Using LS-OPT for meta-model based global sensitivity analysis

Uwe Reuter, Zeeshan Mehmood, Clemens Gebhardt
Department of Civil Engineering, TU Dresden, Germany

Martin Liebscher, Heiner Müllerschön, Ingolf Lepenies
DYNAmore GmbH, Dresden, Germany

Summary:

Popular sensitivity analysis methods such as ANOVA and SOBOL indices are widely used in LS-OPT in order to measure the importance of different input variables with respect to the model response. These methods are applied using meta-models in LS-OPT. In contrast, sensitivity information can be directly extracted from the meta-models using weight-based and derivative-based approaches. Meta-models capture the non-linear relationship of the underlying input parameters to the design response. In this paper, powerful sampling and pre-processing capabilities of LS-OPT are coupled with a user-defined neural network based meta-model in order to perform weight based and derivative based sensitivity analysis. The results of these sensitivity measures are compared with the default SOBOL approach by using an analytical as well as an industry relevant crash analysis example.

Keywords:

Sensitivity analysis, variable screening, neural networks.

1 Introduction

Sensitivity analysis helps in identifying the most significant model parameters affecting a specific model response. Each individual model parameter $x_{i,\, i=1,2,\cdots,n}$ is compared with the other remaining model parameters $x_1, \cdots x_{i-1}, x_{i+1}, \cdots, x_n$ in order to evaluate its influence on the model response z.

The normalized sensitivity measure S_i for a model parameter $x_{i,\, i=1,2,\cdots,n}$ is given by:

$$S_i = \frac{\widetilde{S}_i}{\sum_{j=1}^{n} \widetilde{S}_j} \tag{1}$$

where $\widetilde{S}_i$ represents the influence of x_i on z according to a specific sensitivity measure.

Already existing variance based sensitivity analysis techniques such as ANOVA and SOBOL indices in LS-OPT provide global sensitivity information for linear/quadratic and non-linear models respectively [9, 5, 11]. These techniques are applied in LS-OPT using a meta-model. In this paper, properties of neural networks are exploited for directly extracting sensitivity information. For this, a tool named PySen is developed which contains a neural network implementation and a sensitivity analysis module. The sensitivity analysis module contains the weight-based and derivative based method implementations. The tool is attached to LS-OPT using a well-defined meta-model interface. Finally, a wrapper is written which automates and simplifies the use of LS-OPT for calculating sensitivity information together with PySen. The wrapper also facilitates the comparison of different methods and the generation of the results. In section 2, different global sensitivity analysis methods are briefly explained. Section 3 highlights the software system architecture including LS-OPT, Interface and PySen which is used to perform sensitivity analysis for the examples presented and evaluated in section 4. Section 5 concludes the paper.

2 Global sensitivity analysis methods

2.1 Variance based methods

Advancement in computing power has facilitated the use of variance based methods that can accommodate non-linearity and interactions within a model and its variables. These methods are usually model independent so they can be used on models whose algorithms are complex or are not understood deterministically. One such method is the variance based decomposition [12, 10]. A response $y = f(x)\,, x = (x_1, \cdots, x_n)$ can be represented as

$$f(x) = f_o + \sum_{i}^{n} f_i(x_i) + \sum_{1 \le i < j \le n} f_{ij}(x_i, x_j) + \cdots + f_{1,2,\cdots,n}(x_1, \cdots, x_n) \tag{2}$$

Such a decomposition of $f(x)$ is termed as variance based decomposition. The function $f(x)$ is characterized by its variance V which can be decomposed into partial variances associated with $x_1, x_2 \cdots, x_n$ according to Eq. (2) as

$$V = \sum_{i=1}^{n} V_i + \sum_{1 \le i < k \le n} V_{i,k} + \cdots + V_{1,2,\cdots,n} \tag{3}$$

Each partial variance $V_{i_1,\cdots,i_s}$ can be related to each of the sensitivity measures $S_{i_1,\cdots,i_s}$ as

$$S_{i_1,\cdots,i_s} = \frac{V_{i_1,\cdots,i_s}}{V},\ 1 \le i_1 < \cdots i_s \le n,\ s = 1,2,\cdots,n \tag{4}$$

In order to evaluate the total effect of a single variable x_i, all partial sensitivity measures S_i involving x_i are summed up to define total sensitivity measure S_{T_i}. The total sensitivity measures consider the interactions among all model parameters. In order to quantify which amount of variance V is caused due to a single variable x_i, the corresponding total sensitivity measures S_{T_i} can be normalized as

$$norm\, S_{T_i} = \frac{S_{T_i}}{\sum_{k=1}^{n} S_{T_k}} \tag{5}$$

The total sensitivity measure S_{T_i} as shown in Eq. (5) can be numerically computed using the SOBOL approach [12, 10] which uses the Monte Carlo simulation. S_{T_i} associated with each input variable x_i can be computed as $S_{T_i} = 1 - S_{\sim i}$, where $S_{\sim i} = \frac{V_{\sim i}}{V}$ and

$$V_{\sim i} \approx \frac{1}{N} \sum_{k=1}^{N} f\left(x_{\sim ik}^{(1)}, x_{ik}^{(1)}\right) f(x_{\sim ik}^{(1)}, x_{ik}^{(2)}) - f_0^2 \tag{6}$$

The superscripts (1) and (2) indicate that two different samples are generated and mixed. $x_{\sim ik}^{(1)}$ denotes the kth sample point with $x_{\sim ik}^{(1)} = (x_{1k}^{(1)}, \cdots, x_{(i-1)}^{(1)}, x_{(i+1)k}^{(1)}, \cdots, x_{nk}^{(1)})$ and $f_0 = \frac{1}{N} \sum_{m=1}^{N} f(x_k)$ is the mean and N is the number of simulation.

2.2 Neural Network based methods

A trained neural network can reason about the behavior of the model. Features of the neural network can be used to determine the sensitivity measures [3, 7]. Values stored in the static matrix of weights can be used to determine the relative influence of each input variable on the network response. Different equations have been proposed which calculate the products of the weights of the network and then obtaining the sum of the calculated products according to a certain criteria [13, 4]. These methods are broadly characterized as weight based sensitivity measures. On the other hand, using the derivability property of the neural network, derivative based sensitivity measures can also be determined [6].

2.2.1 Weight based sensitivity measures

The procedure termed as weights method was first introduced for a single hidden layer neural network by Garson [2] in which hidden-output connection weights are partitioned into components associated with each input neuron using absolute values of connection weights as

$$S_{ik} = \frac{\sum_{j=1}^{L}\left(\frac{w_{ij}}{\sum_{r=1}^{N} w_{rj}} w_{jk}\right)}{\sum_{i=1}^{N}\left(\sum_{j=1}^{L}\left(\frac{w_{ij}}{\sum_{r=1}^{N} w_{rj}} w_{jk}\right)\right)} \tag{7}$$

where w_{ij} is the weight associated with the input neuron i and the hidden neuron j and w_{jk} is the weight associated with the hidden neuron j and output neuron k, $\frac{w_{ij}}{\sum_{r=1}^{N} w_{rj}}$ is normalized value of the connection weight, N is the total number of input neurons and L is the total number of hidden neurons. If there is only one output neuron or response variable, as in the case of the examples used in this paper, the subscript k can be dropped.

Another equation has been proposed by Tchaban et al **[13]** in which the impact of each input is evaluated on the neural network's output through the connections with the hidden layer. A neuron input has a positive impact on the neuron output if the input value multiplied by the weight is positive and vice versa. This impact is established as a proportion of a particular input in the neuron output and stated as $S_{ij} = \frac{x_i w_{ij}}{o_j}$ where S_{ij} is the impact of the input i on the output of the neuron j, x_i is the value of the input i, w_{ij} is the weight between the input neuron i and the hidden neuron j, and o_j is the output from the hidden neuron j. Similarly the impact of the of the hidden neuron on the network output is given by $S_{jk} = \frac{o_j w_{jk}}{o_k}$. Altogether, the impact of the of the input i on the output neuron k through the hidden neuron j is equal to the product of the impact of the input i on the output of the hidden neuron j with the impact of the hidden neuron j on the network output neuron k. Thus, $S_i = \sum_j^L S_{ij} S_{jk}$, where S_i is the overall impact of the input i on the output neuron k. Substituting the value of S_{ij} and S_{jk} yields

$$S_i = \frac{x_i}{o_k} \sum_{j=1}^{L} w_{ij} w_{jk} \tag{8}$$

Equation (8) has been modified in [8, 14] for more than one hidden layers in which the impact or the sensitivity measure S_i is formulated for the number of layers $k \in \{1, \cdots, s\}$ and with $j_k \in \{1, \cdots, N_k\}$ neurons per layer k respectively as

$$S_i = \sum_{j_{s-1}=1}^{N_{s-1}} \cdots \sum_{j_2=1}^{N_2} \left| w_{i,j_2}^1 . w_{j_2,j_3}^2 . \cdots . w_{j_{s-1},1}^{s-1} \right| \tag{9}$$

Weights in Eq. (12) are taken as absolute values in contrast to Eq. (8) in order to evaluate the total influence. This method is taken as the standard weight based sensitivity measure in this paper.

Another sensitivity measure termed as activation weight based method is mentioned in [8, 14] in which the values of the activations behind neurons $K \geq 2$ are also considered due to the fact that weights are always multiplied by the activation y_p^k of the previous neuron. Equation (10) is used for calculating activation weight based sensitivity measures in this paper where s is the total number of hidden layers and N is the total number of neurons in each hidden layer.

$$S_i = \sum_{j_{s-1}=1}^{N_{s-1}} \cdots \sum_{j_2=1}^{N_2} \left| w_{i,j_2}^1 \right| . \left| y_{j_2}^2 \right| . \left| w_{j_2,j_3}^2 \right| . \left| y_{j_3}^3 \right| . \cdots . \left| y_{j_{s-1}}^{s-1} \right| . \left| w_{j_{s-1},1}^{s-1} \right| \tag{10}$$

2.2.2 *Derivative based sensitivity measures*

Partial derivatives are quite useful in order to determine the importance of the input variables because they represent the instant slope of the underlying function between each pair of input x_i and output y_k, where i is the number of input neurons and k is number of output neurons. Montano et al [6] presented a partial derivative based method using a single hidden layer neural network for determining sensitivity measures with the help of the following equation.

$$S_{ik} = \frac{\partial y_k}{\partial x_i} = f'(net_k) \sum_{j=1}^{L} w_{ij} f'(net_j)\, w_{jk}. \tag{11}$$

In Eq. (11) S_{ik} denotes the sensitivity of the output y_k due to changes in the input variable x_i, $f'(net_j)$ and $f'(net_k)$ are the derivatives of the activation function of the hidden neuron j and the output neuron k respectively. net_k is the output of the k^{th} neuron of the output layer. Equation (11) is further elaborated and extended for a single layer and a two hidden layers network in [14] . For a single hidden layer network the formula is

$$\widetilde{S}_i^{3Layer} = \sum_{k=1}^{N_2} \left| w_{ik}^1 . w_{k1}^2 . der \left(w_{0k}^2 + \sum_{j=1}^{N_1} w_{jk}^1 . o_j^1 \right) \right| \tag{12}$$

where, $w_{0k}^2 + \sum_{j=1}^{N_1} w_{jk}^1 . o_j^1$ is the input to the single neuron in output layer K. Eq. (12) is extended for a two hidden layers network as:

$$\widetilde{S}_i^{4Layer} = \sum_{l=1}^{N_3} \left| w_{l1}^3 . der \left[w_{0l}^3 + \sum_{k=1}^{N_2} w_{kl}^2 . act \left(w_{0k}^2 + \sum_{j=1}^{N_1} w_{jk}^1 . o_j^1 \right) \right] \cdots \sum_{k=1}^{N_2} w_{ik}^1 . w_{kl}^2 . der \left(w_{0k}^2 + \sum_{j=1}^{N_1} w_{jk}^1 . o_j^1 \right) \right| \tag{13}$$

3 Using LS-OPT with user-defined meta models

3.1 System Architecture

The software system consists of four components namely: Wrapper, LS-OPT, Interface and PySen, see Fig. 1. The wrapper automates and coordinates the execution of the test examples and cases. LS-OPT acts as an intermediate layer for generating the sample points and the responses with the help of a solver (LS-DYNA, Perl script etc.). Interface binds PySen with LS-OPT, while PySen acts as a user-defined meta-model with sensitivity analysis capabilities. All components excluding LS-OPT are implemented in Python 2.6.

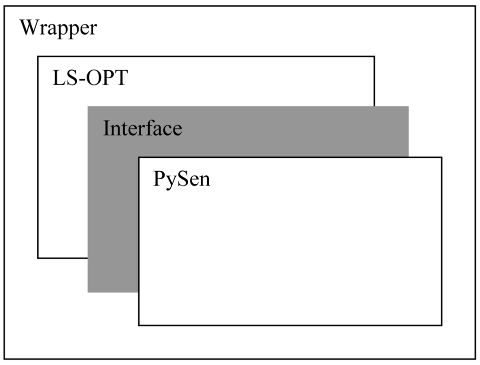


Figure 1: Architecture Overview

3.2 LS-OPT

LS-OPT is an optimization tool which helps in design optimization, design of experiments, reliability studies and sensitivity analysis. It uses meta-models for optimization and sensitivity analysis. Since the accuracy of the meta-models is dependent on many factors such as the size of the sub-region and the number and distribution of the design points, LS-OPT provides several point selection procedures such as factorial, composite, D-optimal, Latin Hypercube and space filling for this purpose. LS-OPT acts as a powerful design point and response generator with its parallel processing and diverse remote job scheduling and queuing facilities. LS-OPT provides facility for attaching a user-defined meta-model which can use the sampling strategies of LS-OPT.

3.3 Interface

The Interface acts as a connection component between LS-OPT and PySen. It is built using well-defined interface functions provided by LS-OPT for connecting user-defined meta-models. It is a dynamic link library/shared object file (Linux) which contains embedded python calls for invoking the meta-model and sharing design points with PySen. Definitions for the interface functions are stated in Tab. 1.

void Build(float coordinates[][], float values[]) Call for building meta-model based on sampling points given by LS-OPT.	
coordinates:	List of the sample points containing coordinates. Every point is a list of coordinates as well. For example: [[1,2,2],[4,3,5]]
values:	List of response values for the sample points.

float FuncVal(float coordinates[]) Returns the response value of the meta-model at a specific point.	
coordinates:	List of the coordinates of a sample point. For example: [1,2,2]

float FuncGrad(float coordinates[]) Returns the derivative of a specific point in the meta-model.	
coordinates:	List of the coordinates of a sample point. For example: [1,2,2]

Table 1: Definitions of interface functions

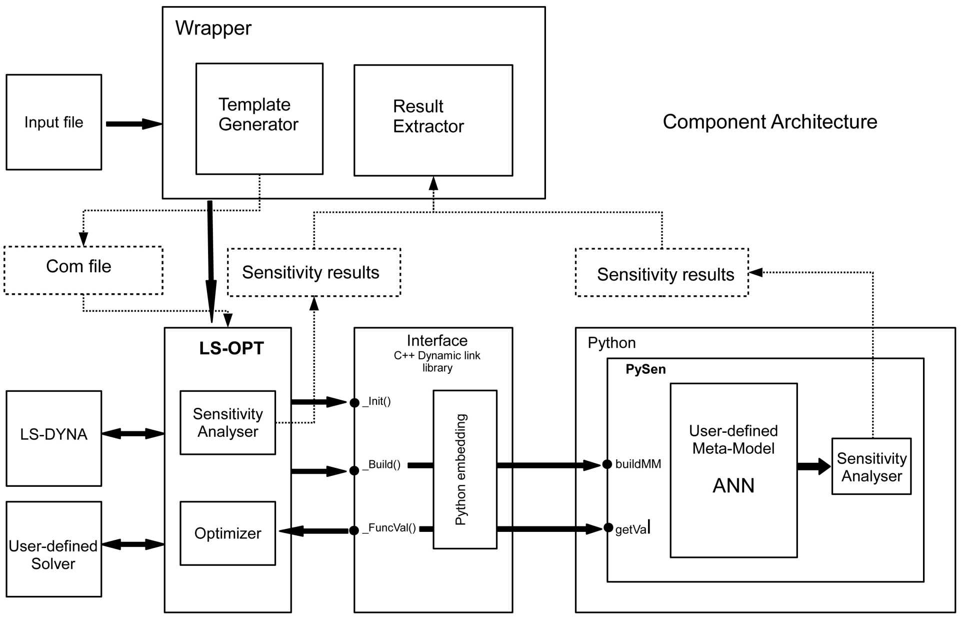


Figure 2: Software components and their interaction

3.4 PySen

PySen is a meta-model based sensitivity analysis tool which is developed for performing sensitivity analysis with the help of different methods. It is written in Python programming language. It contains a multi-layered neural network implementation and modules for calculating different neural network based sensitivity measures. These include derivative based, weight based and activation based methods. It also includes a SOBOL implementation. It has a flexible design and contains a Template and Hook pattern architecture for connecting new methods.

3.5 Wrapper

The Wrapper is an umbrella layer that automates the execution of different test cases for sensitivity analysis, see Fig. 2. It contains a template generator which accepts input files with simple statements and invokes LS-OPT with automatically generated ”command” files. It provides a simplified interface to LS-OPT focusing on executions related to sensitivity analysis. Fig. 3 shows a simple input file for wrapper and the generated COM file with text in highlight showing the default options. The language of the input file consists of the valid Python statements. The wrapper also automates the execution of the test cases for different number of design points in order to analyze their effects on the model sensitivity.

```
# Input script for Wrapper. Case: Ishigami.
# Variables
# Continuous: for continuous variables = [name, init value, min value, max value].
# Discrete: for discrete variables = [name, init value, [range]].
# ----------------------------------------
Continuous('var1', 0, -3.14, 3.14)
Continuous('var2', 0, -3.14, 3.14)
Continuous('var3', 0, -3.14, 3.14)
# ----------------------------------------
# Response Variables.
Response('Response', 1, 0)
# ----------------------------------------
# Solver: Contains the solver parameters.
# arg1: name of the solver.
# arg2: The file which stores and returns the response variable values.
# ----------------------------------------
Solver('Ishigami.perl', 'Response')
# ----------------------------------------
# Case(name)
Case('solver')
# ----------------------------------------
# Number of design points for testing.
Points(400,200,50)
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Command file "poly-2-eval-bat"
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Generated using Wrapper
"Ishigami"
solvers 1
responses 1
$ DESIGN VARIABLES
variables 3
Variable 'var1' 0
  Lower bound variable 'var1' -3.14.
  Upper bound variable 'var1' 3.14.

Variable 'var2' 0
  Lower bound variable 'var2' -3.14.
  Upper bound variable 'var2' 3.14.

Variable 'var3' 0
  Lower bound variable 'var3' -3.14.
  Upper bound variable 'var3' 3.14.

Optimization Method SRSM
$      SOLVER "Ishigami"
$ DEFINITION OF SOLVER "Ishigami"
 solver own 'Ishigami'
  solver command "perl"
  solver input file "Ishigami.perl"
$ ------ Metamodeling ---------
solver order user
  solver user metamodel 'pyInterface'
  solver experiment design space_filling
   solver number experiments 400
$ ------ Job information ------
  solver concurrent jobs 1
response 'Response' 1 0 "type Response"
 objectives 1
objective 'Response' 1
 constraints 0
Metamodel Optimization Strategy DOMAINREDUCTION
  iterate param design 0.01
  iterate param objective 0.01
  iterate param stoppingtype and
Optimization Algorithm simulated annealing
 DOE
STOP
```

Figure 3: Input file for the wrapper and the generated COM file for Ishigami example

4 Examples

In this section, different sensitivity analysis methods are applied for determining the variable contributions for an analytical as well as for a standard crash analysis example named US NCAP. As an analytical example, the popular Ishigami function is taken which is often used in literature to study the impact of input variables on the function response and whose sensitivity measures for each variable are well known [9]. The sensitivity results are calculated with LS-OPT and with user-defined meta-model in PySen.

4.1 Ishigami function

Ishigami function is often used as a benchmark in order to test the results of different sensitivity methods. The function is given as:

$$f(x_1,x_2,x_3) = \sin(x_1) + a\,.\,\sin^2(x_2) + b\,.\,x_3^4\,.\,\sin(x_1). \tag{14}$$

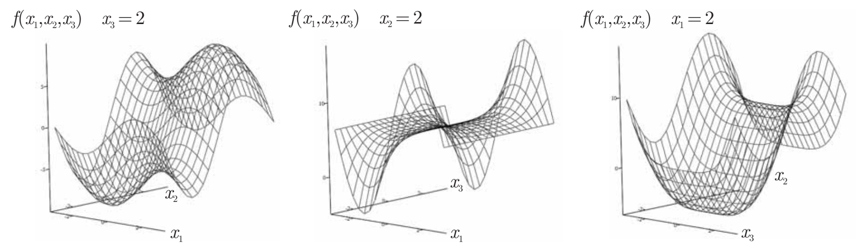


Figure 4: Cross-sectional plots of the Ishigami function

The example is carried out using the numerical values of $a = 7$ and $b = 0.1$. The plot of this nonlinear and non-monotonic function is shown in Fig. 4. The function is approximated with a meta-model using 50, 200 and 400 points respectively each within the interval $[-\pi, \pi]$ for each variable $x_i, i = 1,2,3$. The points are sampled using the space filling method in LS-OPT. A user-defined Perl script is used to emulate the solver which contains the implementation of an Ishigami function. Tab. 2 shows the sensitivity plots for derivative based, weight based, activation weight based methods as well as for SOBOL calculated with PySen and for SOBOL calculated with LS-OPT respectively. Tab. 3 summarizes the sensitivity values for each variable according to the different methods. In PySen, the user-defined neural network is trained with two hidden layers of 13 and 7 neurons each. For each set of training points, two different neural networks are trained with random initial weight settings in order to take into account the effects of training on sensitivity measures. Therefore, Tab. 2 shows the mean and the variance for the sensitivity measures associated with each variable. Regardless that a direct comparison between neural network based methods and variance based methods is not feasible because of the different underlying theoretical approaches [8], neural network based sensitivity measures are quantitatively more closer to the actual analytical SOBOL indices calculated directly on the function as compared to the meta-model based SOBOL indices. For the case of 50 points, variable contributions are effectively identified by the neural network based methods. A point to note is that training of the neural network has an impact on the quality of the sensitivity measures. Variance based methods were unable to correctly identify the contribution of the variable x_2 even for 400 points because these methods are strongly dependent on the approximation quality of the underlying meta-model. In contrast, on neural network based sensitivity methods the quality of approximation has relatively less effect. These results are initial investigations and require further analysis of the effects of the quality of the meta-model and the convergence criteria.

4.2 US NCAP

As a crash analysis example, the NCAC Ford Taurus model is taken for a US NCAP frontal impact test case [1]. For this test, 27 design variables are taken out of which 21 are sheet thickness variables and 6 are discrete material variables. There are three groups of responses namely: mass, intrusions and

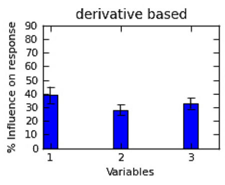
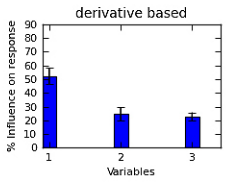
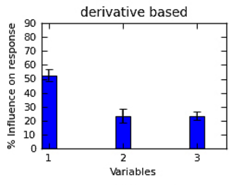
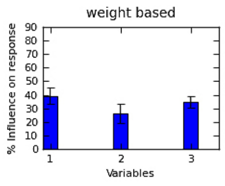
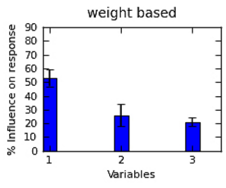
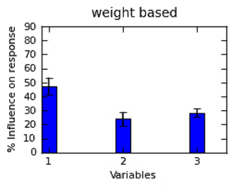
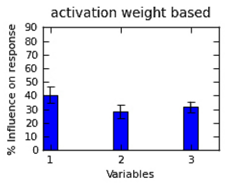
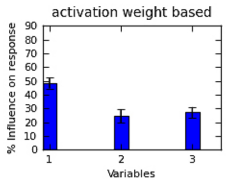
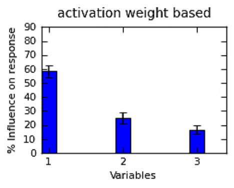
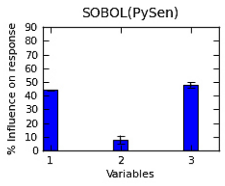
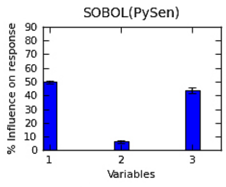
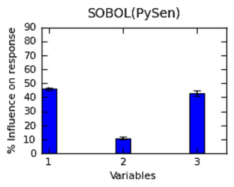
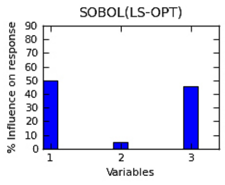
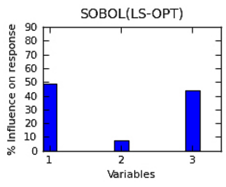
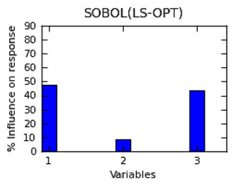
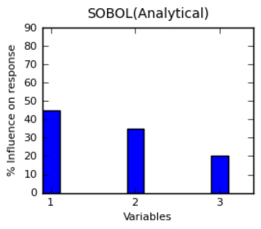
Sensitivity measures for Ishigami function		
50	200	400
Neural Network based methods		
		
		
		
Variance based method		
		
		
		

Table 2: Sensitivity measures for the Ishigami function with 50, 200 and 400 points

			50					200					400				
#	Variables	Analytical	LS-OPT	PySen				LS-OPT	PySen				LS-OPT	PySen			
		SOBOL	SOBOL	SOBOL	derivative based	weight based	activation weight based	SOBOL	SOBOL	derivative based	weight based	activation weight based	SOBOL	SOBOL	derivative based	weight based	activation weight based
1	x_1	0.45	0.49	0.44	0.39	0.39	0.40	0.48	0.49	0.52	0.53	0.48	0.47	0.46	0.52	0.47	0.58
2	x_2	0.35	0.04	0.07	0.28	0.26	0.28	0.07	0.06	0.24	0.25	0.24	0.08	0.10	0.23	0.24	0.25
3	x_3	0.20	0.45	0.48	0.32	0.34	0.31	0.43	0.43	0.22	0.21	0.27	0.43	0.42	0.23	0.28	0.16

Table 3: Tabular values for the sensitivity measures of the Ishigami function

accelerations. In this example, the impact of the 27 design variables (see Tab. 4) on a single response variable ”max_intrusion_firewall_left” is investigated.

In order to evaluate the effects of the number of design points on the sensitivity measures, the analysis is performed for 50, 200 and 400 simulated design points respectively. For the LS-OPT case, feed forward neural network is taken as a meta-model. Default integration points are used for calculating sensitivity indices. For the case of the user-defined meta-model, 10 neural networks are trained each with two hidden layers of 7 and 3 neurons each. The sensitivity measures for each variable are shown in Tab. 5 for all sets of design points. Due to the lack of an analytical benchmark, the results of SOBOL are compared with the neural network based sensitivity measures. For the case of 50 design points, neural network based methods show substantial contributions for almost all variables excluding variables 10, 16 and 27, while LS-OPT shows very strong contributions for only 10 variables (1,2,5,7,9,11,16,21,24,26 and 27). With respect to the increase in number of design points, the variable contributions remain relatively similar for neural network based methods. For the case of 400 points, the contribution of variable 2 is identified to be the most significant by all methods. For variables 9, 11 and 24, LS-OPT shows a higher contribution as compared to the neural network based methods. Also, a very significant change in the contribution of variables relative to the increase in design points is identified by SOBOL approach in LS-OPT while this change is not evident in neural network based methods. These results are initial investigations and require further analysis of the effects of the quality of the meta-model and the convergence criteria.

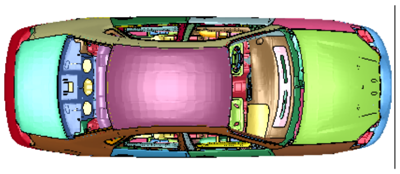


Figure 5: US NCAP - 56.6 km/h [1]

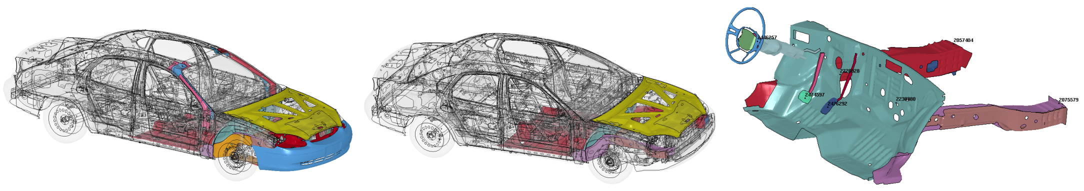


Figure 6: 21 sheet thicknesses, 6 discrete materials, 1 response variable [1]

#	Name	Description	#	Name	Description	#	Name	Description
1	mat_rO	BIW - rail - L - O / - R - O	11	rail_I	BIW - rail - L - I / - R - I	21	sreinf	BIW - toepan sup reinforc 1 - L / - R
2	mat_rI	BIW - rail - L - I / - R - I	12	mreinf2	BIW - rail middle reinfor 2 - L / R - L	22	A_pil_O	BIW - A pillar - L - O / - R - O
3	mat_fw	BIW - firewall	13	subfr	f-mech-subframe-front	23	A_pil_I	BIW - A pillar - L - I / - R - I
4	mat_tp	BIW - toe pan	14	subfr_I	f-mech-subframe-front-lower	24	firewal	BIW - firewall
5	mat_fl	BIW - floor	15	arm_top	f-mc-subframe-arm1-top / -arm2-top	25	toe_pan	BIW - toe pan
6	mat_hd	OB - hood – I	16	arm_bot	f-mc-subframe-arm1-bottom / -arm2-bottom	26	radiat	MC - radiator
7	rplate1	BIW - rail plate	17	tie_bar	BIW - tie bar module	27	t_floor	BIW - floor
8	rplate2	BIW - rail plate	18	bump_FT	OB - bumper - FT			
9	rail_O	BIW - rail - L - O / - R - O	19	hood_I	OB - hood - I			
10	mreinf	BIW - rail middle reinfoce - L / - R	20	hreinf3	BIW - front bumper frame			

Table 4: Design variables of Ford Taurus for US NCAP

5 Conclusions

A software framework is developed which uses LS-OPT in order to perform user-defined meta-model based sensitivity analysis. It provides a possibility for the usage of enhanced meta-model and different methods for sensitivity analysis. Different neural network based sensitivity methods show a good potential for identifying the variable contributions even with very few design points and can be used as an alternative to the variance based sensitivity analysis techniques.

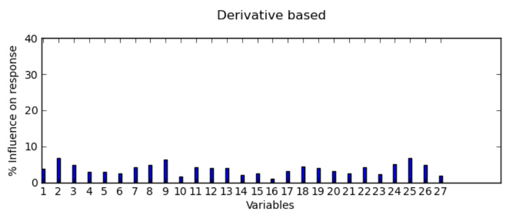
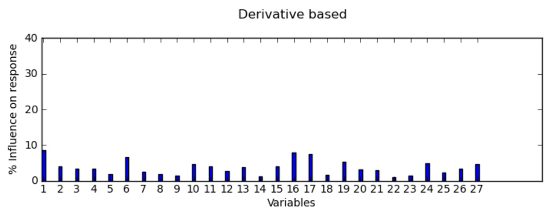
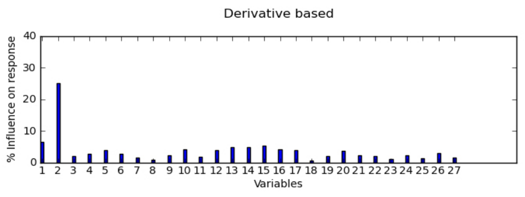
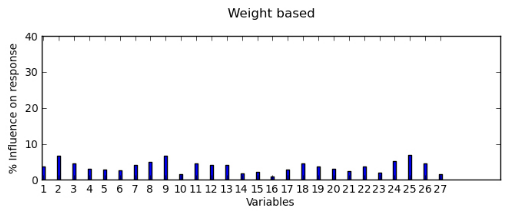
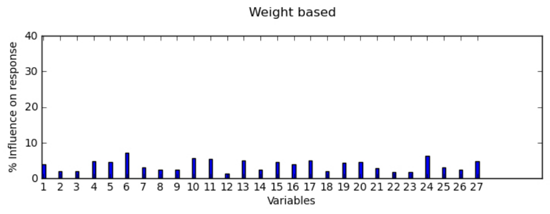
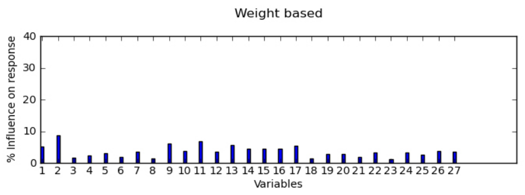
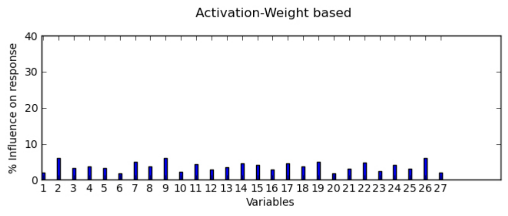
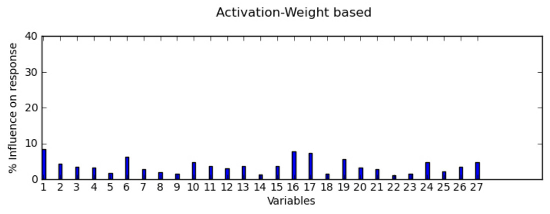
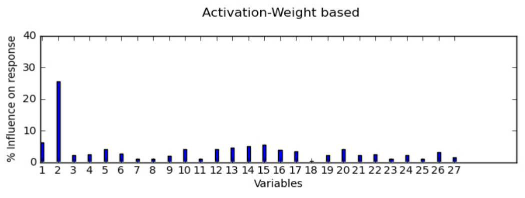
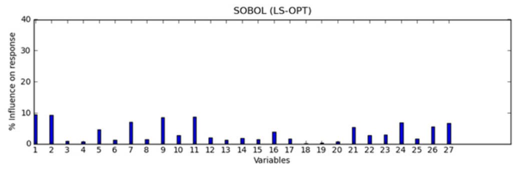
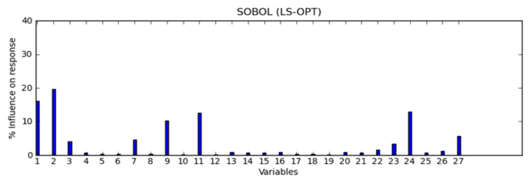
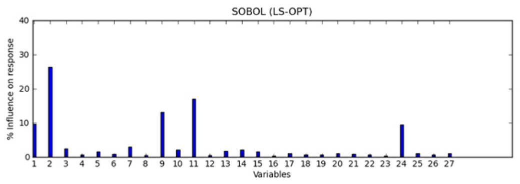
Sensitivity measures for US NCAP		
50	200	400
Neural Network based methods		
		
		
		
Variance based method		
		

Table 5: Sensitivity measures for ”max_intrusion_firewall_left” with 50, 200 and 400 points

Acknowledgment

The authors gratefully acknowledge the financial support of the European Union (EFRE) and the Free State of Saxony as well as the support of the Center for Information Services and High Performance Computing (ZIH) of the Technische Universität Dresden.

References

[1] FHWA/NHTSA National Crash Analysis Center. Finite element model of ford taurus. Technical report. Available at http://www.ncac.gwu.edu/vml/archive/ncac/vehicle/taurus-v3.pdf.

[2] G. D. Garson. Interpreting neural-network connection weights. *AI Expert*, pages 47–51, 1991.

[3] M. Gevrey, I. Dimopoulos, and S. Lek. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological Modelling*, 160(3):249 – 264, 2003.

[4] M. Gevreya, I. Dimopoulosb, and S. Leka. Two-way interaction of input variables in the sensitivity analysis of neural network models. *Ecological Modelling*, 195:43–50, 2006.

[5] B. Hohage, A. Förderer, G. Geissler, and H. Müllerschön. Global sensitivity analysis in industrial application with ls-opt. In *9. LS-DYNA Forum*, Bamberg, 2010.

[6] J. J. Montano and A. Palmer. Numeric sensitivity analysis applied to feedforward neural networks. *Neural Computation and Applications*, 12, 2003.

[7] J. D. Olden, M. K. Joy, and R. G. Death. An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. *Ecological Modelling*, 178(3-4):389–397, 2004.

[8] S. Pannier and W. Graf. Sectional sensitivity measures with artificial neural networks. In *9th LS-DYNA User Forum*, Bamberg, 2010.

[9] U. Reuter and M. Liebscher. Global sensitivity analysis in view of nonlinear structural behavior. In *7th LS-DYNA User Forum*, Bamberg, 2008.

[10] U. Reuter, M. Liebscher, and H. Müllerschön. Global sensitivity analysis in structural optimization. In *7th European LS-DYNA Conference*, Salzburg, 2009.

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

[12] I. M. Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and Computers in Simulation*, 55:271–280, 2001.

[13] T. Tchaban, M. J. Taylor, and J. P. Griffin. Establishing impacts of the inputs in a feedforward neural network. *Neural Computing and Applications*, 178:309–317, 1998.

[14] Bernd Zwingmann. Einsatz neuronaler Netze in der numerischen Simulation - Sensitivitätsanalyse und Netzoptimierung. Diploma Thesis, Institute for Structural Analysis, Faculty of Civil Engineering, TU Dresden, 2009.