



Project acronym: PAMINA

**Plan for benchmark, including specification of synthetic PA cases**

Reference: M2.1.D.3  
Version: 0.02, for review by task leader  
RTDC: 2  
Work package: 2.1.D  
Author: E. Plischke  
Date of working paper: February 14, 2008

# PAMINA milestone M2.1.D.3: Plan for benchmark, including specification of synthetic PA cases

Elmar Plischke\*

February 14, 2008

In the benchmarking exercise in the frame of PAMINA Task 2.1.D we want to gain experience with the tools and techniques of sensitivity analysis (SA) by applying them to analytical test models. In the following, we give a non-exhaustive list of software tools that may be used to analyse the test models. Such an analysis is presented in Section 2 in order to provide a worked example. We then describe the models in Section 3, accompanied by published results from the literature. Some new results are also incorporated. The main sources of test models are [1] and [2].

## 1 The software tools

**SimLab** 2.2 is a software designed for Monte Carlo based uncertainty and sensitivity analysis. It is available from the SimLab Homepage. It has a graphical user interface(GUI) and supports many Uncertainty Analysis(UA)/Sensitivity Analysis(SA) indicators. Small example models can be directly integrated as so-called internal models. The export of sampled data and the import of computed data is supported for external models. Unfortunately, the user interface is unreliable. Moreover, the format of imported and exported files depends on the current locale settings which has implications on data transfer and even on the demos inside **SimLab**.

Its successor, **SimLab** 3.0 comes without GUI, but can be interfaced to MATLAB, FORTRAN and C++. In my experience, the MATLAB interface (Version 7.5) fails to handle external data (i.e., calculations are performed inside MATLAB and then loaded into **SimLab**). On the other hand, **SimLab**'s internal model runner) works reliable inside MATLAB.

The C++ interface runs stable, although the UA module seems to be a bit slow. Some hand tuning is necessary: The include files for xercesc and boost have to be installed manually so that all references can be resolved.

A solution purely depending on MATLAB is EIKOS. The report [4] describes the implementation and the underlying algorithms. It also contains code-snippets which are of use for any matrix-oriented programming environment.

Other software packages are under evaluation by the participants, like SUSA or the JRC Petten subroutine library.

## 2 Presentation of results

In order to provide a common ground for the modelling teams we propose the following presentation of SA results to ensure that these results are comparable between the participants and to simplify the work of the task leader.

### 2.1 General layout

First of all, a problem to be studied should be specified. Problems to be studied might be sampling strategies, sampling sizes, algorithms, implementations, resource usage (speed, memory), ... Depending on this,

---

\*elmar.plischke@tu-clausthal.de

Model 4a LHS		Model 4a LHS																					
A	B	C	D	E		F		G		H		I		J		K		L		M		N	
1	Model 4a LHS	Mean	Variance	R <sup>2</sup> <sub>y</sub>	SRC		PCC		Pearson		First Order Effects		Total Effects										
2	Sample size				X1	X2	X1	X2	X1	X2	X1	X2	X1	X2	X1	X2	X1	X2	X1	X2	X1	X2	
3	(expected)	0.7	0.154444	0.89			0.9078413	0.8660254	0.73455316	0.58764253	0.53956835	0.46043165	0.53956835	0.46043165									
4	100	0.69963	0.155512	0.88637961	0.70100616	0.58632164	0.90202683	0.86749187	0.73542473	0.62472168	0.48711117	0.438793545	0.56120455	0.51288883									
5	100	0.70011	0.16384	0.89102573	0.71941294	0.5696146	0.90938122	0.86600333	0.75111222	0.60842819	0.47788173	0.43271093	0.56728907	0.52211827									
6	100	0.70058	0.151239	0.88068192	0.70772841	0.59662414	0.89950238	0.86672452	0.72139015	0.61233046	0.47576488	0.3994926	0.6005074	0.52423512									
7	100	0.70032	0.169143	0.89430411	0.73732801	0.56182774	0.91550942	0.8664997	0.7588295	0.58900025	0.55285346	0.45491576	0.54508424	0.44714654									
8	100	0.70087	0.171697	0.89474754	0.66991367	0.56364241	0.89844302	0.86521915	0.76244851	0.6738584	0.48837561	0.46229004	0.53770996	0.51162439									
9	100	0.69907	0.140236	0.87299978	0.7781839	0.615006	0.90888375	0.86402731	0.70634805	0.51977354	0.42775817	0.36609711	0.63390289	0.57224183									
10	100	0.69881	0.129369	0.8636937	0.77638299	0.63570019	0.90207998	0.86265929	0.68350325	0.51784249	0.5403294	0.42413262	0.57586738	0.4596706									
11	100	0.69921	0.148195	0.87935441	0.7555156	0.59832112	0.90915792	0.86564827	0.72019093	0.55169123	0.53919132	0.4280868	0.5719132	0.46080868									
12	100	0.70046	0.147708	0.87888952	0.78493009	0.60676873	0.91370486	0.86727678	0.71906519	0.51636671	0.60918673	0.48019071	0.51980929	0.39081327									
13	100	0.69971	0.163686	0.89200622	0.69644642	0.57049916	0.90440183	0.86640995	0.75290815	0.63782009	0.53628775	0.538426	0.461574	0.46371225									
14	1000	0.70003	0.164087	0.89148004	0.71264674	0.57017424	0.90738662	0.86544793	0.75342507	0.62102856	0.517142	0.45093444	0.54906556	0.482858									
15	1000	0.69998	0.149947	0.88137786	0.73912855	0.59617974	0.90646724	0.86593382	0.72512916	0.57859199	0.51206802	0.4675302	0.53294698	0.48793198									
16	1000	0.70001	0.156051	0.88626712	0.71344373	0.58531272	0.90401851	0.86639833	0.73748884	0.61454088	0.55403843	0.48106310	0.51893689	0.44595157									
17	1000	0.69999	0.157767	0.88728507	0.73552213	0.58131464	0.9097853	0.86605732	0.74097272	0.58803603	0.52004278	0.45661442	0.54338558	0.47995722									
18	1000	0.70001	0.153791	0.8842668	0.7344368	0.58893527	0.90745313	0.86601867	0.73286889	0.58686133	0.54909082	0.42701157	0.57298843	0.45099018									
19	1000	0.69998	0.152539	0.88384542	0.71691105	0.59175447	0.90317765	0.86660728	0.73041112	0.60798889	0.54539541	0.41249462	0.58750538	0.45460459									
20	1000	0.69999	0.154282	0.88469354	0.73788538	0.58793636	0.90849031	0.86604535	0.73396985	0.58289619	0.55245354	0.46800269	0.53199731	0.44754646									
21	1000	0.69998	0.148719	0.88035942	0.74227868	0.59864153	0.90640979	0.86584676	0.72251588	0.57397647	0.52290466	0.47215631	0.52784369	0.47709534									
22	1000	0.69999	0.159636	0.88854598	0.72857265	0.57786457	0.90913329	0.86590659	0.74468155	0.59793359	0.56940784	0.42405778	0.57594222	0.43059216									
23	1000	0.69999	0.157996	0.88736839	0.72451987	0.58102885	0.90740499	0.86594994	0.7414236	0.60189727	0.53734756	0.44438996	0.5561404	0.46265244									
24	10000	0.7	0.155195	0.88545281	0.73653811	0.58622037	0.90866653	0.86603981	0.73604721	0.58558574	0.54996704	0.46132166	0.53867834	0.45003296									
25	10000	0.7	0.154933	0.88525537	0.73667044	0.58671978	0.9085585	0.86603762	0.73551521	0.58525074	0.5383341	0.46520661	0.53479339	0.4616659									
26	10000	0.7	0.15383	0.88442137	0.73656492	0.58881745	0.90795747	0.86602367	0.7332742	0.58468535	0.54386632	0.45839097	0.54160903	0.45613368									
27	10000	0.7	0.153218	0.88396498	0.73478912	0.58997817	0.90725689	0.86602437	0.7320268	0.58652232	0.53328798	0.46069106	0.53930894	0.46671202									

Figure 1: Sensitivity analysis data exported into Excel

the documentation shall include the following informations:

- The model (and its variant) under consideration,
- The used software package,
- Sampling method, sampling size, (initial seed for reproduction of sampling data),
- Statistics and sensitivities to be calculated (with a documentation of the used algorithms or tools),
- An Excel-sheet or ASCII-file per problem collecting the results (the layout shall be similar to the one we use in the next subsection),
- Conclusions with optional graphical representation of results,
- Comments, remarks, including expectations which were confirmed or not satisfied.

The participants are requested to bring in their own ideas and to find creative solutions for visualizing big amounts of data.

## 2.2 An example problem analyzed

Let us now analyse an example test case. We take model 4 (a)&(c) (see below) and want to study different sample sizes, sensitivity methods and algorithms. We are interested in a home-brewed implementation in MATLAB which calculates mean  $E(Y)$  and variance  $V(Y)$  via built-in commands. The calculations of regression analysis are done with the help of MATLAB's backslash operator for solving least squares problems, such that  $R_y^2$ , SRC and PCC are available. Furthermore, the correlation ratios Pear are calculated. First order effects  $S_i$  and total effects  $S_{T_i}$  are computed following the algorithm in Appendix C of [3]. We compare part of these results with other tools.

Sampling sizes of  $N = 100, 1000, 10000, 100000, 250000$  are generated inside MATLAB using Latin hypercube sampling. Note that for the calculation of first order effects and total effects, a second sampling set and further calculations of the model are needed. For each sampling size we collect 10 calculations of all indicators. MATLAB completes the sensitivity analysis after about 10s for each model. Hence for real world models, the computational bottleneck is the execution time of each model run. The data from the runs is then exported into an Excel sheet, documenting the used model, sample size and calculated statistics and sensitivities, see Figure 1.

The model is an additive model, so first order effects should equal the total effects. Moreover, one can study if the square of the Pearson correlations coincides with the first order effects. The following conclusions

can be drawn from this example: The statistics and sensitivities based on linear regression or correlation work reliable for sampling sizes of about 1000, sensitivity effects as computed by the cited algorithm are less robust. In configuration (c) it becomes obvious that fixing the precision of non-influential factors requires a massive amount of model evaluations, as there are still negative first order effects for sampling sizes of a quarter million.

We now study the first order effects and total effects as computed by **Simlab 3.0** using the extended FAST algorithm. A C++ program which drives **SimLab**'s sample generator and sensitivity analysis module produces the following sample outputs for configuration (a).

```
-----Case4: Fourth Power (2 factors, 1000 samples)
+++Sample size changed to 994
Model evaluation took 0.015s
SI   :      0.537031      0.454918
STI  :      0.540296      0.463736
Sensitivity analysis took 0.032s
-----Case4: Fourth Power (2 factors, 250000 samples)
+++Sample size changed to 249986
Model evaluation took 35.5s
SI   :      0.538324      0.455833
STI  :      0.539572      0.462665
Sensitivity analysis took 218.313s
```

We see that although the sample size has been changed by orders of magnitude in the two model runs, the first order and total effects remain roughly the same, so it is hard to predict a convergence rate to the true values. Note that computational costs increase considerably compared to the MATLAB implementation. For model 4(c), the following runs are produced.

```
-----Case4: Fourth Power (2 factors, 1000 samples)
+++Sample size changed to 994
Model evaluation took 0.015s
SI   :      0.000067      0.982129
STI  :      0.002127      0.998834
Sensitivity analysis took 0.063s
-----Case4: Fourth Power (2 factors, 100000 samples)
+++Sample size changed to 99986
Model evaluation took 10.5s
SI   :      0.000075      0.984079
STI  :      0.000095      0.998830
Sensitivity analysis took 43.203s
```

In contrast to the algorithm used in our MATLAB program, the first order effect of the unimportant factor is well approximated even for small sampling sizes.

For a more complete picture of the available tools we test the EFAST implementation in MATLAB from [4] which was slightly modified to take functional arguments into account. The following session log shows the computations with a requested size of 2000 samples.

```
>> model4=inline('x(:,1)+x(:,2).^4','x');
>> trafo=inline('x','x');
>> tic();[si,sti]=efast(2,2000,model4,trafo),toc()
si =
    0.538341049823859    0.463092139893829
sti =
    0.539583538136977    0.470023772131451
Elapsed time is 0.011895 seconds.
>> trafo=inline('5*x','x');
>> tic();[si,sti]=efast(2,2000,model4,trafo),toc()
si =
    0.000076592413786    0.984551778884310
sti =
    0.000083142251702    0.999288724689167
Elapsed time is 0.011507 seconds.
```

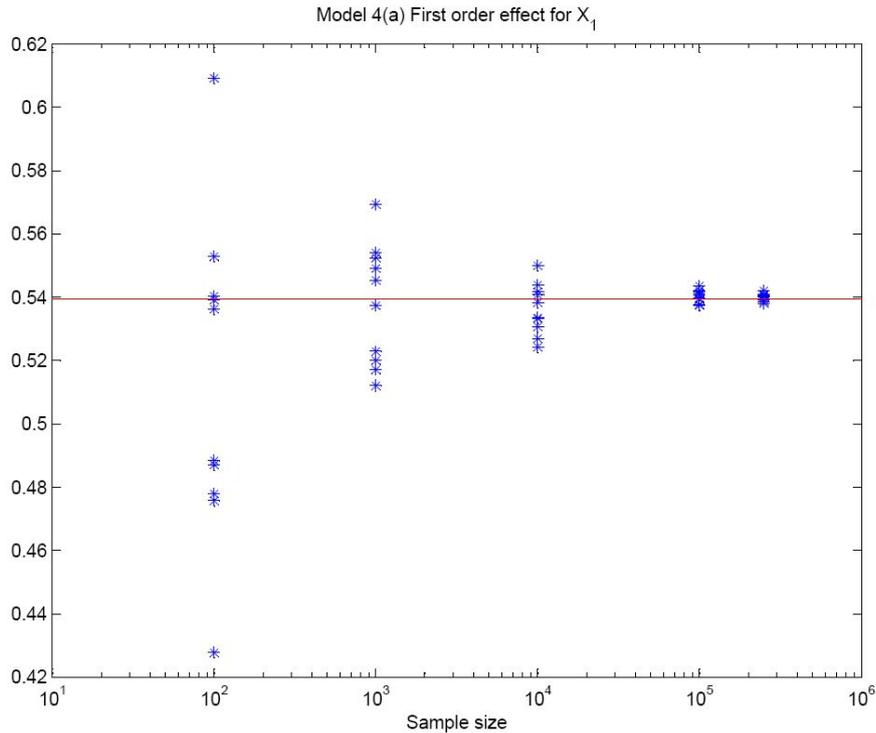


Figure 2: Convergence of the first order effect

This implementation is on par with the **SimLab** version, so we might consider including it in our home-brewed MATLAB version. The currently implemented brute-force method of [3] is still faster and does not need a special sampling scheme, but lacks the precision.

For a complete discussion of this model the results using alternative simulation tools should also be tabulated in form of an Excel sheet to ease the comparison of the results. As an example for the visualization of results, Figure 2 shows the convergence of the first order effect for different sampling sizes using the algorithm from [3]. For comparison, the same graphics would be created with data from the EFAST method runs.

Further remarks: During the creation of this example, an earlier version of the Simlab C++ program was missing 1%–2% of variance in the calculation of first order effects. Due to the additive nature of the model this was an indicator that something went terribly wrong. A revised model calculation loop now gives correct results. (More interna: memory allocation for input and output matrices is done by C++, then **SimLab** fills the input with the requested sample methods, the model is evaluated in C++ and stored in the output matrix, the output is loaded into **SimLab** which then performs the sensitivity analysis.)

### 3 The benchmark models

This section gathers the published results for our test models. For quick comparisons with model runs, all values have been given decimally.

### 3.1 Model 1 – obligatory

The first model is Model 1 of [1, §2.9.1]. It is a purely linear model given by

$$Y = \sum_{i=1}^3 X_i \quad (1)$$

where  $X_i \sim U(3^{i-1}/2, 3^i/2)$ ,  $i = 1, 2, 3$ . This function can be used to test the software setup. It is also available in the Demo Mode of **SimLab 2.0**. The values in Table 1 are taken from the literature.

Table 1: Statistics and Sensitivities for Model 1

	<table border="1"> <tr> <td>E(Y)</td> <td colspan="2">13</td> </tr> <tr> <td>V(Y)</td> <td colspan="2">91/12=7.5833</td> </tr> <tr> <td><math>R_y^2</math></td> <td colspan="2">1</td> </tr> </table>			E(Y)	13		V(Y)	91/12=7.5833		$R_y^2$	1	
E(Y)	13											
V(Y)	91/12=7.5833											
$R_y^2$	1											
	1	2	3									
Pear(Y, $X_i$ )	$1/\sqrt{91} = 0.10483$	$3/\sqrt{91} = 0.31449$	$9/\sqrt{91} = 0.94346$									
PCC(Y, $X_i$ )	1	1	1									
$V(E(Y X_i))/V$	$1/91 = 0.010989$	$9/91 = 0.098901$	$81/91 = 0.890109$									
$E(V(Y X_i))$	$90/12 = 7.5$	$82/12 = 6.8333$	$10/12 = 0.8333$									

### 3.2 Model 2 – voluntary

Model 2 is again taken from [1, §2.9.1]. It is also a linear model, but now the distributions of the factors are not independent. We consider

$$Y = X_1 + X_2 \quad (2)$$

where the joint probability density function is given by

$$p(x_1, x_2) = \begin{cases} 2 & \text{if } 0 \leq x_1, x_2 \leq 0.5 \text{ or } 0.5 \leq x_1, x_2 \leq 1, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

This test case is marked optional, as the generation of dependent factors is not always supported. Tabulated values are available in Table 2.

One can try to generate samples via a reject method, or to implement the probability of  $x_2$  given  $x_1$  and then follow the ideas from [5] to compute the variance of the conditional expectations.

Table 2: Statistics and Sensitivities for Model 2

	<table border="1"> <tr> <td>E(Y)</td> <td>1</td> </tr> <tr> <td>V(Y)</td> <td><math>7/24 = 0.29167</math></td> </tr> <tr> <td>Pear(<math>X_1, X_2</math>)</td> <td><math>3/4 = 0.75</math></td> </tr> <tr> <td><math>R_y^2</math></td> <td>1</td> </tr> </table>		E(Y)	1	V(Y)	$7/24 = 0.29167$	Pear( $X_1, X_2$ )	$3/4 = 0.75$	$R_y^2$	1	<table border="1"> <tr> <td></td> <td>1,2</td> </tr> <tr> <td>Pear(Y, <math>X_i</math>)</td> <td><math>\sqrt{7/8} = 0.93541</math></td> </tr> <tr> <td>PCC(Y, <math>X_i</math>)</td> <td>1</td> </tr> <tr> <td>SRC</td> <td><math>\sqrt{2/7} = 0.53452</math></td> </tr> <tr> <td><math>V(E(Y X_i))</math></td> <td><math>13/48 = 0.27083</math></td> </tr> <tr> <td><math>E(V(Y X_i))</math></td> <td><math>1/48 = 0.02083</math></td> </tr> </table>			1,2	Pear(Y, $X_i$ )	$\sqrt{7/8} = 0.93541$	PCC(Y, $X_i$ )	1	SRC	$\sqrt{2/7} = 0.53452$	$V(E(Y X_i))$	$13/48 = 0.27083$	$E(V(Y X_i))$	$1/48 = 0.02083$
E(Y)	1																							
V(Y)	$7/24 = 0.29167$																							
Pear( $X_1, X_2$ )	$3/4 = 0.75$																							
$R_y^2$	1																							
	1,2																							
Pear(Y, $X_i$ )	$\sqrt{7/8} = 0.93541$																							
PCC(Y, $X_i$ )	1																							
SRC	$\sqrt{2/7} = 0.53452$																							
$V(E(Y X_i))$	$13/48 = 0.27083$																							
$E(V(Y X_i))$	$1/48 = 0.02083$																							

### 3.3 Model 3 – voluntary

The next model from [1, §2.9.1] is a linear model due to Sobol'. It has 22 parameters.

$$Y = \sum_{i=1}^k c_i (X_i - 0.5) \quad (4)$$

where  $k = 22$ ,  $X_i \sim U(0, 1)$ , and  $c_i = (i - 11)^2$ .

Tabulated values are available in Table 3. For convenience, the sensitivities of the most important factors are given decimally. This is an optional exercise.

Table 3: Statistics and Sensitivities for Model 3

E(Y)	0			
V(Y)	65307/12=5442.3			
$R_y^2$	1			
	i	22	1,21	2,20
Pear(Y, $X_i$ )	$c_i/\sqrt{\sum_{i=1}^k c_i^2} = c_i/255.55$	0.47348	0.39131	0.3169
PCC(Y, $X_i$ )	1	1	1	1
V(E(Y  $X_i$ ))	$c_i^2/12$	1220.1	833.33	546.75
E(V(Y  $X_i$ ))	$\sum_{j \neq i} c_j^2/12$	4222.2	4608.9	4895.5

### 3.4 Model 4 – obligatory

The fourth model is from [1, §2.9.2],

$$Y = X_1 + X_2^4. \quad (5)$$

To put the non-linearity into work, the parameter distributions are varied as follows.

- (a)  $X_i \sim U(0, 1)$
- (b)  $X_i \sim U(0, 3)$
- (c)  $X_i \sim U(0, 5)$

Model 4(b) is optional, cases (a) and (c) are obligatory tests. Tabulated values for cases (a) and (c) can be found in Table 4. As these models are non-linear, rank statistics have been supplied.

Table 4: Statistics and Sensitivities for Model 4

	(a)		(c)	
E(Y)	7/10 = 0.7		255/2 = 127.5	
V(Y)	139/900 = 0.15444		40003 · 25/36 = 27780	
$R_y^2$	0.89		0.75	
$R_y^{2*}$	0.89		0.98	
	(a)		(c)	
	1	2	1	2
Pear(Y, $X_i$ )	$15/\sqrt{417}=0.735$	$4\sqrt{3/139}=0.588$	$\sqrt{3/40003}=8.66 \cdot 10^{-3}$	$100\sqrt{3/40003}=0.866$
PCC(Y, $X_i$ )	$15/\sqrt{273}=0.908$	$\sqrt{3}/2=0.866$	$\sqrt{3}/10003=0.017$	$\sqrt{3}/2=0.866$
V(E(Y  $X_i$ ))/V	$75/139=0.540$	$64/139=0.460$	$3/40003=7.50 \cdot 10^{-5}$	$40000/40003=0.9999$
Sppear(Y, $X_i$ )	0.76	0.55	0.08	0.99
PRCC(Y, $X_i$ )	0.91	0.85	0.50	0.99

### 3.5 Model 5 – voluntary

Model 5, again found in [1, §2.9.2], is an exponential function due to Sobol', given by

$$Y = \exp\left(\sum_{i=1}^k b_i X_i\right) - \prod_{i=1}^k \frac{\exp(b_i) - 1}{b_i}. \quad (6)$$

We consider two different sets of parameters.

- (a)  $X_i \sim U(0, 1)$  and  $k = 6, b_1 = 1.5, b_2 = \dots = b_k = 0.9,$
- (b)  $X_i \sim U(0, 1)$  and  $k = 20, b_1 = \dots = b_{10} = 0.6, b_{11} = \dots = b_k = 0.4.$

Tabulated values can be found in Table 5. This is an optional benchmarking exercise.

Table 5: Statistics and Sensitivities for Model 5

	(a)		(b)			
			1	> 1	<= 10	>= 11
$E(Y)$	0	0				
$V(Y)$	427.28	18022				
$R_y^2$	0.80	0.81				
$R_y^{2*}$	0.97	0.96				
Pear( $Y, X_i$ )			0.51	0.32	0.24	0.16
PCC( $Y, X_i$ )			0.76	0.58	0.47	0.32
$V(E(Y X_i))/V$			0.2870	0.1057	0.0562	0.0250
Spearm( $Y, X_i$ )			0.59	0.35	0.26	0.17
PRCC( $Y, X_i$ )			0.96	0.88	0.76	0.61

### 3.6 Model 6 – obligatory

Model 6 is found in [1, §2.9.2]. It is a quotient of powers given by

$$Y = X_2^4/X_1^2 \quad (7)$$

where we have two configurations exploiting the non-linearity of the model.

(a)  $X_i \sim U(0.9, 1.1)$ ,

(b)  $X_i \sim U(0.5, 1.5)$ .

Note that the explicit formulas for  $V(Y)$  published in [1, §2.9.2] miss some summands and factors, see Table 6 for corrected values. Model 6(b) is an optional exercise, while 6(a) is required for the benchmarking test.

Table 6: Statistics and Sensitivities for Model 6

	(a)		(b)	
$E(Y)$	$100/99(1.1^5 - 0.9^5) = 1.0303$		$4/15(1.5^5 - 0.5^5) = 2.0167$	
$V(Y)$	$25/27(1.1^9 - 0.9^9)(0.9^{-3} - 1.1^{-3}) - E^2(Y) = 0.070442$		$1/27(1.5^9 - 0.5^9)(208/27) - E^2(Y) = 6.9012$	
$R_y^2$	0.98		0.675	
$R_y^{2*}$	0.99		0.98	
	(a)		(b)	
	1	2	1	2
Pear( $Y, X_i$ )	-0.45	0.89	-0.47	0.67
PCC( $Y, X_i$ )	-0.98	0.99	-0.64	0.76
$V(E(Y X_i))/V$	0.2023	0.7690	0.26191	0.51098
Spearm( $Y, X_i$ )	-0.42	0.90	-0.43	0.89
PRCC( $Y, X_i$ )	-0.97	0.99	-0.95	0.99

### 3.7 Model 7 – obligatory

The non-monotonic Sobol' g-function is given by

$$Y = \prod_{i=1}^k \frac{|4X_i - 2| + a_i}{1 + a_i} \quad (8)$$

where  $k = 8$ ,  $(a_i) = (0, 1, 4.5, 9, 99, 99, 99, 99)$  and  $X_i \sim U(0, 1)$ , see [1, §2.9.3]. This function is also available in the Demo Mode of **SimLab 2.0**.

Tabulated values are presented in Table 7.

We skip model 8 in our discussion.

### 3.8 Model 9 – obligatory

We continue our analysis with Model 9 from [1, §2.9.3], the so-called Ishigami function

$$Y = \sin X_1 + 7 \sin^2 X_2 + 0.1 X_3^4 \sin X_1 \quad (9)$$

$E(Y)$	1
$V(Y)$	0.4652
$R_y^2$	0

Table 7: Statistics and Sensitivities for Model 7

	1	2	3	4	5-8
$V(E(Y X_i))/V$	0.7165	0.1791	0.0237	0.0072	0.0001

where  $X_i \sim U(-\pi, \pi)$ . This function is also available in the Demo Mode of **SimLab 2.0**. For tabulated values see Table 8.

Table 8: Statistics and Sensitivities for Model 9

$E(Y)$	3.5		1	2	3	
$V(Y)$	$\pi^4/50 + \pi^8/1800 + 53/8 = 13.845$		$V(E(Y X_i))/V$	0.3139	0.4424	0
$R_y^2$	0.19		SRC	0.435	0	0
$R_y^{2*}$	0.19		SRRC	0.436	0	0

### 3.9 Model 10 – obligatory

Model 10 is called the Morris function, see [1, §2.9.3]. As the summation notation of the published version is non-standard, we interpret this as

$$Y = \beta_0 + \sum_{i=1}^{20} \left( \beta_i \omega_i + \sum_{j>i}^{20} \left( \beta_{ij} \omega_i \omega_j + \sum_{k>j}^{20} \left( \beta_{ijk} \omega_i \omega_j \omega_k + \sum_{\ell>k}^{20} \beta_{ijkl} \omega_i \omega_j \omega_k \omega_\ell \right) \right) \right) \quad (10)$$

where  $X_i \sim U(0, 1)$  and

$$\omega_i = \begin{cases} 22X_i/(10X_i + 1) - 1 & \text{for } i = 3, 5, 7 \\ 2X_i - 1 & \text{otherwise.} \end{cases}$$

The  $\beta$  parameters are given by

$$\beta_i = 20 \text{ for } i = 1, \dots, 10, \quad \beta_i \sim N(0, 1), i = 11, \dots, 20. \quad (11)$$

$$\beta_{ij} = -15 \text{ for } i, j = 1, \dots, 6, \quad \beta_{ij} \sim N(0, 1) \text{ otherwise.} \quad (12)$$

$$\beta_{ijk} = -10 \text{ for } i, j, k = 1, \dots, 5, \quad \beta_{ijk} = 0 \text{ otherwise.} \quad (13)$$

$$\beta_{ijkl} = 5 \text{ for } i, j, k, \ell = 1, \dots, 4, \quad \beta_{ijkl} = 0 \text{ otherwise.} \quad (14)$$

Note that with respect to (10), factors like  $\beta_{ij}$  where  $j < i$  are never used. For a deterministic model the non-determined first-order and second order parameters are given by

$$\beta_i = (-1)^i, \beta_{ij} = (-1)^{i+j}. \quad (15)$$

Tabulated values for this function are not available.

### 3.10 Model 11 – obligatory

Model 11 is taken from [2, §3.1]. Setting the constants mentioned there to  $2g/k_{el} = 13$  yields

$$Y = X_1 - 13 * X_2/X_3 \quad (16)$$

where  $X_1 \sim U(40, 60)$  (height),  $X_2 \sim U(67, 74)$  (weight),  $X_3 \sim U(20, 40)$  (no. of strands). The output  $Y$  models the minimal height of a bungee jump.

After some calculations we arrive at the statistics of Table 9, whereas the sensitivities have been published in [2].

Table 9: Statistics and Sensitivities for Model 11

E(Y)	$50 - 1833 \log(2)/40 = 18.237$		1	2	3
V(Y)	$(1734658 - 3359889 \log(2)^2)/1600 = 75.2433$	$V(E(Y X_i))/V$	0.44	0.01	0.55

### 3.11 Model 12 – voluntary

Model 12 measures the distance of a sample to two spheres.

$$Y = \left( \sqrt{X_1^2 + X_2^2 + X_3^2} - R_1 \right)^2 / A_1 + \left( \sqrt{X_4^2 + X_5^2 + X_6^2} - R_2 \right)^2 / A_2 \quad (17)$$

where  $R_1 = R_2 = 0.9$ ,  $A_1 = A_2 = 1/1000$  and

- (a)  $X_i \sim N(0, 0.35)$ ,
- (b)  $X_i \sim U(-1, 1)$ .

This is the only example where a normal distribution is requested. In the second case we study what happens if we replace the (truncated) normal distribution by a uniform distribution with a comparable support.

No pretabulated data are available. This exercise is optional.

### 3.12 Model 13 – voluntary

Model 13 is a switching example given by

$$Y = (\tanh(k(X_1 - 0.5))) + s)X_2 \quad (18)$$

where  $X_i \sim U(0, 1)$  and  $k = 50$ .

Depending on  $X_1$ , we treat  $X_2$  differently. A smooth switching behaviour is modelled with the help of the tanh function. This formulation is therefore compatible with **Simlab**'s internal model runner.

We consider two variants:

- (a)  $s = 1.0$  which switches between 0 and  $2X_2$ ,
- (b)  $s = 0.0$  which switches between  $-X_2$  and  $X_2$ .

The statistics and sensitivities for (18) are found in Table 10. From the formulas therein the behaviour of a discontinuous switch becomes clear for  $k \rightarrow \infty$ . This example is again optional for the benchmarking.

Table 10: Statistics and Sensitivities for Model 13

	(a)		(b)	
E(Y)	0.5		0.0	
V(Y)	$5/12 - 2/3 \tanh(k/2)/k = 0.40333$		$1/3 - 2/3 \tanh(k/2)/k = 0.3200$	
	(a)		(b)	
	1	2	1	2
$V(E(Y X_i))$	$1/4 - 1/2 \tanh(k/2)/k = 0.24$	$1/12 = 0.0833$	$1/4 - 1/2 \tanh(k/2)/k = 0.24$	0.0
$V(E(Y X_i))/V$	0.59505	0.20661	0.75	0.0

### 3.13 Level E

We decided to use PSACOIN Level E code as a synthetic PA benchmark case since it is well-studied and widely available, see [6] or <http://www.nea.fr/html/science/docs/pubs/psace.pdf>. For those teams who want to participate in the 'Level-E' exercise, some details follow on how to obtain and run the code. The source code for the Level-E test case can be found on the **SimLab** tutorial pages under 'A REAL TEST CASE', see <http://sensitivity-analysis.jrc.ec.europa.eu/tutorial/>. It compiles under gfortran/linux and under g95(MinGW)/WindowsXP after minor

modifications (adding LOGICAL MXFL, changing one == to .eqv., line continuation character was misaligned, some lines were too long).

In its published form, the Level-E code takes the file LE-Fast-6000.sam and generates out\_le\_fast-6000.dat. Note that it will not overwrite an existing file. A comparison with the supplied out\_le\_fast-6000.dat file shows that the binary does not fully reproduce the results given there (as newly calculated DoseUp\* values always give 0.0).

**SimLab 2.0** comes with a data set from Level-E which is available under the Demo menu. **SimLab 3.0** contains a dynamic loadable library named levele.dll which contains one user-callable function named LEVELE() that reads in a user-supplied sample file and writes the simulation results into a specified file. If this code performs the same as the code from the tutorial pages is still under investigation and should be seen as part of the exercise.

## References

- [1] A. Saltelli, K. Chan, E.M.Scott (eds.), Sensitivity Analysis, Chichester: Wiley, 2000.
- [2] A. Saltelli, S. Tarantola, F. Campolongo, M. Ratto, Sensitivity Analysis in Practice, Chichester: Wiley, 2004.
- [3] A. Saltelli, M. Ratto, S. Tarantola, F. Campolongo, Sensitivity analysis practices: Strategies for model-based inference, Reliability Engineering and System Safety 91(2006), 1109-1125.
- [4] P.-A. Ekström, Eikos – A Simulation Toolbox for Sensitivity Analysis, (Direct Link), 2005.
- [5] B. Krzykacz-Hausmann, The principal variance-based sensitivity indices can be estimated without assuming independence between the input variables, Fifth International Conference on Sensitivity Analysis of Model Output (SAMO 2007), Budapest.
- [6] Nuclear Energy Agency, PSACOIN Level E Intercomparison, Paris: OECD, 1989.