(y1)-j+1);
    end%end for
    % Create axes
    axes1 = axes('Parent',figure1,'FontSize',8,'FontName','Times');
    set(gca,'YTick',[1:length(y1)],'YTickLabel',nxticklab2);
    xlim(axes1,[0 max(ydat1)+0.5*max(ydat1)]);
    box(axes1,'on');
    hold(axes1,'all');
    %Create a bar diagram of the first order effect
    bar(y1,...
    'FaceColor',[0.450980392156863 0.262745098039216...
    0.262745098039216],...
    'Horizontal','on',...
    'BarWidth',0.1,...
    'BarLayout','stacked');
    %create xlabel
    xlabel('Regression coefficient of the metamodel','FontSize',9,...
    'FontName','Times','fontweight','b');
    % Create ylabel
    ylabel('Decreasing order of the dominating parameters',...
    'FontSize',9,...
    'FontName','Times','fontweight','b');
    % Create title
    title({['Comparison of the  most influent parameters of the'...
```

```
       modnam ] },...
    'FontSize',9,...
    'FontName','Times','fontweight','b');
    % Create textbox
    annotation(figure1,'textbox',...
    [0.659928571428571 0.133333333333334 0.225785714285714...
    0.0857142857142879],...
    'String',{'(+) Positive effect on the output',...
    '(-) Negative effect on the output'},...
    'HorizontalAlignment','center',...
    'FontWeight','light',...
    'FontSize',7,...
    'FontName','Times',...
    'FitHeightToText','on');
% Create textbox
    annotation(figure1,'textbox',...
    [0.660714285714286 0.22857142857143 ...
    0.223214285714286 0.130952380952382],...
    'String',{' surveyed output : ',[ modnam ' output']},...
    'FontSize',6,...
    'FontName','Times',...
    'FitHeightToText','off');
    %annotation
    %write on the figure the name of the factor corresponding to the effect
    for i=1:long1
    if (Coefficient(ind(i))>0)
       [figx,figy]=dsxy2figxy(gca,[ max(y1(i))+0.09*max(ydat1)...
          y1(i)],[i i]);
       annotation(figure1,'textarrow',figx,...
        figy,'TextEdgeColor','none','String',['<' ...
        name{Hq(ind(i))} '>-- (+)'],...
        'HeadLength',6,'HeadWidth',6,...
       'HeadStyle','vback1','FontSize',8);
    end%end if
       if(Coefficient(ind(i))<0)
       [figx,figy]=dsxy2figxy(gca,[max(y1(i))+0.09*max(ydat1)...
          y1(i)],[i i]);
       annotation(figure1,'textarrow',figx,...
        figy,'TextEdgeColor','none','String',['<' ...
```

```
        name{Hq(ind(i))} '>-- (-)'],...
        'HeadLength',6,'HeadWidth',6,...
          'HeadStyle','vback1','FontSize',8 );
       end%end if
    end%end for
end%end if
%second order effects
if (long2>1)
    figure2 = figure;
    set(gcf, 'Color',[0.50 0.50 0.50]);
     set(gcf,'Name','Sensivity Analysis Tool','NumberTitle','off');
    [y2,ind2] = sort(ydat2);
    for j = 1 :length(y2)
      nxticklab1{j} = num2str(length(y2)-j+1);
    end%end for
    % Create axes
axes1 = axes('Parent',figure2,'FontSize',8,'FontName','Times');
    set(gca,'YTick', [ 1:length(y2)] ,'YTickLabel',nxticklab1);
     xlim(axes1,[0 max(ydat2)+0.4*max(ydat2)]);
    box(axes1,'on');
    hold(axes1,'all');
    %Create a bar diagram of the second order effect
    bar(y2,...
    'FaceColor',[0.450980392156863,0.262745098039216,...
0.262745098039216],...
    'Horizontal','on',...
    'BarWidth',0.1,...
    'BarLayout','stacked');
    xlabel('Regression coefficient of the metamodel','FontSize',9,...
    'FontName','Times','fontweight','b');
    % Create ylabel
    ylabel('Decreasing order of the parameter''s order 2 interaction',...
    'FontSize',9,...
    'FontName','Times','fontweight','b');
    % Create title
    title({'Comparison of the order 2 interaction of the model'},...
    'FontSize',9,...
    'FontName','Times','fontweight','b');
%write on the figure the set of factors corresponding to the interaction
   %effects
```

```
    for i=1:long2
        if (Coefficient(long1+ind2(i))>0)

            [figx,figy]=dsxy2figxy(gca,[ max(ydat2)+0.1*max(ydat2)...
               y2(i)],[i i]);
            annotation(figure2,'textarrow',figx,...
            figy,'TextEdgeColor','none','String',...
             ['<' name{Hm(ind2(i),1)} ...
            '>*<' name{Hm(ind2(i),2)} '> -- (+)'],...
            'HeadLength',6,'HeadWidth',6,...
            'HeadStyle','vback1','FontSize',8);
        end% end if
        if(Coefficient(long1+ind2(i))<0)

        [figx,figy]=dsxy2figxy(gca,[max(ydat2)+0.1*max(ydat2)...
            y2(i)],[i i]);
        annotation(figure2,'textarrow',figx,...
         figy,'TextEdgeColor','none','String', ['<' name{Hm(ind2(i),1)}...
         '>*<' name{Hm(ind2(i),2)} '>-- (-)'] ,...
         'HeadLength',6,'HeadWidth',6,...
          'HeadStyle','vback1','FontSize',8);
        end%end if
    end% end for
```