Technische Universiteit Delft
Faculteit Elektrotechniek, Wiskunde en Informatica
Delft Institute of Applied Mathematics

Prestaties van de Morris method met copula's

(Engelse titel: Performance of the copula-based Morris Method)

Verslag ten behoeve van het
Delft Institute of Applied Mathematics
als onderdeel ter verkrijging

van de graad van

BACHELOR OF SCIENCE
in
TECHNISCHE WISKUNDE

door

Rutger van Beek

Delft, Nederland
Juni 2019

BSc verslag TECHNISCHE WISKUNDE

"Prestaties van de Morris method met copula's."
(Engelse titel: " Performance of the copula-based Morris Method)"

Rutger van Beek

Technische Universiteit Delft

Begeleider

Dr. D. Kurowicka

Overige commissieleden

Dr. G.Y.H. El Serafy

Dr. J.G. Spandaw

Juni, 2019

Delft

Abstract

The Morris method ([Morris, 1991]) is a widely used screening method in sensitivity analysis. The method assumes that the input parameters are independent of each other. To overcome the assumption a copula-based Morris method is proposed ([Ţene et al., 2018]). In this report the results of taking the dependencies into account are analyzed for the Morris method. For two examples sensitivity analysis is performed with the Morris method, with copula-based Morris method and by calculating sample correlations with a Monte Carlo simulation. From the analysis it follows that taking dependencies into account can have varying effects for different methods. It turns out that a straight-forward implementation makes the method often practically unusable. The sampling of model evaluation points becomes too computer expensive. The amount of copula evaluations is growing exponentially with the dimension and for copulas without an analytic expression these are already lengthy. The computational intensity can be reduced in two ways. First, one can approximate the probabilities. Different ways of approximating the probabilities are researched. Numerically integrating with the midpoint rule seems to be the best way of approximating the probabilities in the copula-based Morris method. Next to approximating the probabilities, one can also use the independent groups when implementing the method. When the input parameters are correlated there are usually a few groups of correlated parameters rather than that all the parameters are correlated with each other. This can be utilized to more efficiently implement the copula-based Morris method. When the group sizes are not increasing the computational intensity depends linearly instead of exponentially on the number of model parameters. By using both improvements the method can generally be applied to tens or hundreds of parameters in reasonable time, which is desired for a screening method.

Contents

Chapter 1

Introduction

Nowadays mathematical modelling is widely used in the natural sciences, engineering and the social sciences. The idea is to make a phenomenon abstract by describing it in mathematical concepts. The mathematical model is then a function, $y = f(x_1, \ldots, x_d)$, describing the phenomenon. The function, $f : \mathbb{R}^d \to \mathbb{R}$, has d input parameters, x_i, and 1 output, y. In many cases a mathematical model does not have one output. In that case, one can define a summarizing function. Such a function could be an error function or an average of all the outputs. Another possibility is to choose one of the outputs to investigate the sensitivity of. The function can be represented in different ways. It can be a simple mathematical expression, a set of equations, computer code or a black-box-model. In general, one puts some numbers into the function and gets a number out. In a black-box-model the internal workings of the model are unknown, because they are too complicated or are not accessible. However, one can compute the output given the input.

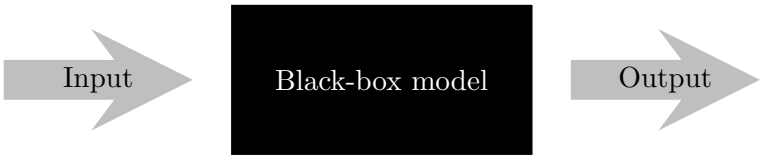


Figure 1.1: Visualization of a black-box-model. The input goes in the black-box, the output comes out. But what happens in between can be everything.

Often computers are used to solve mathematical models. In the last decades, the capacities of computers have increased a lot. This allowed for computers to solve very complicated mathematical models. A more complicated model can grasp more details of the phenomenon. It also means that it is harder to understand the relation between the input and the output. All the details in the model make it impossible to oversee all the different calculations and their effects. Not overseeing the internal workings is not so important if the input is exactly known. One can just use the output of the model. However, most of the time model parameters are estimated based on data or expert judgments. Both methods lead to uncertainty in the parameters. There is not one value for the parameters, but rather a probability distribution on a set of possible values. The uncertainty can then for instance be measured by the variance or entropy of the distribution. A different value for an input parameter generally leads to another value for the output. Therefore the uncertainty in the input propagates to the output of the model. It is hard to predict how the uncertainty propagates when it is not possible to oversee the internal workings of the model or when the model is too complex. The input and output of the model are thus random variables and we write $Y = f(\mathbf{X})$. The effect of uncertainty in the input on the

model output could be very different for different parameters or different models. The analysis of these different effects is called sensitivity analysis.

Sensitivity analysis provides methods to study the relation between the uncertainty of the input and output. In [Saltelli et al., 2004] sensitivity analysis is defined as follows:

"The study of how the uncertainty in the output of a model (numerical or otherwise) can be apportioned to different sources of uncertainty in the model input"

There are plenty of reasons to do sensitivity analysis, but four will be named here.

- The most important reason is when one wants to reduce the uncertainty in the model. Parts of the input that contribute a lot to the uncertainty of the model output can be identified by sensitivity analysis. These parameters should be analyzed further. One can for instance collect more data or ask more experts to reduce the uncertainty of these inputs. As a consequence, the uncertainty in the model output also decreases. Analyzing parameters that are not influential would not have the same effect.

- Furthermore models are often over complicated or over-parametrized. This means that there are parts of the model or parameters which do not have a significant influence on the model output. Over-parametrization is a widespread problem in ecological models ([van Griensven et al., 2006]). Sensitivity analysis methods can indicate whether parameters or parts of the model have negligible effect. These parameters or parts can be simplified or set to a constant value, which reduces the complexity of the model.

- Furthermore one can get a better understanding of the workings of the model by knowing the influence of different input parameters.

- Finally it is also possible to find errors in the model by finding unexpected relations between the input and output.

These are the four main reasons to analyze the sensitivity of the model. As there are many reasons to do sensitivity analysis there are different settings for which different methods are appropriate. One of these methods is the Morris method ([Morris, 1991]). It is a widely used method in sensitivity analysis ([Iooss and Lemaître, 2015]) and can be applied for an efficient preliminary analysis. An assumption of the method is that all the parameters, X_i, are independent. Independence of the parameters is assumed in all major sensitivity methods. There are examples of models for which there are dependencies between the input parameters. Examples are mentioned in [Ţene et al., 2018], [Saltelli et al., 2004] and [Li et al., 2011]. The problem of sensitivity analysis with dependent input is not researched much yet, but for the variance-based methods it is solved ([Mara and Tarantola, 2012]). In [Jacques et al., 2006] the correlated inputs are treated as a group. This is not a real solution, but rather a workaround. A Morris method without the independence assumption was proposed in [Ţene et al., 2018]. Here copulas are used to describe the dependencies between the input parameters. These dependencies are then taken into account in the method.

It turns out that in some cases the implementation of the copula-based Morris method is too computational intensive. This problem was mentioned in [Ţene et al., 2018]. For the implementation a lot of computer evaluations of a copula are necessary. For some copulas such an evaluation is lengthy. Often a more efficient implementation is necessary for the method to be practically useful. The central question in this report is: "Are there more efficient implementations for the copula-based Morris method?". Two ways of increasing the performance of

the method are researched. First of all, probabilities calculated by evaluating copulas can be approximated instead of exactly calculated. Several ways of approximating the probabilities are compared. Next to that, the possibility to use the structure of the dependencies is researched. The effect of differently implementing the method when there are independent groups is researched.

The goal of this report is to present, analyze and improve the copula-based Morris method. The report is organized as follows. First, an introduction in sensitivity analysis is given. One can get a feeling for the desired properties of methods in sensitivity analysis. Next to that the two important examples used throughout the report are presented and a sensitivity analysis is applied to them. In the third chapter the Morris method is presented in two ways. First, the original way of Morris using sampling matrices is presented. Next, a more intuitive geometric interpretation is given. This interpretation is also used to extend the Morris method. We discuss the interpretation of the results with some examples and the choices for the parameters in the Morris method. In the fourth chapter the extended Morris method will be presented. First, an introduction into copulas is given. Here the necessary properties of copulas for this report will be treated. The copula-based Morris method is also applied to an example, where one can see the difference between taking the dependencies in input parameters into account and not doing this. In the fifth chapter the problem of the amount of lengthy copulas evaluations, mentioned before, will be addressed. A solution proposed in [Ţene et al., 2018] is compared to several other approximation of probabilities computed with copulas. Next to that, an implementation using independent groups is presented. Finally, a conclusion is mode on when to use which implementations.

Chapter 2

Sensitivity Analysis

Sensitivity analysis consists of many different methods. There are also different ways to measure sensitivity. In this chapter, we will name the most important groups of methods. The first two groups we distinguish are the local and global methods. Also, we will do a simple sensitivity analysis on two examples.

2.1 Local methods

In a local method the sensitivity of a model is analyzed only in a certain point. The point is called the base value. The most common approach is to calculate or estimate the partial derivatives in the base value with respect to the input parameters. A partial derivative indicates how much the model output changes when the corresponding input changes. Partial derivatives are therefore one way to measure the sensitivity of a model. Parameters with partial derivatives close to zero are not so influential. The model output does not change much when the value of the parameter changes. High values (both positive and negative) of the partial derivative indicate an influential parameter. A lot of uncertainty in an influential parameter will lead to a lot of uncertainty in the model input. The method of calculating the partial derivatives is a One-At-a-Time(OAT) method. These methods change only one parameter at a time to isolate the effects. The effects can thus easily be assigned to an input parameter. The disadvantages are that for each parameter the calculations should be repeated and the interactions between input parameters cannot be measured. The counterpart of local methods are the global methods, which will be discussed next.

2.2 Global methods

Global methods allow for the exploration of a whole variation range of the inputs. One can use bounds for each parameter. The sensitivity of the model will then be analyzed for the entire range between these bounds. The values of the input variables are unknown and thus random variables. Within the global sensitivity analysis methods there are a lot of different methods for different situations. These methods have different sensitivity measures. A few sensitivity measures used in global methods will be presented in the next section.

2.2.1 Sensitivity measures

The first sensitivity measure that will be discussed are the partial derivatives of the output with respect to the different inputs. These were already mentioned in section 2.1 where the local methods were discussed. The partial derivatives compare a change in the input with a change in the output. In a local method, it is obvious in which point the derivatives should be calculated. In global methods, there are many possible values for the parameters, so there is not one derivative. Partial derivatives are calculated in different points. The idea is to analyze the distribution of the derivatives per parameter. A simple summary of the distribution is the average. The average gives a good idea about the general effect of the parameter. The variance and other properties of the distribution can also be used to analyze the effects of parameters.
Another way to measure the sensitivity is by calculating the correlation of the model output with the model input. All correlations are defined to be a measure of the dependence or association of two variables in some sense. One can calculate the correlation of the model output, Y, with all the model input parameters, X_i. This will then indicate how dependent the model output is on the different parameters. Parameters with a correlation close to 1 have a high positive dependence, whereas parameters with a correlation close to -1 have a high negative dependence. Both are considered influential parameters. Correlations close to zero indicate that there is almost no dependence of that type. So the (absolute) values of the correlations are compared to see what the most influential parameters are. The sign of the correlation indicates the sign of the general effect. There are different types of correlations that can be used.

- The most commonly used correlation coefficient is the Pearson product-moment correlation coefficient. This correlation measures linear dependence.
- Another type of correlations are the rank-based correlations in which the ranks of both sets are compared. For instance, Kendall's τ and Spearman's rho are examples of rank correlations. In this report Spearman's ρ is used in the sensitivity analysis to measure rank correlation. All rank correlations will rank have comparable values. The usage of rank correlations in sensitivity analysis is very elaborately discussed in [Saltelli and Sobol, 1995].

Other correlations and more information about the correlations above can be found in [Nelsen, 2006]. They can all be used as a sensitivity measure when applied to the model output and a model parameter.

The last type of sensitivity measure that will be discussed is the variance decomposition. Here the uncertainty in the model output is measured by the variance. The sensitivity measure is then the part of the variance in the model output that can be apportioned to the variance in a set of input parameters. The set can be taken to be a single parameter, if one wants to know the influence of individual parameters. The method was introduced by Sobol in 1993 and is in detail presented in [Sobol, 1993].

2.2.2 Sensitivity of a linear model

In the previous section different measures for sensitivity were introduced. In this section all of these sensitivity measures will be calculated for the simple linear model:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_d X_d \tag{2.1}$$

It is assumed that the input parameters are independent and that they are distributed uniformly on $[0, 1]$. Intuitively the sensitivity of parameters in a linear model should be closely related

to the corresponding β as this coefficient describes the relation between the model input and output.

Partial derivatives The partial derivatives of a linear function are just equal to the corresponding coefficients.

$$\frac{\partial Y}{\partial X_j} = \beta_j \tag{2.2}$$

The distribution of partial derivatives is equal to a single value.

Correlations The linear correlations can be calculated using the independence of the parameters.

$$\begin{aligned}
\rho(Y, X_j) &= \frac{\mathrm{Cov}(Y, X_j)}{\sqrt{\mathrm{Var}(Y)}\sqrt{\mathrm{Var}(X_j)}} \\
&= \frac{\mathrm{Cov}(\sum_{i=1}^{d} \beta_i X_i, X_j)}{\sqrt{\mathrm{Var}(\sum_{i=1}^{d} \beta_i X_i)}\sqrt{\mathrm{Var}(X_j)}} \\
&= \frac{\mathrm{Cov}(\beta_j X_j, X_j)}{\sqrt{\sum_{i=1}^{d} \beta_i^2 \mathrm{Var}(X_i)}\sqrt{\mathrm{Var}(X_j)}} \\
&= \frac{\beta_j \mathrm{Var}(X_j)}{\sqrt{\sum_{i=1}^{d} \beta_i^2 \mathrm{Var}(X_i)}\sqrt{\mathrm{Var}(X_j)}} \\
&= \frac{\beta_j \sqrt{\mathrm{Var}(X_j)}}{\sqrt{\sum_{i=1}^{d} \beta_i^2 \mathrm{Var}(X_i)}}
\end{aligned} \tag{2.3}$$

Apportioning of variance For the apportioning of the variance we first observe the following:

$$\mathrm{Var}(Y) = \mathrm{Var}(\sum_{i=1}^{d} \beta_i X_i) = \sum_{i=1}^{d} \beta_i^2 \mathrm{Var}(X_i) \tag{2.4}$$

Now one can see that the fraction of the variance of the model output explained by parameter j is equal to $\frac{\beta_j^2 \mathrm{Var}(X_j)}{\sum_{i=1}^{d} \beta_i^2 \mathrm{Var}(X_i)}$. This sums to ones as desired.

All the three sensitivity measures were calculated for the linear model. In all the cases the sensitivity was closely related to the corresponding coefficient. And in all cases a bigger β corresponded to a bigger influence. This should not come as a surprise as the different sensitivity measures all try to measure (roughly) the same concept.

2.3 Monte Carlo simulation

In the previous section, we have seen that calculating the sensitivity measures is easy for the linear model. For complicated models calculating the sensitivity measures will come down to estimating them. This will often be done using some form of Monte Carlo simulation ([Metropolis and Ulam, 1949]). More information about the implementation of Monte Carlo simulation can be read in [Robert and Casella, 2013]. In Monte Carlo simulation a random sample of the parameter is used to simulate the model. By doing this many times the sensitivity measures can be estimated from the samples. The law of large numbers makes sure that the estimate converges to the real value. The estimate will differ from the exact value, but for a large

sample the difference will be small. Different implementations for different sensitivity measures result in different sensitivity analysis methods. For correlations it is intuitive how Monte Carlo simulation can be used. We can use a lot of Monte Carlo samples to calculate the sample correlation. The sample correlation is an unbiased estimator for the actual value of the correlation. Monte Carlo simulation simulations can also be used when there are dependencies between the parameters. We should then use copulas to do the random sampling in the parameter space. Copulas are introduced and explained in section 4.1.

2.4 Screening methods

In the previous section, an idea was given for methods that can analyze the sensitivity of more complex models. One part of the global method,s where random sampling is often used, are the screening methods. These are methods that estimate the sensitivity measures very efficiently. They allow for the analysis of models with tens or hundreds of parameters ([Saltelli et al., 2004]). With efficient we mean that not many model evaluations are necessary to produce a meaningful result. Screening methods give a qualitative output rather than a quantitative input. In most sensitivity analysis methods the parameters get scores and these values have a direct meaning related to the sensitivity of the model. Screening methods usually provide more of a quantitative output. They can rank the parameters or give the nature of the effects of certain parameters, but cannot exactly quantify the sensitivity.

There are many different screening methods. Most screening methods are based on the idea of fractional factorial designs ([Box and Hunter, 1961]). A few discrete levels for each variable are considered. For instance for a certain parameter with lower bound 0.5 and upper bound 0.9 one only considers the values $0.5, 0.6, 0.7, 0.8, 0.9$ instead of the whole range. The number of levels can of course vary. A full factorial design is when all the combinations of values for all the parameters are used. In a fractional factorial design a subset of the full factorial design is used. Different ways of choosing this subset lead to different methods. Generally, the subset is (partially) random. Then a fractional factorial design is closely related to a Monte Carlo simulation. Only considering each parameter at a few levels makes sure that the coverage of the space is good and the results will be more reliable. By only using a fraction of the full factorial not many computations are needed.

Another often chosen method to speed up the analysis is to do the screening by groups. By changing multiple values at the same time the amount of model evaluations is lowered. Of course the causes of the effect cannot be assigned to a single variable. Therefore often the process is repeated a few times with different partitions each time. By averaging over the different runs the effects of a single parameter can be estimated. This is different from the OAT method as mentioned in section 2.1. In contrary to OAT methods the effects of a single parameter are harder to isolate, but fewer computations are necessary.

Examples of screening methods can be found in [Cotter, 1979], [Andres and Hajas, 1993] and [Bettonvil and Kleijnen, 1997]. There are many more, but in general they allow for a quick exploration of the parameter space. They give not so much information, but allow for an indication of the types of the effects and the ranking of parameters based on their influence. Most of the methods make assumptions on the underlying model that is analyzed. The Morris method does not make such assumptions and is therefore widely used.

2.5 Examples

The two examples will be described first and a sensitivity analysis is performed. The examples will be used throughout the report to compare different methods in sensitivity analysis.

2.5.1 Linear model with interactions

The first example will consist of three linear functions with interactions. These functions all have three input parameters, but different coefficients. The coefficients are chosen such that the nature of the effects of the parameters is very clear. We hope to see these effects then clearly in the sensitivity analyses. The functions, $f_i : [0,1]^3 \to \mathbb{R}$, will be of the following form:

$$f_i(x_1, x_2, x_3) = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{1,2} x_1 x_2 + \beta_{1,3} x_1 x_3 + \beta_{2,3} x_2 x_3 \tag{2.5}$$

A few different sets of parameters will be used. In the first function the parameters will be chosen such that function has no interactions. The coefficients corresponding to interaction effects will be set to almost zero. The size of the influence will increase, so x_1 has the smallest influence and x_3 the biggest, but the influence of x_3 will be negative. The second function will consist of a variable, x_1 with negligible effects and two parameters, x_2 and x_3 who also have interaction. The non-interaction effect of x_3 will be negative. Then the overall effect of this parameter is non-monotonic. The last function will have negative interaction effects between the parameters x_1 and x_2. The effect of these two parameters are then non-monotonic. x_3 will have a no interaction effects. The corresponding values of β are summarized in figure 2.1.

function	f_1	f_2	f_3
β_1	1	0.1	0
β_2	5	5	5
β_3	-10	-5	10
$\beta_{1,2}$	0.1	0.3	-10
$\beta_{1,3}$	0.1	0.3	0.3
$\beta_{2,3}$	0.1	10	0.3

Figure 2.1: The coefficients for the three different test functions.

In this report great emphasize is put on making it possible to handle dependent input parameters in sensitivity analysis. Therefore some correlations are defined for the input parameters. For the methods that allow for inputs to be correlated, the correlations will be set to the values in figure 2.2.

	x_1	x_2	x_3
x_1	1	0.9	-0.5
x_2	0.9	1	0
x_3	-0.5	0	1

Figure 2.2: The correlations of the input parameters for the functions f_1, f_2 and f_3.

2.5.2 Sensitivity of the linear model with interactions

We will now do a sensitivity analyses of these simple functions. The method described in section 2.3 will be used. Random samples from the sample space will be used to create a sample of

model evaluations. And the sample correlation with the different inputs will be calculated. The Pearson and Spearman correlations are calculated. The ranks are based on the absolute value of the Spearman correlation coefficient. Next to that, we will also perform the analysis while taking the dependencies into account. The resulting sample correlation will be compared. For both simulations a sample size of $N = 10000$ was used.

	Pearson	Spearman	ranks
X_1	0.0965	0.08999	3
X_2	0.4453	0.4227	2
X_3	-0.8828	-0.8887	1

	Pearson	Spearman	ranks
X_1	0.8223	0.8163	2
X_2	0.5021	0.4823	3
X_3	-0.8660	-0.8700	1

Figure 2.3: The results of the sensitivity analysis for f_1 using sample correlations with a sample size of 10000. In the left gable the dependencies are not taking into account, whereas in the right table they are. The ranking is based on the absolute values of the Spearman correlation coefficient.

The results of the sensitivity analysis for the first function can be seen in figure 2.3. The first function was constructed such that there would be no interactions. The correlations resulting from the simulation with independent parameters are as expected. The size of the correlation is related to the size of the coefficient, β, and the signs and ranks are correct. We can also compare the correlation with the Pearson correlations computed in 2.2.2 for the linear model without interactions and independent variables. The variance of each variable is $\text{Var}(U[0,1]) = \frac{1}{12}$, because they are all assumed to be uniformly distributed on $[0,1]$. The Pearson correlation coefficients for the model without interaction can be computed with equation 2.3 and are:

$$\rho(X_1, Y) = 0.0891 \quad \rho(x_2, Y) = 0.4454 \quad \rho(X_3, Y) = -0.8909$$

The correlations are very similar to the sample correlations, because the interactions are very small for f_1. When we look at the right table of figure 2.3, where the dependencies are taken into account, we see that X_1 is now very much correlated with the model output. This is because it is correlated with X_2 and X_3 with the signs such that variable is very much correlated with the model output. Here the dependencies have a big effect on the results of the sensitivity analysis.

	Pearson	Spearman	ranks
X_1	0.0381	0.0365	2
X_2	0.961	0.9738	1
X_3	0.0297	0.0242	3

	Pearson	Spearman	ranks
X_1	0.7714	0.7793	2
X_2	0.9634	0.9753	1
X_3	-0.0130	-0.0146	3

Figure 2.4: The results of the sensitivity analysis for f_2 using sample correlations with a sample size of 10000. In the left gable the dependencies are not taking into account, whereas in the right table they are. The ranking is based on the absolute values of the Spearman correlation coefficient.

For function f_2 only X_2 seems to have a significant effect in the case of independent variables. The other two variables seem to have negligible effects. We know from the function that X_3 has a big effect, but that the effects are have different signs. The linear part is negative and the interaction effects is positive. The parameter is important, but this is not seen in the sensitivity analysis. With dependencies X_1 becomes important, because it is correlated with X_2 which has

a big effects. The effects of X_3 still do not come up in the analysis.

	Pearson	Spearman	ranks
X_1	-0.3384	-0.3166	2
X_2	0.0297	0.0295	3
X_3	0.9047	0.9132	1

	Pearson	Spearman	ranks
X_1	-0.6765	-0.6677	2
X_2	-0.2317	-0.2184	3
X_3	0.9434	0.9491	1

Figure 2.5: The results of the sensitivity analysis for f_3 using sample correlations with a sample size of 10000. In the left gable the dependencies are not taking into account, whereas in the right table they are. The ranking is based on the absolute values of the Spearman correlation coefficient.

Lastly, for function 3 we see that X_3 is very important as expected from the coefficients. In the case with independent parameters the interaction effect of X_1 and X_2, which is quite big, is not so clearly visible. Similar as with the previous function, a parameter with effects of opposite signs is unjust indicated as not important. The interaction effect becomes a lot more obvious, when the big dependency between X_1 and X_2 is taken into account.

2.5.3 Population model

In the previous section, an example of sensitivity analysis was given where the effects of the different parameters were clear by looking at the model function. We will now consider an example were the effects of different parameters is not so clear. We will consider a model that describes the development of different populations of animals that compete for resources. These developments can be described with the competitive Lotka-Volterra equations ([Zhu and Yin, 2009]). The population sizes will grow logistically when there is no competition. However, as both animals compete for the same resources, the existing of one has a harmful effect on the growth of the other. The equations describing the growth for the population sizes of species x_1 and x_2 are as follows.

$$\begin{cases} \frac{dx_1}{dt} = r_1 x_1 \left(1 - \frac{x_1 + \alpha_{1,2} x_2}{K_1}\right) \\ \frac{dx_2}{dt} = r_2 x_2 \left(1 - \frac{x_2}{K_2}\right) \end{cases} \tag{2.6}$$

In equation 2.6 the competitive Lotka-Volterra equations are given, where the r's are the growth rates for the corresponding species. This means that when there would be no competition with other animals (from other or the same species) the population would grow with this rate. $\alpha_{1,2}$ indicates the size of the harmful effect of x_2 on x_1. When $\alpha_{1,2}$ is large it means that x_2 uses a lot of resources x_1 needs. In combination with a big population size of x_2 it will decrease the growth of x_1. The K's are the carrying capacities of the environment under investigation. This is the maximum population size of the corresponding animal an environment can sustain. Also the starting population sizes, $x_1(0)$ and $x_2(0)$ are input parameters. The values of the parameters are not exactly known, but bounds can be estimated. A uniform distribution is then assumed between these bound. All the parameters and their bounds are summarized in figure 2.6. The values for the parameters are chosen similar to the example studied in [Vano et al., 2006]. The model output we will focus on the ratio of the two population sizes after five years, $x_2(5)/x_1(5)$. Similarly as in the example in the previous section we draw 10000 random samples form the parameter space and calculate the resulting model output each time. In figure 2.7 the uncertainty in the model output is visualized by a histogram of the simulations.

parameter	interpretation	lower bound	upper bound
r_1	per capita growth rate species 1	0.8	1.2
r_2	per capita growth rate species 2	0.5	0.8
$\alpha_{1,2}$	the harmful effect of species 2 on 1	0.2	0.8
K_1	Carrying capacity for species 1	0.9	1.1
K_2	Carrying capacity for species 2	0.8	1
$x_1(0)$	starting population species 1	0.3	0.7
$x_2(0)$	starting population species 2	0.2	0.5

Figure 2.6: The different parameters for the competitive Lotka-Volterra model with an estimate for their value.

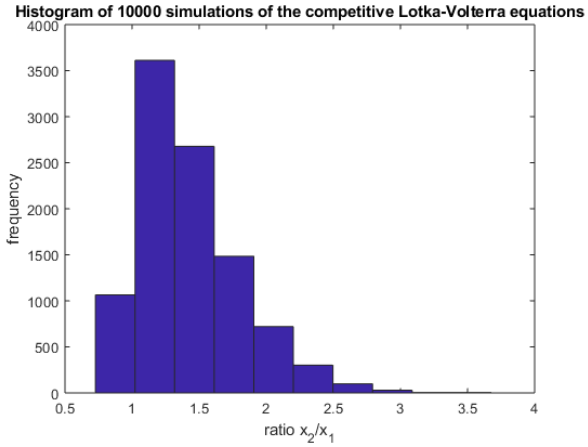


Figure 2.7: The distribution of the model output (ratio x_1/x_2 after five years) of 10000 simulations. The parameters were assumed to be independent for these simulations.

Until now independence of the input parameters was assumed. However, for biological reasons there are correlations between certain parameters. For instance, if carrying capacities are closely correlated. The species compete for roughly the same resources. When the environment turns out to contain more resources for one species it probably also contains more resources for the other species. The correlations would have to assessed by experts. In figure 2.8 the estimated correlations are summarized. The correlations between parameters and themselves are of course 1.

parameters	r_1	r_2	α_{21}	K_1	K_2	$x_1(0)$	$x_2(0)$
r_1	1	0	-0.5	0	0	0.5	0
r_2	0	1	0.5	0	0	0	0.5
α_{21}	-0.5	0.5	1	0	0	-0.5	0.5
K_1	0	0	0	1	0.5	0	0
K_2	0	0	0	0.5	1	0	0
$x_1(0)$	0.5	0	-0.5	0	0	1	-0.5
$x_2(0)$	0	0.5	0.5	0	0	-0.5	1

Figure 2.8: The estimated correlations in the parameters.

Simulations from a set of dependent parameters can be done using copulas, which will be discussed in section 4.1. In figure 2.9 the histogram for the ratio of two populations after five

years can be seen. The histogram looks very similar to the histogram of the simulations where the correlations were not taken into account. The uncertainty in the model output with and without dependencies in the input parameters is thus very similar. In the next section we will see whether the dependencies matter for the sensitivity analysis.

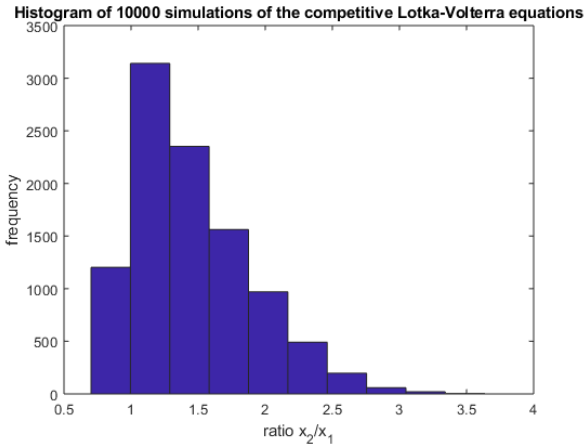


Figure 2.9: The distribution of the model output (ratio x_1/x_2 after five years) of 10000 simulations where the correlations between the input parameters are taken into account.

2.5.4 Sensitivity of the population model

A sensitivity analysis of the model described in the previous section will now be performed. For this we will use the same method as used in the other example. We used a Monte Carlo simulation to create a sample and calculated the correlations of the model output with different inputs. We will do this here for the model with and without correlated input parameters. Next to that, we are going to calculate the correlation of the model output with different inputs as a sensitivity analysis.

A sample size of 10000 will be used. Both the linear correlation and Spearman's ρ will be estimated from this sample. 10000 is a big enough sample size to come close enough to the actual values. The first observation that we can make from figure 2.10 is that taking into account

parameter	Pearson	Spearman	ranks
r_1	0.0159	0.0069	7
r_2	0.2044	0.2046	5
$\alpha_{1,2}$	0.5941	0.7999	1
K_1	-0.3388	-0.3299	2
K_2	0.3165	0.3117	3
$x_1(0)$	-0.0632	-0.0589	6
$x_2(0)$	0.2647	0.2550	4

parameter	Pearson	Spearman	ranks
r_1	-0.3348	-0.3494	5
r_2	0.5996	0.6236	2
$\alpha_{1,2}$	0.7130	0.8977	1
K_1	-0.1784	-0.1653	6
K_2	0.1486	0.1306	6
$x_1(0)$	-0.4101	-0.4143	4
$x_2(0)$	0.5853	0.5992	3

Figure 2.10: The results of the sensitivity analysis of the competitive Lotka-Volterra. The sample linear correlation and Spearman's *rho* are calculated based on a Monte Carlo sample. Also the parameters are ranked based on their influence. On the left the correlations between the model input parameters are not taken into account. On the right the model input parameters are taken into account.

the dependencies in the model really makes a big difference. The parameters r_1, r_2, $x_1(0)$ and $x_2(0)$ are a lot more correlated with the model output, whereas the correlation of K_1 and K_2 has decreased. The parameter $\alpha_{1,2}$ is the only parameter that is not much affected. All the parameters for which the correlations increased a lot are exactly the ones correlated with $\alpha_{1,2}$. Because $\alpha_{1,2}$ is such an influential parameter the parameters correlated with $\alpha_{1,2}$ are now also well correlated with the model output. The changes in the correlations indicate that in this case it was important to take the dependencies into account.

Another observation that can be made is that the linear correlation coefficient is very similar to Spearman's ρ. It can be concluded from this that the effects are almost linear. Only for $\alpha_{1,2}$ the values of the two correlations coefficients differ much. This indicates that the effect of $\alpha_{1,2}$ is partially non-linear. Lastly we can draw conclusions about the sensitivity of the model. We will look at the right table. As this is the table were the correlations of the input parameters are taken into account. The most important parameter seems to be $\alpha_{1,2}$. If one wants to decrease the uncertainty in the model output than one should investigate this parameter. Other important parameters are K_1 and K_2. It seems that the parameters related to species 2 are more important then those related to species 1. The effects of K_1 and K_2 are the smallest but not negligible. This was an example of how sensitivity analysis could be applied to a simple model and what conclusion can be made from the analysis. The example also showed that in this case it was important to take the dependencies into account.

In this chapter a general introduction into sensitivity analysis was given. The most important sensitivity measures were defined. Next to that it was shown how these can be calculated for a simple linear model. And an idea was given how the sensitivity measures can be estimated in more complicated models. Two examples of a sensitivity analysis were given. First on simple functions were the effects were obvious and then on a more realistic, but sill small model. In the next chapter the Morris method is introduced. This is a method that can estimate the distribution of partial derivatives for all types of models.

Chapter 3

Original Morris Method

Sensitivity analysis and its main ideas were introduced in the previous chapter. In this chapter a global sensitivity analysis method will be presented. The method is the Morris method, which was proposed by Max D. Morris in 1991 ([Morris, 1991]). It is an OAT screening method. At the end of the chapter the method will be applied to both the examples that were introduced in the previous chapter.

3.1 Elementary effects

In the Morris method the following setting is assumed. The range defined on the parameters can be used to scale the parameter space to the unit hypercube, $\tilde{f} : [0,1]^d \to \mathbb{R}$. Because this can be done without loss of generality, the parameter space will be assumed to be $[0,1]^d$ for the rest of the report. As only bounds for the parameters are known a uniform distribution is assumed between the bounds, $X_1 \sim U[0,1]$.

The Morris method uses the distribution of derivatives as the sensitivity measure. It was shown in section 2.2.1 how the (partial) derivatives are related to the sensitivity of the model. In most models calculating exact derivatives is not possible. Many models have no explicit expression of partial derivatives, because they consist of computer code for example. Therefore not the exact, but an approximation of the partial derivative is used. This approximation is called an elementary effect and is defined as follows:

$$\mathrm{EE}_j(x_1, \ldots, x_d) = \frac{f(x_1, \ldots, x_j + \delta, \ldots, x_d) - f(x_1, \ldots, x_j, \ldots, x_d)}{\delta} \tag{3.1}$$

is the elementary effect in the point $(x_1, \ldots, x_d)$ in the direction of input variable x_j. The parameters are changed one-by-one, so that the effect of the individual parameters is isolated as generally happens in an OAT method. The idea in the Morris Method is to sample these elementary effects and analyze their distribution. It will be possible to distinguish three types of influence:

- negligible effects
- (close to) linear effects
- non-linear or interaction effects.

3.2 Sampling elementary effects

In equation 3.1 it can be seen that the calculation of an elementary effect uses two model evaluations. In this section it will be shown in which two points the model should be evaluated. Especially how these points can be chosen such that the model evaluations can be used efficiently. The Morris method is meant to be a screening method. The number of model evaluations should be kept to the minimum. The number of samples of elementary effects per parameter that is calculated is denoted as r. For the sampling of elementary effects a fractional factorial design will be used. This is used often in screening methods and the basic idea is already mentioned in section 2.4. Each parameter is only considered at a few different levels in the range. The full factorial design can be represented as a grid. In a fractional factorial design we sample a subset of the grid points to evaluate the model. These points will then be used to calculate elementary effects. Morris used sampling matrices to explain and notate the sampling of elementary effects. Next to that, there is also a more intuitive geometric interpretation as given in [Ţene et al., 2018]. Both will be presented in this section. The geometric approach will be used in chapter 4 to extend the Morris method.

3.2.1 Sampling on a grid

Instead of looking at the problem as sampling in a full factorial design we will look at the problem as sampling from a grid, which in the end comes down to the same thing. However, the grid can be visualized. For the grid we choose a regular grid with p levels in each dimension. The step over which the elementary effects will be calculated is equal to $\delta = \frac{k}{p-1}$ for some $k \in \{1, 2, \ldots, p-1\}$. k is thus over many different levels one elementary effect is calculated. In the visualizations in this section $k = 1$ is chosen, to make the figures clear. A discussion about choosing these parameters as well as r can be found at the end of this chapter. The grid is chosen such that elementary effects with step δ are calculated between grid points. The grid points will be $\{0, \frac{1}{p-1}, \ldots, \frac{p-2}{p-1}, 1\}^d$. The discretization of the 3-dimensional cube is shown in figure 3.1, where $p = 3$ is used. The axis shown in the picture will be used throughout this section for all pictures. In some pictures they are left out because of readability.

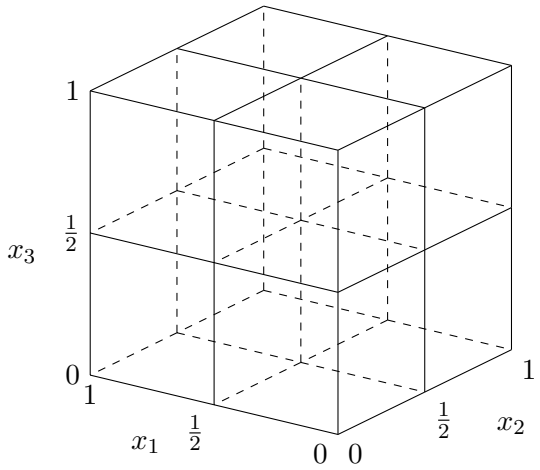


Figure 3.1: The grid of the 3-dimensional unit hypercube with 3 levels in each dimension

In general, one needs two model evaluations to calculate one elementary effect as one is calculating a change. This can also be seen in equation 3.1, where the model function f is evaluated in two points. However, by efficiently using the grid, model evaluations can be reused. The

endpoint of the previous calculation is used as a starting point for the next calculation. This effect is visualized in figure 3.2. In the figure, points used to calculate elementary effects are joint by coloured lines, where the start and endpoint of the line represent a point in which the model should be evaluated. In the left subfigure the elementary effects are calculated apart. For each elementary effect two points next to each other are used. This requires 12 model evaluations for the 6 elementary effects. In the right subfigure the elementary effects are calculated with the same equation 3.1, only now the model evaluations are reused. After an elementary effect is calculated, a point next to the last point is chosen to calculate another elementary effect together with a point used in the previous calculation. In the right figure only 8 model evaluations are needed for the same amount of elementary effects.

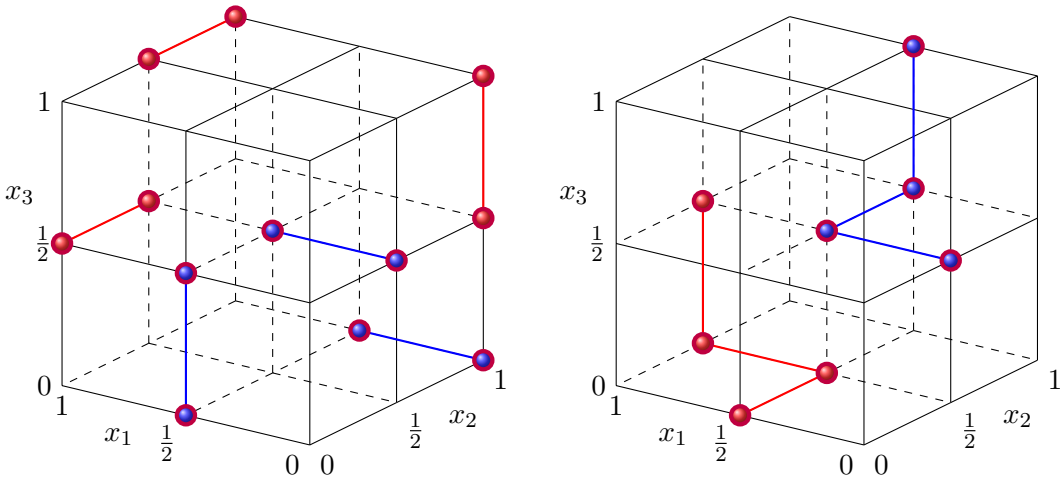


Figure 3.2: The difference between calculating elementary effects apart (left) or over paths (right).

We will now look at the blue paths in figure 3.2 more closely. The following elementary effects are calculated with the blue paths. In both the left and right picture an elementary effect in each direction can be calculated over the blue paths.

Elementary effect calculations (left)	Elementary effect calculations (right)
$\mathrm{EE}_1(0, \frac{1}{2}, \frac{1}{2})) = \frac{f(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}) - f(0, \frac{1}{2}, \frac{1}{2})}{\frac{1}{2}}$	$\mathrm{EE}_1(0, \frac{1}{2}, \frac{1}{2})) = \frac{f(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}) - f(0, \frac{1}{2}, \frac{1}{2})}{\frac{1}{2}}$
$\mathrm{EE}_2(1, 0, \frac{1}{2})) = \frac{f(1, \frac{1}{2}, \frac{1}{2}) - f(1, 0, \frac{1}{2})}{\frac{1}{2}}$	$\mathrm{EE}_2(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})) = \frac{f(\frac{1}{2}, 1, \frac{1}{2}) - f(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})}{\frac{1}{2}}$
$\mathrm{EE}_3(0, 1, \frac{1}{2})) = \frac{f(0, 1, 1) - f(0, 1, \frac{1}{2})}{\frac{1}{2}}$	$\mathrm{EE}_3(\frac{1}{2}, 1, \frac{1}{2})) = \frac{f(\frac{1}{2}, 1, 1) - f(\frac{1}{2}, 1, \frac{1}{2})}{\frac{1}{2}}$

Here one can see that in both cases a elementary effects using the points $(0, \frac{1}{2}, \frac{1}{2})$ and $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$. However, in the right picture the point $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ is reused to calculate the second elementary effects. In the left example this is not the case and the two new model evaluations are used. The same applies to the calculation of the third elementary effect. This can be generalized by calculating the elementary effects over paths that have a change in each direction exactly once. These paths allow for the calculation of one elementary effect in each direction. One should therefore sample r of these paths to obtain all the necessary elementary effects. It would result in a total of $r(d+1)$ instead of $2dr$ model evaluations. This is a big improvement, especially if d is large. Sampling these paths will be discussed in the next sections.

3.2.2 Sampling paths using matrices

Above the sampling of elementary effects is reduced to the sampling of r paths over a grid. A path is a set of points on the grid where each point only differs from the previous in one dimension and a distance δ. Morris used sampling matrices to describe the sampling of elementary effects. In the sampling matrices each row represents a point of the path. Two consecutive rows of the sampling matrices should differ in only one column and this difference should be equal to δ. Next to that, there should be exactly one such difference in each column. These matrices can be constructed in the following manner. First a random point on the grid, called a base value $\mathbf{x}^*$, is chosen. Let B be a $d+1$ by d matrix of 0's and 1 's with the property that for every column there are two rows of B, which only differ in that particular column. An example of such a matrix would be:

$$B = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix} \tag{3.2}$$

Let D^* be a diagonal matrix with values randomly 1 or -1. Let $J_{d+1,d}$ be an $d+1$ by d matrix of all ones. Then $\frac{1}{2}[(2B - J_{d+1,d})D^* + J_{d+1,d}]$ is a matrix similar to B, but now for each column either the column is the same or all the 0's and 1's are switched. To give equal probability to all the points a d by d random permutation matrix P^* is applied as well. This leads to the following matrix formula:

$$B^* = \left(J_{d+1,1}\mathbf{x}^* + \frac{\delta}{2} \right) [(2B - J_{d+1,d})D^* + J_{d+1,d}]P^* \tag{3.3}$$

The rows of B^* form a path on the grid with the desired properties. Each matrix or vector with a star indicates that it is generated randomly. By independently sampling r samples for each of these random matrices, r matrices B^* are obtained. Each B^* corresponds to one path in the grid described in section 3.2.1. A row of each B^* corresponds to a point in which the model should be evaluated.

We will again look at the blue paths in the right picture of figure 3.2. The sampling matrix belonging to this path is written down in equation 3.4

$$B^* = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{2} & 1 & 1 \end{bmatrix} \tag{3.4}$$

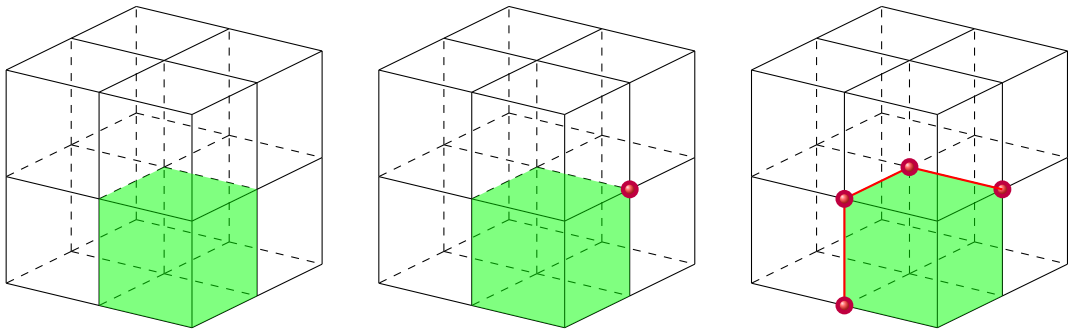
This matrix can be constructed by using equation 3.3 and the following matrices:

$$\mathbf{x}^* = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \qquad D^* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad P^* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

All the diagonal entries of D^* are 1, because the paths only changes in the positive direction. If the path would change in the negatives direction for a certain parameter then the corresponding entry on the diagonal of D^* would be negative. Next to that, the permutation matrix is equal to the identity matrix. This is because coincidentally the elementary effects are calculated in the order 1, 2, 3. Another order could be sampled. That would results in a different matrix for P^* and a different path will result. Using sampling matrices to obtain a path in the grid is not very intuitive. A geometric approach is more intuitive and this will be presented next.

3.2.3 Sampling paths, geometric interpretation

Now a geometric interpretation of the same process will be given. This interpretation was presented in [Ţene et al., 2018]. In order to sample such a path one should first sample a cell in the grid (green in figure 3.3). The cell should be a hypercube with the Morris step δ as size in each dimension and grid points as corners. There are $(p-k)^d$ of such cells and each cell should have equal probability. Next one of the 2^d corner of the cell should be sampled with equal probability. This corner will be the starting point of the path. Finally one should use a random permutation of the dimensions to decide in which order all the dimensions are traversed. So if a certain dimension is the first in the permutation than the starting point of the path is changed δ in the corresponding dimension and such that a new corner of the original cell is obtained. This process is repeated for the entire permutation. The process of sampling a path is illustrated in figure 3.3, where we see the three steps above. First the cell is sampled, then a starting point is sampled and lastly a permutation is used to create a path.



(a) Sampling a cell from the grid.

(b) Sampling a corner of the cell as starting point of the path.

(c) Sampling a permutation to obtain a path on the cell.

Figure 3.3: Visualization of the geometric interpretation of choosing a path. The axis correspond to the axis used in figure 3.1 and have been left out for readability.

This way of sampling paths can be done independently r times. After the model is evaluated in each point all the desired elementary effects can be calculated. To sample cells Latin Hypercube Sampling can be used to obtain a good spread over the parameter space. A good spread is especially important when r is small. Latin Hypercube Sampling was first presented in [McKay et al., 1979]. In Latin Hypercube Sampling (LHS) all the dimensions are divided into n equally sized intervals. A Latin Hypercube Sample is then a sample of n random numbers with exactly one sample in each interval for every dimension. Per dimension the random points are spread out over all the intervals. For a d-dimensional hypercube with n intervals in each dimension this can be arranged in the following way: First generate d independent permutations $\pi^1, \ldots, \pi^d$ of $\{1, \ldots, n\}$. Furthermore let $U_i^j \sim U[0,1]$ then

$$V_i^j := \frac{\pi_i^j - 1}{n} + \frac{U_i^j}{n}, \quad j = 1, \ldots, d, \quad i = 1, \ldots, n \tag{3.6}$$

An example of this transformation is given in figure 3.4. The permutations indicate in which intervals the sample will be. Then in each interval a point is uniformly sampled. We can leave the second term of equation 3.6 out, because we need cells instead of samples in the Morris method. We can use the permutations to obtain n cells who are all different in each dimension. To apply LHS for the Morris method we should set $n = p - k$ to let the borders of the LHS

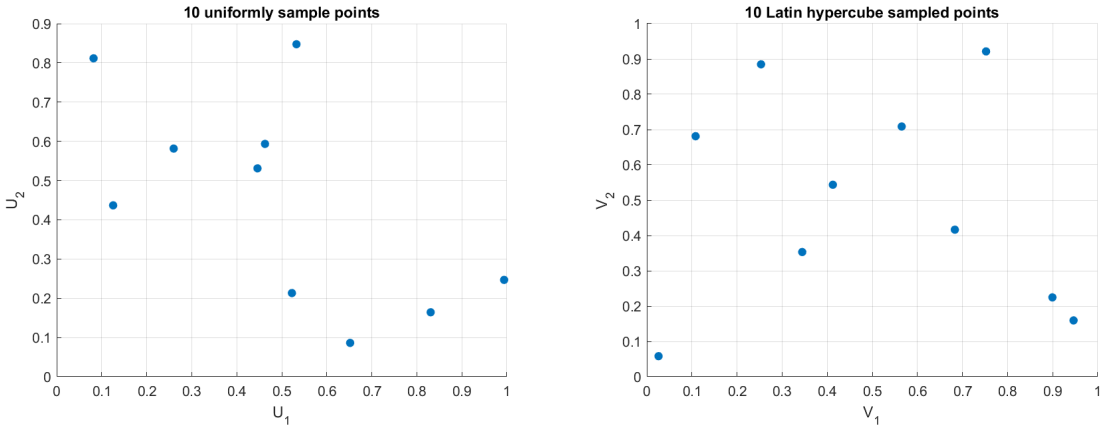


Figure 3.4: The uniform sample on the left is transformed to a LHS sample using equation 3.6

overlap with the borders of the cells. Possibly r of the Morris method is bigger than n, the amount of samples generated. In this case the process can be repeated until enough cells are obtained. Then more than one sample can occur per interval, but overall a good spread of the cells is maintained. Now that it is clear how to obtain all the necessary elementary effects their distribution will be analyzed to make a conclusion about the sensitivity of the model.

3.3 Analyzing the distribution of elementary effects

Using the above procedure to obtain model evaluation points, elementary effects can be calculated. This would result in r sampled elementary effects for each parameter. Several sensitivity measures are defined to indicate properties of the parameter from the distribution of elementary effects. We denote the elementary effect of parameter j in path i as $\mathrm{EE}_j^{(i)}$. In the original paper two sensitivity measures were proposed, the average and standard deviation ([Morris, 1991]).

$$\mu_j = \frac{1}{r}\sum_{i=1}^{r} \mathrm{EE}_j^{(i)} \tag{3.7}$$

$$\sigma_j = \sqrt{\frac{1}{r-1}\sum_{i=1}^{r}(\mathrm{EE}_j^{(i)} - \mu_j)^2} \tag{3.8}$$

These sensitivity measures allow for the categorization of the input parameters based on their effects on the model output. μ_j gives an idea of the general effect of changing the parameter. σ_j indicates whether the elementary effects change much over the grid. When μ_j is close to zero it indicates that there is no general effect on the model output. This can be due to the parameter having a negligible effect. That would be indicated by a low value for σ_j as well. If a low value of $|\mu_j|$ is accompanied with a high value of σ_j then the effects are important but non-monotonic. The different elementary effects cancel each other out in computing the average. It can be due to the effect being non-linear or because of interactions with other parameters. When μ_j is big (either positive or negative) there is a clear general effect. Together with a low value of σ_j this indicates a (close to) linear effect. A high value for both μ_j and σ_j indicates non-linear effects or interactions with other parameters. This interpretation is summarized in table 3.5. Making a scatter plot with for each parameter the μ-values on the x-axis and the values of σ on the y-axis gives a good overview. In that case points close to the origin indicate negligible effects.

Other points close to the x-axis indicate linear effects. Points further from the x-axis indicate non-linear effects or interactions with other parameters. One can say that the further from the origin the more important the parameter. An example of such a graph will be given in section 3.4. Other sensitivity measures are proposed to rank parameters on their importance. The

$\sigma_j \backslash \lvert\mu_j\rvert$	low	high
low	negligible effects	(close to) linear effects
high	non-monotonic effects that are non-linear and or interaction with other parameters	non-linear effects and or interaction with other parameters

Figure 3.5: Summary of the indications of the different effects parameters can have

first, μ^*, is the mean of the absolute elementary effects proposed by [Campolongo et al., 2007]. Although the information about the sign of the general effect is removed it can also indicate effects with opposite signs as important.

$$\mu_j^* = \sum_{i=1}^{d} |\mathrm{EE}_j^{(i)}| \tag{3.9}$$

Also D is proposed as a sensitivity measure to rank the importance of the parameters by [Portilla et al., 2009]. This is the euclidean distance from the origin in the $\mu - \sigma$-graph.

$$D_j = \sqrt{\mu_j^2 + \sigma_j^2} \tag{3.10}$$

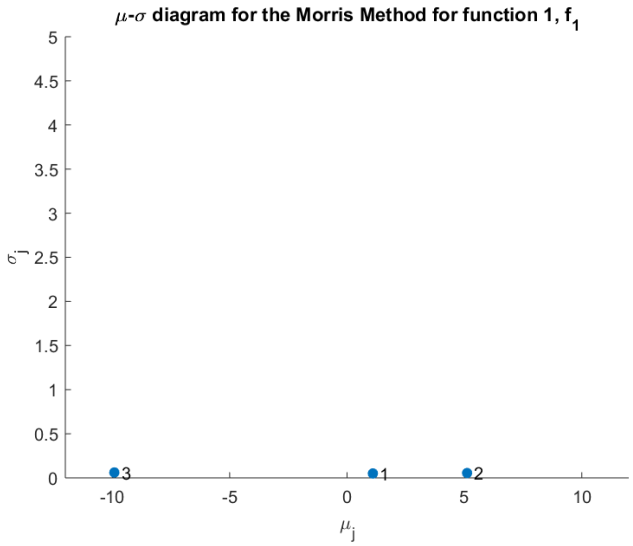
Both μ_j^* and D_j can be used to rank the parameters based on their influence on the model outcome. For μ_j^* it is shown in [Campolongo et al., 2007] that the sensitivity measure achieves similar results as the variance-based method, which is a lot more computationally intensive. By ranking the parameters one can differentiate between the important and the unimportant parameters. Now that we know how to draw conclusions from the distribution of elementary effects the whole Morris method is known. In the next section examples will be given on how to categorize the input variables in the three categories using a μ-σ-diagram.

3.4 Examples

The Morris method will be applied to the same examples for which the sensitivity was already analyzed using correlations. The input parameters are assumed to be independent, because the Morris method cannot handle dependencies.

3.4.1 Linear model with interactions

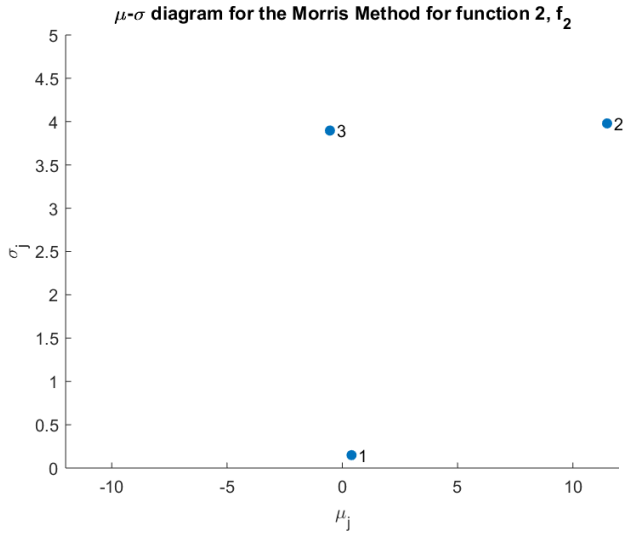
First we apply the Morris method to the linear model with interactions, which was described in 2.5.1. For applying the Morris method to these function we will use the following parameters: $r = 20$, $p = 4$, $k = 1$ and thus $\delta = \frac{1}{3}$. The calculation of the ranks will be based on the value of μ^*.



parameter	X_1	X_2	X_3
μ	1.1000	5.1133	-9.9133
σ	0.0497	0.0549	0.0613
μ^*	1.1000	5.1133	9.9133
D	1.1011	5.1136	9.9135
ranks	3	2	1

Figure 3.6: The results of the sensitivity analysis of f_1. Left one sees the $\mu - \sigma$-diagram and on the right one sees the table with the values for all the sensitivity measures.

From figure 3.6 a lot can be concluded. For the first function the standard deviation is very low for each of the variables. It means that all the effects are close to linear, which is indeed the case. This can also be seen in the $\mu - \sigma$-diagram. the points all lie close to the horizontal axis, which indicates that the effects are almost linear. The point corresponding to variable one is very close to the origin. Therefore it might be considered that variable 1 has negligible effects. Furthermore both μ^* and D rank the parameters based on their influence as expected from the coefficients.

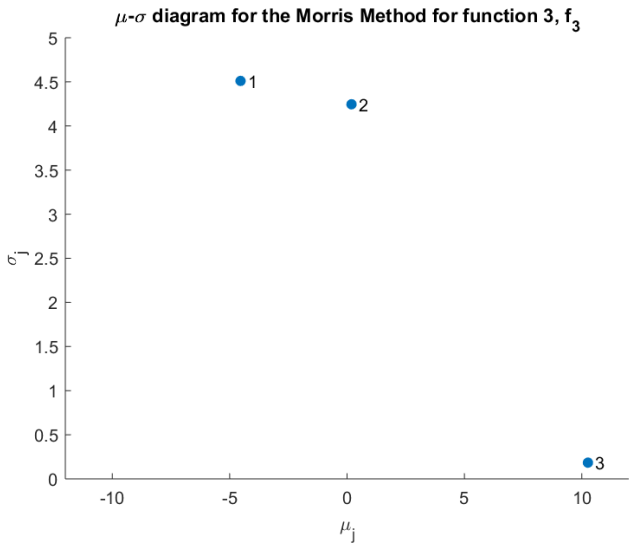


parameter	X_1	X_2	X_3
μ	0.4000	11.4833	-0.5367
σ	0.1491	3.9788	3.8963
μ^*	0.4000	11.4833	3.3433
D	0.4269	12.1531	3.9331
ranks	3	1	2

Figure 3.7: The results of the sensitivity analysis of f_2. Left one sees the $\mu - \sigma$-diagram and on the right one sees the table with the values for all the sensitivity measures.

Parameters X_2 and X_3 of the second function have a high standard deviation. This is because these two parameters have interactions with each other. μ_3 is close to zero, although X_3 has a

big influence. The effects are just not monotone. This is also why both μ^* and D rank it as quite an important variable. From the graph, but also from the values of μ^* and D it can be concluded that the effects of x_1 are negligible.



parameter	X_1	X_2	X_3
μ	-4.5367	0.1900	10.2600
σ	4.5107	4.2458	0.1838
μ^*	5.2367	3.6567	10.2600
D	6.3975	4.2500	10.2616
ranks	2	3	1

Figure 3.8: The results of the sensitivity analysis of f_3. Left one sees the $\mu - \sigma$-diagram and on the right one sees the table with the values for all the sensitivity measures.

For the last function it can easily be seen from the graph that the effect of the third parameter is indeed linear. The overall effect of x_1 is quite negative, but the high standard deviation indicates that this is mainly an interaction effect. For x_2 it can be concluded that in general it has no effect, but the standard deviation shows that this is because the effects are with interaction or non-linear, which is indeed the case. For the previous two functions μ_j^* and D_j were very similar here small differences are visible. We can see D gives a bit more weight to parameters with interaction effects. Still the differences are minimal and for the ranks of inputs using the Morris method it will in general not matter.

In these examples we have seen that the results were equally similar in the sensitivity analysis with correlations as expected from the coefficients. However, from the Morris method a lot more can be concluded. The interaction effects are more clearly visible. Next to that, parameters with opposite effects are also indicated as influential. In the sensitivity analysis with sample correlations in the previous chapter, these seemed not so influential. The Morris method randomly samples the paths to calculate the elementary effects. Different runs of the method will therefore lead to different results. To get a feeling for the uncertainty in the method, the analysis is repeated 100 times for function f_3. The results can be seen in figure 3.9. One can see that there is some uncertainty especially in the value for the standard deviation. However, the differences are small and would not lead to a different characterization of the effect.

For these simple examples the effects were clear by looking at coefficients of the function. In general one applies this method when these effects cannot be seen so easily. For the Morris method the results were again as expected and similar to the sensitivity analysis with correlations. In the next we will see whether it is also the case for the population model.

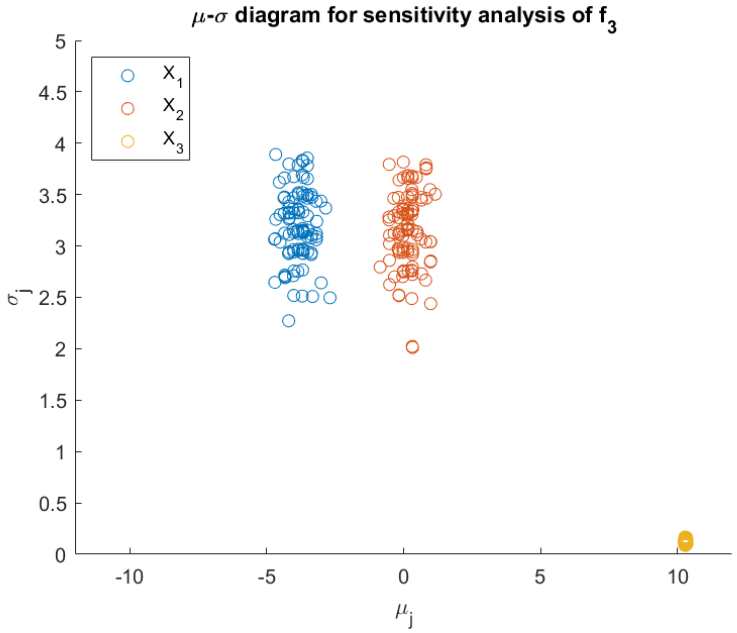


Figure 3.9: The results 100 repetitions of the Morris method on function f_3.

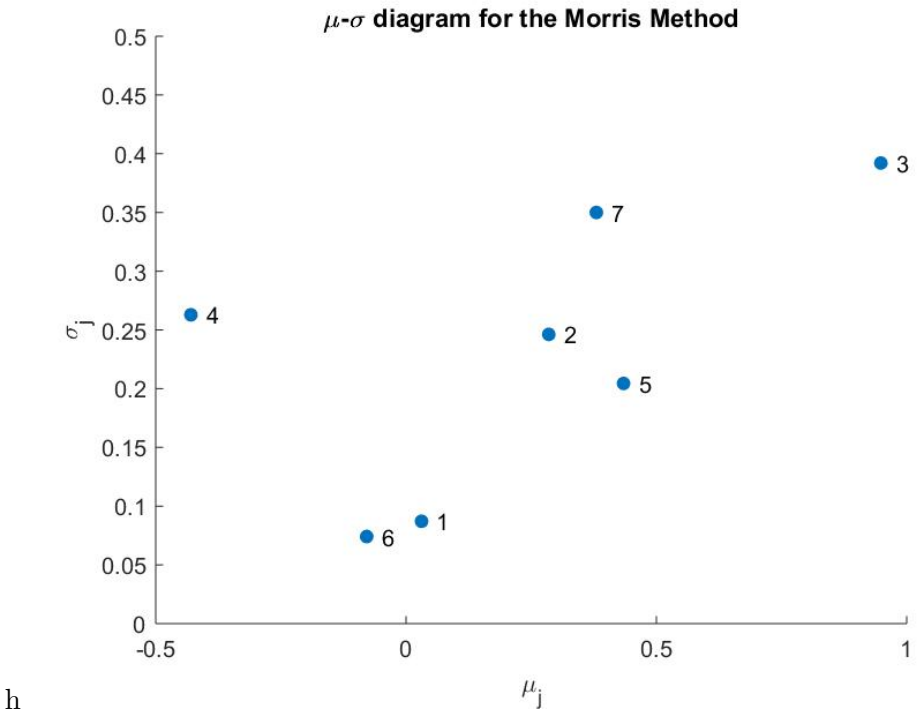
3.4.2 Morris method on the population model

In this section we will apply the Morris method to the population model described in section 2.5.3. We cannot take the dependencies of the model into account as the Morris method assumes that the input is independent. The parameters in the Morris method are chosen to be $r = 20$ $p = 4$ and $\delta = \frac{1}{3}$. Choosing these parameters is elaborately discussed in the next section.

	μ	σ	μ^*	D	ranks
r_1	0.03095	0.08711	0.04571	0.09245	7
r_2	0.2851	0.2462	0.2851	0.3767	5
$\alpha_{1,2}$	0.9484	0.392	0.9484	1.026	1
K_1	−0.4297	0.2629	0.4297	0.5038	3
K_2	0.4344	0.2044	0.4344	0.4801	2
$x_1(0)$	−0.07865	0.07412	0.07865	0.1081	6
$x_2(0)$	0.3801	0.35	0.3801	0.5167	4

Figure 3.11: Values for the different sensitivity measures resulting from applying the Morris method to the population model. The ranking of the model parameters based on their influence is based on μ^*.

In figures 3.10 and 3.11 the results of the sensitivity analysis are summarized. Both in the graph and the table one can see that $\alpha_{1,2}$ is the most important parameter. Both r_1 and $x_1(0)$ are not so influential. The ranking of the parameters is very much the same as in the sensitivity analysis where the correlations were calculated. Only the ranks of K_1 and K_2 are reversed, but one can see that the values that lead to that ranking are very close. Furthermore if we would have chosen D as the sensitivity measure to base the rankings on the two would not be reversed. However, then the ranks of K_2 and $x_2(0)$ would be reversed. In general we can say that the ranks, both based on μ^* and D are very similar. For all parameters the standard deviation is of comparable



h

Figure 3.10: The $\mu - \sigma$-diagram resulting from applying the Morris method to the population model.

size as the average. The effects are thus not close to linear. This is different from the other analysis were all effects seemed to be close to linear. Furthermore no major differences in the results can be spotted between doing sensitivity using Monte Carlo or with the Morris method. However, for this Morris method $20 \cdot (7 + 1) = 160$ model evaluations were used, whereas with the Monte Carlo simulation we used 10000 model evaluations.

3.5 Parameters of the Morris method

In this section the reader will be given an idea about how p, δ and r in the Morris should be chosen based on information of the model. In the original paper by Morris it was proposed to use p even and $\delta = \frac{p}{2(p-1)}$. Then the computation would be less complicated. First p and δ which are closely related will be discussed and next r will be discussed.

3.5.1 Choosing p and δ

The choices for p and δ go hand in hand, because $\delta = \frac{k}{p-1}, \quad$ for $k \in \{1, \ldots, p-1\}$. These parameters have two main influences on the performance of the method. The first main influence is the accuracy of the approximation of the derivatives. In the Morris method we are approximating derivatives with elementary effects. The smaller the step of the elementary effect the better the approximation. A simple function where we can see the effect: $f(x_1, x_2) = \sin(10x_1)\cos(10x_2)$ on $[0, 1]^2$. This function has derivatives in the interval $[-10, 10]$, which change a lot over small intervals. The function is plotted in figure 3.12.

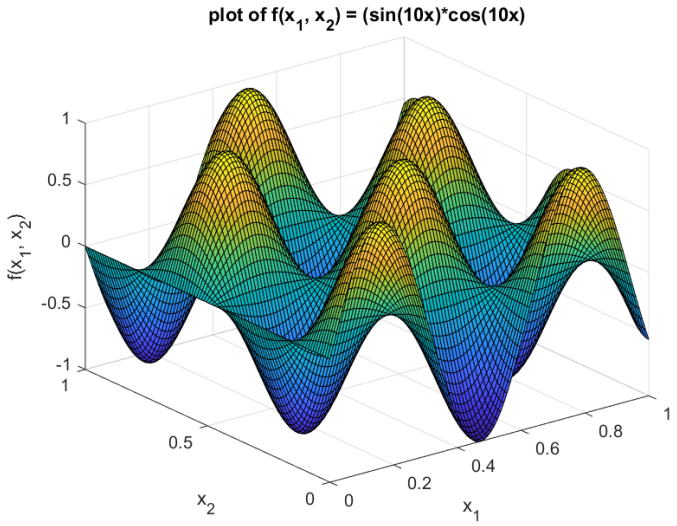


Figure 3.12: The function $f(x_1, x_2) = \sin(10x_1)\cos(10x_2)$ on $[0, 1]^2$

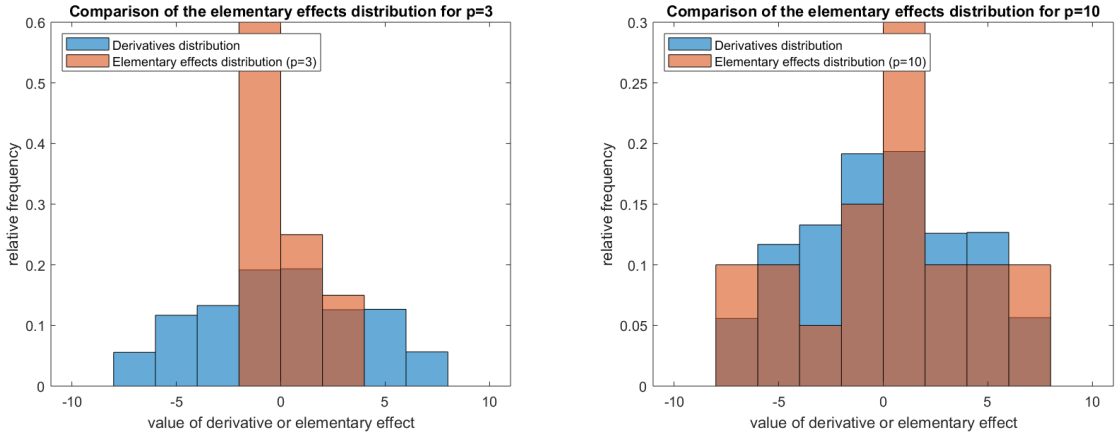


Figure 3.13: The empirical distribution of the elementary effects for p=3 (left) and p=10 (right). The distributions are compared with a Monte Carlo estimate of the distribution of derivatives. The derivatives and elementary effects are with respect to x_1 in $f(x_1, x_2) = \sin(10x_1)\cos(10x_2)$.

In figure 3.13 the distribution of the elementary effects according to the Morris method is graphed for two values of p, 3 and 10. For comparison one can see the exact distribution in red. Calculating the exact distribution is in general not possible, but in this example it can be computed easily. One can see that in the left histogram the higher values of the derivatives are completely missed, because they are averaged out with smaller effects. The influence of the parameter seems a lot less than it in fact is. In the right histogram we see that the accuracy is a lot better when δ is decreased. There is, however, also a downside to decreasing δ. Decreasing δ means that p increases, which results in a worse coverage of the parameter space if the number of samples r is kept the same. Increasing p means that the amount of cells increase exponentially, because this amount is equal to $(p-1)^d$. The fraction of the parameter space that is investigated is at most $\frac{r}{(p-1)^d}$. So in high dimensions only a very small part of the parameter space is investigated. The Latin Hypercube Sampling, as described in section 3.2.3, takes care of this problem partially, by making sure the chosen cells are spread out over the parameter space.

However, still only a small fraction of the cells is investigated.

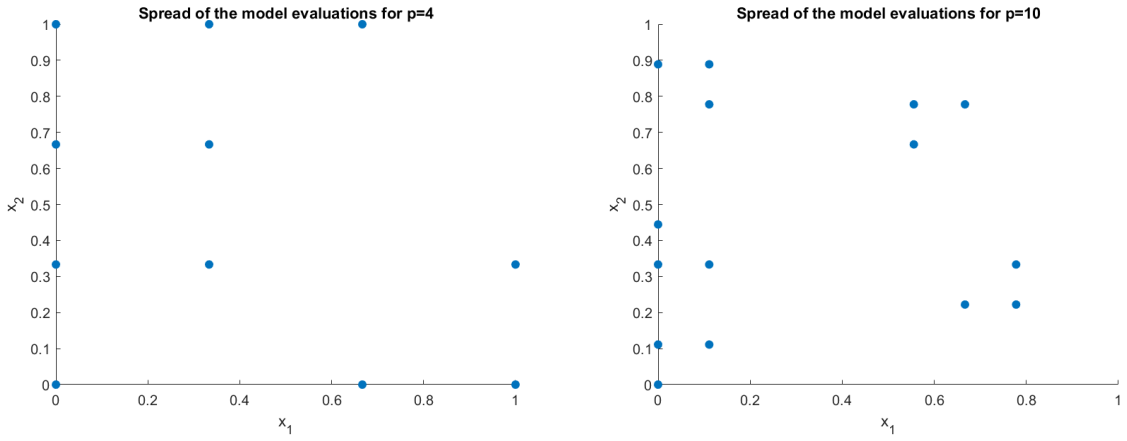


Figure 3.14: The dots represent model evaluation points in the Morris method for $p = 4$ (left) and $p = 10$ (right). Also $r = 5$ was used.

In figure 3.14 this is visualized for two dimensions where 5 cells were chosen. One can see that for a small value of p the spread of the model evaluation points over the parameter space is a lot better. In the right subfigure for $p = 10$ there are a lot of white spaces, where there are no model evaluation points. This picture shows the effect in only two dimensions, when the dimensions increase the amount of cells increases exponentially and this becomes more of a problem. There is thus a trade-off between the coverage of the Morris method and the accuracy of the derivatives estimations. Choosing k large is not a good idea, because then one has bad coverage and bad derivative approximations. Generally we would set $k = 1$, $p = 4$ and thus $\delta = \frac{1}{3}$. For specific models and also for depending on the value of r other parameter values might be better. Choosing a value for r will be discussed in the next section.

3.5.2 Choosing r

In the previous section the trade-off when choosing a value for p was described. For r the choice depends on the available computational power. As was shown in section 3.2.1 the amount of model evaluations is $r(d + 1)$. In general sensitivity analysis is applied when the model is very complex. Therefore most of the time a lot of model evaluations is not possible, because one model evaluation already takes a long time. The value of r is therefore mostly dependent on the available computational power and the computational intensity of the model. It seems that r should be at least 10 for $p = 4$ and a high dimension d, such that different parts of the parameter space are covered. If $r < 10$ already is too costly then the number of parameters that is investigated should be decreased to still reach $r = 10$. Results that are obtained with a very low r probably have no meaning. An r higher then 10 is better because there is a better coverage of the parameter space. If r can be chosen a lot larger then 10 than p can be increased, because a good coverage is maintained. The value of r therefore also influences the trade-off described in the previous section. For the rest of this report generally the values $r = 20$, $p = 4$ and thus $\delta = \frac{1}{3}$ are used. These values are similar to often used values.

Now one has a full picture of the workings and abilities of the Morris method. As said in the introduction this method assumes independence between the input parameters. The cells and corners are all sampled with equal probability. These probabilities should not be all equal

when there are dependencies between the input parameters. In the next chapter the Morris method will be extended such that dependencies in the input can be taken into account.

Chapter 4

Copula-based Morris method

In the previous chapter the Morris Method was presented. A big emphasize was put on the geometric interpretation of the Morris method. This will be used in this chapter. Here the copula-based Morris method as introduced in [Ţene et al., 2018] will be presented. We start with an introduction into copulas. Next, the actual extension of the Morris method that allows dependencies between parameters is presented. In the end we will analyze the effect of taking the dependencies into account for the two examples for which the sensitivity was already analyzed.

4.1 Describing dependence: copulas

Copulas are functions that describe dependence. They will be used in the extension of the Morris method, which takes dependencies into account. The introduction to copulas is based on [Nelsen, 2006], where a more in-depth presentation of copulas can be found.

A transformation that can be applied to continuous random variables is the probability integral transform. This is a transformation between the unit uniform distribution and another continuous distribution using the cumulative density function (CDF), notated as F_X.

$$\tilde{X} = F_X(X) \sim U([0,1]) \tag{4.1}$$

A copula is a multidimensional distribution function with uniform marginals. The Sklar Theorem ([Sklar, 1959]) then states that by using the probability integral transform every multidimensional distribution can be rewritten as a copula and its marginals. For a random vector $\mathbf{X}$ with distribution function $F_{\mathbf{X}}$ and corresponding copula C one can write

$$F_{\mathbf{X}}(\mathbf{x}) = C(F_{X_1}(x_1), \ldots, F_{X_d}(x_d)) \tag{4.2}$$

What we can see from the equation is that we can change the marginals without changing the copula or the dependence. We can model the dependencies between different variables and their marginals. This property makes them very useful, especially for high dimensional statistical problems. First one can estimate the marginals and next one can separately estimate the dependence structure. Fitting a multidimensional distribution function with different marginals would be very cumbersome.

4.1.1 Types of Copula

There are many types of copula all describing different dependence structures. The three used in this thesis will be introduced here.

Independence copula The simplest copula is the independence copula. The distribution of a vector of independent random variables is the product of the marginals. The copula corresponding to independence is therefore a function that takes the product of all the inputs.

$$C(u_1, \ldots,$$

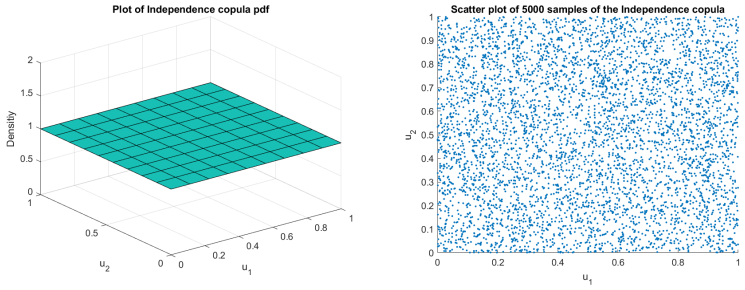


Figure 4.1: The visualization of the two-dimensional independence copula with the PDF (left) and a sample (right).

Gaussian copula An often used copula is the Gaussian copula. As the name suggests it is closely related to the normal distribution. The parameter of the Gaussian copula is a correlation matrix R. The Gaussian copula is given by

$$C_R^{\text{Gaussian}}(\mathbf{u}) = \Phi_R(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d)) \tag{4.4}$$

Where Φ is the cumulative distribution function of the standard normal,

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-t^2/2}\, dt.$$

$$\Phi_R(x_1, \ldots, x_d) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_d} \frac{1}{\sqrt{(2\pi)^k \det R}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T R^{-1}(\mathbf{x}-\mu)\right) d\mathbf{x}$$

is the multivariate normal distribution with covariance matrix R. The Gaussian copula cannot be written as an explicit function, because Φ is not a explicit function. The Gaussian copula is used a lot, because of its relation with the normal distribution. In this report the dependencies between the input parameters are also modelled with a Gaussian copula.

Archimedean copula The Archimedean copulas are a whole class of copulas. They all have a different generator function, ψ with usually one parameter (θ).The generator function should be d-monotone ([McNeil et al., 2009]). The Archimedean copulas can generally be written as follows:

$$C(\mathbf{u}; \theta) = \psi^{[-1]}(\psi(u_1; \theta) + \cdots + \psi(u_d; \theta); \theta) \tag{4.5}$$

An example of an Archimedean copula is the Gumbel copula. This copula has generator function $\psi(t) = (-\log(t))^\theta$. The parameter θ should be in $[1, \infty)$. For two dimension the copula has the following explicit formula:

$$C(\mathbf{u}; \theta) = \exp\left[-((-\log(u_1)\theta + (-\log(v))^\theta)^{1/\theta}\right]$$

For many generator functions the Archimedean copulas have an explicit expression, which makes computations easier.

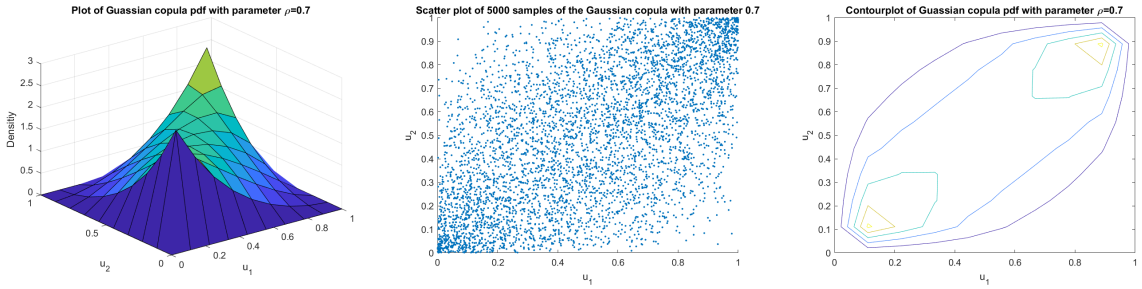


Figure 4.2: The visualization of the Gaussian copula with the PDF (left), a sample (middle) and the contourplot of the PDF (right).

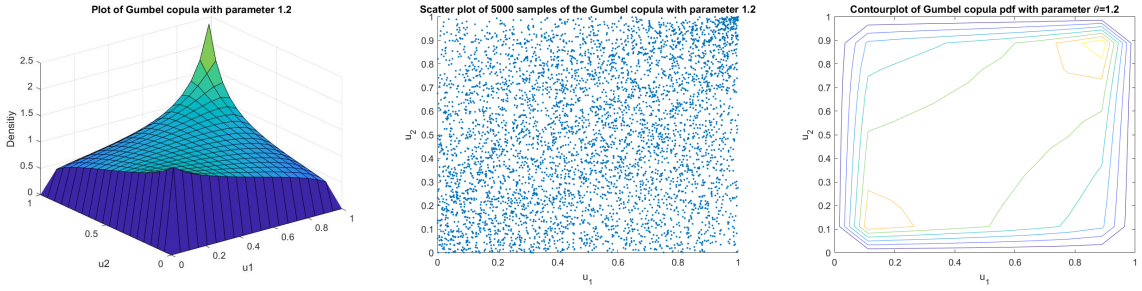


Figure 4.3: The visualization of the Gumbel copula with the PDF (left), CDF(middle) and contourplot of the PDF (right).

4.1.2 Finite difference formula

The problem of calculating probabilities using a copula is now addressed. Only the simple cases of calculating the probabilities over hyperrectangles is discussed. In one dimension calculating probabilities over intervals is trivial. Suppose one has a random variable Y with cumulative distribution function (CDF) $F_Y = \mathbb{P}(Y \leq y)$ and wants to know the probability of Y being in some interval $[a, b]$. The following applies

$$\mathbb{P}(Y \in [a, b]) = \mathbb{P}(Y \leq b) - \mathbb{P}(Y \leq a) = F_Y(b) - F_Y(a). \tag{4.6}$$

The idea behind equation 4.6 can be extended to more dimensions. Probabilities of random variables over hyperrectangles can be calculated using the CDF and the finite difference formula as given by [Nelsen, 2006]. The probability of a random vector $\mathbf{X}$ with as distribution a copula C over a hyperrectangle between the points $\mathbf{a} = (a_1, \ldots, a_d)$ and $\mathbf{b} = (b_1, \ldots, b_d)$ is given by

$$\mathbb{P}(a_1 \leq x_1 \leq b_1, \ldots, a_d \leq x_d \leq b_d) = \Delta_{a_1}^{b_1} \ldots \Delta_{a_d}^{b_d} C \tag{4.7}$$

where the Δ is the difference operator defined by $\Delta_{a_j}^{b_j} C = C(u_1, \ldots, b_j, \ldots, u_d) - C(u_1, \ldots, a_j, \ldots, u_d)$.

As an example the probability of a random vector $\mathbf{X}$ with as distribution the Gaussian copula with correlation $\rho = 0.5$ is calculated, $X \sim C = C_{0.5}^{\text{Gaussian}}$. The probability is calculated over the rectangle with corners $(0.8, 0.6)$ and $(0.4, 0.1)$.

$$\begin{aligned} \mathbb{P}(0.4 \leq X_1 \leq 0.8, 0.1 \leq X_2 \leq 0.6) &= C(0.8, 0.6) - C(0.4, 0.6) - C(0.8, 0.1) + C(0.4, 0.1) \\ &= 0.5380 - 0.3162 - 0.0974 + 0.0758 \\ &= 0.2002 \end{aligned} \tag{4.8}$$

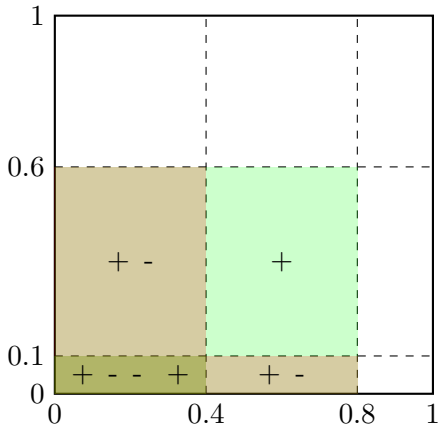


Figure 4.4: The example above illustrated. Only the area over which it was the goal to calculate the probability has a positive count.

This example is illustrated in figure 4.4. It shows how by calculating probabilities over different areas and adding and subtracting them, the probability of the right area is obtained. The pluses and minuses indicate how often each area is added or subtracted. The basic properties of copulas are now given. In the rest of this chapter these will be used to extend the Morris method to take dependencies into account.

4.2 Copula-based sampling of paths

The Morris method will be extended now by taking dependencies into account. The method described in [Ţene et al., 2018] will be explained and elaborated. It is assumed that next to the usual setting a copula is defined on the input parameters. All copulas are allowed, but if the independence copula is chosen, the method comes down to the original Morris method. The only thing that will be altered to the Morris method is the sampling of the paths over which the elementary effects are calculated.

4.2.1 Sampling a cell

We follow the process of sampling paths in the geometric interpretation of the Morris method. We therefore sample a cell first. In section 3.2.3 it was mentioned that in the original Morris method this can be done using Latin Hypercube Sampling (LHS). In [Packham and Schmidt, 2008] it was presented how LHS can be altered to take dependencies into account. Instead of random permutations as used in equations 3.6, permutations are constructed using the copula. Again each of the d dimensions is divided into n subintervals. In the extended Morris method n should be chosen to be equal to $p - k$. Now a sample, $\mathbf{X_1}, \ldots, \mathbf{X_n}$ of the copula is drawn. In each dimension the sample is ordered. The index in the ordered sample is called the rank. The rank R_i^j is the rank of sample $i, 1 \leq i \leq n$ in dimension $j, 1 \leq j \leq d$. These ranks form a permutation, but this permutation is based on the copula. The ranks can then be used as the permutation in equation 3.6. In figure 4.5 it is visualized how a copula sample is transformed to a Latin Hypercube Sample of cells. But the Latin Hypercube Sample still shows the dependencies of the copula. The method is called Latin Hypercube Sample with Dependencies (LHSD). Equally as in the original method we don't mind about the samples in the cell as long as we know the cells. Note that using LHSD with the independence copula is equivalent to LHS. By using LHSD the

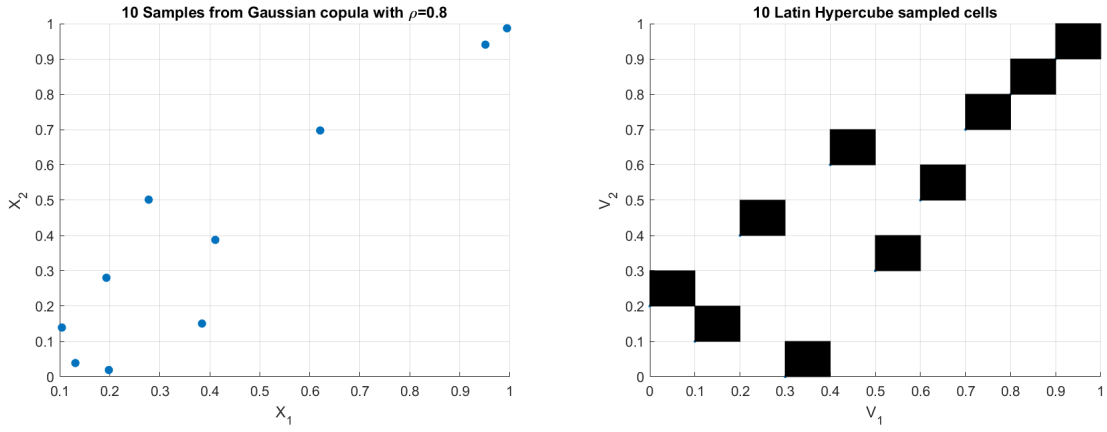


Figure 4.5: The copula sample on the left is transformed to a LHSD sample using equation 4.9. For these pictures the Gaussian copula with parameter $\rho = 0.8$ was used.

cells are spread over the entire hypercube, but with taking the dependencies into account. The corners of cell i are in the end given by

$$\text{Cell}_i = \prod_{j=1}^{d} \left\{ \frac{R_i^j - 1}{p-1}, \frac{R_i^j - 1}{p-1} + \delta \right\} \tag{4.9}$$

In this equation the ranks are based on a copula sample. Sampling cells with LHSD leads to $p - k$ cells per sample and can should be repeated until r cells are obtained.

4.2.2 Sampling a starting point

Once a cell is sampled a corner of this cell should be sampled as starting point of the path. The copula is a distribution function on the unit hypercube, but we only need a distribution of the grid points. We will transform the copula to a distribution on the grid points. The bases for this should be that points with a lot of density from the copula close to it, should be more probable in the grid point distribution. If $\mathbf{x}$ is a point in the grid we choose the following volume to belong to that point, $V_{\mathbf{x}} = \left(\prod_{j=1}^{d}[x_j - \frac{1}{2}\delta, x_j + \frac{1}{2}\delta]\right) \cap [0,1]^d$. The last intersection is to make sure that everything is well defined also on the boundary. Now each point has a volume assigned to it we can say that the probability of that corner is equal to the probability of the volume according to the copula.

$$\mathbb{P}(\mathbf{x}) = \int_{V_{\mathbf{x}}} C(\mathbf{u}) d\mathbf{u} \tag{4.10}$$

This is now a discrete distribution were each corner has a probability. To sample a corner after choosing a cell, the probability distribution should be conditioned on the cell that is chosen. Let the cell be given by $\text{Cell} = \prod_{j=1}^{d}\{a_j, b_j\}$ with the corresponding volume $V_{\text{Cell}} = \prod_{j=1}^{d}[a_j, b_j]$. We then define the conditional probability of the corner given the cell by

$$\mathbb{P}(\mathbf{x}|\text{Cell}) := \mathbb{P}(V_{\mathbf{x}}|V_{\text{Cell}}) = \frac{\mathbb{P}(V_{\mathbf{x}} \cap V_{\text{Cell}})}{\mathbb{P}(V_{\text{Cell}})} \tag{4.11}$$

The areas in this equation are illustrated in figure 4.6. The calculation comes down to the following. After choosing the cell, each dimension is divided in two. This gives 2^d hyperrectangles each with exactly one corner of the original cell. The probability of that corner is then the

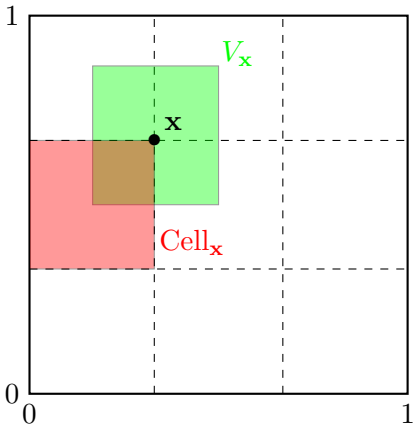


Figure 4.6: Illustration of the areas and points mentioned in this section

probability of the corresponding hyperrectangle according to the copula. The calculation of these probabilities can be done using the finite differences formula as presented in 4.1.2. Now that a distribution of the corners of the chosen cell is obtained it is easy to sample one of the corners as the starting point of the path. This way a corner is sampled by using the copula.

4.2.3 Permutation

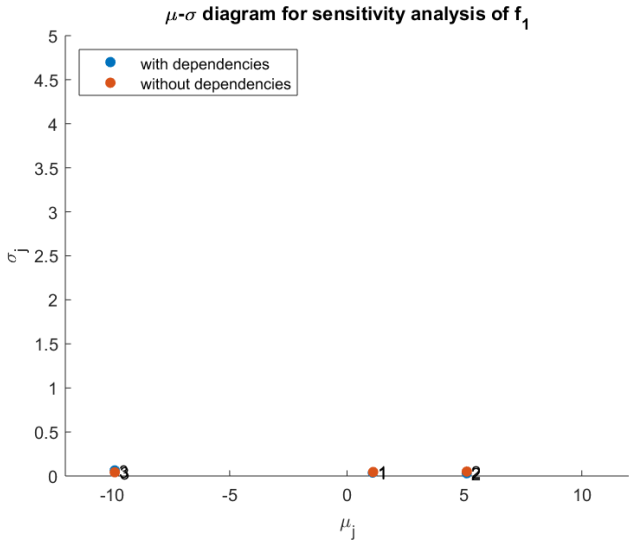
Once a starting point is obtained a permutation of the dimensions is needed to get a full path. In this section we diverge from the method described in [Ţene et al., 2018]. In that paper it was proposed to randomly permute the dimensions as was done in the original Morris method. Here it is chosen to use the distribution of the corners obtained in the previous section for the permutation as well. The second point in the path is a corner of the cell adjacent to the starting point. This point is obtained by sampling from these adjacent corners. The probability used is the distribution of the corners as obtained in the previous section, conditioned on the corners being adjacent to the starting corner. We can do the same for the other points of the path. Each time the probability distribution on the corners, conditioned on being adjacent to the previous point, is used. Next to that the dimension in which the corners differ should not have been used in the path before. This way a path over the cell is obtained. These are the same kind of paths as obtained in the Morris method. But very importantly the path is now sampled with taking the copula as much as possible into account. The paths are not sampled with the same probability for all the paths as in the Morris method, but the copula favours certain paths above others. Still all paths can occur only some will be unlikely. The paths should be used in the same way to obtain the elementary effects as in the Morris method. Also the analysis of the distribution of the elementary effects is equal. But now the dependencies between the input parameters have been taken into account.

4.3 Examples

We will now apply the extended Morris method to the two examples to which we applied the Morris method. For these examples we can see what the results are of taking the dependencies into account. The standard parameters are used: $r = 20$, $p = 4$, $k = 1$ and thus $\delta = \frac{1}{3}$. The ranking of the input variables will be based on μ^*.

4.3.1 Copula-based Morris method on linear model

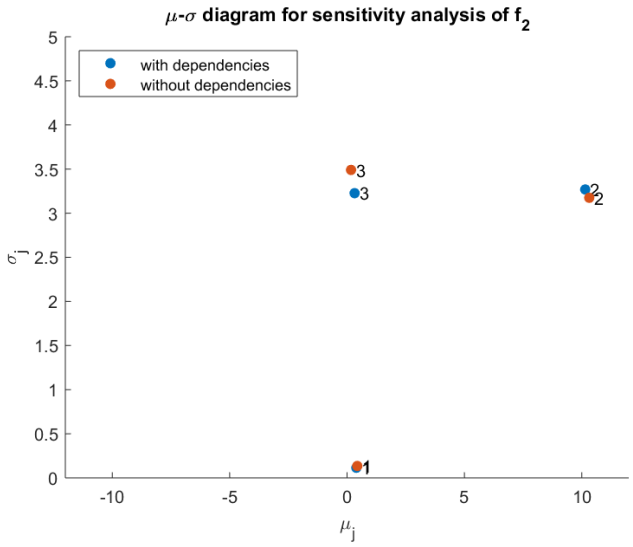
We apply the copula-based Morris method to the linear model with interactions, which can be found in 2.5.1. The results of the original Morris method applied to this model can partially be seen in the graphs, but are most extensively analyzed in 3.4.1.



parameter	X_1	X_2	X_3
μ	1.098	5.098	-9.897
σ	0.0382	0.03148	0.06297
μ^*	1.098	5.098	9.897
D	1.099	5.098	9.897
ranks	3	2	1

Figure 4.7: The results of the sensitivity analysis of f_1 with the correlated parameters. The results without the correlated parameters are also plotted as a reference. Left one sees the $\mu-\sigma$-diagram and on the right one sees the table with the values for all the sensitivity measures.

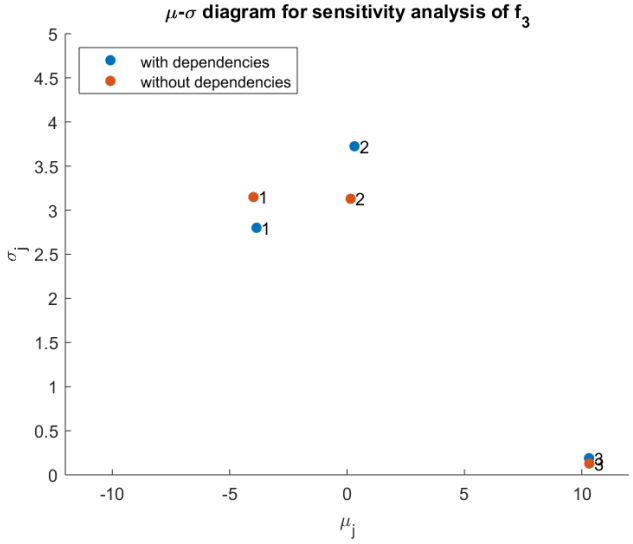
In figure 4.7 we see the results of applying the copula-based Morris method on f_1 as given in 2.5.1. We see in the $\mu - \sigma$-diagram that the points coincide. The correlations have almost no effect on the results of the sensitivity analyses. This is not surprising as we choose the function such that it has almost no interactions. When there are no interaction between parameters the values of other parameters do not matter for the effects of a parameter. Sampling paths based on correlations between parameters then has no effect.



parameter	X_1	X_2	X_3
μ	0.395	10.14	0.3217
σ	0.1146	3.269	3.227
μ^*	0.395	10.14	2.742
D	0.4113	10.66	3.243
ranks	3	1	2

Figure 4.8: The results of the sensitivity analysis of f_2 with the correlated parameters. The results without the correlated parameters are also plotted as a reference. Left one sees the $\mu-\sigma$-diagram and on the right one sees the table with the values for all the sensitivity measures.

In figure 4.8 one can see the results of applying the copula-based Morris method to function f_2. There are some differences between the results with and without dependencies now. This is because the interaction effects are a bit bigger. The effect of parameter 1 was negligible and still is. The effect is negligible over the entire parameter space. It therefore doesn't matter which paths we sample with a higher probability. The parameters 2 and 3 have a big interaction effect, however, these were chosen uncorrelated. Therefore the results are still similar as when there were no interactions.



parameter	X_1	X_2	X_3
μ	-3.855	0.3167	10.31
σ	2.801	3.724	0.1889
μ^*	4.075	3.207	10.31
D	4.765	3.738	10.31
ranks	2	3	1

Figure 4.9: The results of the sensitivity analysis of f_3 with the correlated parameters. The results without the correlated parameters are also plotted as a reference. Left one sees the $\mu-\sigma$-diagram and on the right one sees the table with the values for all the sensitivity measures.

Figure 4.9 shows the results of the sensitivity analysis of function f_3. Here changes between

the sensitivity analysis with and without dependencies can be seen. Parameters 1 and 2 have an interaction effect and they are correlated. This results in a different value for the sensitivity measures. Also both μ^* and D indicate that X_1 and X_2 are less important relative to X_3 now. This is because the interaction effect is more important now and that cancels out the linear effects of the parameters. For parameter three the effects are again minimal.

In general we see that taking dependencies into account only has an effect if the parameters have an interaction effect. This is in contrast to the sensitivity analysis with the correlations of the model output with different inputs. There, being correlated with an influential parameter made the parameter itself also influential. The Morris method is made global by summarizing a set of One-At-the-Time effects. Although taking the dependencies into account leads to the sampling of different elementary effects, still the effect of a single parameter is measured. The influence of the taking the dependencies into account is therefore not so big. If one compares the effects with the uncertainty in the method is visualized in figure 3.9 it can be questioned whether the effects are significant.

4.3.2 Copula-based Morris method on population model

Now we have seen that how the Morris method can be applied in the case of dependencies we can apply it to the example of the population model as described in 2.5.3. The results of applying the Morris method with and without dependencies will be compared. The results will also be compared to the sensitivity analysis with correlations on the populations, which can be found in section 2.5.4. We will use the standard parameters again: $r = 20$, $p = 4$ and $\delta = \frac{1}{3}$.

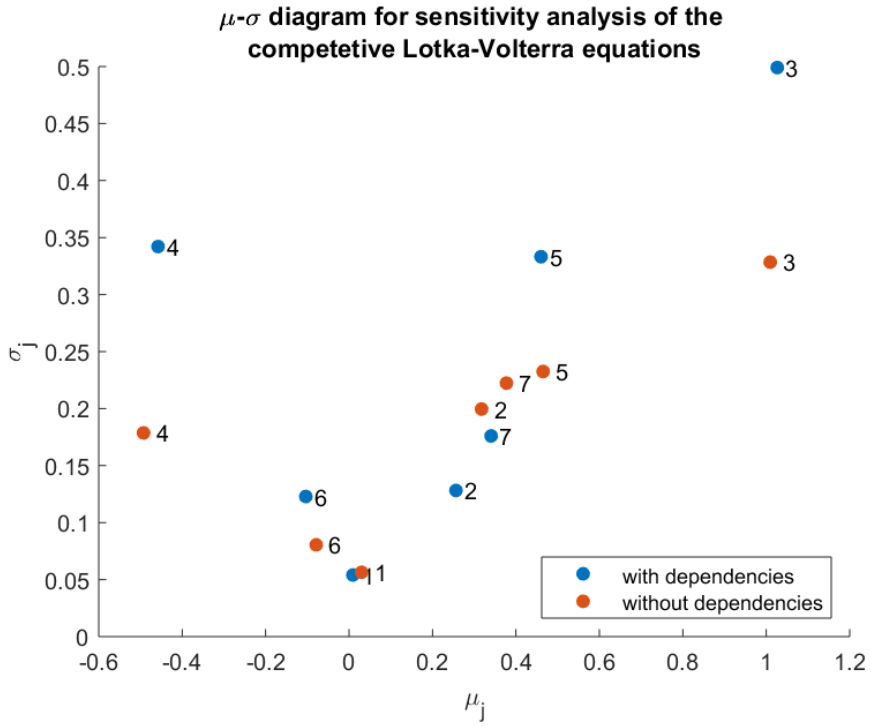


Figure 4.10: The $\mu - \sigma$-diagram for the sensitivity of the competitive Lotka-Volterra equations as described in section 2.5.3. The Morris method is applied both with and without taking into account the dependencies.The corresponding parameters are $r = 20$, $p = 4$ and $\delta = \frac{1}{3}$ for both cases.

	μ	σ	μ^*	D	ranks
r_1	0.01562	0.07398	0.04405	0.07562	7
r_2	0.2263	0.08347	0.2263	0.2412	5
$\alpha_{1,2}$	1.114	0.5208	1.114	1.23	1
K_1	−0.486	0.215	0.486	0.5315	3
K_2	0.4982	0.2781	0.4982	0.5706	2
$x_1(0)$	−0.09104	0.09816	0.09104	0.1339	6
$x_2(0)$	0.2925	0.1233	0.2925	0.31747	4

Figure 4.11: Values for the different sensitivity measures resulting from applying the Copula-based Morris method to the population model with dependent input parameters. The ranking of the model parameters based on their influence is based on μ^*.

In figure 4.10 and one can see the results of applying the Morris method both with and without taking dependencies into account. The difference between taking into account the dependencies is visible in the graph, but it is not that big. The differences do not lead to a different categorization of parameters. The distribution of elementary effects is different when taking into account the dependencies. The ranks are very similar. Only K_1, K_2 and $x_2(0)$ are switched around a bit. It is still very clear that $\alpha_{1,2}$ is the most important parameter in this model.

The results of the sensitivity analysis can also be compared to the sensitivity analysis with Monte Carlo simulation as in section 2.5.4. The results for the cases without dependencies turned out to be very similar. In the previous section we already saw that the influence of taking dependencies into account is very different for the Morris method and the Monte Carlo sample correlation. For the population model this is also the case. The influence of parameters does not change that much in the population model, whereas parameters correlated with $\alpha_{1,2}$ became a lot more important in the other sensitivity analysis. If we look at the ranking for the different parameters the only thing that is the same is that $\alpha_{1,2}$ is still the most important parameter. The other ranks are different.

In this chapter the finite differences formula given by equation 4.7 is used. It turns out that in some cases the finite difference formula is too computationally intensive for practical use in the Morris method. In the next chapter this problem will be analyzed.

Chapter 5

Improvements in calculations of corner probabilities

In the previous chapter it is shown how the Morris Method can be adapted to take dependencies between parameters into account. There the finite differences formula was used to calculate the probabilities of the corners. The problem is that the method is too slow for copulas, without an explicit expression for the copula CDF. For instance, applying the method to a problem with 8 parameters takes already about an hour to find the points in which the model should be evaluated. The Morris method is often used as a screening method. Therefore most of the time tens or hundreds of parameters are used. The problem is that the computational intensity increases exponentially with the dimensions. First of all, if the dimension increases one, the number of corner probabilities to calculate doubles. Next to that, the amount of copula CDF evaluations per corner doubles. For non-explicit copulas one such evaluation is already lengthy and for high dimensions this becomes problematic. The calculation of corner probabilities is the part of the method that is too computationally intensive, we therefore focus on improving the implementation of this.

5.1 Approximating corner probabilities

5.1.1 One-cell approximation

In [Ţene et al., 2018] it was proposed to compute the corner distribution only once. The idea is to calculate the corner probabilities as if the cell chosen would be the entire cube. The corner probabilities would be calculated for the entire unit hypercube. Then for each sampled cell the corner probabilities are assumed to be the same. This way the corner probabilities only have to be calculated once, but it uses the assumption that the copula has a similar behaviour among all cells. This can be the case for copulas with a lot of symmetry. At the end of the section, the method will be compared with some numerical integration techniques to see whether it can be used in general.

5.1.2 Numerical integration

As one can see in figure 4.4 when using the finite differences formula the probabilities over certain areas are calculated often and then added and subtracted. This is because a copula distribution function gives the probability of the hyperrectangle between a point and the origin. However we only need the probability over a small hyperrectangle. In this section we will calculate the probabilities only over the hyperrectangle by numerically integrating the copula

probability density function. For copulas without an explicit copula CDF numerically integrating the PDF will often be faster than using the finite differences formula. However, there will be an error, because we numerically integrate. In this section we will present four ways of numerical integration. At the end of the section, the numerical integration methods together with the one-cell approximation will be compared based on error and speed.

Midpoint rule

The first type of numerical integration that is discussed will be *midpoint integration*. For example in [Vuik et al., 2007] a 1-dimensional midpoint rule is mentioned. The idea is that when one has to integrate a function over an interval the function value in the middle of the interval is a good representation for the function value over the interval. One can then multiply the middle function value with the size of the interval. Midpoint integration can be written in formula's as follows:

$$\int_a^b f(x)dx \approx f\left(\frac{a+b}{2}\right)(b-a) \tag{5.1}$$

How equation 5.1 approximates integrals can be seen in figure 5.1, where it visualized for an exampled. When the interval is wide or the function differs a lot over the interval the approximation is not that good. It can, however, be improved by dividing the interval into n smaller subintervals and applying the midpoint rule to each subinterval. This idea is illustrated in the right subfigure of figure 5.1. In figure 5.1 it is illustrated how $\int_1^3 x^2 dx$ can be approximated

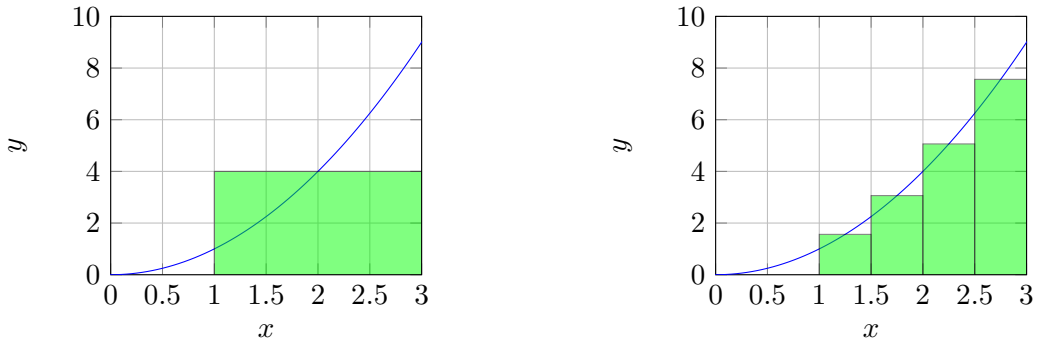


Figure 5.1: The midpoint rule with 1 (left) and 4 (right) function evaluations illustrated. The integral $\int_1^3 x^2 dx$ is approximated.

with the midpoint rule. We will calculated the green areas to see what the approximation error is. First we notice that the actual value is $\int_1^3 x^2 dx = 8\frac{2}{3}$. In applying the midpoint rule we use $a = 1$, $b = 3$ and $f(x) = x^2$:

$$\int_1^3 x^2 dx \approx \left(\frac{1+3}{2}\right)^2 (3-1) = 8 \tag{5.2}$$

The approximation error is thus equal to $\frac{2}{3}$. We are also going to apply to midpoint rule where we use four subintervals.

$$\begin{aligned}\int_1^3 x^2 dx &= \int_1^{1.5} x^2 dx + \int_{1.5}^2 x^2 dx + \int_2^{2.5} x^2 dx + \int_{2.5}^3 x^2 dx \\ &\approx \left(\frac{1+1.5}{2}\right)^2 (1.5-1) + \left(\frac{1.5+2}{2}\right)^2 (2-1.5) \\ &+ \left(\frac{2+2.5}{2}\right)^2 (2.5-2) + \left(\frac{2.5+3}{2}\right)^2 (3-2.5) \\ &= (1.5625 + 3.0625 + 5.0625 + 7.5625) \cdot 0.5 = \frac{69}{8} \end{aligned} \tag{5.3}$$

The approximation error is now equal to $\frac{1}{24}$. By increasing the amount of subintervals the approximation error was decreased and this works in general. The midpoint rule can be extended to higher dimensions. As V is a hyperrectangle with equal length in each dimension the middle is easily found. Then again the function is evaluated in the middle point. As in the 1-dimensional case smaller subvolumes can be created to decrease the error. Now in each dimension the interval should be split in an equal amount of subintervals. Again one evaluates the function in each of the midpoints and multiplies with the corresponding volume. One would now obtain a approximation for the integral by summing over the results of all volumes.

Trapezoidal rule

A more sophisticated algorithm is the *trapezoidal rule* ([Vuik et al., 2007]). This algorithm uses the average of the two outer function values as an overall average. The method converges faster than the midpoint rule. The calculation is as follows:

$$\int_a^b f(x)dx \approx \frac{f(a)+f(b)}{2}(b-a) \tag{5.4}$$

This method is illustrated in figure 5.2 for the same example as with the midpoint rule. As with

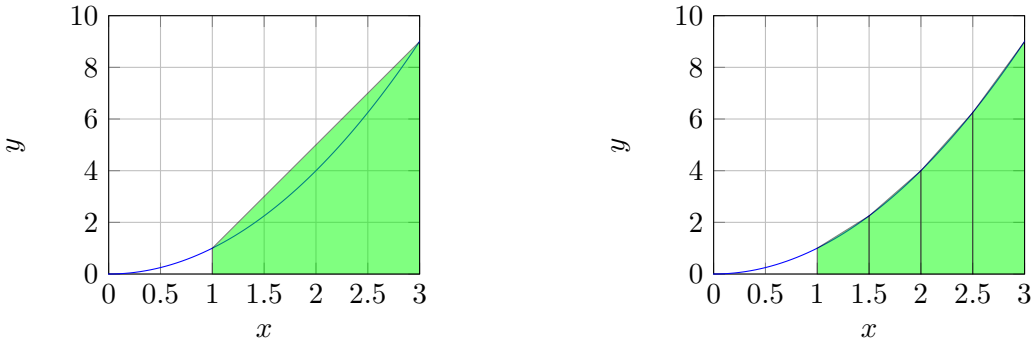


Figure 5.2: The trapezoidal rule for integration illustrated with 2 (right) and 5 (right) model evaluations. The integral $\int_1^3 x^2 dx$ is approximated.

the midpoint rule we can calculate the approximations to see what the error is. Again $a = 1$, $b = 3$ and $f(x) = x^2$ are used and the actual value is still $8\frac{2}{3}$.

$$\int_1^3 x^2 dx \approx \frac{1^2+3^2}{2}(3-1) = 10 \tag{5.5}$$

The approximation error is $1\frac{1}{3}$. Again we also apply the method with 4 subintervals.

$$\begin{aligned}\int_1^3 x^2 dx &= \int_1^{1.5} x^2 dx + \int_{1.5}^2 x^2 dx + \int_2^{2.5} x^2 dx + \int_{2.5}^3 x^2 dx \\ &\approx \frac{1^2 + 1.5^2}{2}(1.5 - 1) + \frac{1.5^2 + 2^2}{2}(2 - 1.5) \\ &+ \frac{2^2 + 2.5^2}{2}(2.5 - 2) + \frac{2.5^2 + 3^2}{2}(3 - 2.5) \\ &= 0.8125 + 1.5625 + 2.5625 + 3.8125 = \frac{35}{4} \end{aligned} \tag{5.6}$$

Again the approximation error is decreased. Now it is equal to $\frac{1}{12}$. In this example the midpoint rule was more accurate, but in general the trapezoidal rule converges faster than the midpoint rule. So when more and more subintervals are used the error will decrease faster. There is one problem. A lot of copulas do not behave well at the boundary of the unit hypercube. In figure 4.2, where the Gaussian copula is displayed it can be seen that at the exact boundary the value of the PDF is zero everywhere, although at some boundaries the value of the copula close to the boundary tends to infinity. With the trapezoidal method, the PDF is also evaluated at the boundary of the unit hypercube, but its value there is not representative for the value in the region. Therefore the approximation will be off.

Monte Carlo Integration

Monte Carlo integration uses random sampling for calculating an integral. This idea was introduced in [Metropolis and Ulam, 1949]. More information about this method can be read in [Robert and Casella, 2013]. The idea is to construct a random variable, Y, with as expectation the desired quantity. The law of large numbers states that in the cases of i.i.d. random variables that the sample mean converges to the average. If one wants to integrate a function over an hyperrectangle $V = \prod_{i=1}^{d}[a_i, b_i]$ one can apply this in the following way. First N uniform samples from the hyperrectangle are drawn. Next, the function is evaluated in these points and multiplied with the volume of V. Then the sample mean is calculated. According to the weak law of large numbers this than converges to the expected value, which is the exact integral. This can be written down in equations as follows where $\mathbf{X_i} \sim U(V)$ i.i.d., $Y_i = \|V\| f(\mathbf{X_i})$ and $\|V\|$ denotes the volume of area we are integrating over:

$$\frac{\|V\|}{N}\sum_{i=1}^{N} f(\mathbf{X_i}) = \frac{1}{N}\sum_{i=1}^{N} Y_i \to \mathbb{E}(Y) = \int_V f(\mathbf{x}) d\mathbf{x} \tag{5.7}$$

An estimation for the standard error of the estimate is $\|V\|\sqrt{\frac{\mathrm{Var}(Y)}{N}}$. The error decreases thus with square root of the number of samples. In general Monte Carlo integration works well for integrals in high dimensions, which we are dealing with in the copula-based Morris method.

Antithetic variates

Sometimes the standard error in Monte Carlo can be decreased by using antithetic variates ([Hammersley and Morton, 1956]). The idea is that if one uses a randomly generated point that one also uses the opposite point (whatever that may be according to the setting). Then the

average of the estimate based on those two points is used instead of the estimate based on the single point. For the example above with calculating the integral over a hyperrectangle this can be done as follows. First one draws a uniformly distributed vector $\mathbf{U}$. One then transforms this to the hyperrectangle, $\mathbf{X_i} = [a_1 + u_1(b_1 - a_1), \ldots, a_d + u_d(b_d - a_d)]$, over which we are integrating. Next to this point one also uses $\tilde{\mathbf{X}}_\mathbf{i} = [a_1 + (1 - u_1)(b_1 - a_1), \ldots, a_d + (1 - u_d)(b_d - a_d)]^T$. This is the opposite point in our setting. All the coordinates are flipped with respect to the center of the cell. The variance of the combined estimate is now as follows:

$$\mathrm{Var}(\bar{Y}) = \mathrm{Var}\left(\frac{f(\mathbf{X_i}) + f(\tilde{\mathbf{X}}_i)}{2}\right) = \frac{\mathrm{Var}(f(\mathbf{X_i})) + \mathrm{Var}(f(\tilde{\mathbf{X}}_i)) + 2\mathrm{Cov}(f(\mathbf{X_i}), f(\tilde{\mathbf{X}}_i))}{4} \tag{5.8}$$

So the variance is reduced if the covariance is negative between the function value and the function value in the 'opposite' point. A reduced standard error means that the estimate is more accurate. This method is thus preferred over standard Monte Carlo when the covariance between the two function values is negative.

Comparison of numerical integration techniques

Four ideas for numerical integration were presented. The numerical integration methods are similar in the sense that they all evaluate the copula PDF in several points and use a weighted average of the PDF values in these points as an estimate for the value over the entire volume. The methods differ in which points the PDF should be evaluated. In the midpoint rule the points are in the middle of smaller hyperrectangles. In the trapezoidal method the points are also fixed, but points at the border are also used. In Monte Carlo integration the points in which the PDF is evaluated are random. By using antithetic variables the points are still random, but a bit better spread is achieved. The points in which the PDF is evaluated are visualized in figure 5.3. Note that for the last two figures the algorithms are (partially) random and the distributions of the points is therefore just an example. However, also these figures should give an idea of what the distribution of model evaluation points should look like.

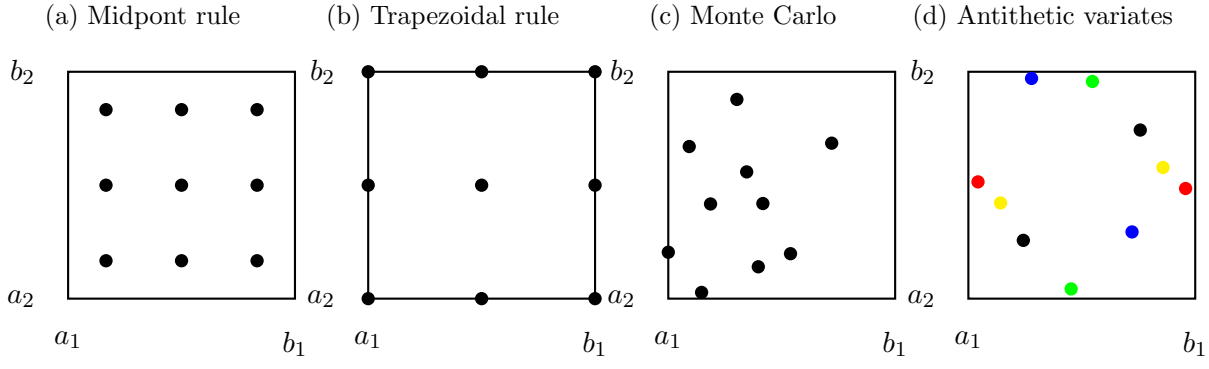


Figure 5.3: The function evaluation points for different numerical integration techniques. In the figure for the antithetic variates the colours corresponds to sets of points and their 'opposite' point.

5.1.3 Conclusion on approximating probabilities

To compare the methods two quantities are used. The first quantity is the speed of the algorithm. The algorithms are used to speed up the copula-based Morris method, therefore speed is important. The second quantity is the error the approximations causes. As we are calculating

probabilities instead of general real numbers we will use a non-standard error measure. We will measure the error by the average change in log-odds. Function 5.9, the change in log-odds, has desired properties in comparing probabilities, for instance $E(x, y) = E(1 - x, 1 - y)$.

$$E : [0, 1]^2 \to [0, \infty) \text{ by } E(x, y) = \left| \log\left(\frac{x}{1-x}\right) - \log\left(\frac{y}{1-y}\right) \right| \tag{5.9}$$

To compare the different methods of approximating the probabilities it is chosen to apply the methods to calculate all corner probabilities in grid with $p = 5$ and $d = 5$. The speed is measured and the average change in log-odds is calculated. In the first table the underlying copula is the Gaussian copula with correlation parameter $\rho = 0.5$ for all the parameters. In the second table the underlying copula is the Gumbel copula with parameter $\theta = 2$. $\theta = 2$ comes down to a Kendall's τ equal to 0.5. The linear correlation cannot be calculated from θ. The important difference is that the Gumbel copula has an explicit expression, whereas the Gaussian does not. For the numerical methods in brackets is the amount of PDF evaluations per probability calculation.

method	speed (in seconds)	average error (difference in log odds)
Finite differences	668.533	0
one-cell approximation	0.608	0.6515
Midpoint (N=1)	4.395	0.0313
Midpoint (N=3125)	66.069	0.0150
Trapezoidal (N=32)	7.7376	0.2202
Trapezoidal (N=1024)	60,916	0.0775
Monte Carlo (N=10)	5.122	0.1129
Monte Carlo (N=1000)	32.332	0.0111
Antithetic variates (N=10)	7.834	0.1130
Antithetic variates (N=1000)	37.592	0.0110

Figure 5.4: Results for simulating the calculation of all the corner probabilities in a grid with $p = 5$ and $d = 5$ for the Gaussian copula with correlation parameter $\rho = 0.5$ for all parameters. The parameters in brackets indicate the amount of function evaluations used per probability calculation.

method	speed (in seconds)	error
Finite differences	9.918	0
one-cell approximation	0.051	1.0747
Midpoint (N=1)	144.194	0.0728
Trapezoidal (N=1)	1419.007	0.1582
Monte Carlo (N=4)	368.240	0.3361
Antithetic variates (N=4)	385.616	0.3359

Figure 5.5: Results for simulating the calculation of all the corner probabilities in a grid with $p = 5$ and $d = 5$ for the Gumbel copula with parameter $\theta = 2$.

First of all one can see that all the approximation algorithms provide a speed improvement for the case with the Gaussian copula. This can be explained by the fact that PDF of the Gaussian copula can be calculated a lot easier than the CDF as there is no explicit expression for the CDF

in contrary to the Gumbel copula. Exactly calculating the probabilities using the finite differences formula is therefore already quite fast. Using a lot of PDF evaluations to estimate these probabilities is taking more time and there is also an approximation error. One can conclude that the numerical integration techniques only make sense when the PDF is easy to calculate and the CDF not, like with the Gaussian copula. In cases with an explicit formula for the CDF as with the Gumbel copula, numerical integration should not be used. For these cases one can either use the once-cell approximation or the finite differences formula. The one-cell approximation is very fast, but also very inaccurate compared to the numerical integration methods. As a reference the error of the exact solution with giving all corners the same probability is 0.4655. It is therefore best to use the finite differences formula. It was suggested in section 5.1.1 that the error of the one-cell approximation would decrease when there are more symmetries. The results in the tables are in line with that. The error when the one-cell approximation is applied to the more symmetrical Gaussian copula is lower.

In comparing the numerical integration methods one can see that using antithetic variates in Monte Carlo does not add accuracy. This should therefore not be used. As expected the trapezoidal method does not perform well, because at the boundaries of the unit hypercube the value of the PDF at the boundary is often a lot different from the value of PDF close to the boundary. This is the case for many copulas. Trapezoidal integration should therefore not be used. The midpoint method has the best accuracy when not many PDF evaluations are used. When more computational power is available the Monte Carlo integration using is very accurate, because it converges faster.

We will now see what the effect of these speed improvements is in a more practical setting. The Morris method is generally applied to models with tens or hundreds of parameters. The points in which the model should be evaluated should be calculated in a reasonable time. The reasonable order of magnitude for finding the model evaluation points seems to be a couple of hours. The computational time for the model evaluations is included in this. We will therefore now test which problem size we can handle in about one hour. We will use the standard parameters ($r = 20$, $p = 4$, $\delta = \frac{1}{3}$) and the Gaussian copula. For the straight-forward implementation with finite differences this is about 8 dimensions. Calculating all the model evaluation points in 8 dimensions took 3258.715 seconds or 0.905 hour. By implementing the method using the midpoint rule for calculating probabilities this can be more than doubled. It is possible to handle 20 dimensions, because that took 2586.431 seconds or 0.718 hour. With 20 dimensions one comes close to values where one has memory issues. In *Matlab* it is for instance not possible to save the distribution of the corners in dimensions higher than 29, because for 30 this is an array of size 2^{30} or 8.0 GB. 30 is a lot less then the standard screening methods which can handle tens or hundreds of input parameters.

In general the midpoint integration rule seems the best way to approximate the corner probabilities in order to decrease the computer intensity of the method. Therefore we will also look at another way to speed up the implementation. Not by approximating the probabilities, but rather by making sure we have to do fewer computations.

5.2 Independent Groups

One can easily imagine that when there is dependence among parameters that this is only among certain groups. Especially models with a lot of parameters usually have a lot of parameters that are unrelated to each other and therefore also independent of each other. The example of the population model has, for instance, two independent groups of correlated variables. In this section it is showed how in these cases the copula-based Morris method can be implemented more efficiently.

Say the model parameters can be divided into two arbitrary independent groups. So $\mathbf{X^1} \in \mathbb{R}^{d_1}$ and $\mathbf{X^2} \in \mathbb{R}^{d_2}$ and the modeling being $f : [0,1]^{d_1+d_2} \to \mathbb{R}$. By saying the groups are independent it is meant that all pairs of $\mathbf{X^1}$ and $\mathbf{X^2}$ are independent: $\forall 1 \leq i \leq d_1$, $\forall 1 \leq j \leq d_2$, X_i^1 is independent of X_j^2. Because of this independence we can write $C(u_1, \dots u_{d_1+d_2}) = C_1(u_1, \dots u_{d_1})C_2(u_{d_1+1}, \dots, u_{d_1+d_2})$. The probability of independent event is equal to the product of the marginal probabilities. This idea can of course be extended by using induction to combine more independent groups. If $(C_i)_{i\in\{1,\dots,n\}}$ a sequence of copulas, then $C = \prod_{i=1}^{n} C_i$ is again a copula.

We will apply the extended Morris method to each group separately. The groups are independent, so the value of a parameter in one group does not influence the sampling of a path in another group. The parameters from different groups can, however, have interaction effects in the model. The different paths should therefore be combined to get a path in the entire parameter space. How this should be done is not so obvious. A single point in the parameter space can be obtained by appending a point from all of the groups together. For a path over the grid each point should only differ from the previous in one dimension. Next to that, we want the combining of the paths to be randomized. Parameters from different independent groups can still have interaction effects. To not bias certain effects the combining should be randomized. If we only look at one group we want the path to be equal to the original path. We assume to have a vector $\mathbf{v}$ that indicates to which group a variable belongs. $v_i = j$ means that variable i belongs to group j. We will then first show how the different sampling matrices (section 3.2.2) can be randomly appended in an example. After that we will summarize this in an algorithm.

5.2.1 Example

Here it will be showed how the different sampling matrices can be appended in the smallest case possible. This example will consist of four parameters (x_1, x_2, x_3, x_4). The first two parameters will be in the first group and the third and fourth parameter will be in the second group. This gives the following vector $\mathbf{v} = [1, 1, 2, 2]$. Say we obtained the following two sampling matrices in applying the copula-based Morris method to the groups separately.

$$B_1 = \begin{bmatrix} B_1^{(1)} \\ B_1^{(2)} \\ B_1^{(3)} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{3} \\ 0 & \frac{2}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix}, \quad B_2 = \begin{bmatrix} B_2^{(1)} \\ B_2^{(2)} \\ B_2^{(3)} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & 1 \\ 0 & 1 \\ 0 & \frac{2}{3} \end{bmatrix} \tag{5.10}$$

It will be illustrated how these two sampling matrices can be randomly appended to form an overall sampling matrix in the entire parameters space of (x_1, x_2, x_3, x_4).

1. We randomly permute $\mathbf{v}$ to obtain $\mathbf{v}^* = [2, 1, 1, 2]$.

2.
$$B_1 = \begin{bmatrix} 0 & \frac{1}{3} \\ 0 & \frac{2}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix}, \quad B_2 = \begin{bmatrix} \frac{1}{3} & 1 \\ 0 & 1 \\ 0 & \frac{2}{3} \end{bmatrix} \longrightarrow \quad \tilde{B} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & 1 \\ & & & \end{bmatrix}$$

3.
$$B_1 = \begin{bmatrix} 0 & \frac{1}{3} \\ 0 & \frac{2}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix}, \quad B_2 = \begin{bmatrix} \frac{1}{3} & 1 \\ 0 & 1 \\ 0 & \frac{2}{3} \end{bmatrix} \quad v_1^* = 2 \longrightarrow \quad \tilde{B} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & 1 \\ 0 & \frac{1}{3} & 0 & 1 \\ & & & \end{bmatrix}$$

4.
$$B_1 = \begin{bmatrix} 0 & \frac{1}{3} \\ 0 & \frac{2}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix}, \quad B_2 = \begin{bmatrix} \frac{1}{3} & 1 \\ 0 & 1 \\ 0 & \frac{2}{3} \end{bmatrix} \quad v_2^* = 1 \longrightarrow \quad \tilde{B} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & 1 \\ 0 & \frac{1}{3} & 0 & 1 \\ 0 & \frac{2}{3} & 0 & 1 \\ & & & \end{bmatrix}$$

5.
$$B_1 = \begin{bmatrix} 0 & \frac{1}{3} \\ 0 & \frac{2}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix}, \quad B_2 = \begin{bmatrix} \frac{1}{3} & 1 \\ 0 & 1 \\ 0 & \frac{2}{3} \end{bmatrix} \quad v_3^* = 1 \longrightarrow \quad \tilde{B} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & 1 \\ 0 & \frac{1}{3} & 0 & 1 \\ 0 & \frac{2}{3} & 0 & 1 \\ \frac{1}{3} & \frac{2}{3} & 0 & 1 \end{bmatrix}$$

6.
$$B_1 = \begin{bmatrix} 0 & \frac{1}{3} \\ 0 & \frac{2}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix}, \quad B_2 = \begin{bmatrix} \frac{1}{3} & 1 \\ 0 & 1 \\ 0 & \frac{2}{3} \end{bmatrix} \quad v_4^* = 2 \longrightarrow \quad \tilde{B} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & 1 \\ 0 & \frac{1}{3} & 0 & 1 \\ 0 & \frac{2}{3} & 0 & 1 \\ \frac{1}{3} & \frac{2}{3} & 0 & 1 \\ \frac{1}{3} & \frac{2}{3} & 0 & \frac{2}{3} \end{bmatrix}$$

In the above example first the vector $\mathbf{v}$ is randomly permuted to obtain $\mathbf{v}^*$. In the other steps two rows of the small sampling matrices are appended to form one row of the overall sampling

matrix. Which rows of the sampling matrices are used for this is determined by $\mathbf{v}^*$. The overall sampling matrix obtained is

$$\tilde{B} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & 1 \\ 0 & \frac{1}{3} & 0 & 1 \\ 0 & \frac{2}{3} & 0 & 1 \\ \frac{1}{3} & \frac{2}{3} & 0 & 1 \\ \frac{1}{3} & \frac{2}{3} & 0 & \frac{2}{3} \end{bmatrix} \tag{5.11}$$

$\tilde{B}$ matrix has all the desired properties of a sampling matrix. Two consecutive rows only differ in one column and that difference is always equal to $\frac{1}{3}$. Next to that, in each column there is exactly one time such a difference. This means that when the rows of this matrix are interpreted points of a path the matrix represents a path on the grid.

5.2.2 Algorithm

The example above can be generalized to an algorithm. We will have a sampling matrix for each group. Let $\mathbf{v}$ be the vector that indicates to which group a variable belongs. $v_i = j$ means that variable i belongs to group j. We randomly permute the vector and notate the new vector with $\mathbf{v}^*$. Entry i of this vector will indicate from which group the change between the points i and $i+1$ from the path comes. Because $\mathbf{v}^*$ is a permutation of $\mathbf{v}$ the amount of times there is a change from a group is equal to the number of variables in the group. By saying that a change comes from a group we mean that the next point of a path is obtained from the previous point by only changing the value in a dimension corresponding to that specific group. The other values are kept the same. The change in the group comes from the sampling matrix belonging to that group, where we move to the next row, which is the next point. We can write that as an algorithm. An implementation of this algorithm can be found in appendix A

With this algorithm a sampling matrix is obtained that fulfills all demands. It will be a $d+1$ by d matrix, where two consecutive rows differ only δ in one column. And for every column the value is changed exactly once. Furthermore if one looks only at the columns belonging to one group and then deletes the duplicate rows then one retrieves the original sampling matrix belonging to that group.

Algorithm 1 Combining sampling matrices of independent groups.

1: Apply the copula-based Morris method to each group to obtain a sampling matrix B_i.
2: Permute the vector $\mathbf{v}$ to obtain $\mathbf{v}^*$.
3: The first point in the path is equal to the first row of each sampling matrix appended to each other.
4: **for** $i := 1$ **to** d **do**
5: Row $i+1$ from the sampling matrix is equal to the previous except for the values belonging to group v_i. For these values use the next row of B_{v_i}, the sampling matrix belonging to group v_i.
6: **end for**

5.2.3 Results

There are several positive effects of applying the copula-based Morris method with this implementation. First of all the copula evaluations are in lower dimensions. The copulas are evaluated in a dimension equal to the size of the group and not in the dimension of the entire parameter

space. Next to that, a lot fewer evaluations of the copula are necessary. Normally to calculate the probability density of the corners for each of the 2^d corners 2^d copula evaluations are necessary. Now d can be replaced with the group size. The method has to be several times, but per group there is an exponential decrease. This leads to a large decrease in total computational time. Next to that, there are no memory issues. A probability density of all the corners was stored. For more than 30 dimensions this was not possible anymore. With this implementation only the probability density of the corners per group needs to be stored. So there are no problems as long as the groups are not bigger than 30 parameters.

The speed advantage can be tested in a simulation. Earlier we have seen that in about one hour the model evaluation points in the problem with 8 dimensions could be calculated. By using the midpoint rule this was already increased to 20 dimensions. The simulations were also done for different group sizes and both the finite differences and midpoint implementations. In

Type	group size	time in seconds	time in minutes
Finite differences	6, 6, 7, 6, 5	5277.195	1 hour and 28.2 minutes
Finite difference	50 groups of four parameters	2594.216	43.2 minutes
Midpoint rule	19, 17, 4, 13, 17	3682.162	1 hour and 1.4 minutes
Midpoint rule	250 groups of 4	52.339	0.87 minutes

Figure 5.6: Simulation of the copula-based Morris method using the independent groups implementation. The usual parameters in this report were used: $r = 20, p = 4, \delta = \frac{1}{3}$.

figure 5.6 the results of the simulations are summarized. One can see what can be achieved in approximately one hour. The last simulation did not take approximately one hour, but this already consisted of 1000 variables. There are not many cases in which models have more than 1000 parameters. As expected from the implementation the total time is closely related to the sum of the time it would cost to apply the method to each group separately. Of course some overhead is also necessary now, but this is still very small compared to the time evaluation copula PDF's and CDF's. This shows that by using independent groups the computational intensity of the method does not grow exponentially with the dimension. If the groups are of the same size than the computational intensity is linear with the dimensions. As long as groups do not become bigger than about 15 parameters the method can be applied to models with tens or hundreds of parameters, without having ridiculously long computational times. For a screening method this is really beneficial as these are most of the time applied to models with that amount of parameters. Finding the model evaluation points was only a part of the method, but it was that part that became too computational intensive, because of the additions of copulas. Of course a lot will computational time will go into model evaluations, however the number of model evaluations grows linearly with the dimensions so that will not cause problems so fast.

Chapter 6

Conclusion

In mathematical modeling sensitivity analysis can be used for the following reasons:

- knowing which parameters to investigate;
- model simplification;
- model understanding;
- finding errors in the model.

To do sensitivity analysis many different methods for calculating or estimating different sensitivity measures can be used. The screening methods are a subset of the sensitivity analysis methods. Screening methods are used for a fast preliminary analysis of models with tens or hundreds of parameters. One of the screening methods is the Morris method. It estimates derivatives based on a fraction factorial design. The model evaluation points can be obtained by constructing sampling matrices. There is also a more geometric approach. The fractional factorial design is then represented as a grid and paths on the grid are sampled to obtain the model evaluation points. By analyzing the distribution of elementary effects the parameters can be ranked and the following type of effects can be identified:

a) negligible effects

b) (close to) linear effects

c) non-linear effects and/or interaction with other parameters

The results of the Morris method were compared to the results of the method where a Monte Carlo simulation is used to calculate the sample correlation. For the two examples the results were in line, but the Morris method was more informative. It was possible to identify certain effects, whereas these effects were missed by the method using sample correlations. In the applications of the Morris method hardly any differences between D and μ^*, which suggest there is no reason to prefer one more than the other.

In some models the input parameters are correlated. In the Morris method the model evaluation points are sampled assuming that the input parameters are independent. A copula-based Morris method was presented and elaborated in this report. The method can handle dependencies between input parameters. In each step of sampling a path, a copula is used to give more probability to some paths. The results of taking the dependencies into account were different

for the Morris method and the Monte Carlo method. In the Monte Carlo method variables correlated with influential variables were automatically also identified as influential. In the Morris method the effect of taking the dependencies into account was only notable when there was a significant interaction effect between the parameters. Even then it can be questioned whether these effects were significant in comparison with the randomness in the method. A general conclusion about taking the dependencies into account cannot be made. There are lots of different models and dependency structures, for which different things might hold. Especially as the method was not applied to very complex models in this report. However, when one can apply the method with dependencies one should always do so.

The goal of the research was to improve the performance of the copula-based Morris method, because calculating the probabilities of the corners is very computer intensive. For non-explicit copulas these computations are lengthy. The amount of copula evaluations increases exponentially with the number of input parameters. Five methods for approximating the corner probabilities were compared. The one-cell approximation was very fast, but also not so accurate. The midpoint integration rule had the best accuracy when not much computation time was available. The Monte Carlo integration method converged more quickly. For speeding up the algorithm the midpoint rule was the best option. By using the midpoint rule the amount dimension of the problem that could be handled in about one hour increased from 8 to 20. An exact performance improvement cannot be stated as this would depend on the parameters of the Morris method and the dependencies. With approximating the corner probabilities a small error was created. The effect of this error and whether this has a significant influence on the method is still open for research. Using Latin Hypercube might also be an idea to use for numerical integration. This can be investigated in future research.

Next to approximating the probabilities another way of speeding up the algorithm was proposed. In many models with correlated input, the input will only be correlated within some groups, where the groups would be independent. An implementation utilizing this was presented. It turned out to be very beneficial. First of all, it overcame the memory issue that might occur when the probability distribution of a lot of points is to be saved. Next to that, the computer intensity was decreased to the sum of the computer intensity for the different groups. So as long the groups do not become bigger the computation intensity increases linearly instead of exponentially with the dimension. It was shown that if the groups are not so big (less then 10 or 15) all the model evaluation points can be calculated in a reasonable time even when there tens or hundreds of input parameters. For a screening method, such as the Morris method this is important as these are the orders of problems to which the method is often applied.

In general when one has problems with the computational intensity of calculating the model evaluation points in the copula-based Morris method these problems can be overcome. First of all, one can implement the method by using the independent groups. If the groups are not so large this should really reduce the computational intensity. If the computational intensity is still too large one can use the midpoint rule to approximate the corner probabilities. At the cost of a little accuracy the computer intensity is further decreased. So the method should always be applicable in a reasonable time when the groups of correlated input do not consist of more than 20 parameters. The method will now also be applied to the model mentioned in [Ţene et al., 2018] without full correlations using the implementation mentioned in the report. The faster implementation of the copula-based Morris method also allows for further research whether there is a significant effect of taking the dependencies into account. The method can now be applied to more complex models to test whether the dependencies make a difference.

Appendix A

Matlab implementation of indendent groups

```
%This is the implementation of the copula-based Morris method using the
%independent groups. The user should define the groups and the copula's on
%these groups. The methods than samples paths as neccessary in the Morris
%method. For each group this is done using the copula-based Morris method.
%The copula-based Morris method is implemented using the midpoint rule
%instead of the finite differences formula.
%For the group with all the independent parameters this is done using the
%original Morris method. At the end the paths per group are combined to
%form a set of sampling matrices. These can be found in the variable paths,
%were path(i, j, k) is the k-th coordinate of the j-th point of path i.

%% initialization
%method parameters
p = 4; %the number of discretizations levels
r = 20; %the number of EE samples per parameter
k = 1;
step = k/(p-1); %the Morris step, also referred to as delta
rng(15); %set seed for reproducibility

%% define the model parameters here
totalParameter = 1; %set the total number of parameters
parameterPerGroup = zeros(totalParameters) %make a vector with for each parameter the corresponding group.
% Set this to one if the parameter is independent of all groups. The
```

```
% parameters do not have to be ordered.
lowerBound = [0];
upperBound = [1];
%initization of the groups
totalVector = 1:totalParameter;
for i = 1:nGroups
    parameterGroups(i).nParameters = sum(parameterPerGroup == i);
    parameterGroups(i).parameters = totalVector(parameterPerGroup==i)
        ;
end
%next for each group define a copula except for the first group
%parameterGroups(i).copula = Copula("Gaussian", Sigma);

%% Creating values for the independent parameters
%Here a sampling matrix for all the independent parameters is
    obtained.
nSample = ceil(r/(p-k)); %the amount of LHS(D) samples.

%use permutation to decide on the cell
randomRanks = zeros(parameterGroups(1).nParameters, p-k);
nIndependent = parameterGroups(1).nParameters;
iPath =0;
for iSample = 1:nSample
    for j = 1:nIndependent
        randomRanks(j, :) = randperm(p-k);
    end
    for i = 1:p-k
        cellA = ((ranks(:, i)-1)./(p-1)); %defines the cell. cellA is
            the point in the cell with the smallest indices for all
        corner = cellA + (rand(nIndependent, 1)>0.5)*step; %the
           starting corner of
        perm = randperm(nIndependent); %permutation of the dimensions
            .
        signs = (corner == cellA).*2-1;
        for j = i:nIndependent
            corner(perm(j)) = corner+signs(perm(j))*step;
            paths(iPath, j+1, :) = corner;
        end
        iPath = iPath+1;
        if (iPath>r)
            break;
        end
    end
%% Creating the paths for each group
%In this section the individual sampling matrices are obtained.

for g = 1:nGroups
    %for each group for each sample a sampling matrix is created and
        stored
```

```
67      %in parametersPerGroup(g).paths this is done in the usual way in
            the
68      %copula-based Morris method
69      nParameters = parameterGroups(g).nParameters; %the amount of
            parameters in this group
70      parameterGroups(g).paths = zeros(r, nParameters+1, nParameters);
            %initialize the sampling matrices
71
72      %define the grids (these are matrices with in the rows all the
            points
73      %in the grid.
74      cornerGrid = getPoints([0, step], nParameters);
75      cornerGridSmall = cornerGrid./2;
76
77
78      iPath = 1; %index keeps track of the path
79      %next the LHSD are created by sampling (p-k) values from the
            copula and
80      %calculating their ranks.
81      for iSample = 1:nSample
82          u = parameterGroups(g).copula.rnd(p-k);
83          ranks = zeros(nParameters, p-k);
84          for iParameter = 1:nParameters
85              ranks(iParameter,:) = tiedrank(u(:,iParameter));
86          end
87          for i = 1:p-k
88              %cellA is the smallest point of the cell, this uniquely
                    defines
89              %the cell
90              cellA = ((ranks(:, i)-1)./(p-1))';
91
92              %calculate corner probabilities;
93              %all cdf values are calculated once. Some can be reused.
94              cdfPoints = cellA + cdfGrid;
95              cdfValues = parameterGroups(g).copula.cdf(cdfPoints);
96
97              %assign probabilities to corners
98              cornerProbabilities = zeros(1, 2^nParameters);
99              for c = 1:2^nParameters
100                 %midpoint
101                 %a = cellA + ones(1, nParameters)*step/4+getCorner(c,
                        nParameters)*step/2;
102                 %cornerProbabilities(c) = parameterGroups(g).copula.
                        pdf(a);
103                 %finite differences
104                 a = cellA + cornerGridSmall(c,:);
105                 corners = a + cornerGridSmall;
106                 b = a + cornerGridSmall(2^nParameters, :);
107                 s = 0;
```

```
108                 for d = 1:2^nParameters
109                     sign = (-1)^(sum(b ~= corners(d,:)));
110                     o = cdfValues(getIndex(corners(d, :), cdfPoints))
                            ;
111                     s = s+sign*o;
112                 end
113                 cornerProbabilities(c) = s;
114             end
115 
116             %sample a corner as starting point of the path
117             cornerIndex = randsample(2^nParameters, 1, true,
                    cornerProbabilities);
118             corner = cellA + step*getCorner(cornerIndex, nParameters)
                    ;
119             parameterGroups(g).paths(iPath, 1, :) = corner;
120             todo = 1:nParameters; %each dimension should be used once
                    , this keeps track of that
121             for j = 1:nParameters
122                 possibleCorners = cornerIndex + parameterGroups(g).
                        signs(iPath, todo).*2.^(nParameters-todo);
123                 if isscalar(possibleCorners) %if only one dimension
                        is left, this dimension should be used.
124                     cornerIndex = possibleCorners;
125                 else
126                     cornerIndex = randsample(possibleCorners, 1, true
                            , cornerProbabilities(possibleCorners));
127                 end
128                 parameterGroups(g).paths(iPath, j+1, :) = cellA+step*
                        getCorner(cornerIndex, nParameters); %add the
                        corner to the sampling matrix
129                 q = find(parameterGroups(g).paths(iPath, j, :) ~=
                        parameterGroups(g).paths(iPath, j+1, :));
130                 todo(todo == q) = []; %remove the dimension from the
                        todo vector
131             end
132             iPath = iPath+1;
133             if (iPath > r) %because each time (p-k) cells are sampled
                     at a time, this might be too much and the algoritm
                    can be stopped
134                 break;
135             end
136         end
137     end
138 end
139 
140 %% The different sampling matrices are now appended to form a full
        sampling matrix.
141 %here the method ouptut is created. These consist mainly of the paths
142 %matrix. The permutations and sign matrices are used to correctly
```

```
143 %calculated the elementary effects afterwards.
144 paths = zeros(r, totalParameter+1,totalParameter);
145 permutations = zeros(r, totalParameter);
146 signs = zeros(r, totalParameter);
147
148 %in this for loop the sampling matrices are constructed by appending
      the
149 %matrices in the right way.
150
151 %change the permuation here.
152 for i = 1:r
153     perm = parameterPerGroup(randperm(totalParameter));
154     s=0;
155     for g = 1:nGroups
156         changeIndices = totalVector(perm==g);
157         paths(i, 1:changeIndices(1), parameterPerGroup==g) = repmat(
              parameterGroups(g).paths(i, 1, :), 1, changeIndices(1), 1)
              ;
158         for j = 1:length(changeIndices)-1
159             paths(i, (changeIndices(j)+1):changeIndices(j+1),
                  parameterPerGroup==g) = repmat(parameterGroups(g).
                  paths(i, j+1, :), 1, changeIndices(j+1)-changeIndices(
                  j), 1);
160         end
161         paths(i, (changeIndices(end)+1):totalParameter+1,
              parameterPerGroup==g) = repmat(parameterGroups(g).paths(i,
               end, :),  1, totalParameter+1-changeIndices(end), 1);
162     end
163
164 end
165
166
167 %here the sign and permutation matrices are constructed.
168 diff = paths(:, 2:totalParameter+1, :)-paths(:, 1:totalParameter, :);
169 for i=1:r
170     for j =1: totalParameter
171         permutations(i, j) = find(diff(i, :, j));
172         signs(i, j) = sign(diff(i, permutations(i, j), j));
173     end
174 end
175
176 modelEvaluationPoints = zeros(r, totalParameter+1, totalParameter);
177 for i = 1:totalParameter
178     modelEvaluationPoints(:, :, i) = lowerBound(i)+ uppperBound(i)*
        paths(:, :, i);
179 end
```

Bibliography

[Andres and Hajas, 1993] Andres, T. H. and Hajas, W. C. (1993). Using iterated fractional factorial design to screen parameters in sensitivity analysis of a probabilistic risk assessment model.

[Bettonvil and Kleijnen, 1997] Bettonvil, B. and Kleijnen, J. P. (1997). Searching for important factors in simulation models with many factors: Sequential bifurcation. *European Journal of Operational Research*, 96(1):180–194.

[Box and Hunter, 1961] Box, G. E. and Hunter, J. S. (1961). The 2 kp fractional factorial designs. *Technometrics*, 3(3):311–351.

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

[Cotter, 1979] Cotter, S. C. (1979). A screening design for factorial experiments with interactions. *Biometrika*, 66(2):317–320.

[Hammersley and Morton, 1956] Hammersley, J. and Morton, K. (1956). A new monte carlo technique: antithetic variates. In *Mathematical proceedings of the Cambridge philosophical society*, volume 52, pages 449–475. Cambridge University Press.

[Iooss and Lemaître, 2015] Iooss, B. and Lemaître, P. (2015). A review on global sensitivity analysis methods. In *Uncertainty management in simulation-optimization of complex systems*, pages 101–122. Springer.

[Jacques et al., 2006] Jacques, J., Lavergne, C., and Devictor, N. (2006). Sensitivity analysis in presence of model uncertainty and correlated inputs. *Reliability Engineering & System Safety*, 91(10-11):1126–1134.

[Li et al., 2011] Li, L., Lu, Z., and Zhou, C. (2011). Importance analysis for models with correlated input variables by the state dependent parameters method. *Computers & Mathematics with Applications*, 62(12):4547–4556.

[Mara and Tarantola, 2012] Mara, T. A. and Tarantola, S. (2012). Variance-based sensitivity indices for models with dependent inputs. *Reliability Engineering & System Safety*, 107:115–121.

[McKay et al., 1979] McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245.

[McNeil et al., 2009] McNeil, A. J., Nešlehová, J., et al. (2009). Multivariate archimedean copulas, d-monotone functions and 1-norm symmetric distributions. *The Annals of Statistics*, 37(5B):3059–3097.

[Metropolis and Ulam, 1949] Metropolis, N. and Ulam, S. (1949). The monte carlo method. *Journal of the American statistical association*, 44(247):335–341.

[Morris, 1991] Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174.

[Nelsen, 2006] Nelsen, R. B. (2006). *An introduction to copulas*. Springer, second edition edition.

[Packham and Schmidt, 2008] Packham, N. and Schmidt, W. M. (2008). Latin hypercube sampling with dependence and applications in finance. *Available at SSRN 1269633*.

[Portilla et al., 2009] Portilla, E., Tett, P., Gillibrand, P., and Inall, M. (2009). Description and sensitivity analysis for the lesv model: water quality variables and the balance of organisms in a fjordic region of restricted exchange. *Ecological Modelling*, 220(18):2187–2205.

[Robert and Casella, 2013] Robert, C. and Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

[Saltelli and Sobol, 1995] Saltelli, A. and Sobol, I. M. (1995). About the use of rank transformation in sensitivity analysis of model output. *Reliability Engineering & System Safety*, 50(3):225–239.

[Saltelli et al., 2004] Saltelli, A., Tarantola, S., Campolongo, F., and Ratto, M. (2004). Sensitivity analysis in practice: a guide to assessing scientific models. *Chichester, England*.

[Sklar, 1959] Sklar, A. (1959). Fonctions dé repartition à n dimension et leurs marges. *Université Paris*, 8(3.2):1–3.

[Sobol, 1993] Sobol, I. M. (1993). Sensitivity estimates for nonlinear mathematical models. *Mathematical modelling and computational experiments*, 1(4):407–414.

[Ţene et al., 2018] Ţene, M., Stuparu, D. E., Kurowicka, D., and El Serafy, G. Y. (2018). A copula-based sensitivity analysis method and its application to a north sea sediment transport model. *Environmental Modelling & Software*, 104:1–12.

[van Griensven et al., 2006] van Griensven, A. v., Meixner, T., Grunwald, S., Bishop, T., Diluzio, M., and Srinivasan, R. (2006). A global sensitivity analysis tool for the parameters of multi-variable catchment models. *Journal of hydrology*, 324(1-4):10–23.

[Vano et al., 2006] Vano, J., Wildenberg, J., Anderson, M., Noel, J., and Sprott, J. (2006). Chaos in low-dimensional lotka–volterra models of competition. *Nonlinearity*, 19(10):2391.

[Vuik et al., 2007] Vuik, C., Van Beek, P., Vermolen, F., and Van Kan, J. (2007). *Numerical Methods for Ordinary differential equations*. VSSD.

[Zhu and Yin, 2009] Zhu, C. and Yin, G. (2009). On competitive lotka–volterra model in random environments. *Journal of Mathematical Analysis and Applications*, 357(1):154–170.