School of Computation, Information and Technology
Department of Computer Science

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Computational Science and Engineering

A Framework for Variance Based Sensitivity Analysis Applied to Stochastic Modelling of Human Driver Behavior in Traffic Simulations

Theresa Marie Hefele

School of Computation, Information and Technology
Informatics

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Computational Science and Engineering

A Framework for Variance Based Sensitivity Analysis Applied to Stochastic Modelling of Human Driver Behavior in Traffic Simulations

Ein Framework für varianzbasierte Sensitivitätsanalysen, angewandt auf stochastische Modelle für menschliches Fahrerverhalten in Verkehrssimulationen

Author:	Theresa Marie Hefele
Supervisor:	Univ.-Prof. Dr. Hans-Joachim Bungartz
Advisors:	Dr. rer. nat. Tobias Neckel
	Arun Das, M.Sc. (BMW Group)
Date:	05 October 2023

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 05 October 2023 Theresa Marie Hefele

Acknowledgements

I wish to express my gratitude to my academic advisor, Tobias Neckel, for his ongoing support during the last six months. I also thank Prof. Dr. Hans-Joachim Bungartz for supporting this research topic with an external company. I enjoyed working with this chair a lot.

I am very grateful to all my colleagues at BMW: Team Dijkstra, FG-463, EG-342, Vasco, Alex, and especially Arun, who made this work possible and always found time for inspiring technical discussions.

I also want to thank the Max Weber Program for supporting my Master's studies financially and with numerous inspiring new impulses from outside my bubble.

Thank you Vivi, Martini, Kevin, Mélise, and Sebastian for making me laugh until my stomach hurts and sticking around, no matter what.

Das Wichtigste zum Schluss: Danke Mama, Papa und Matthias, dass ihr alle meine Vorhaben bedingungslos bestärkt und unterstützt.

Abstract

An important research topic in the field of Automated Driving (AD) is its interaction with human behavior. This requires algorithms for modeling human driver behavior, pedestrians, bikes, modeling of vehicle dynamics, and much more. One approach to this is to base the traffic simulations and traffic participant models on stochastic processes with stochastic distributions as input parameters. This leads to new traffic situations and edge cases that can be analyzed. Obtaining the input distributions and understanding how they influence the simulation result can be challenging. However, understanding the importance of input parameters can be crucial to further improve simulation models.

This thesis develops a framework for measuring the effect of input parameters for stochastic models on the results of traffic simulations. Global variance-based sensitivity analysis, specifically first- and total-order Sobol' indices, measure the sensitivity of the simulation models in this framework. The focus lies on the total-order effects because these models are supposed to have non-negligible parameter interactions and strong nonlinearities. Monte Carlo approximation of the Sobol' indices addresses the potentially high number of parameters, i.e., more than 28. The samples are generated quasi-randomly to improve the convergence of standard pseudo-random sampling. To further reduce computation time, the execution of the simulations is parallelized.

The framework is applied to BMW's Stochastic Cognitive Model (SCM) in multiple traffic scenarios generated by the Open Platform for the Assessment of Safety Systems (openPASS). This application demonstrates the plausibility of the framework's analysis results and showcases the usage and interpretation of the sensitivity indices. The results show that the parameter's Sobol' index highly depends on various conditions, such as the traffic scenario, the analyzed time step of the simulation, and how the parameter was defined. The differences between the analysis results for first- and total-order indices indicate strong higher-order parameter interactions. This justifies the computational costs necessary for calculating total-order effects.

With the total-order effects, researchers and developers can identify interactions between parameters in complex traffic scenarios and, therefore, obtain their influence with certainty. This supports the understanding of the way a model works. It can discover parameters with too much or too little influence and parameters that can be eliminated to simplify the model.

Contents

Part I.

Introduction

As automated driving systems evolve, simulating and virtually validating these systems is crucial to ensure their safety and reliability. Simulation methods provide means to thoroughly test and evaluate automated driving functions prior to real world tests. A fundamental aspect of these simulations involves accurately modeling human driver behavior. Understanding and replicating the interactions between human behavior and automated systems is vital for developing these systems. By virtually replicating real-world conditions, traffic simulations enable researchers, car manufacturers, insurance companies, and others to analyze how automated vehicles interact with human-driven vehicles, evaluate traffic flow, and identify risks. A promising platform for conducting such analyses is the Open Platform for Assessment of Safety Systems (openPASS) [1], an open-source tool developed with the Eclipse Foundation. OpenPASS is used throughout this work.

An important aspect of automated driving simulations is the interaction between human drivers and automated systems. Therefore, human driver models are employed within simulation platforms. These models aim to replicate the diverse behaviors observed in real-world driving scenarios, ranging from simple rule-based models to more complex machine-learning algorithms. Within the openPASS platform, BMW has developed a Stochastic Cognitive Model (SCM) to replicate human driving behavior based on stochastic decision-making. Understanding human driver models' characteristics, strengths, and limitations is crucial for accurately assessing the performance and impact of automated driving systems.

Developing, validating, verifying, and maintaining human driver behavior models is important. The accuracy of these algorithms impacts the reliability, safety, and effectiveness of the automated driving systems under test. However, this poses a challenging task for several reasons: Firstly, human driving behavior is highly complex and variable because many factors, such as individual characters, environmental conditions, and social interactions, influence it. Capturing this complexity and variability in a model requires extensive data collection, analysis, and calibration. Secondly, validating the algorithms necessitates comparing their outputs against real-world driving data or expert knowledge. Acquiring reliable ground truth data is often difficult due to the inherent risks and costs associated with on-road testing. Additionally, human driving behavior is subjective and can vary between individuals and cultures, making it challenging to establish a definitive benchmark for validation. Thirdly, new data is becoming available as understanding all these aspects of human driving behavior evolves. Consequently, this dynamic nature of validation requires constant maintenance and further development of the model, which again increases its complexity.

Conducting sensitivity analysis for human driver models in automated driving simulations can help keep and establish an understanding of a model: It is essential for identifying influential factors, assessing model robustness, understanding parameter interactions, optimizing computational resources, and enhancing interpretability. By incorporating sensitivity analysis into the development and validation process, researchers can improve the accuracy, reliability, and trustworthiness of automated driving simulations, ultimately contributing to the advancement and safe deployment of autonomous vehicles.

Goal: Analyzing Input Parameters of Stochastical Black Box Models

This thesis seeks to develop a framework to perform sensitivity analysis for simulation models in traffic simulations. Those models can be arbitrarily complex and potentially only provided as binaries without further insights into model structure or implementation. A generic approach is necessary to enable applicability to any of such black box models. The rationale behind this stems from several considerations:

1. **Credibility of the Simulation Model.** The ability to quantify and rank parameters allows for the validation of the model. It aids in identifying whether a specific parameter holds the influence it should or whether its effect is over- or under-represented.
2. **Enhancing Understanding.** This effort aids in exploring which parameters significantly influence the model. This not only deepens the understanding of the model but also assists in simplifying the tool through factor fixing.
3. **Comparative Analysis.** We also aim to establish how a parameter's influence might vary across different scenarios, offering insights into the model's adaptability and dynamic nature.
4. **Unveiling Parameter Interactions.** The tool should reveal and highlight interactions between parameters and generate an understanding of the more nuanced aspects of model performance and operation.

To show the capabilities of the framework, it is applied to the human driver model SCM within openPASS. The main interest is to investigate the human driver behavior's input parameters on longitudinal driving behavior. This will be demonstrated with specific emphasis on velocity-related, reaction-, and following behavior parameters of SCM.

Thesis Outline

Apart from the introduction and conclusion, this thesis is divided into two parts: Part II is meant to give a detailed overview of the technical concepts applied in the framework, as well as its internal structure and implementation. Part III demonstrates how those methods are implemented, how their convergence and performance can be improved and how they can be applied to SCM.

Part II: Technical Background

- **Chapter 1** provides an overview over SCM and openPASS. The main purpose of this chapter is to summarize the important model characteristics to consider when performing a sensitivity analysis of simulation algorithms based on stochastics.
- The first part of **Chapter 2** gives context to global variance-based sensitivity analysis by comparing the most common methods of conducting sensitivity analysis. The second part explains the global variance-based sensitivity analysis concept, how to compute it, and how to improve the convergence rates.
- **Chapter 3** explains the Ishigami function, commonly used to validate sensitivity analysis tools and methods because it is strongly nonlinear and its sensitivity indices are computable analytically. Therefore, it provides a good benchmark method.

Part III: Main Part

- **Chapter 4** describes how the theory of variance-based sensitivity analysis from Chapter 1 applies to the models in the stochastic traffic simulation context.
- **Chapter 5** investigates convergence and performance behavior. This chapter merges concepts from part two of Chapter 2 and Chapter 3.1 and applies them to the framework.
- **Chapter 6** represents both plausibility checks for the methods implemented and showcases how to interpret the results from the tooling.

Info Boxes

For users of the framework, there are some info boxes at locations where user-specific information needs to be pointed out. Readers only interested in the methodology can skip the user-information boxes:

User Information 0.1: Information Boxes

This is an exemplary information box to show what they look like. You will find more of these in the remaining parts of this thesis.

Part II.

Technical Background

1. Technical Aspects of Stochastic Traffic Simulation Models

Driver behavior models are heavily used in traffic simulations. SCM is a complex model to simulate human driving behavior during traffic simulations to develop Automated Driving (AD) functions. This chapter aims to provide a comprehensive overview of SMC, drawing primarily upon sources [2]. This includes a summary of openPASS, the 'home' of SCM. It is important to note that these descriptions remain high-level, as this analysis treats any model as a black box.

There exist many methods for evaluating the safety of AD algorithms via computational simulations. P.E.A.R.S [3], a consortium looking into the simulation-based assessment of traffic safety, has outlined four key concepts for assessing the safety of Advanced Driver Assistance Systems (ADAS):

- **A.** Testing the AD algorithms using existing real-world cases (i.e., reconstructed crash or field data) without any changes. An example is given in [4].
- **B.** Usage of real-world cases plus varying the initial values using distributions to test more than the available cases. An example is [5].
- **C1.** Deriving scenario mechanisms and distributions from real-world cases and selecting a low number of representative cases. This approach has been applied in [6].
- **C2.** Deriving scenario mechanisms and distributions from real-world cases and applying sampling to generate multiple cases. Examples are [7].

The authors of [8] consider simulation approach C2 the most promising for assessing the safety performance of autonomous driving. Unlike approaches A and B, C2 does not rely directly on real-world cases but establishes a connection through distributions. Only the starting conditions of road users are sampled from input distributions derived from accident or field data. The decisions and actions of traffic participants are modeled using behavior models. Therefore, anything happening during simulation happens naturally with this approach. OpenPASS follows this C2 modeling approach.

1.1. Simulation Framework: openPASS

OpenPASS simulates traffic scenarios for evaluating the performance of AD functions. Figure 1.1 depicts a screenshot of an exemplary traffic simulation in openPASS. It models vehicles, drivers, and pedestrians in a virtual environment. Traffic participants are called agents. SCM steering a vehicle is one of those agents. Just like the C2 model approach, the simulations are stochastic, meaning that multiple variations of the system and agent positions are tested,

leading to a range of possible outcomes. The information in this section is based on [9] if not stated otherwise.

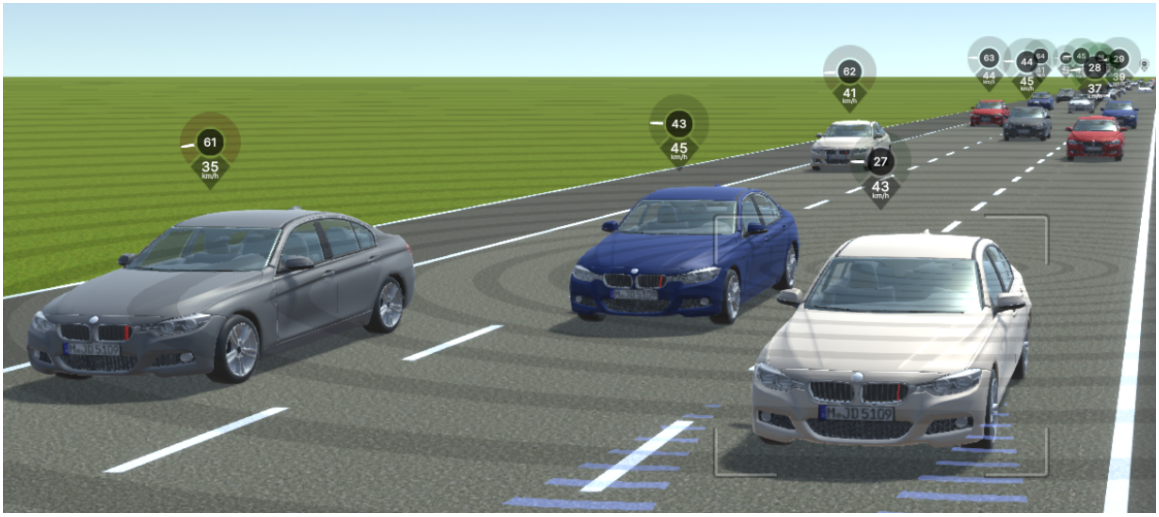

Figure 1.1.: OpenPASS simulation result visualized [10].

The core algorithm consists of several classes and abstracted steps, also sketched in Algorithm 1. OpenPASS executes Tasks, which can be various computations necessary during the simulation. They can be linked to a single agent or core functions within the simulation kernel. The simulation process can be divided into three main steps:

1. It begins with the initialization tasks and placing agents into the environment randomly. Most initialization procedures are random decisions based on distributions or probabilities defined by the user or internally.

2. The time-stepping phase computes the time-dependent part of the simulation. SpawningAgentTasks create new agents as they enter the observed location in space. The update step divides the agent's computational logic into NonRecurringAgentTasks and RecurringAgentTasks. PreAgentTasks publish global simulation data to specific classes for outputting results. SynchronizeAgentTasks synchronize data among agents, and a clearing step releases temporary simulation data. Most of the tasks in this step include random processes which are - to some extent - controllable by input parameters. Some random variables that are the basis for decision-making are redefined at every time step.

3. Finally, FinalizationTasks write the simulation output.

Algorithm 1: Main openPASS simulation algorithm [10].

```
t ← t_0
Execute BootstrapTasks
while t ≤ T do
    Execute SpawningAgentTasks
    update agents
    execute PreAgentTasks
    execute NonRecurringAgentTasks
    execute RecurringAgentTasks
    execute SynchronizeAgentTasks
    clear
    t ← t + δt
Execute FinalisationTasks
```

1.2. Structure of the Stochastic Cognitive Model

The challenge for the C2 modeling approach lies in the absence of predefined trajectories of traffic participants according to [8]. Therefore, their movement results from the starting conditions and must be derived during the simulation. This necessitates a driver behavior model representing everyday driving, human mechanisms, and reactions in critical scenarios to ensure realistic traffic scenarios. SCM stems from the P.h.D of [11] as a response to the mentioned need for an algorithm that simulates human driving behavior. SCM is structured to function on motorways within openPASS [1], described in the previous section. The model will be considered as a black box. However, it is important to account for certain aspects of the model's functionality when developing the analysis. As such, this chapter summarizes the necessary background information to frame subsequent analyses in a meaningful context. This comprises two aspects: firstly, the core principles that guide the modeling of SCM are presented. Secondly, the inputs that SCM relies on will be summarized and classified into groups to clarify which can and cannot be analyzed with this tool.

1.2.1. Building Blocks and Determinism

SCM is a complex, non-linear model that uses stochastic algorithms to model human driving behavior. Its' structure consists of six key components depicted in Figure 1.2: Information Acquisition, Mental Model, Situation Manager, Action Manager, Action Implementation, and Driver Characteristics. Each component employs stochastic processes to replicate human decision-making and physiology in the perception and execution of actions in driving scenarios. The Information Acquisition and Mental Model modules serve as the perceptual and cognitive centers of the model, while the Action Manager and Action Implementation modules translate decisions into vehicle commands. The Driver Characteristics module underpins all processes by providing parameter distributions. As specified in the goal of this thesis, outlined in Part I, SCM will be treated as a black box for this study. More in-depth insights into SCM are available in prior works by [8], [12], and [13].
The probabilistic nature of openPASS and SCM is one of their key characteristics. It introduces controllable variability into the simulations: openPASS and SCM allow the

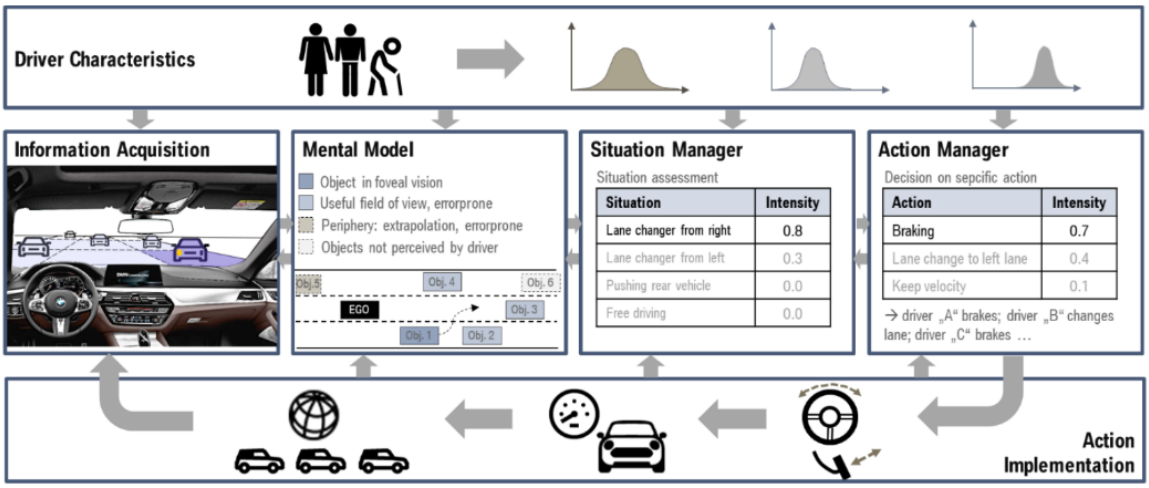

Figure 1.2.: Overview of the information flow within SCM [8].

definition of a random seed, ensuring the stochastic processes' reproducibility. When a particular seed is employed consistently, the stochastic model behaves deterministically, leading to identical simulation outcomes with the same input parameters and seed. Expanding on this last point, the determinism achieved through consistent random seed usage holds significant implications for the analysis and validation possibilities of the model. Analysts can conduct repeatable and reliable simulations by controlling the random number generator of the stochastic processes. This feature also allows for systematic exploration of parameter spaces. Chapter 1.3 gives more information on the random number generator used in SCM and openPASS.

1.2.2. In- and Output of SCM

Input

In the context of this thesis the parameters of SCM are grouped into the following:

- **User-Defined Parameters.** The user can define these parameters via an input file called ProfilesCatalog.xml. Most of them are defined via distributions.
- **Internal Parameters.** Certain parameters are randomly assigned by SCM internally. Some parameters are drawn from a distribution at the beginning of the simulation, and some are drawn repeatedly in each time step. This makes it especially important to be careful with how the random base generator works to ensure that the final result stays deterministic.
- **Environment-, and Vehicle-Based Inputs.** SCM's most interesting aspect is its interaction with its environment. Trivially, the surrounding traffic situation represents a significant input for SCM. Another category would be vehicle-specific responses to the actions of SCM. For simplicity, those are also included when talking about environment-based parameters in the upcoming chapters.

Output

An openPASS simulation generates a time-dependent output, capturing the states of all participating agents at each simulated time step. This includes, for example, speed-related data such as velocity and acceleration. These results can be visualized as a virtual map, displaying traffic participants and their interactions. It is the user's responsibility to define which statistics represent valuable information. This task is scenario-specific. For instance, a key metric of interest in an emergency brake scenario would typically be the Time to Collision. In a scenario where SCM follows another vehicle, the primary metric of interest is the Time Gap (TGap) - the temporal distance between two vehicles.

1.3. Random Number Generator

As detailed in the preceding sections, openPASS and SCM require - amongst others - random variables as input parameters and operate based on further stochastic decisions during runtime. In other words: input parameters can change their value between time steps. This underlines the necessity to evaluate if a particular input value can be defined and maintained consistently throughout the simulation: to quantify the influence of certain input parameters, it is required to alter only one parameter at a time, ensuring the remaining input space and decisions made during simulation, which are not affected by that parameter, remain deterministic. Both openPASS and SCM employ the mt19937 generator from the C++ standard library [14] to generate pseudo-random numbers. Based on the Mersenne Twister Algorithm [15], this generator uses a seed to establish its initial state and produces a sequence of pseudo-random numbers. This leads to two key aspects:

1. **Initialisation.** The mt19937 generator is initiated with a seed value, which sets the initial state. The generator transitions through its internal state upon each invocation, producing a new pseudo-random number. This state transition is deterministic and reproducible, guaranteeing consistent sequences when the same seed is used.
2. **Order Dependency.** The sequence of generated numbers varies depending on the order of invocation. Different invocation orders yield distinct pseudo-random sequences even when identical seeds are used. This is attributable to the progression of the generator's internal state, which is influenced by each call [15].

Integration of mt19937 into openPASS and SCM

As work on this thesis progresses, SCM is transitioning from exclusive access within BMW to open-source availability. During this transition, it is being encapsulated from the openPASS framework into its own Functional Mockup Unit (FMU) to facilitate the use of SCM in other simulation platforms outside openPASS. This transition affects the use of the stochastic library within SCM: while it was directly embedded in openPASS and used the same stochastic library (and thus, the same base generator), post-transition SCM contains its own stochastic library and base generator.

2. Sensitivity Analysis

This chapter provides an overview of sensitivity analysis, which holds different interpretations across different contexts and modeling communities. The overarching objective of this chapter is to establish an understanding of the term *sensitivity analysis* in the context of this thesis. Therefore the following sections explain the benefits and drawbacks of existing sensitivity analysis methods, present the algorithm used in this work, and discuss the problem and possible solutions of convergence in Monte Carlo methods, which is the underlying concept of the analysis used in this thesis.

2.1. Approaches on Sensitivity Analysis

This section utilizes several resources to provide a comprehensive understanding of sensitivity analysis. Chapter 9 of Neckel's "Lecture on Algorithms for Uncertainty Quantification" [16] offers both a general overview of sensitivity analysis and detailed insights into global variance-based sensitivity analysis and its calculation using polynomial chaos expansion. Straub's lecture notes on Engineering Risk Analysis [17] features a chapter that describes sensitivity analysis and its applications in engineering contexts. Saltelli's book [18] on global sensitivity analysis, serves as a reference for many research papers and is a detailed reference for any available methods for conducting sensitivity analysis including discussions about rationals and applicability. Smith's book [19] on uncertainty quantification provides a broader perspective on the overarching subject of Uncertainty Quantification, effectively positioning sensitivity analysis within this wider context.

Definition of sensitivity analysis

Sensitivity Analysis *provides information on how the input parameters* $\boldsymbol{X}$ *and their uncertainties influence the model output* $\boldsymbol{Y}$, [17]. Furthermore [17] summarizes the following motivations for conducting sensitivity analysis from [18].

1. **Robustness.** Assert whether a model is robust or overly fragile concerning assumptions made on various input parameters.
2. **Screening.** Spot negligible parameters and give justification to simplify a given model by fixing those insensitive inputs.
3. **Ranking.** Rank input factors according to their importance.
4. **Mapping.** Identify parameter regimes that have a large impact on a model's sensitivity possibly leading to unwanted behavior, and on the other hand find out which regimes optimally impact responses of the model.
5. [18] and [17] list many more depending on the context.

2.1.1. Local and Global Sensitivity Analysis

Sensitivity analysis can be classified as local or global. If not stated otherwise, information in this chapter is based on [18]. Figure 2.1 depicts a generic model

$$Y = f(\mathbf{X}) = f(X_1, X_2, ...X_k), \tag{2.1}$$

where $\mathbf{X}$ are independent random variables. This model is be the basis for the current chapter.

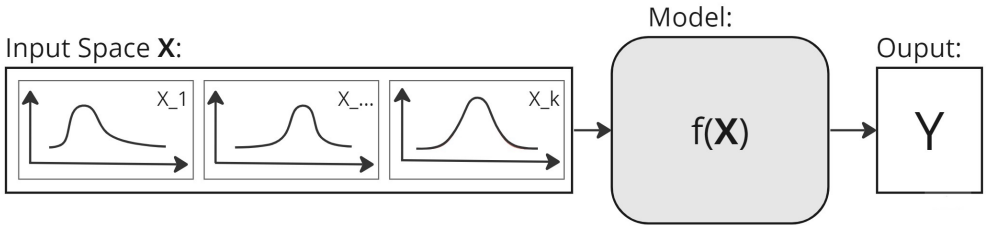

Figure 2.1.: Schematic of a generic model definition with random variables $\mathbf{X}$ as input for $f(\mathbf{X}) = Y$, where $\mathbf{Y}$ is a scalar value.

Local sensitivity analysis

Local sensitivity analysis concentrates on assessing the variability in a model response upon the introduction of small perturbations to inputs, centered around a nominal value. In cases where the input is represented as a distribution, the mean value is frequently used as the nominal value. According to [18], the foundation of most local sensitivity analyses lies in derivatives. In this case, the derivative $\frac{\partial Y_i}{\partial X_i}$ of an output Y_i with respect to an input X_i can be construed as a mathematical embodiment of the sensitivity of Y_i relative to X_i. While derivative-based approaches possess the merit of computational efficiency, they fall short when confronted with models of uncertain linearity and indeterminate inputs. This limitation stems from the fact that the derivatives capture sensitivity information only at the precise location they are computed, leaving the remaining parameter space unexplored. This phenomenon is less problematic for linear systems, where extrapolation of model behavior is feasible based on the first-order derivative. However, it poses considerable difficulties for non-linear systems, [19].

Global sensitivity analysis

In contrast to local sensitivity analysis, global sensitivity analysis analyses the contribution of the whole range of values in a parameter's space towards the output, not only one nominal value. Additionally, it facilitates the identification of uncertainties resulting from combinations of parameters across the analyzed parameter space. The selection of appropriate measures and methods for performing global sensitivity analysis heavily depends on the characteristics of the model under consideration and the questions the sensitivity analysis aims to answer. The subsequent chapter will introduce existing methods. In the context of this work, a variance-based measure will be utilized and contextualized amongst these global sensitivity analysis methods.

2.1.2. Methods for Global Sensitivity Analysis

One-at-a-time analysis (OAT)

This section follows the description from [19], which says that One-at-a-Time Analysis (OAT) offers a straightforward and intuitive approach to sensitivity analysis. The method involves changing a single factor within a specified range while holding all other factors constant. This approach allows researchers to investigate how changes in individual parameters separately affect the output. OAT analysis has several advantages: firstly, it is straightforward to apply and understand. Additionally, the method has relatively low computational costs.

Despite its benefits, OAT analysis has several drawbacks. A significant limitation is that it does not allow for exploring interactions between input parameters. Furthermore, for models with a high number of dimensions, this method can be inefficient as each parameter must be evaluated individually. Additionally, ranking the importance of parameters against each other can be more challenging with this method than others. While OAT analysis is typically conducted at a specific value, making it a local method, there are variations of OAT that can perform global sensitivity analysis. One such method is Morris Screening, which can rank parameters according to their importance without quantifying the degree of their importance. For a detailed explanation of the Morris Screening algorithm, readers can refer to Chapter 15.2 of Smith's book on uncertainty quantification [19].

Graphical sensitivity analysis: scatterplots

Following the descriptions from [18] and [17], Scatterplots offer one of the simplest and most qualitatively informative ways to conduct sensitivity analyses. The scalar results can be plotted over the input values by applying Monte Carlo sampling. The distribution of these points indicates the influence of the parameter: the narrower the distribution, the greater the parameter's influence in that particular regime. According to [18], many sensitivity analysis measures condense scatterplot information. However, scatterplots can only provide a qualitative global sensitivity analysis. For sets of parameters that significantly influence the results, the two-dimensional scatterplot may be insufficient for comprehensive analysis.

Linear regression coefficients

Chapter 1.2.5 [18] explains that the idea underpinning the use of linear regression for sensitivity measurement is to utilize the results of a Monte Carlo simulation and re-describe the model by fitting the resulting data to a linear model of the form: $Y = b_0 + \sum_{j=1}^{k} b_{X_j} * X_j^i$. In this expression, Y represents the model solution, X the input, and b_i the constants. The coefficients b_0, b_{X_i} serve as effective sensitivity measures for linear or near-linear models. These coefficients can quantify and rank the effects of an input parameter on the output. However, like OAT, this method neglects the interactions between coefficients for non-linear models. For black box models, where assumptions about linearity are impossible, reliance on linear regression coefficients for quantifying the importance of input parameters is not recommended.

Variance-based methods

Variance-based methods are common tools for analyzing parameter interactions within global sensitivity analysis. Here, the uncertainty of the output is represented by its variance [20]. A standard sensitivity measure involves examining the contribution of an input parameter X_i to the output variance $V(\mathbf{Y}) = V(f(\mathbf{X}))$. This measure forms the basis of the work conducted in the main chapter and will thus be elaborated on in greater detail in the following section.

2.2. Variance Based Sensitivity Analysiss

Before looking at the formal definition of variance-based sensitivity measures, this chapter follows the rationals established in Chapter 1 of [18] to provide an intuition about what they actually mean: Given the generic model defined in Equation 2.1, where each $\mathbf{X_i}$ has some uncertainty, it can be interesting to see what happens to Y if one of the input factors $\mathbf{X_i}$ could be fixed at $\mathbf{X_i} = x_i$. Comparing $V(Y|X_i = x_i)$, which is the variance of the solution when it was fixed at $X_i = x_i$ and when it was not ($V(Y)$) can give some indication about how important X_i's contribution is to the variance of the output: the smaller $V(Y|X_i = x_i)$, the more important the parameter. This approach only measures a local effect, which is why it makes sense to compute this variance for the whole set of values in $\mathbf{X_i}$ and take its average: $E(V(Y|X_i))$. This is always lower or equal to $V(Y)$, actually [21] proved that

$$V(E(Y|X_i)) + E(V(Y|X_i)) = V(Y). \tag{2.2}$$

Therefore, a small $E(V(Y|X_i))$ and a large $V(E(Y|X_i))$ imply a big influence of the parameter $f(\mathbf{X_i})$ on Y. For simplicity, and as generally referred to in the literature, the remaining part of this work will abbreviate

$$V_i = V(E(Y|X_i)), \forall i \in \{1, 2, \ldots, k\}. \tag{2.3}$$

First-order Sobol' indices

These considerations lead to the definition of the so-called *first-order Sobol' index*:

$$S_i = \frac{V_i}{V(Y)}, \forall i \in \{1, 2, \ldots, k\}, \tag{2.4}$$

which relates the earlier explained V_i to $V(Y)$. It only measures the effect of a single parameter without any interactions with other parameters.

Higher-order Sobol' indices

Higher-order Sobol' indices are defined similarly. To give an example, second-order Sobol' indices

$$S_{i1,\ldots,ik} = \frac{V_{i1,\ldots,ik}}{V(Y)}, \forall i \in \{1, 2, \ldots, k | i \neq i\} \tag{2.5}$$

measure the contributions of second-order interaction between two input parameters.

Total-order Sobol' indices

$$S_i^T = \sum_{i \in \{i_1,\ldots,i_s\}} S_{i_1,\ldots,i_s} \tag{2.6}$$

represent the sum of all orders of contributions of a parameter. These allow meaningful conclusions about the impact of an input parameter on a result. [20] defines the total-order Sobol' indices using the following relation:

$$S_i^T = \sum_{i \in \{i_1,\ldots,i_s\}} S_{i_1,\ldots,i_s} = \frac{E(V(Y|\mathbf{X} \sim i)}{V(Y)} = 1 - \frac{V(E(Y|\mathbf{X} \sim i)}{V(Y)}, \tag{2.7}$$

which can be interpreted as considering that $V(E(Y|\mathbf{X} \sim i)$ is the first-order effect of $\mathbf{X}_{\sim \mathbf{i}}$, so that $V(Y) - V(E(Y|\mathbf{X} \sim i)$ must give the contribution of all terms in the variance decomposition which do include X_i. This definition makes it possible to compute the total-order effects in the same computation as the first-order effects.

Sobol' index considerations

There are a few properties of Sobol' indices to keep in mind for verification: The sum of all first- and higher-order Sobol' indices (excluding the total-order) equals one because Sobol' indices decompose the function's variance into its contributions. If a model has only first-order effects then the sum of all first-order indices (Z) would be 1. The value of Z also gives information about higher-order effects: the impact of higher-order effects on a systems's sensitivity rises with decreasing Z. On the other hand, if Z exceeds one then there might be too few samples, numerical errors or a bug in the framework as this is theoretically impossible.

The sum of all total-order indices must be at least one. This is because if there is an interaction between parameter U and parameter W, then this contribution is accounted for in both the total-order Sobol' index of U and parameter W.

To compute S_i and S_i^T, the challenging/costly part is to compute $V_i = V(E(Y|X_i))$ and $V(E(Y|\mathbf{X} \sim i)$. There are generally two methodologies for computing the Sobol' indices: Polynomial Chaos Expansion (PCE) uses a surrogate model by redescribing it's output variance using the ANOVA decomposition described in [19]. However, according to [20], the complexity of this approach is $O(2^k)$, which causes computational challenges for larger numbers of parameters (k). Alternatively, the indices can be numerically approximated using techniques such as Monte Carlo sampling. The complexity of this method is $O(k * N)$, where N represents the number of samples. A drawback to this approach is the theoretical convergence rate of Monte Carlo sampling, which is $O(\frac{1}{\sqrt{N}})$. This convergence can be improved to $O(\frac{log(N)^k}{N})$ by applying quasi-random sampling. This is explained in Section 2.2.3. Given that SCM's parameter space can exceed 30 potential input parameters, the Polynomial Chaos Expansion approach is impractical due to its computational complexity.

2.2.1. Approximating Sobol' Indices Using Monte Carlo Sampling

The paper by Saltelli et al. [20] provides a detailed explanation and comparison of various mechanisms for approximating the Sobol' indices, serving as the source of all information presented in this chapter. The current section will focus on the specific definitions employed in this thesis.

The first step involves sampling each parameter $2 * N$ times to generate two matrices:

$$\mathbf{A} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_i^{(1)} & \dots & x_k^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_i^{(2)} & \dots & x_k^{(2)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_1^{(N-1)} & x_2^{(N-1)} & \dots & x_i^{(N-1)} & \dots & x_k^{(N-1)} \\ x_1^{(N)} & x_2^{(N)} & \dots & x_i^{(N)} & \dots & x_k^{(N)} \end{bmatrix} \in \mathbb{R}^{k \times N} \tag{2.8}$$

$$\mathbf{B} = \begin{bmatrix} x_{k+1}^{(1)} & x_{k+2}^{(1)} & \dots & x_{k+i}^{(1)} & \dots & x_{2k}^{(1)} \\ x_{k+1}^{(2)} & x_{k+2}^{(2)} & \dots & x_{k+i}^{(2)} & \dots & x_{2k}^{(2)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_{k+1}^{(N-1)} & x_{k+2}^{(N-1)} & \dots & x_{k+i}^{(N-1)} & \dots & x_{2k}^{(N-1)} \\ x_{k+1}^{(N)} & x_{k+2}^{(N)} & \dots & x_{k+i}^{(N)} & \dots & x_{2k}^{(N)} \end{bmatrix} \in \mathbb{R}^{k \times N} \tag{2.9}$$

Then, for each parameter, one new matrix is generated,

$$\mathbf{A}_\mathbf{B}^\mathbf{i} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_{k+i}^{(1)} & \dots & x_k^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_{k+i}^{(2)} & \dots & x_k^{(2)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x_1^{(N-1)} & x_2^{(N-1)} & \dots & x_{k+i}^{(N-1)} & \dots & x_k^{(N-1)} \\ x_1^{(N)} & x_2^{(N)} & \dots & x_{k+i}^{(N)} & \dots & x_k^{(N)} \end{bmatrix} \in \mathbb{R}^{k \times N}, \tag{2.10}$$

where all samples are taken from $\mathbf{A}$ but the ith parameter sample, which is taken from matrix $\mathbf{B}$. This process results in a total of $k * (N + 2)$ model evaluations. [18] explains why the computational effort required to compute second-order indices would be $k*(2*N+2)$.

An advantage of the above-described method is that it enables the simultaneous computation of the first-, S_i and total-order S_i^T Sobol' indices. The definitions of the first- and total-order Sobol' indices 2.4 and 2.7 in Chapter 2.2 require the computation of $V(E(Y|X_i))$ and $V(E(Y|X \sim i))$. The paper provides the following approximation method:

$$For\ S_i : V(E(Y|X_i)) \approx \frac{1}{N} * \sum_{j=1}^{N} f(\mathbf{B})_j * (f(\mathbf{A_B}^{(i)})_j - f(\mathbf{A})_j) \tag{2.11}$$

$$For\ S_i^T : E(V(Y|\mathbf{X} \sim i)) \approx \frac{1}{2 * N} * \sum_{j=1}^{N} (f(\mathbf{A})_j - f(\mathbf{A_B}^{(i)})_j)^2 \tag{2.12}$$

Remarks on Equation 2.11:
There are two things to consider when applying Equation 2.11: first, more computationally efficient methods exist to approximate V_i via Monte Carlo sampling. One possibility involves taking the averages over slices in scatter plots from Monte Carlo sampling, explained in Chapter 1.2.7 of [18]. Another method based on Monte Carlo sampling is described in Chapter 5.6.4 of [17]. The problem with these two methods is that they only provide first-order effects. However, the above-described method allows for the simultaneous computation of both the first- and total-order indices.

The second thing to be aware of is that there is no guarantee for positive S_i estimates. This becomes clear when looking at the inner substraction in Equation 2.11: $f(\mathbf{A_B}^{(i)})_j - f(\mathbf{A})_j$. That means that for small sampling sizes, it can happen that the approximations of S_i are negative. However, S_i estimations below zero should be very close to zero and its confidence intervals should overlap with zero. If this is the case and N cannot be increased, the result can be considered as zero according to [22].

2.2.2. Convergence of Standard Monte Carlo Sampling

The basic idea behind Monte Carlo estimation is to use pseudo-random sampling to generate points within the integration domain and estimate the integral based on the average value of the integrated function over these sampled points. The Monte Carlo method provides a stochastical approach to estimate the integral value with an associated uncertainty. Appendix A of [23] is this chapter's source of information.

Monte Carlo

An integral over a multidimensional domain $\Gamma \subset \mathbb{R}^m$ with volume V,

$$I = \int_{\Gamma} f(\mathbf{X}) d\mathbf{X}, \tag{2.13}$$

can be reformulated to

$$I = E[f(\mathbf{X})] * V. \tag{2.14}$$

The standard Monte Carlo approach is to sample points pseudo-randomly on the domain: given N randomly distributed samples $X_1, X_2, \ldots, X_N \in \Gamma, E[f(\mathbf{X})$ can be approximated by

$$E[f(\mathbf{X})] \approx \overline{f(\mathbf{X})} = \frac{1}{N} * \sum_{i=1}^{N} f(X_i) \tag{2.15}$$

because the law of large numbers ensures that

$$\lim_{N \to \infty} \overline{f(\mathbf{X})} = E[f(\mathbf{X})]. \tag{2.16}$$

Error and Convergence Rate of Monte Carlo

A common metric for the error of the Monte Carlo estimator is the Mean Squared Error (MSE):

$$MSE[\hat{\sigma}_f] = E[(\hat{\sigma}_f - \sigma)^2], \tag{2.17}$$

which quantifies a difference between the Monte Carlo estimation $\hat{\sigma}_f$ and the true value $\hat{\sigma}$. Further analysis of this definition while assuming that Monte Carlo is an unbiased estimator results in

$$MSE[\hat{\sigma}_f] = V(\hat{\sigma}_f). \tag{2.18}$$

Further analysing the definition of variance, the Definition 2.18 boils down to

$$MSE[\hat{\sigma}_f] = V(\hat{\sigma}_f) = \frac{\sigma^2}{N}. \tag{2.19}$$

Chapter 4 of [10] elaborates on these definitions in more detail.

Therefore, to reduce the error of Monte Carlo estimators, one needs to reduce its variance by increasing the number of samples. The definition of Monte Carlo leads to the following relation between the true variance and the variance of the sampled values. According to Equation 2.19 Monte Carlo estimators converge with $O(\frac{1}{\sqrt{N}})$.

2.2.3. Methods to Improve the Convergence of Monte Carlo

As the chapter above indicates, one needs to reduce the variance or increase the number of samples to accelerate the convergence of Monte Carlo estimators. One way to do that is by replacing the pseudo-random sampling in Standard Monte Carlo approaches with deterministic sampling patterns, also mentioned in [18], which will be elaborated in this section.

Quasi-random Monte Carlo sampling

Quasi-random Monte Carlo (QMC) sampling improves upon traditional Monte Carlo sampling by using low-discrepancy sequences to generate sample points that are "sampled" according to specific rules for numerical integration. Low discrepancy means that a measure between the discrete sampled values is as close as possible to the underlying continuous probability function. Unlike pseudo-random sampling, quasi-random generally aims for uniform coverage, enhancing convergence rates and accuracy. Figure 2.2 illustrates that idea.

Sobol' sampling method

The framework will utilize the sampling and analysis functionality of SALib, a Python library for sensitivity analysis. Therefore, this chapter gives an overview and justification for how the techniques in this library work. There are multiple different low-sequences available. [25] gives an overview and compares them. As proceeded in [20], which is also the basis for

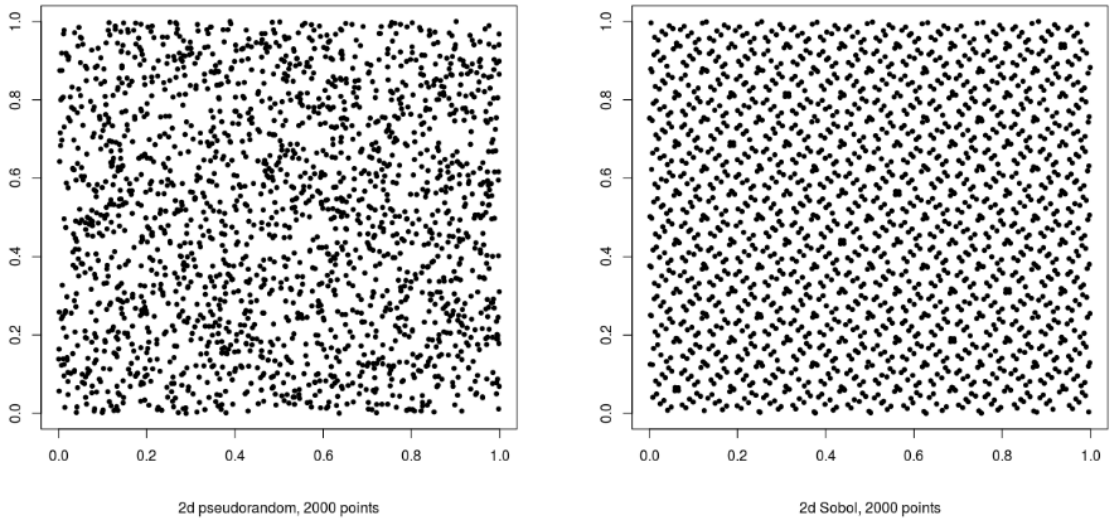

Figure 2.2.: Comparison: bivariate uniform samples - pseudo-random (left) and quasi-random (right) [24].

the SALib library [26], we are working with Sobol' sequences. [25] describes an algorithm to generate them, and [27] gives an intuition on how they are defined. More specifically, SALib enables scrambling the samples to avoid repetitions in the Sobol' sequence, which can be found in Table 2 of [28]. According to [29], QMC can speed up the convergence rate from $O(\frac{1}{\sqrt{N}})$ for standard Monte Carlo sampling to $O(\frac{log(N)^k}{N})$, where N is the number of samples and k the number of parameters to analyse.

Non-uniform distributions

The descriptions provided so far are based on uniform distributions. However, there will be non-uniform distributions in the Stochastic Cognitive Model (SCM) context. SALib allows the definition of normal and log-normal distributions as outlined in [30]. SCM and openPASS work with truncated normal distributions. As of the submission of this thesis, SALib does not offer truncated versions of such distributions. The source code leads to believe that truncated normal distributions are possible, but this needs further investigation. Therefore, continuing this work, we will work with uniform distributions.

3. Benchmark Options for Computing Sensitivity Analyses

This chapter introduces the Ishigami function introduced by Ishigami et al. in [31]. It includes a presentation of the analytical solution, standard Monte Carlo approximation, and the quasi-random Monte Carlo approximation of both the first- and second-order Sobol' indices. Furthermore, we will compare the convergence of standard- and quasi-random Monte Carlo.

3.1. Ishigami Function

The Ishigami function is well-known as a benchmark function in the research community for uncertainty and sensitivity analysis because of its strong nonlinearity and its second-order influence of x_1 and x_3:

$$f(\mathbf{X}) = sin(x_1) + a * sin^2(x_2) + b * x_3^4 * sin(x_1) \tag{3.1}$$

In this example, we set $a = 5$, $b = 0.1$ and sample the input variables $x_i \sim Uniform(-\pi, \pi)$. The Ishigami function has the following non-zero contributions:

$$f_1(x_1) = sin(x_1) \tag{3.2}$$
$$f_2(x_2) = a * sin^2(x_2) \tag{3.3}$$
$$f_{13}(x_1, x_3) = b * x_3^4 * sin(x_1) \tag{3.4}$$

Figure 3.1 visualises these non-zero contributions: looking at f_1 and f_2, x_1 and x_2 have non-zero components, x_3 does not. However, f_{13} depends on the second-order interaction of x_1 and x_3. Therefore, S_3 should converge towards zero, S_1 and S_2 are expected to have a non-zero value and S_3^T and S_1^T should be considerably higher than their first-order effects.

3.2. Sobol' Indices of Ishigami Function

As described in Chapter 15 of [19], the total and partial variances of Equation 3.1 are

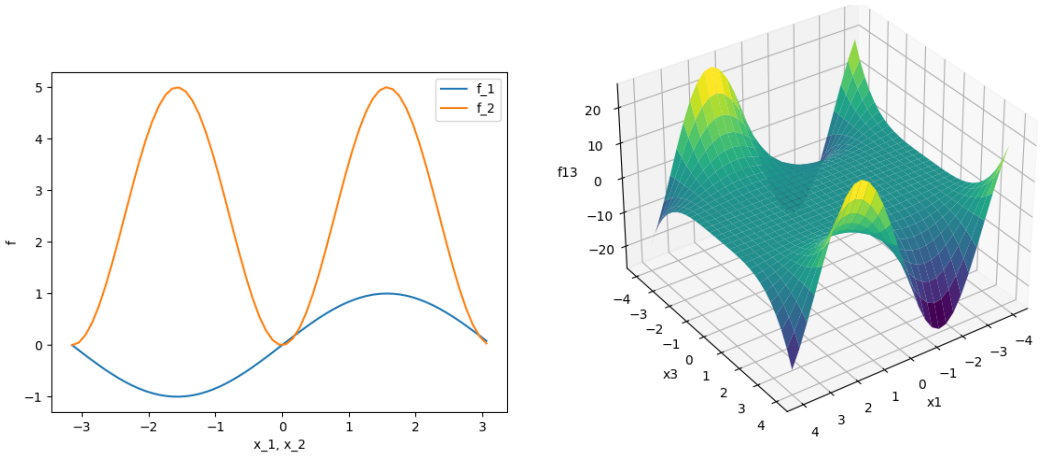

Figure 3.1.: Graphs for nonzero components of the Ishigami function with a = 5 and b = 0.1.

$$V(f(\mathbf{X})) = \frac{a^2}{8} + \frac{b * \pi^4}{5} + \frac{b^2 * \pi^8}{18} + \frac{1}{2} \approx 10.84 \tag{3.5}$$

$$V_1 = \frac{b * \pi^4}{5} + \frac{b^2 * \pi^8}{50} + \frac{1}{2} \approx 4.345 \tag{3.6}$$

$$V_2 = \frac{a^2}{8} \approx 3.125 \tag{3.7}$$

$$V_{13} = \frac{b^2 * \pi^8}{18} - \frac{b^2 * \pi^8}{50} \approx 3.374 \tag{3.8}$$

$$V_3 = V_{12} = V_{23} = V_{123} = 0 \tag{3.9}$$

We can see that $V_1 + V_2 + V_{13} = V(f(\mathbf{X}))$. According to the Definitions 2.4, 2.5, 2.7 and $S_i = \frac{V_i}{V(f(\mathbf{X}))}$ given in Chapter 2.2 the resulting Sobol' indices are

$S_1 \approx 0.40$	$S_2 \approx 0.28$	$S_3 = 0$
$S_1^T = S_1 + S_{13} \approx 0.712$	$S_2^T = S_2$	$S_3^T = S_{13} \approx 0.311$

Table 3.1.: Analytic results of Sobol' indices for Ishigami function.

Part III.

Implementation of the Framework

4. Implementation of Sensitivity Analysis Framework

The framework enables global variance-based sensitivity analysis by calculating first- and second-order Sobol' indices. Therefore, this chapter explains how the concepts from Part 2 are applied to the framework: Chapter 4.1 discusses the necessity of global variance-based sensitivity analysis to achieve this thesis's objectives. Additionally, it applies this approach to stochastic traffic models using the example of SCM. The chapter also addresses considerations for maintaining control over the randomness of other parameters. Chapter 4.2 unveils the implementation's architecture and explains the conceptual implementation of its building blocks. Chapter 4.3 highlights the necessary steps for extending this framework to other models and identifies its associated limitations.

4.1. Application of Global Variance-Based Sensitivity Analysis to Stochastic Traffic Simulations

This chapter describes how variance-based global sensitivity analysis can be applied to stochastic simulation models. Therefore, let's begin by looking at reasons that justify the costly computation of total-order Sobol' indices:

Parameter interactions. Interactions between input parameters are expected. For example, consider a free-driving scenario on a straight street with a speed limit of 100 $\frac{km}{h}$. The interaction between the driver's comfort velocity and willingness to drive faster than allowed should influence the actual velocity. Driver A, with a comfort driving velocity of 120 $\frac{km}{h}$ and a 0 $\frac{km}{h}$ willingness to drive faster than what is allowed ends up with a final velocity of 100 $\frac{km}{h}$. Driver B has the same comfort driving velocity but is willing to drive 10 $\frac{km}{h}$ faster than the speed limit. Therefore, driver B will end up with an actual velocity of 110 $\frac{km}{h}$. As a result, the interaction between those two parameters will considerably influence the resulting velocity. That means that only looking at first-order influences will not suffice the analysis of SCM. Chapter 6.2 presents the Sobol' indices for the described situation.

Remaining goals. Ranking the parameters, enhancing understanding, and comparing influences between different scenarios build up on unveiling those higher-order influences of parameters. More specifically, total-order Sobol' indices are the most reliable source of information for getting a trustworthy estimation for all those goals.

Nonlinearity. On top of these considerations, as described in Chapter 1.2.2, SCM is a nonlinear model and is considered a black box for this analysis. Therefore, a sensitivity analysis method that suits nonlinear models is necessary.

Considerations about computational effort

As described in Chapter 2.2, computing total-order Sobol' indices is by far the most computationally expensive among standard options for sensitivity analysis. This fact requires additional thoughts on the efficiency of both the method itself and its implementation: Several efficiency improvement options apply to this problem, both the method and the implementation. One simulation usually takes between a few seconds and two to three minutes to compute. The following chapter will elaborate on how many simulations are necessary and how to estimate the simulation time and computational bottlenecks, which will be the basis of considerations for further efficiency strategies.

4.1.1. Parameter-Sampling-Invocation Space

This chapter defines the idea of the analysis setup. Therefore it is necessary to start with the analysis input requirements. Figure 4.1 sketches the input space for the analysis:

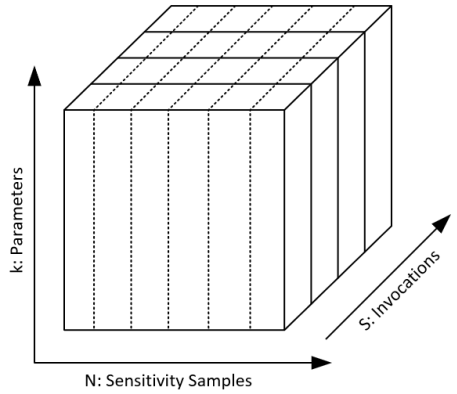

Figure 4.1.: Sketch of analysis input concept.

k: Analysis parameters. As described in Chapter 1.2.2, many more factors influence the result than just the parameters defined by a user. Moreover, those cannot be analyzed at the same time. Consequently, analyzing meaningful subsets of that input space is necessary. From now on, this space will be called *Analysis parameter space*. These parameters in that space must be independent of each other.

P: Sensitivity Samples. As indicated in the introduction to openPASS, the simulation framework and SCM represent a sort of Monte Carlo procedure. Therefore for N Monte Carlo samples, Saltelli's approach described in Chapter 2.2.1 requires

$$P = k * (N + 2) \tag{4.1}$$

samples and model evaluation at each of those samples [20]. The SALib sampling module generates the samples quasi-randomly using the Sobol' sequence.

S: Invocations. To evaluate an AD Algorithm / Functionality, it is necessary to run multiple simulations and average the results, look at edge cases, and more. Between those multiple runs, called *Invocations*, the random seed in openPASS is incremented by one. That means all the other input factors outside the sensitivity analysis parameters space can change. Changing boundary conditions lead to different results and, therefore, different sensitivity indices. That is why, another sampling dimension called *Invocations* must be introduced. In conclusion, each of the P samples will be simulated S times. Each new invocation will be executed by openPASS, which saves the overhead of setting up the simulations. Still, they do increase simulation time. To speed up the computation time, the execution of the simulations will happen in parallel.

$$Number \quad of \quad Simulations = S * P = S * k * (N + 2) \tag{4.2}$$

4.1.2. Considerations about Simulation Result Dimensions

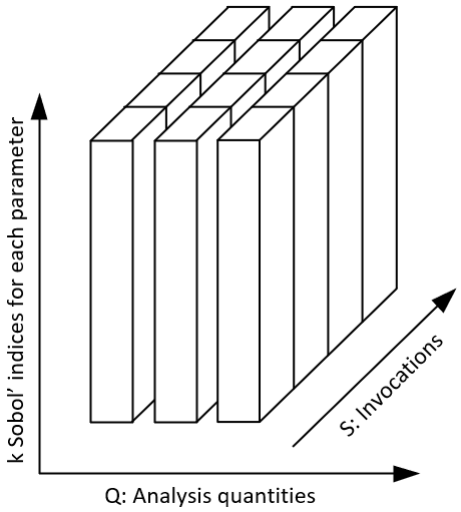

Figure 4.2.: Sketch of sensitivity index result structure.

Calculating Sobol' indices requires scalar results in the form $f(\mathbf{X}) = \mathbf{Y}$ like presented in Figure 2.1. OpenPASS results include various data about all agents in the simulation for each timestep, such as velocity, acceleration, agents in front, position, and many more. In short, openPASS results are not scalar values, but time dependent. Extending the model from Figure 2.1 would look like this:

$$f(\mathbf{X}, t) = Y(Q, t), \tag{4.3}$$

were t represents time and Q a quantity, such as velocity or acceleration. These results are from now on referred to as **time-dependent raw data, $\mathbf{Y}(\mathbf{Q}, \mathbf{t})$**.

Choosing a one-dimensional quantity from these time-dependent raw results is necessary to calculate Sobol' indices. This one-dimensional quantity can be directly extracted or calculated from the raw results and is called **analysis quantity, $\mathbf{Y_{q=q,t=t}}$**. Applying these considerations to Figure 4.1 leads to Figure 4.2: For each invocation slice, there will be the Sobol' indices for each analysis quantity $Y_{q=q,t=t}$.

User Information 1.1: Available Analysis Quantities

For now it is possible to extract data directly available in the result files for certain timesteps. It is possible to add more analysis functionality to the framework to analyze more quantities of interest, such as reaction times between when a driver sees an object until they react.

4.1.3. Kepping Determinism

Applying Saltelli's variance-based global sensitivity analysis approach and considering black box models requires sampling the scenario parameter space outside the openPASS/SCM platform. However, in the configuration files for openPASS, those parameters are defined as a distribution. This platform-specific setup leads to two challenges:

Algorithm 2: Calling mt19973 pseudo-random generator in openPASS

```
1 Function StochasticsImplementation::GetNormalDistributed(mean,
    stdDeviation):
2     if 0 > stdDeviation then
3         return mean
4     draw = normalDistribution(baseGenerator)
5     return stdDeviation * draw + mean
```

1. It is necessary to define the sample generated in the sensitivity framework as a scalar value in the openPASS config files. Setting the mean value to the scalar value and the standard deviation results effectively in a scalar parameter value.
2. However, this could influence the order-dependency of the random number generator. Therefore it is necessary to ensure that the random number generator is still called, even though it should not influence the parameter's value. Figure 2 shows the pseudocode of the implementation inside openPASS when a new sample is generated.

Conclusion

Looking at Figure 2 and assuming stdDeviation = 0, then the if condition in line 2 is false, so the baseGenerator is still called in line 4. Its multiplication with stdDeviation = 0 in line 5 eliminates the drawn value, such that the mean value is returned. Therefore, setting their standard deviation to zero turns parameters defined as distributions into scalar values. This strategy does neither affect the random generator's initialization nor order-dependency.

We can conclude that the determinism outside the analyzed parameter space is unaffected. Other distributions than the normal distribution apply the same logic.

4.2. Architecture and Components

This chapter presents the implementation of the analysis tool. Chapter 4.2.1, describes the inputs to the framework and the outputs it produces. More in-depth information about the framework's building blocks is explained in Chapter 4.2.2. Finally, Chapter 4.3 discusses how the framework can be adapted for different models, along with examining its limitations.

4.2.1. In- and Output

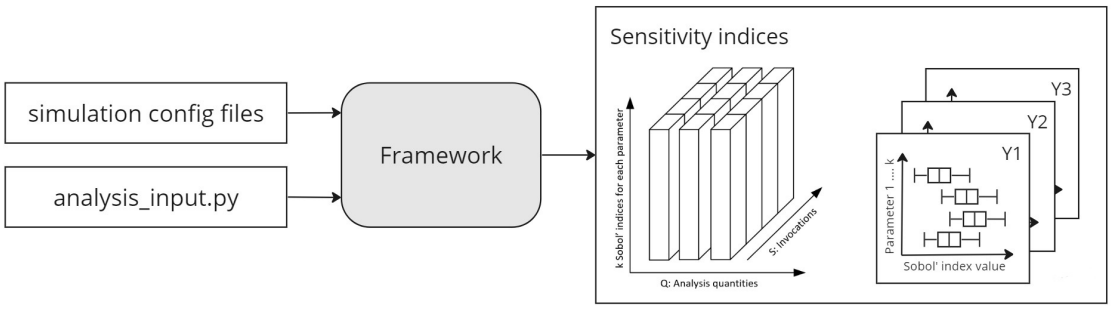

Figure 4.3.: Framework's in- and output.

Inputs

A) Simulation config files. The user needs to define the traffic simulation by providing the respective *simulation configuration files*. The simulation configuration files have to be compatible with the simulation software version defined in the *analysis_input.py.* file.

B) analysis_input.py This file defines the analysis to be performed. The following inputs need to be specified:

1. Number of samples and invocations
2. The paths to the simulation config files, the destination folder (which will be created by the framework and must not exist before the simulation)
3. A path to the simulation software
4. Each parameter must have a lower and an upper bound, and all parameters will be sampled by a uniform distribution. The parameter's name must match the name specified in the corresponding config file. Each parameter must have a mapping to its location in the *parameterParser.py* file.
5. Analysis quantities $Y_{q=q,t=t}$

Output

Based on this input information, the framework generates a folder containing the time-dependent raw data simulation results. There will be as many result files as function evaluations, which can be calculated according to Equation 4.2
As described in Chapter 4.1.2, calculating sensitivity indices requires scalar analysis quantities $Y_{q=q,t=t}$. Each of those quantities will have a designated folder containing a .csv file storing the first- and total-order Sobol' indices with the corresponding confidence intervals for each parameter defined in *analysis_input.py*. Each invocation will correspond to one line in this .csv file. Additionally, the framework generates a boxplot of SI.csv and stores it in the same result folder.

4.2.2. Building Blocks

The algorithm is mainly divided into a pre- and a postprocessing unit where the postprocessing unit needs the output of the preprocessor as an input. An executor module prepares the input and initializes the analysis. Figure 4.4 provides a visual overview of the building blocks explained in this section.

Preprocessor

The *Preprocessor* class generates simulation results necessary for calculating Sobol' indices based on user inputs. Therefore it needs to sample all parameters specified in *analysis_input.py* and calculate the simulations for each of those samples. The following modules explain this procedure more particularly:

Sampling. Once instantiated by the executor, the preprocessor reads the user input and generates samples: the samples are quasi-randomly generated based on Sobol' sequences by calling the SALib sample (salib.sample.sobol) module. Chapter 5.1 compares the convergence behavior of quasi-random and pseudo-random sampling. Using the external library requires translating the parameters specified in the *analysis_input.py* file into a format readable by SALib. The so-called *problem* is a dictionary generated upon instantiating a preprocessor object. The sampling function returns a Numpy array containing all the samples.

Generating configs. In the next step, the *generate_configs* module reads the user-defined simulation config files. It generates a new config folder for each sample by copying the original simulation config files and replacing the parameters to analyze with the respective sampled values. Parameters defined as distributions in openPASS will have a standard deviation of zero to not influence the baseGenerator as discussed in Chapter 4.1. The manipulation and copying for the new configuration files happen in *xml_util.py*, which originates from *pyOpenPASS*, a module in openPASS. This file needs the *parameterParser.py*, which translates the parameter's name to the file's name and its location for the simulation setup files.

Run the simulations. This function loops over all config files generated in the previous steps to start the simulations. This step can potentially generate data that is not necessarily needed. Therefore, as default, once the simulations finish, the config files are deleted, and

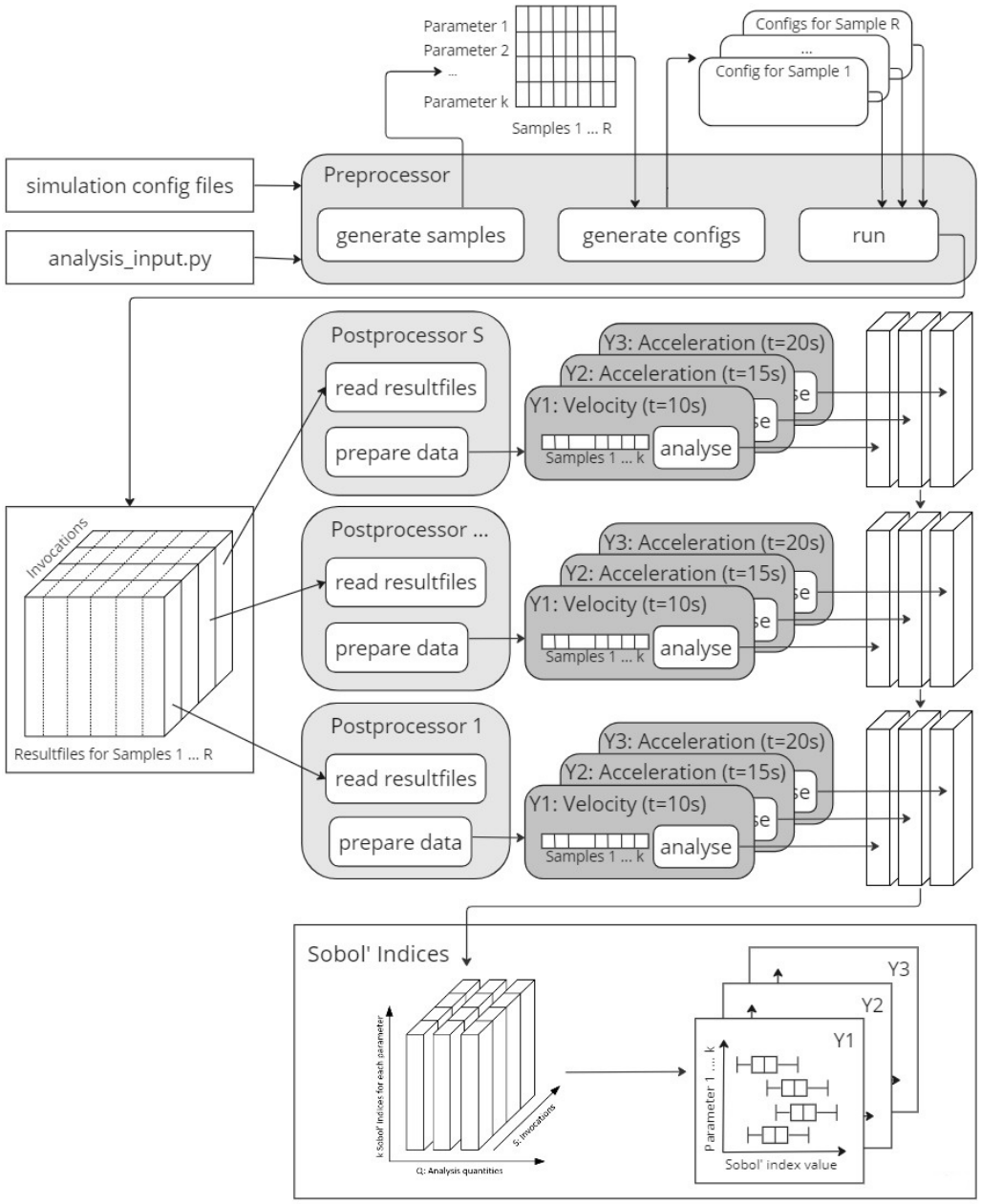

Figure 4.4.: Framework's building blocks.

from the result files only the *Cyclics.csv* files are kept[1]. The simulation process is happening in parallel, using Pythons concurrent.futures.ThreadPoolExecutor functionality. Chapter 5.3 presents how much speedup the parallelization can bring.

[1]Cyclics contain all temporal information about all agents involved in a simulation, such as e.g. velocity, acceleration, position, agents in front and much more

Postprocessor

As pointed out in Chapter 4.1.2, one invocation resembles one set of Sobol' indices. Therefore the executor instantiates one postprocessor for each invocation. The postprocessor class prepares the time-dependent raw simulation data from the preprocessor for further analysis. Out of this prepared data, the postprocessor instantiates a *result object* for each $Y_{q=q,t=t}$ specified by the user.

Analyzing results. The result objects then extract the necessary data from the postprocessor. Each object transforms its data into SALib-compatible format to call the *sobol.analyze* SALib functionality, which estimates the Sobol' indices for all scalar outputs. The *sobol.analyze* module approximates the Sobol' indices using Equations 2.11 for S_i and 2.12 for S_i^T. The executor generates a folder for each analysis quantity where the respective Sobol' indices are stored in a .csv file. There is further functionality that generates box plots out of these .csv files.

4.3. Further Development and Applicability of the Framework

Concluding Chapter 4, this section elaborates on the modularity of the framework and how it can be applied to models other than SCM. Therefore it also summarizes limitations that need to be kept in mind when workig with the framework and propositions for further development.

Adaptivity

Generally, the tool is implemented modularly, such that single components can be exchanged. For example, to use a different simulation model, the user needs to adapt the preprocessors sampling and running functions to work for the respective in- and output, as well as simulation executions requirement and the *read_resultfiles* function in the postprocessor. If a user wants to analyze a new parameter, then the *analysis_input.py* file can be extended with the respective value. Additionally, in *parameter_parser.py*, the location of that parameter in the config files must be linked. Additional analysis quantities $Y_{q=q,t=t}$ can be added by extending the results dictionary in *analysis_input.py*.

Propositions on further development

Sampling distributions. It is only possible to sample in a uniform distribution because SALib officially only supports standard normal and log-normal distributions. OpenPASS works with truncated distributions, which cannot directly be sampled in SALib. It is possible to adapt this by applying inverse transform sampling on the returned value of *generate_samples* within the preprocessor shown in Figure 4.4. More information is given in Chapter 2 of [32].

Scenarios. At the moment, scenarios must not include more than two agents, one SCM, and more than one lane. More complex scenarios, meaning having more lanes or SCMs, can result in a more complex simulation output, which is currently not supported by the

framework. This would affect three components: First of all the *generate_configs* function in the preprocessor. Secondly, if the user requires analysis of the additional components, the *analysis_input.py* must be extended by the additional parameters. If those modifications lead to a different simulation output structure, the *read_resultfiles* function in the postprocessor must be adapted accordingly as well.

Analysis Quantities. As of now, the user can analyze any result given directly in the simulation output file at any point of time, the TTC[2] and TGap[3] in case two agents are present and there is only one lane. Extending this with further statistics depending on other scenarios and average results over a specific time interval is possible. This can be integrated into the result objects functionalities depicted in Figure 4.4: the result object accesses the simulation data from the preprocessor. Adding a function that returns a one-dimensional quantity for each simulation can be used to generate result statistics.

Second-order parameter interactions. Sometimes it could be interesting to analyze the influence of the interaction of two specific parameters towards the output. In that case, the option *calculate_second_order* in the result objects analyze functionality can be set to True. This will double the function evaluations described in Equation 4.2. It will also require adding functionality to read and plot the additional second-order indices. An alternative way of investigating this is to analyze only two parameters. The total-order index will be the sum of the first and the second-order effect.

Polynomial Chaos Expansion. For the current simulation times, the computation time is acceptable. However, if performance becomes an issue for future work, the analysis could be performed with a few samples to identify important parameters. The next step would be to use polynomial chaos expansion to compute the Sobol' indices. In that case, one can use Python libraries like Chaospy. It is important to note, that this method is bound by the number of parameters to be analyzed and has a complexity of $O(2^k)$ described in Chapter 2.2.

[2]Time To Collision: time it would take for two agents to collide based on their current velocities
[3]Time Gap: time gap between two agents

5. Numerical Experiments on Convergence Rates

This chapter will establish an intuition for different convergence and computational speed improvements. Therefore, two comparisons between pseudo-random Monte Carlo sampling and quasi-random Monte Carlo sampling using Sobol' sequences are presented in this chapter: The first example is a general convergence comparison in applying both methods to the Ishigami function described in Chapter 3.1. The second example applies both sampling strategies to a speed limit scenario in openPASS, where one SCM drives on a single lane with a speed limit. Finally, the effect of parallelizing the simulation procedure, described in Chapter 4.2.2, is demonstrated on the same speed limit scenario.

5.1. Comparison: Quasi- and Pseudo-Random Sampling for Monte Carlo Estimations of Sobol' Indices for the Ishigami Function

In this section we compare the convergence of sensitivity indices based on the quasi-random sampling approach described in Section 2.2.3 to pseudo-random sampling. In this convergence study, we analyzed $M = [128, 1024, 8192, 131072, 524288]$[1] different numbers of samples.

Convergence measure

The ground truth, $\hat{Y}_i$, is the analytical solution to the Ishigami function described in Chapter 3.1. N is the number of samples, $f(X_i)$ the function evaluation at X_i with $i \in \{1, 2, \ldots, N\}$. The Mean Squared Error,

$$MSE = \frac{1}{n}\sum_{i=1}^{N}(Y_i) - \hat{f}(X_i))^2, \tag{5.1}$$

is going to measure the difference between the estimation and the ground truth.

Convergence behavior

Figure 5.1 shows that the quasi-random sampling approach improved the quasi-random sampling by approximately one order in convergence rate. One can also observe that both approaches converge one order higher than the theoretical worst case mentioned in Chapter 2.2.1, which is $O(\frac{1}{\sqrt{N}})$ for pseudo-random sampling and the quasi-random approach at $O(\frac{log(N)^k}{N})$.

[1] Sobol' sequences show the best effect when the number of samples is a multiple of 2

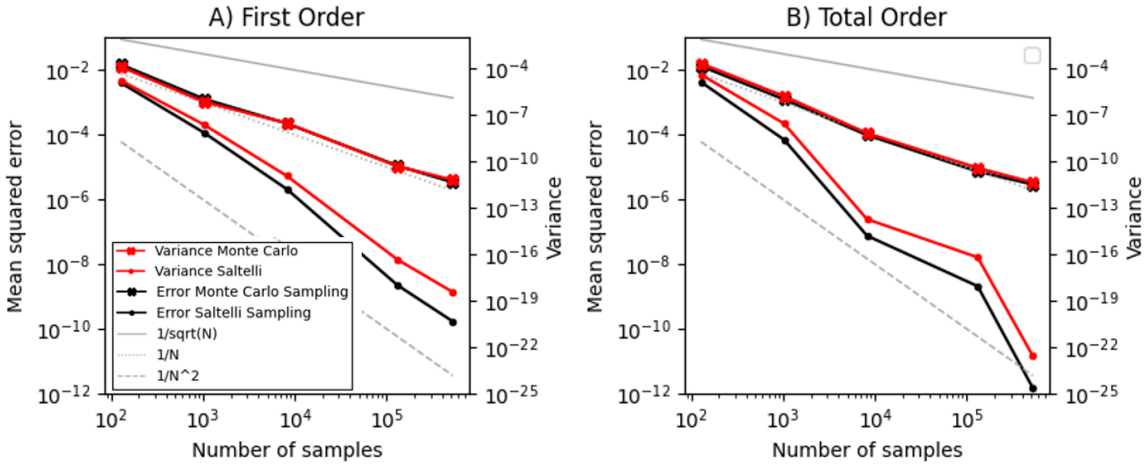

Figure 5.1.: Convergence rate for A) first-, and B) total-order Sobol' indices of Ishigami function.

Conclusion

Based on these observations, quasi-random sampling has the expected convergence improvement. So, the next chapter will examine whether this is true for an application to SCM.

5.2. Comparison: Quasi- and Pseudo-Random Sampling for Monte Carlo Estimations of Sobol' Indices for a Speed Limit Scenario with SCM

To see whether the quasi-random sampling approach also improves convergence for a traffic scenario utilizing SCM, both methods are applied to a speed limit scenario: one SCM is driving on a single lane where there is a speed regulation of 100 $\frac{km}{h}$. The analysis quantity is its velocity at a time in the simulation where the driver is not changing their velocity anymore. The same scenario is further analyzed in Chapter 6.2.

Convergence measure

In this case, there is no ground truth we could compare the resulting sensitivity indices to. Therefore we will look at the respective change between two results: the result of the approximation at $N = 128$ is compared to the result at $N = 1024$ and so on. The less this changes the higher the chance that the results have converged.

Convergence behavior

Figure 5.2 shows a similar trend as Figure 5.1: the quasi-random approach converges at least one order faster than the pseudo-random approach. In both cases, the total-order Sobol' indices converge faster than the first-order indices. Because of this significant improvement in the convergence rate, the framework will employ the quasi-random Monte Carlo approach.

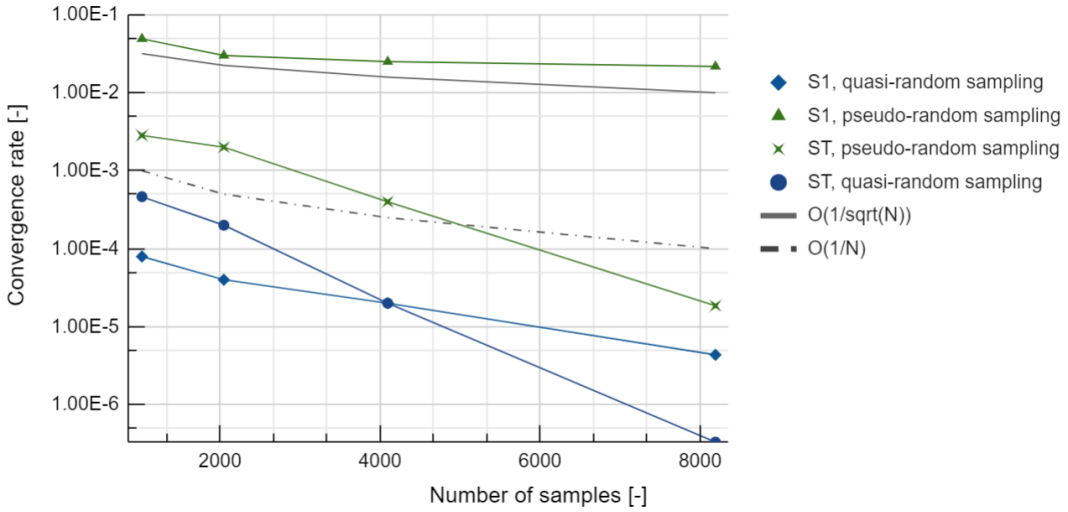

Figure 5.2.: Convergence rates for different sampling strategies applied to a speed limit scenario.

5.3. Parallelising the Execution of the Simulations

The computationally expensive part of the analysis is the execution of the simulations. The simulations have their own configuration files and do not share data dependencies. Therefore they can be executed in parallel. The simulations themselves are not taking a lot of computational effort. However, the amount of simulations makes this a computationally expensive task. That resembles an (input/output) I/O bound problem, which can be distributed and executed by multiple threads in Python, applying the *concurrent.futures* libraries module *concurrent.futures.ThreadPoolExecutor* [33].

Speedup behavior

Parallelising a computation requires the concurrent.futures library to distribute the tasks to different threads. This overhead increases with the number of threads used. That means that increasing the number of threads does not necessarily speed up the execution of the framework in a linear manner. According to Amdahl's Law [34], the theoretical maximum speedup depends on the fraction of the code that cannot be parallelized (s) and the number of threads available (T):

$$SP(s,T) = \frac{T}{(T-1) * s + 1} \tag{5.2}$$

In our case[2] $p = 0.95$. Looking at Figure 5.3 we can see that for the applied problem starting at 27 threads there is a stagnation of speedup at approximately 9.8. This is probably the case because with an increasing number of threads two things are happening:

[2]This is an approximation when looking at the fraction of time it takes the program to set up the simulations compared to running them

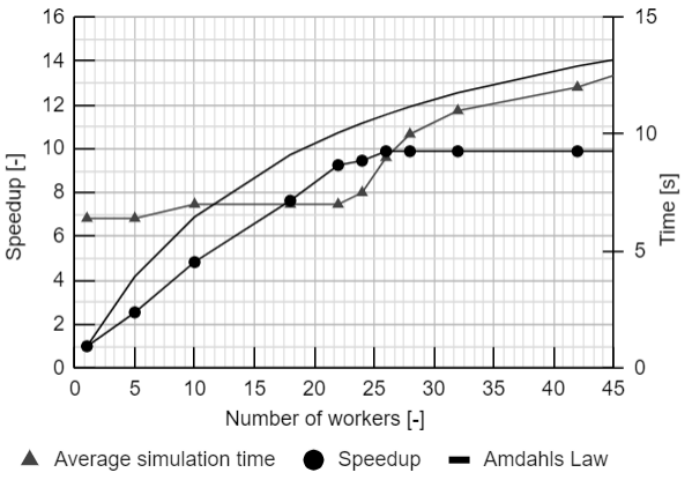

Figure 5.3.: Computation time for different numbers of threads for 32 samples, two parameters, and ten invocations.

1. As indicated before, the increasing number of threads causes an organizational overhead to the framework when distributing the tasks to the threads. At some point, the speedup gained does not make up for the overhead.
2. The more threads, the less computational resources each thread can access. We measured the time of a single simulation. There seems to be a correlation between the number of threads and simulation time: the more threads, the longer a single simulation takes. Looking at Figure 5.3, starting at 24 threads, the time for a single simulation increases significantly.

From these observations, 27 threads are sufficient to achieve the possible speedup for this specific application.

User Information 3.1: Choosing the number of threads prior to simulation

Users of the framework don't have to be worry about choosing a suitable number of threads; they can if necessary. The *concurrent.futures* library in Pyhton adjusts this based on the available threads on the operating computer. However, changing the number of threads in the preprocessor is possible in the *preprocessor.run()* function. If you want to change it, then consider how long a simulation takes and how many simulations there are: calculate the number of simulations according to Equation 4.1. The higher this number, the higher the number of threads. Also, consider the number of invocations: those will not be parallelized, so they increase the simulation time considerably.
A general recommendation is to run the analysis for fewer samples and parameters to see which number of threads gives the best speedup.

6. Exemplary Applications to SCM

This chapter checks the plausibility of the framework's analysis results and demonstrates its usage on the example of four traffic scenarios depicted in Figure 6.1:

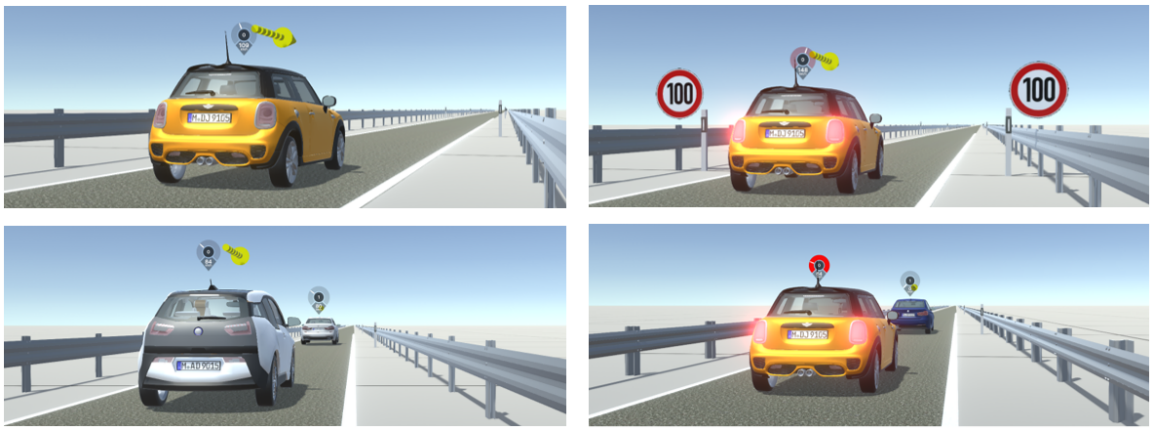

Figure 6.1.: Overview of all traffic simulations: Free driving (upper left), Speed limit (upper right), Approaching (lower left), Emergency brake (lower right).

1. Free driving. For a plausibility check of first-order Sobol' indices, an analysis is performed on the simple example of an SCM driving on one lane without any regulations.

2. Speed limit. The second application is for demonstrating the importance of total-order Sobol' indices as well as convergence behavior based on confidence intervals. This is done by adding a speed limit of 100 $\frac{km}{h}$ to the first example.

3. Approaching another car. Sensible Sobol' indices result - generally speaking - from sensible analysis setup. This is the main message from the third application. Here, another car will drive on the lane instead of a speed limit. There is an example with some of the samples generating blind drivers crashing into the other car and how this is reflected in the Sobol' indices.

4. Emergency brake. The fourth example shows the importance of choosing a suitable point in simulation time for analysis. At the example of a SCM approaching a standing vehicle, different Sobol' indices will be calculated and compared to each other.

How to read box plots in the following sections

The subsections of this chapter all follow a similar structure. They present box plots showing Sobol' indices, like depicted in Figure 6.2:

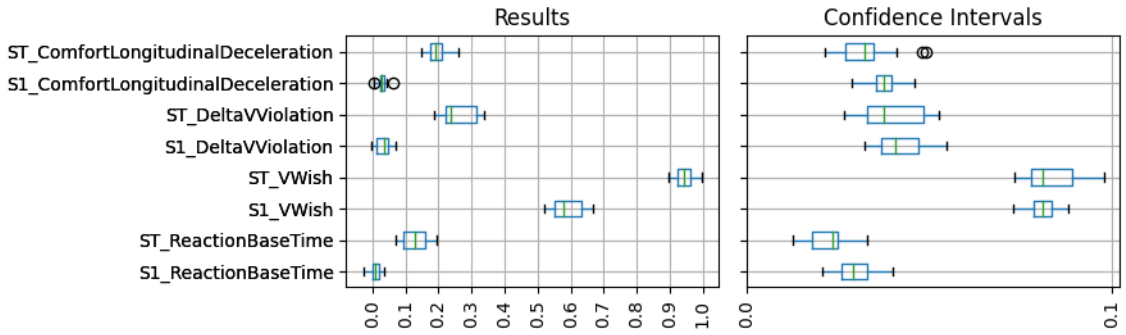

Figure 6.2.: Exemplary box plot showing Sobol' indices for the acceleration at 20 seconds in a scenario with a speed limit.

As described in Chapter 4.1.1, Sobol' indices are generated for each invocation. As the simulation input parameters outside the analysis parameter space change between different invocations, Sobol' indices also vary. The following paragraphs describe how to read the box plots:

Confidence intervals. Each Sobol' index will be shown in a box plot evaluated over all invocations accompanied by its confidence intervals. The indices are sampled with a confidence level of 95%. The confidence intervals do not measure the results' correctness but rather the calculation's convergence. High confidence intervals can resemble a hint that the number of samples is not yet sufficient for this problem. More elaboration and demonstration on this can be found in Chapters 6.2 and 6.3.

Displayed number of parameters. Most diagrams showing Sobol' indices are not displaying all parameters. For simplicity, only the non-zero ones are kept. Detailed result plots showing all analyzed parameters are summarized in the Appendix.

Negative first-order indices. Some results might show negative values very close to zero for first-order indices. These result from effects which stem from mathematical characteristics in the numerical method implemented in SALib. They can be avoided by increasing the number of samples. However, as long as the value is not far from zero and the confidence interval overlaps with zero, the respective values can be assumed to not have an influence [22]. Chapter 2.2.1 explains this numerical phenomenon mathematically.

Overview of analyzed parameters

The following enumeration lists all input parameters that will appear in the later discussions of this chapter. There are many more input parameters for SCM than these, but as they are not relevant for the remaining part of this thesis, they are not listed here.

1. **VWish (V_{Wish}):**
   The velocity the driver wants to drive without any other influences.
2. **DeltaVViolation ($\Delta V_{Violation}$):**
   The amount of velocity by which the driver is willing to violate the speed limit.
3. **ReactionBaseTime (T_{RB}):**
   The value of the driver's reaction time regarding rule-based behavior. [1]
4. **ComfortLongitudinalAcceleration:**
   The comfort longitudinal acceleration of the driver.
5. **ComfortLongitudinalDeceleration:**
   The comfort longitudinal deceleration of the driver.
6. **MaxLongitudinalAcceleration:**
   The maximum longitudinal acceleration of the driver.
7. **MaxLongitudinalDeceleration:**
   The maximum longitudinal deceleration of the driver.
8. **CarQueuingDistance :**
   The distance the driver tries to maintain to its leading vehicle at standstill.
9. **ProportionalityFactorForFollowingDistance:** The proportionality factor for the reduction of following distance at higher speeds (Steven's power law).
10. **PreviewDistance:**
    The distance around the driver's vehicle, which the driver considers relevant. Limited by the visibility distance (upper limit) and **PreviewDistanceMin (lower limit)**.
11. **PreviewTimeHeadway ($T_{PrevHead}$):**
    Time in seconds a driver can see ahead. In the simulation this will translate to a distance, depending on the driver's velocity.

[1] There are also other kinds of reaction times in SCM/openPASS, for example, a skill-based reaction time and a pedal change time. However, these are not modifiable by the user.

6.1. Free Driving

Figure 6.3.: Free driving scenario without speed limit.

This scenario serves as a test scenario to verify if the sensitivity analysis framework delivers reasonable results. Figure 6.3 shows the only agent (which is controlled by SCM) driving on one available lane without any velocity regulations.

N	256
S	20
k	28
$\mathbf{V_{Start}}$	171 $\frac{km}{h}$
$\mathbf{V_{Wish}}$	$\mathcal{U}(80, 160)\frac{km}{h}$

Table 6.1.: Analysis setup for free driving scenario.

6.1.1. Temporal Velocity Development in the Free Driving Scenario

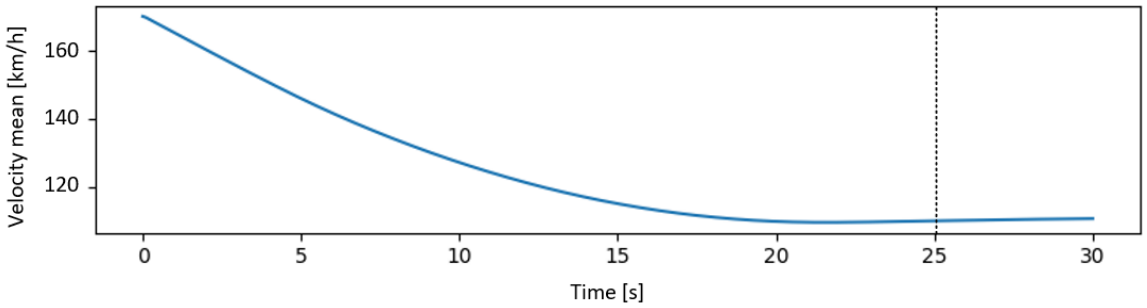

Figure 6.4.: Temporal development of average velocity over all simulations for the free driving scenario.

Figure 6.4 shows the average velocity over time for all simulations performed for the analysis. In the beginning, all vehicles have the same spawning velocity[2] of 171 $\frac{km}{h}$. Depending on their comfort deceleration rates and reaction times they will all end up at their V_{Wish}: this is evident from Figure 6.4 as around 20 seconds the mean velocity does not change anymore and it stabilizes around 120 $\frac{km}{h}$, which is the mean value of a uniform distribution in the interval [80, 160] $\frac{km}{h}$. This is why the time for the velocity at which the Sobol' indices are calculated is set to 25 seconds.

6.1.2. Plausibility Check For First Order Effects on the Example of Free Driving

It is to be expected that at the time when the vehicle reaches its target speed, the Sobol' indices (both first-order and total-order) should be at approximately 1 for V_{Wish} and no other parameter should have an influence, as there are no other restricting factors in this simulation. If the results correspond to this expectation, that will be considered a passed plausibility check.

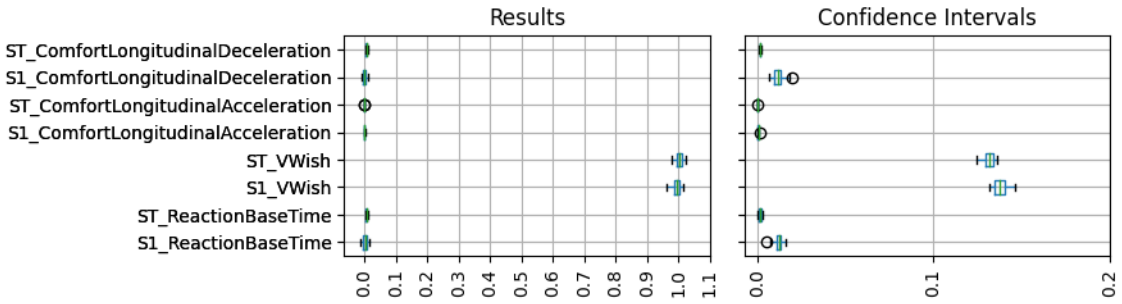

Figure 6.5.: Sobol' indices for velocity at 25 seconds.

Figure 6.5 shows that only V_{Wish} has an impact on the target velocity. $N = 256$ is a relatively low sample size. For such a small sample size, the confidence intervals are quite low. Therefore this analysis suffices for this simple case to prove the point of plausibility. However, running analysis at a much higher number of samples is essential for reliable results that serve as the basis for decision-making.

[2]Spawning velocity is the velocity a car has upon initialization into the simulated environment.

6.2. Speed Limit

Figure 6.6.: Speed limit scenario at 100 km/h.

The speed limit scenario is based on the free driving scenario with the addition of a general speed limit of 100 $\frac{km}{h}$. Its purpose is to showcase the importance of calculating total-order Sobol' indices on the example of the interaction between V_{Wish} and $\Delta V_{Violation}$. Another point of this section is to demonstrate the convergence behavior of the analysis results depending on the number of samples used. The two following setups are compared:

Analysis	**1**	**2**
N	512	1024
S	20	20
k	28	4
$\mathbf{V_{Start}}[\frac{km}{h}]$	171	171
$\mathbf{V_{Wish}}[\frac{km}{h}]$	$\mathcal{U}(80, 160)$	$\mathcal{U}(80, 160)$
$\mathbf{\Delta V_{Violation}}[\frac{km}{h}]$	$\mathcal{U}(0, 36)$	$\mathcal{U}(0, 36)$

Table 6.2.: Analysis setup for speed limit scenario.

6.2.1. Temporal Velocity Development in the Speed Limit Scenario

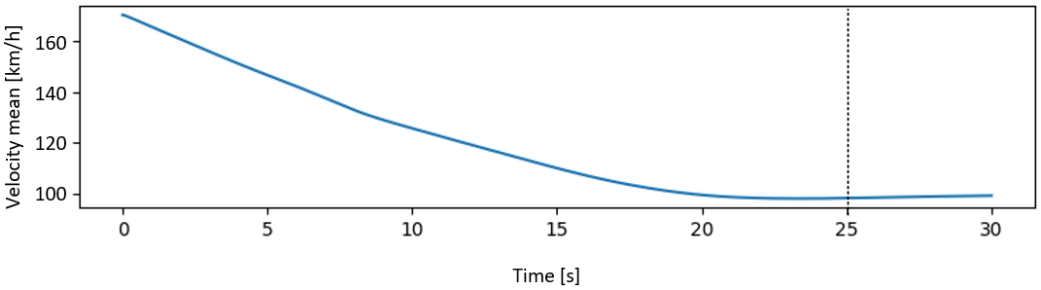

Figure 6.7.: Averaged velocity development for speed limit scenario.

The velocity development visible in Figure 6.7 is very similar to Figure 6.4. The only difference is that it converges towards a lower velocity, approximately 108 $\frac{km}{h}$. This is a

reasonable result considering that the speed limit is 100 $\frac{km}{h}$, the average comfort velocity 120 $\frac{km}{h}$, and the average violation of the speed limit is 18 $\frac{km}{h}$. As in the previous example we will analyze the converged velocity, which is the case at 25 seconds.

6.2.2. Plausibility Check for Total Sobol' indices

As indicated in the paragraph above, it is to be expected that both V_{Wish} and $\Delta V_{Violation}$ are going to have an impact on the final velocity of the driver. They should also have a shared influence: a driver with $V_{Wish} = 120$ $\frac{km}{h}$ and $\Delta V_{Violation} = 20$ $\frac{km}{h}$ is expected to have a final velocity around 120 $\frac{km}{h}$, while a driver with the same V_{Wish} but a zero-$\Delta V_{Violation}$ should have a target velocity of 100 $\frac{km}{h}$.

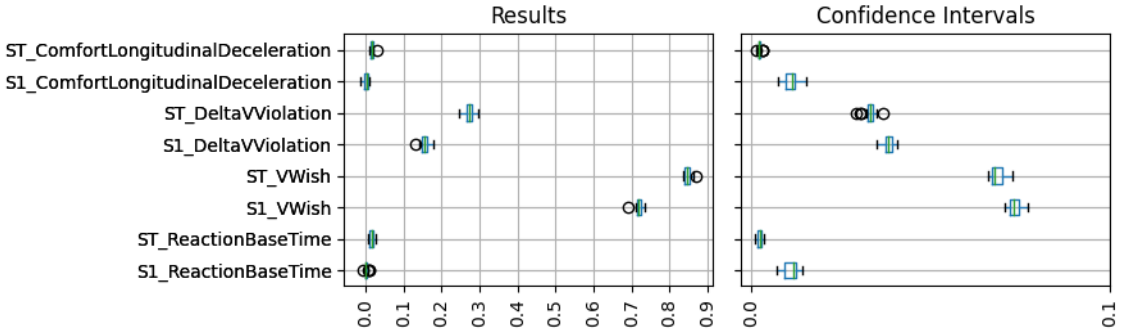

Figure 6.8.: Sobol' indices for the velocity at 25 seconds in speed limit scenario, analysis 1.

The Sobol' indices for this scenario depicted in Figure 6.8 fulfill the expectations explained in the paragraph above: one can see that there is a considerable interaction effect between V_{Wish} and $\Delta V_{Violation}$: in both cases the total-order Sobol' index is approximately 10% higher than the first-order Sobol' index. Recalling the definition of the total-order Sobol' indices given in Equation 2.5, which says that the total-order Sobol' index is the sum of the first- and all higher-order indices of that parameter. As no other parameter shows considerable influence[3], the total-order indices are in this case the first-order index plus the second-order effect between V_{Wish} and $\Delta V_{Violation}$. Therefore, the interaction between these two must have an influence of around 10%. A higher average of $\Delta V_{Violation}$ will probably lead to an even higher interaction. This effect underlines the importance of the calculation of the total-order Sobol' indices for nonlinear models: looking at $\Delta V_{Violation}$, the median first-order effect is approximately 17%, while the total effect is around 27%. Neglecting the total-order effect would mean an error of more than 30%.

6.2.3. Decreasing Confidence Intervals

First of all, a word of warning: a small confidence interval does not justify the correctness of the results but it does measure the tolerance of the resulting value if one would redo this example using other samples. Therefore, it gives a good orientation as to whether the

[3]In these analyses, an influence below 1% will be considered negligible.

analysis has converged. One way to reduce the confidence intervals is by increasing the number of samples. Therefore, two identical speed limit scenarios are analyzed with one difference: Analysis 1 is simulated using $N = 512$ and 2 with $N = 1024$. Following the logic described at the beginning of this paragraph, Analysis 1 should show higher confidence intervals than Analysis 2.

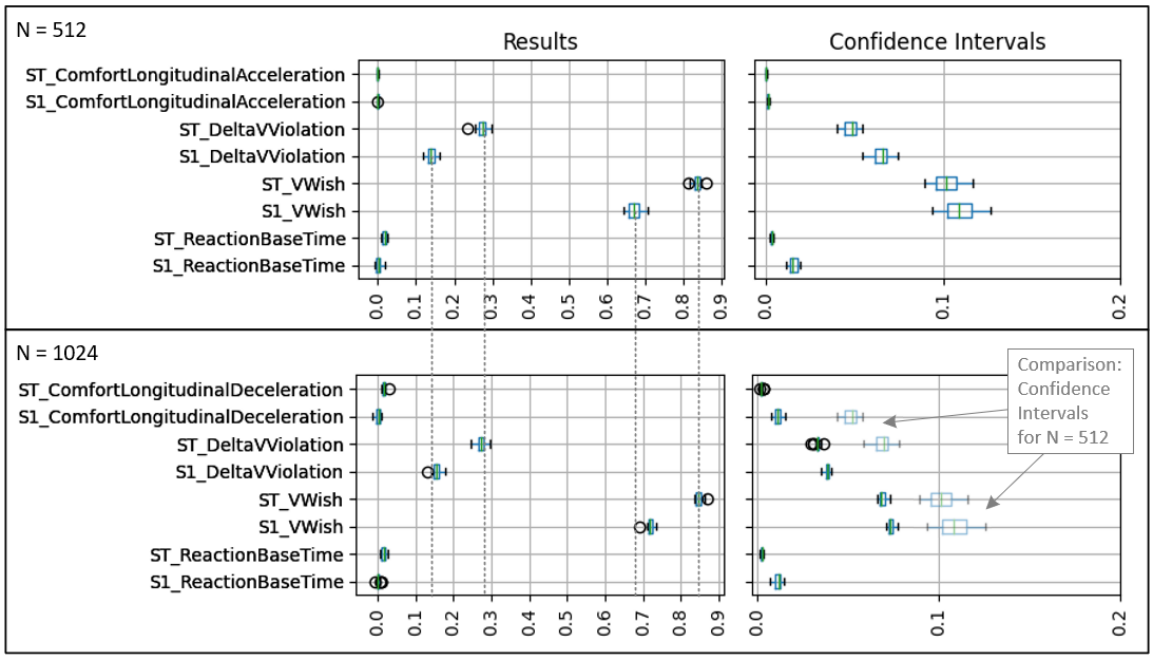

Figure 6.9.: Sobol' indices for speed limit Analysis 1 (upper) and Analysis 2 (lower).

Figure 6.9 shows the Sobol' indices with their confidence intervals for Analysis 1 and 2. As expected, the confidence intervals decreased. Another observation is that the order of magnitude of the Sobol' indices did not change much. However, it is recommended to generate confidence intervals of at most 0.05 for reliable results.

User Information 2.1: Choosing the number of threads prior to simulation

We have restrictions that make increasing the sample size difficult, i.e., computationally costly: firstly, N must be a power of two. Secondly, the computation time grows proportionally with the number of samples. Looking at Equation 4.2, the number of simulations $(S * k * (N + 2))$ decreases linearly with a decreasing number of parameters. Therefore, a recommended procedure is to analyze the simulation at a low number of samples, such as $N = 256$, to identify unimportant parameters to exclude from the analysis. In the next step, one removes all the negligible parameters and increases the number of samples until the largest confidence interval is below the desired tolerance. That is what happened here: Analysis 1 included all 28 parameters, which means Analysis 1 requires $20 * 28 * 514 = 287840$ simulations. Analysis 2 included four parameters but double the number of samples, leading to $20 * 4 * 1026 = 82080$ simulations.

6.3. Approaching and Following Another Agent

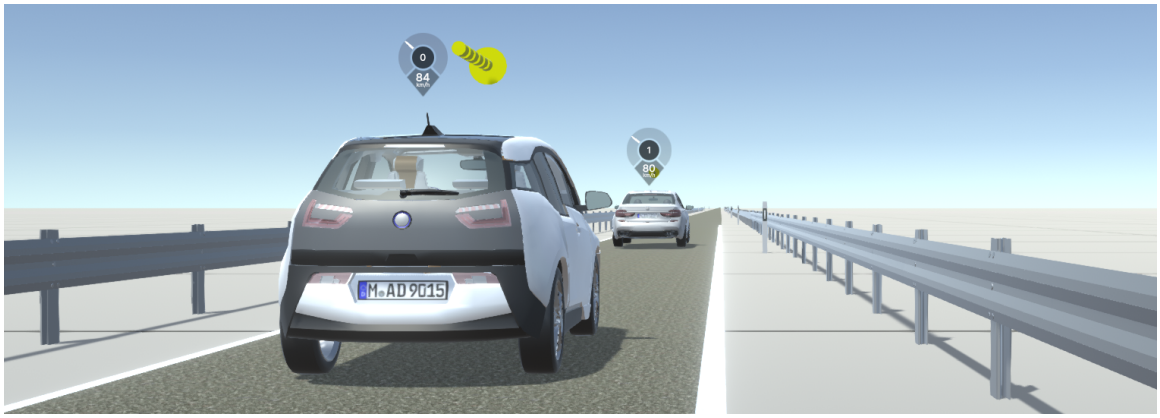

Figure 6.10.: Visualisation of approaching scenario.

The approaching scenario represents another extension to the scenarios before: instead of a speed limit, there is another vehicle driving in the environment at 80 $\frac{km}{h}$. SCM will approach this vehicle and decelerate to follow this other driver. With the approaching scenario, we showcase possible reasons for observing very different Sobol' indices between different invocations. In this example, $T_{PrevHead}$[4] was set to have a lower bound of 0 s, which corresponds to a driver that cannot see ahead. A blind driver cannot see what is ahead and will most likely crash the predecessor. Analyzing the velocity after those potential crashes shows how different the Sobol' indices are, depending on whether it crashed.

Analysis	**1**	**2**	**3**
N	512	2048	512
S	20	20	20
k	28	5	5
$\mathbf{V_{Start}}[\frac{km}{h}]$	171	171	171
$\mathbf{V_{Wish}}[\frac{km}{h}]$	$\mathcal{U}(80, 160)$	$\mathcal{U}(80, 160)$	$\mathcal{U}(80, 160)$
$\mathbf{T_{PrevHead}}[s]$	$\mathcal{U}(0, 15)$	$\mathcal{U}(0, 15)$	$\mathcal{U}(10, 20)$

Table 6.3.: Analysis setup for approaching scenario.

6.3.1. Temporal Velocity Development in the Approaching Scenario

The velocity development visible in Figure 6.11 is again comparable to Figures 6.4 and 6.7. For this scenario, it converges towards approximately 80 $\frac{km}{h}$, which is the velocity of the vehicle in front of SCM. As in the previous example, we will analyze the converged velocity, which is the case at 20 seconds.

[4] $T_{PrevHead}PreviewTimeHeadway$ describes how many seconds a driver can look ahead. Depending on the vehicle velocity, this will be translated into a distance during the simulation.

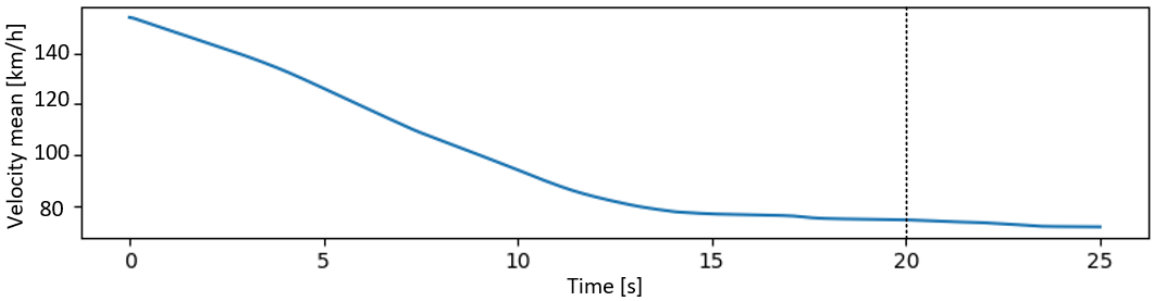

Figure 6.11.: Temporal development of average velocity over all simulations for approaching scenario in Analysis 3.

6.3.2. On Widely Spread Sobol' Indices and Confidence Intervals

Meaningful Sensitivity Input Distributions

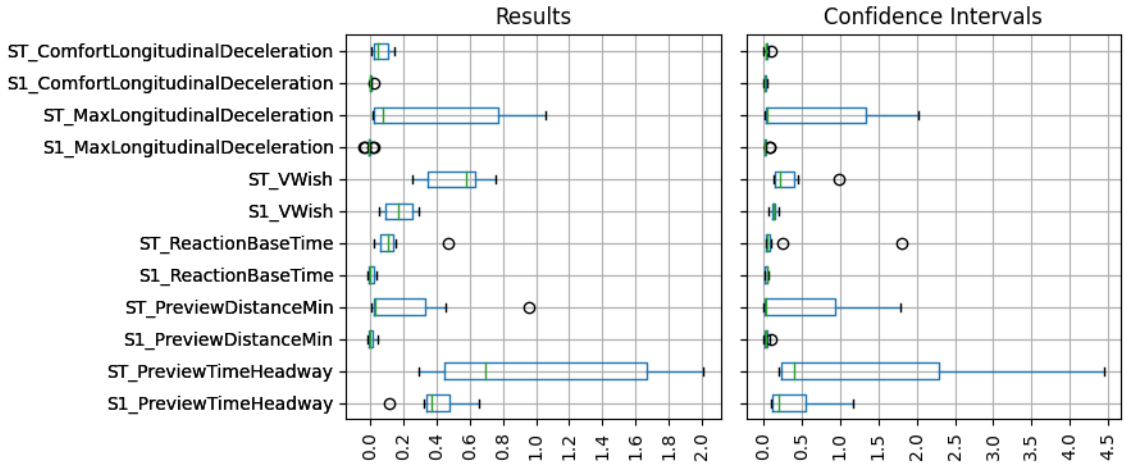

Figure 6.12.: Sobol' indices for the velocity at 20 seconds for Analysis 1 (N=512, I=20, THW$[0, 15]$) in approaching scenario.

Vehicles that crash will not move at the end of the simulation, so their velocity will be zero. Other drivers with a non-zero $T_{PrevHead}$ will slow down and not crash the vehicle in front. They are going to have a velocity of approximately 80 $\frac{km}{h}$ at 20 seconds of simulation time. Depending on what is the case, the influence of $T_{PrevHead}$ is going to be very different, which is evident in Figure 6.12. The same is true for the driver's maximal deceleration: here, one can see the interaction between the $T_{PrevHead}$ and the maximal deceleration. It makes sense that the smaller the distance a driver can see, the later they will start braking and the stronger they need to decelerate to not crash the car in front. However, the confidence intervals are very large, making their interpretation unreliable. This would be a hint to increase the number of samples. Analysis 2 was performed with four times the amount of samples.

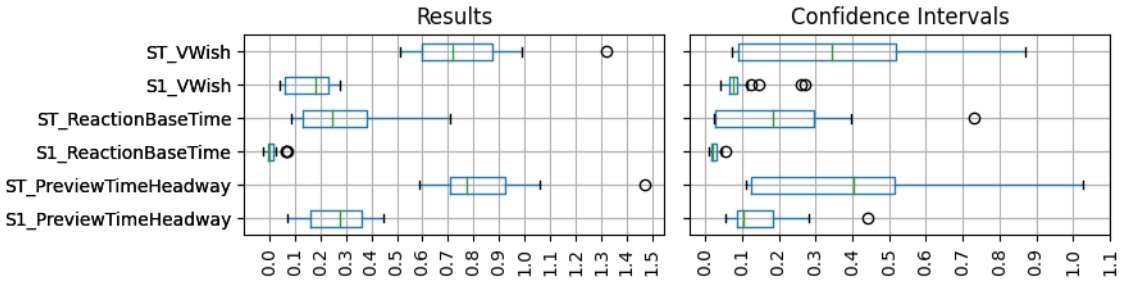

Figure 6.13.: Sobol' indices for velocity at 20 seconds for Analysis 2 (N=2048, I=20, THW=$[0, 15]$).

Figure 6.13 shows approximately a quarter of the confidence intervals compared to Figure 6.12. However, the confidence intervals are still more than ten times too large to be reliable. One can also observe significant differences in the sensitivity indices, especially when comparing the total-order index for $T_{PrevHead}$.

This was an unreasonable simulation case for demonstration purposes. We would not analyze SCM without the ability to look ahead. Therefore, let's check whether a reasonable setup leads to lower confidence intervals and thinner boxes in the sensitivity indices box plots: we therefore perform Analysis 3, which has the same setup as analysis one but more common bounds for $T_{PrevHead}$:

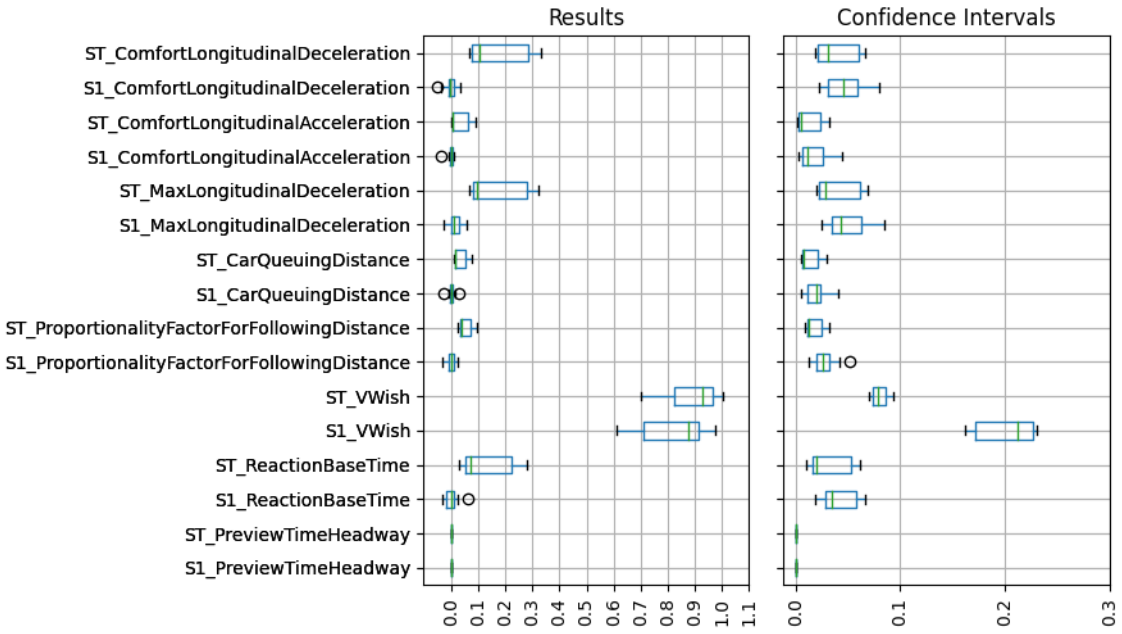

Figure 6.14.: Sobol' indices for the velocity at 20 seconds for Analysis 3 (N=512, I=20, THW$[10, 20]$) in approaching scenario.

Figure 6.14 shows reduced confidence intervals and much thinner ranges for the sensitivity indices. Also, $T_{PrevHead}$ does not influence this analysis. This confirms that the small values for $T_{PrevHead}$ were causing the large confidence intervals, and it also underlines the importance of a correct simulation setup.

User Information 3.1: Think about wether the setup makes sense

This example shows that the setup of the simulation to analyze matters a lot. Encountering such situations should lead the user to consider where such behavior stems from. Of course, this could also stem from something that needs to be analyzed. This analysis could have been split into two parts: one with only critically small distances and one sufficiently large to avoid collisions. In such cases, the user must be aware to increase the sample size until the confidence intervals are reasonable (We recommend a value of approximately 0.05).
Another reason could also stem from side effects of other parameters that might not be of interest. This problem can be solved by filtering the analysis quantities.

6.4. Emergency Brake

Figure 6.15.: Visualisation of emergency brake scenario.

The emergency brake scenario underlines the importance of predefining a sensible analysis quantity and simulation time. In this scenario, SCM is again driving on a single lane and then encounters a standing vehicle on that lane. Some drivers will not react early enough and crash into the vehicle in front, and others will decelerate early enough to avoid the crash.

Analysis	**1**	**2**
N	256	1024
S	40	40
k	28	7
$\mathbf{V_{Start}}[\frac{km}{h}]$	171	171
$\mathbf{V_{Wish}}[\frac{km}{h}]$	$\mathcal{U}(80, 160)$	$\mathcal{U}(80, 160)$
$\mathbf{T_{RB}}[s]$	$\mathcal{U}(0.4, 0.8)$	$\mathcal{U}(0.4, 0.8)$
CarQueuingDistance[m]	$\mathcal{U}(2, 4)$	$\mathcal{U}(2, 4)$
ComfortLongitudinalDeceleration$[\frac{\mathbf{m}}{\mathbf{s^2}}]$	$\mathcal{U}(1.012, 1.185)$	$\mathcal{U}(1.012, 1.185)$
MaxLongitudinalDeceleration$[\frac{\mathbf{m}}{\mathbf{s^2}}]$	$\mathcal{U}(6, 8)$	$\mathcal{U}(6, 8)$

Table 6.4.: Analysis setup for emergency brake scenario.

6.4.1. Temporal Velocity Development in the Emergency Brake Scenario

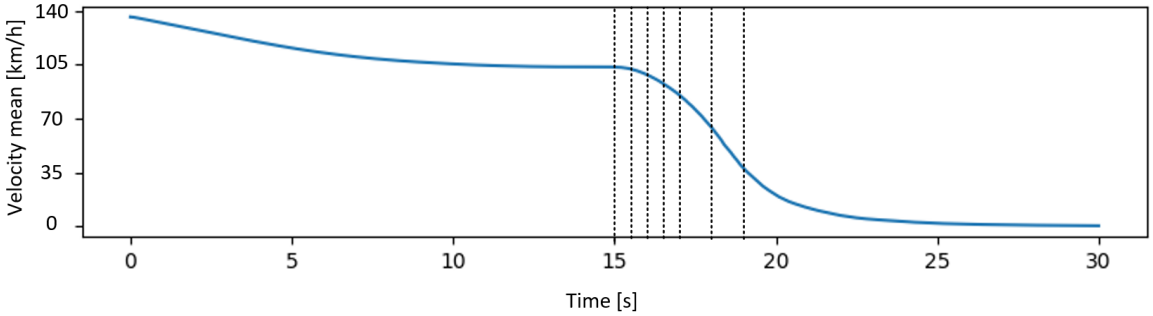

Figure 6.16.: Temporal development of average velocity over all simulations for emergency brake scenario.

At the beginning, SCM enters the simulation at a velocity of 171 $\frac{km}{h}$, then the drivers decelerate until the comfort velocity is reached. Depending on when the standing vehicle is recognized, SCM starts further braking. The Sobol' indices are compared between the normal driving and the braking phases to analyze the different influences of parameters at different times. Looking at the temporal velocity development, most drivers are at their comfortable driving velocity after around 12 seconds and start braking just after 15 seconds. Therefore, the velocity at 15, 15.5, 16, 16.5, 17, 18 and 19 seconds will be analyzed.

6.4.2. Variability of Results based on Different Analysis Times and Measures

Figure 6.17 shows the mean value of the total-order Sobol' indices for the velocity of SCM at different simulation times before the vehicle is standing. Most of the corresponding confidence intervals are below 0.1. The corresponding box plots can be found in the attachment.

The results at 15 seconds are comparable to the free driving scenario: V_{Wish} has an influence of almost one, and all others are relatively small. Starting at 15.5 seconds until 17 seconds the index of V_{Wish} becomes smaller, while all other parameters influences rise: the vehicles

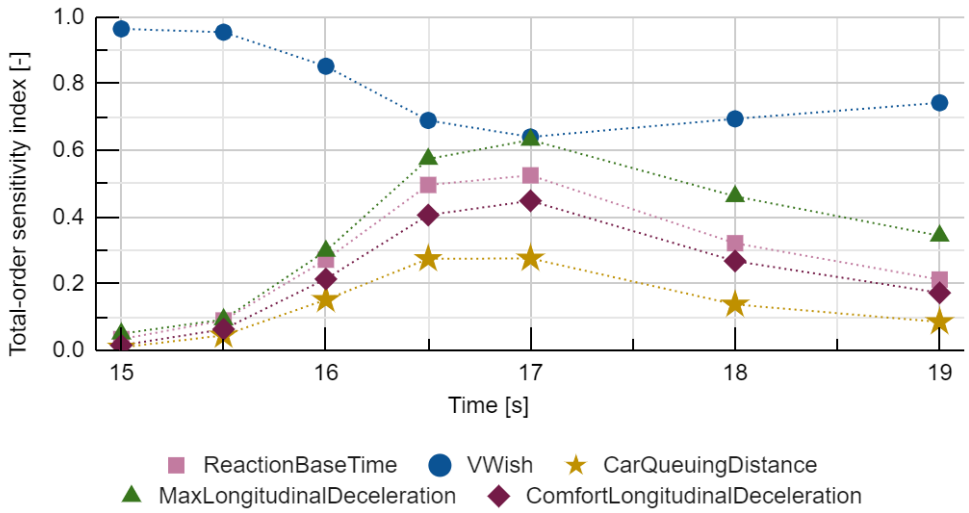

Figure 6.17.: Mean over 40 invocations of total-order Sobol' indices at different times of emergency braking scenario.

brake and reduce their velocity because they see the standing vehicle. However, they start braking sooner or later depending on the reaction time and comfortable distance from other cars. Their comfort- and maximal deceleration rates define how much they can decelerate, which also seems to significantly influence the velocity at those times. Looking at the simulation results, after 17 seconds, some vehicles are standing, and some have not yet approached the standing vehicle, which does not allow for a sensible interpretation of the indices after 17 seconds. They are still depicted here, as this was not evident from Figure 6.16. As there is a peak around 17 seconds, let's have a more detailed look at the box plots for that time: Figure 6.18 underlines once again the importance of calculating the total-order indices: the first-order indices for Reaction Base time, Car Queuing Distance, Comfort longitudinal deceleration are comparably low, if not negligible, while all of their total indices are greater than 30%, most of them even 50%. Calculating only the first-order indices would not have revealed their importance.

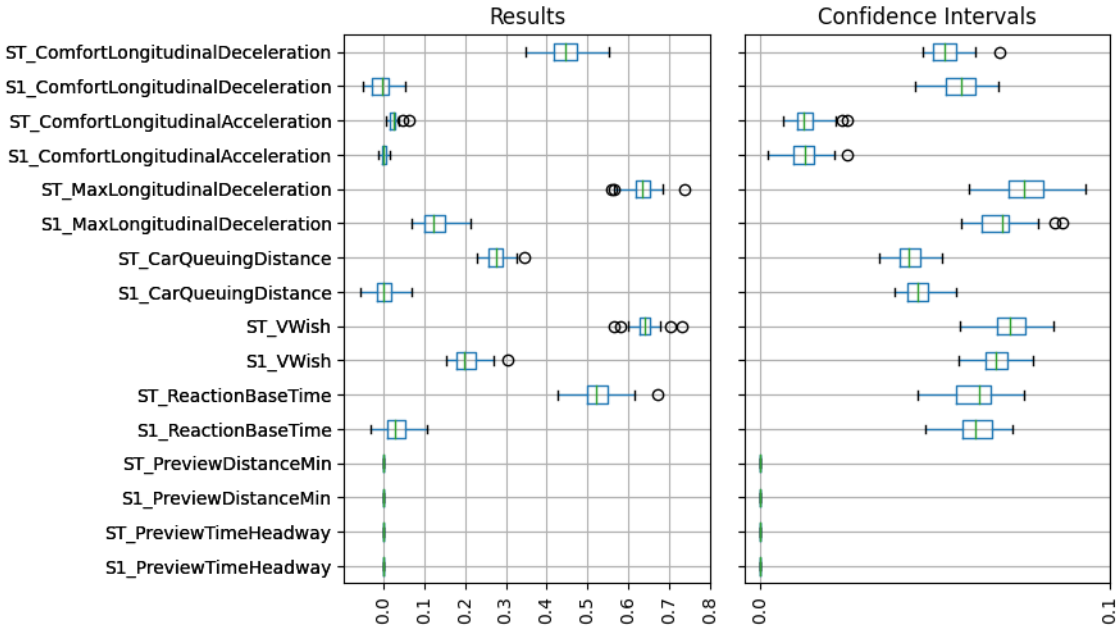

Figure 6.18.: Box plots of Sobol' index estimations in emergency brake scenario for velocity at 17 seconds.

Part IV.

Conclusion and Outlook

Throughout this thesis, a framework for conducting sensitivity analysis on stochastic models in traffic simulations has been developed. This framework is tailored to analyze stochastic black box models within, e.g., openPASS traffic simulations. The framework uses first- and total-order Sobol' indices, a global variance-based approach, to measure sensitivity. However, this methodology had to be extended in this work to account for the stochastic aspects of the models, which is described in Chapters 1.3 and 4.1.1. Comparing the most common methods for conducting sensitivity analysis, summarised in Chapter 2, gobal variance-based sensitivity analysis, especially total-order Sobol' indices, seemed most suitable for the problem to be solved in this work: Assuming strong nonlinearity and higher-order parameter interactions, any method for linear models detecting only first-order effect does not serve the purpose. That is why the relatively expensive computation of total-order Sobol' indices was considered necessary. To prove this assumption, show plausibility, and demonstrate the framework's usage, it was applied to different scenarios with SCM in openPASS. The sensitivity indices for those exemplary applications of SCM (see Chapter 6) show that total-order indices can - depending on the scenario - be very different from first-order effects. This underlines the strong nonlinearity of SCM and how important it is to consider parameter interactions: in the case of the emergency brake, the reaction base time's first-order effect is almost zero, while its total-order sensitivity index was in the range of 0.45 to 0.65 for the velocity at 17 seconds (see Figure 6.18).

Another conclusion from Chapter 6 is the importance of meaningful analysis quantities and sensible input ranges. The Sobol' indices in Figure 6.12 show how unrealistic values for an exemplary parameter can lead to very big ranges in the sensitivity results. Therefore, it must be emphasized that the first step of conducting sensitivity analysis is to set a goal and predefined questions the analysis should answer.

One can also conclude that both convergence and efficiency improvements can lower the computation time significantly. Chapter 5.1 shows that quasi-random sampling for Monte Carlo improved the convergence rate of pseudo-random sampling by one order, which matches with descriptions in, e.g., [20], [18], and [16]. For the example presented in Chapter 5.3 - an analysis with 32 samples, two parameters, and ten invocations - a speedup factor of almost ten was observed.

Propositions on further development of the framework

The current state of the framework enables sensitivity analysis of quantities at specific simulation times. The implemented methodology of Sobol' indices, especially of total-order, enables various possibilities of further development. This thesis focused on implementing the framework, researching the possible sensitivity analysis methods and applying them to stochastic models. The next step is to build up on this basis and extend the software to support more complex and thorough analysis. Chapter 4.3 gives an overview over these extensions, which in short are the following:

- Enabling further **analysis quantities**, which can help understand certain features of simulation models better.

- Enabling additional **analysis input distributions** other than non-uniform distributions, as there are, e.g., also normal and log-normal distributions in the original simulation configurations. This can help the user to analyze the input space more accurately.
- The calculation of the **influence interactions between two specific parameters** towards the output gives further insights into which parameters influence each other.
- Using **Polynomial Chaos Expansion** to compute the Sobol' indices can reduce computation time while improving the accuracy of the results in cases where there is fewer than five parameters to analyse.

However, depending on the users needs there are numerous other possibilities on what could be sensible additions to the framework.

Part V.

Appendix

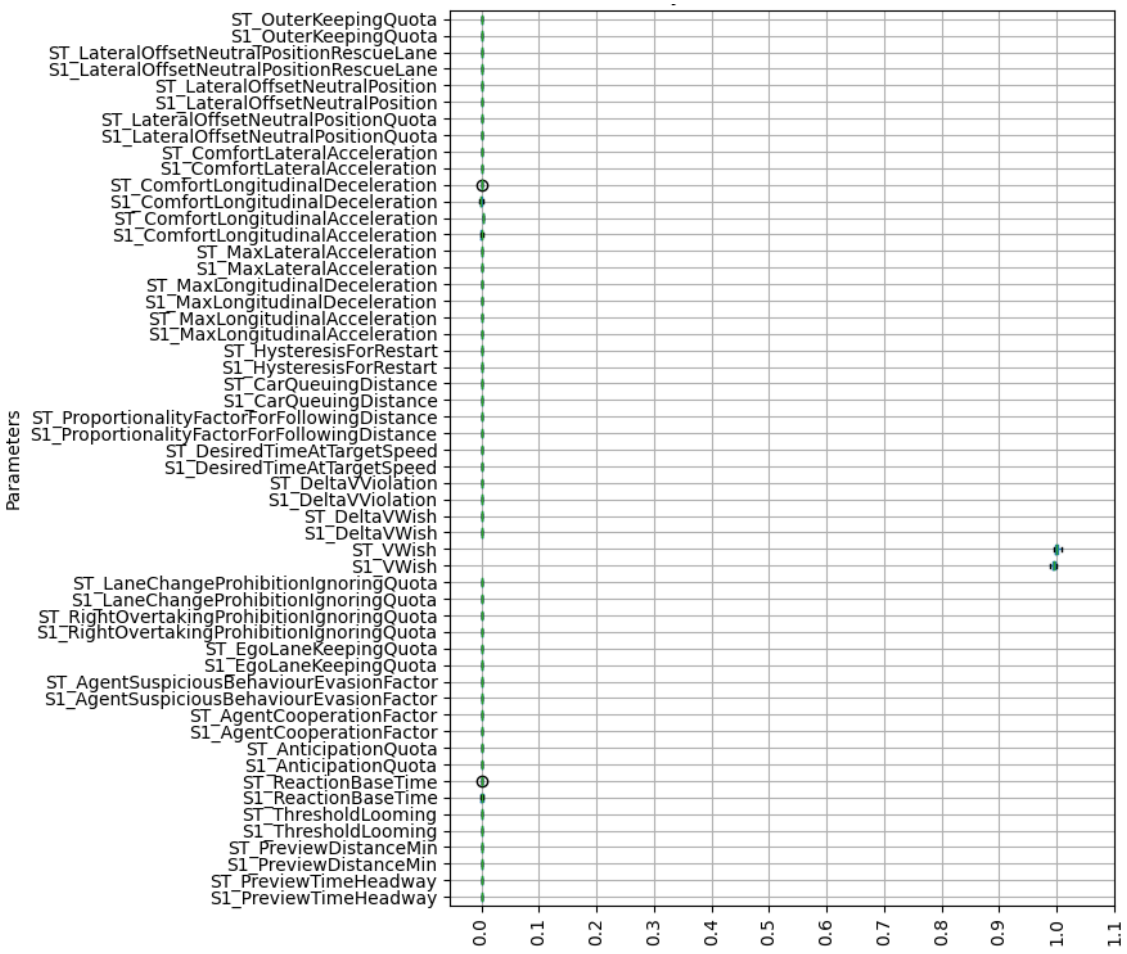

Figure .19.: Complete box plots on sensitivity indices in free driving scenario at 25 seconds.

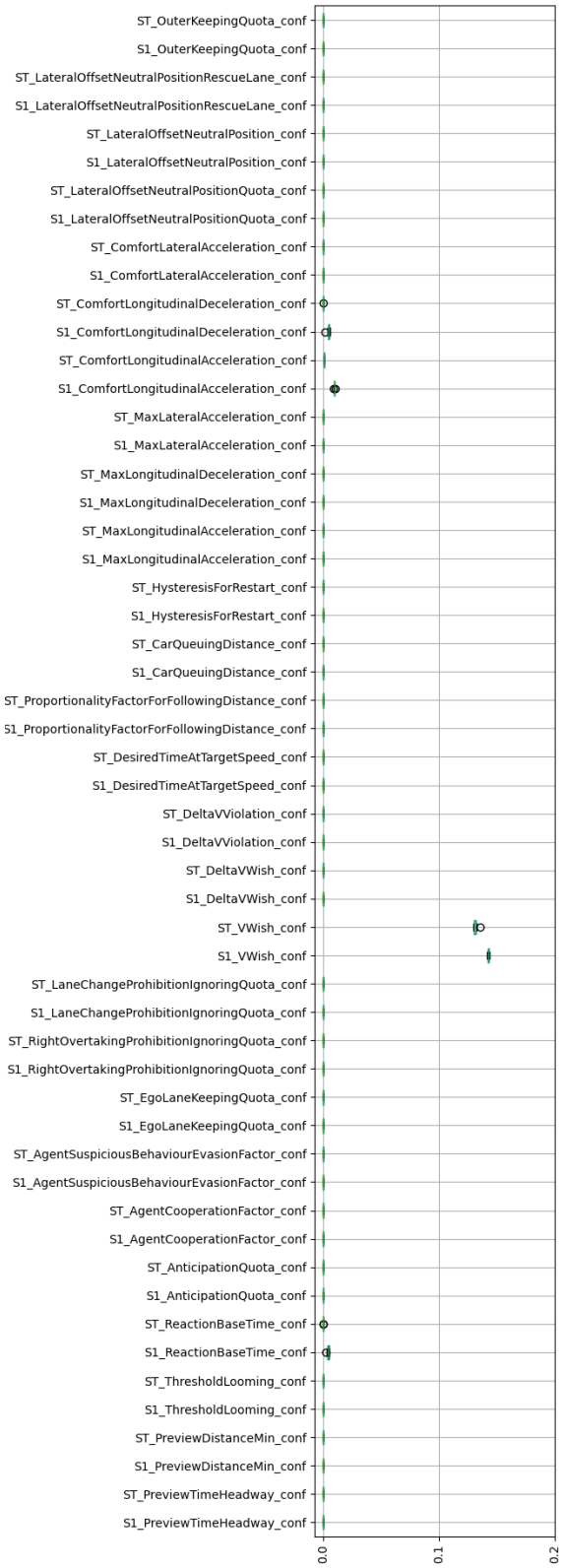

Figure .20.: Complete box plots on confidence intervals in free driving scenario at 25 seconds.

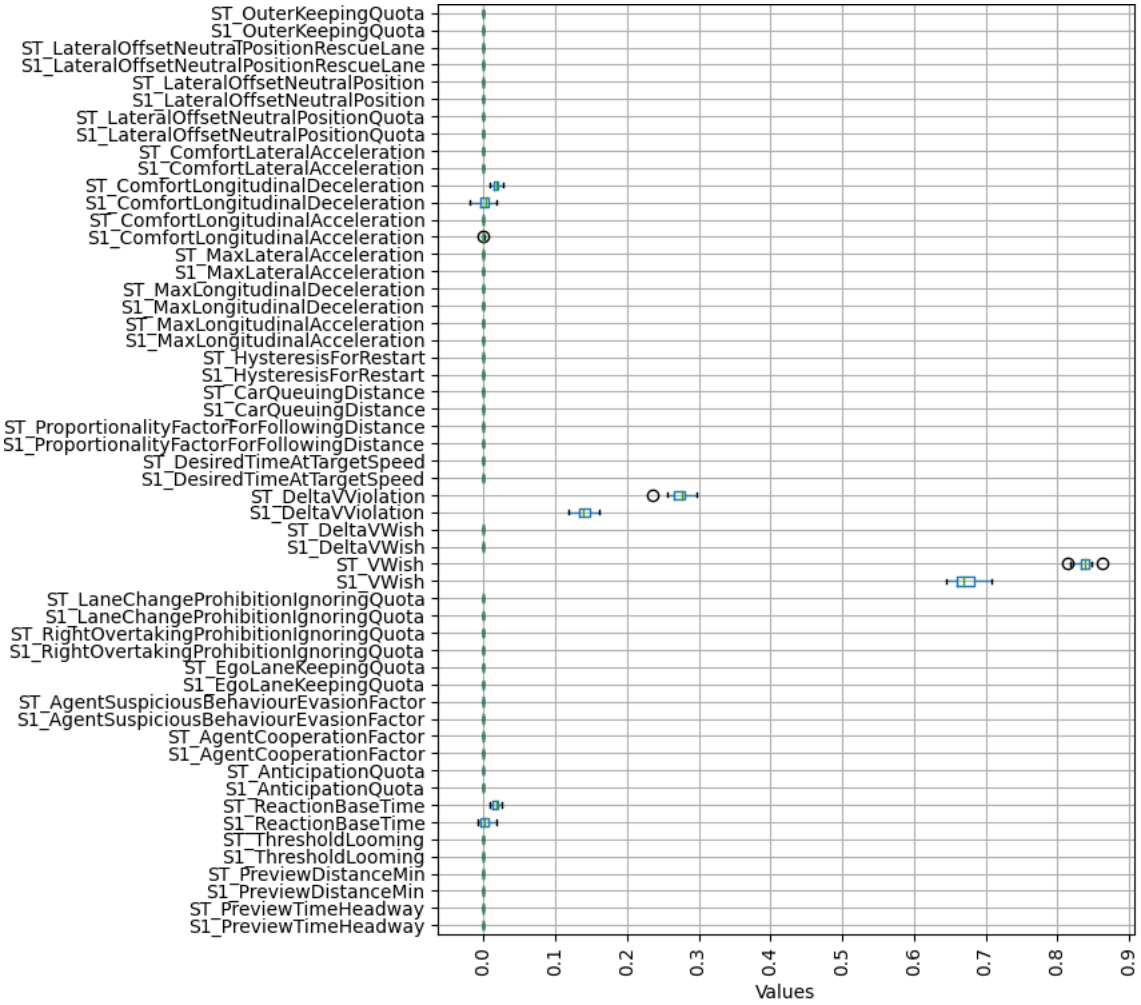

Figure .21.: Complete box plots on sensitivity indices in speed limit scenario at 25 seconds, Analysis 1.

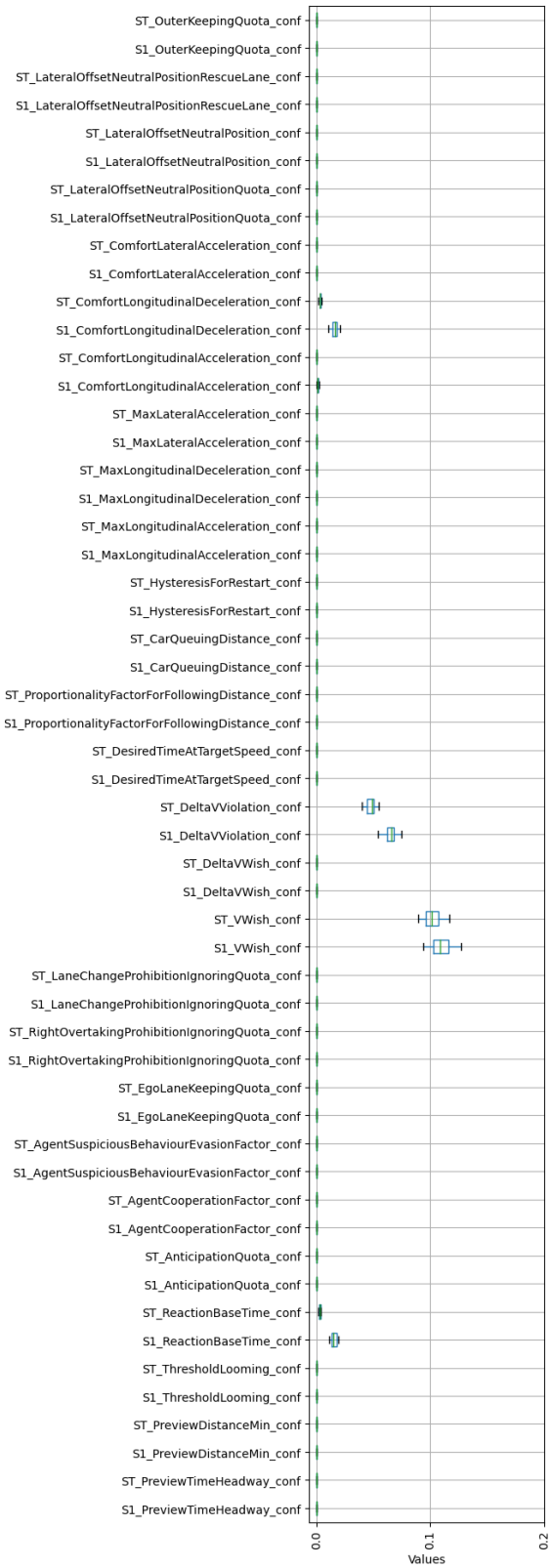

Figure .22.: Complete box plots on confidence intervals in speed limit scenario for velocity at 25 seconds, Analysis 1.

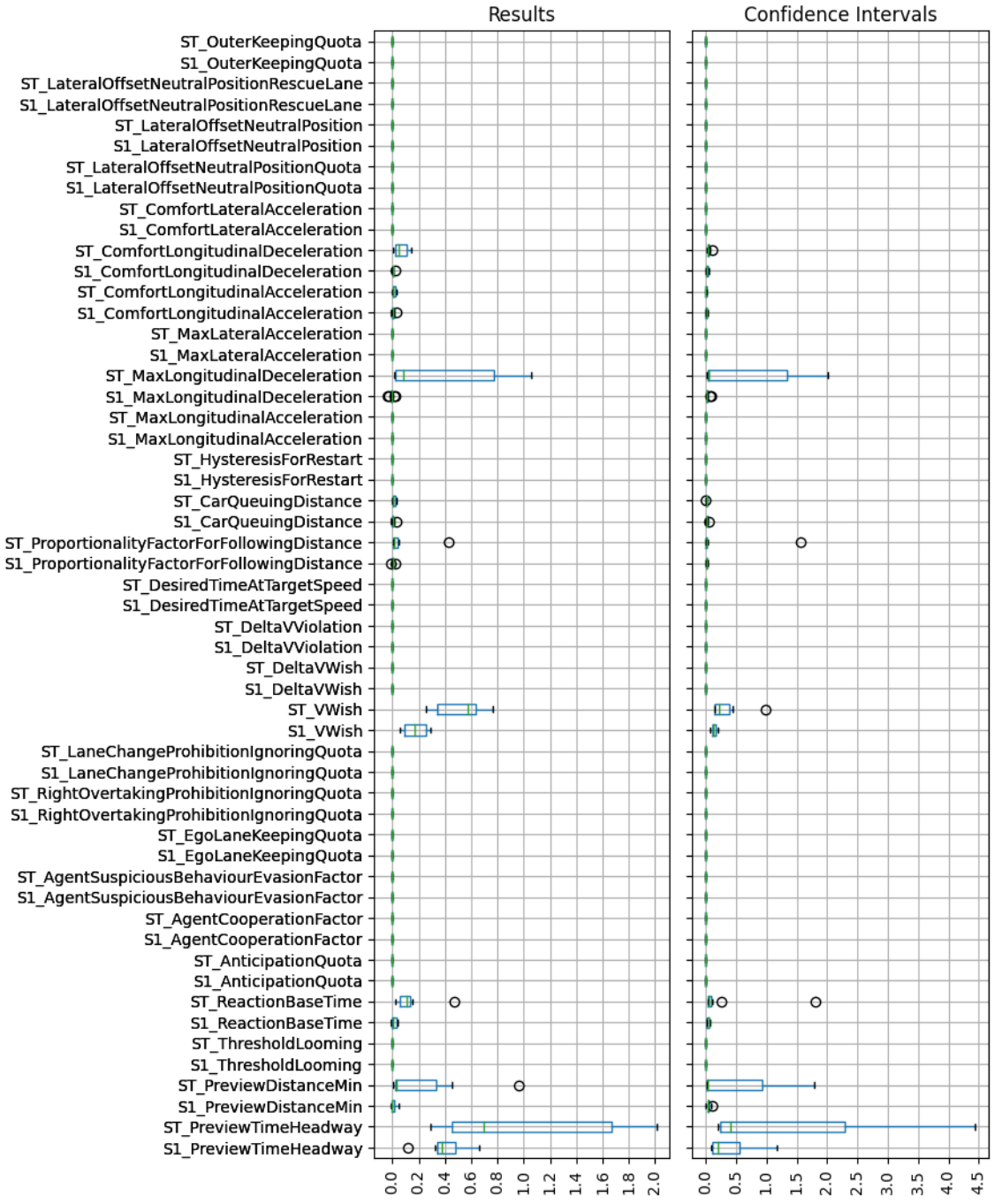

Figure .23.: Complete box plots in approaching scenario for velocity at 20 seconds, Analysis 1.

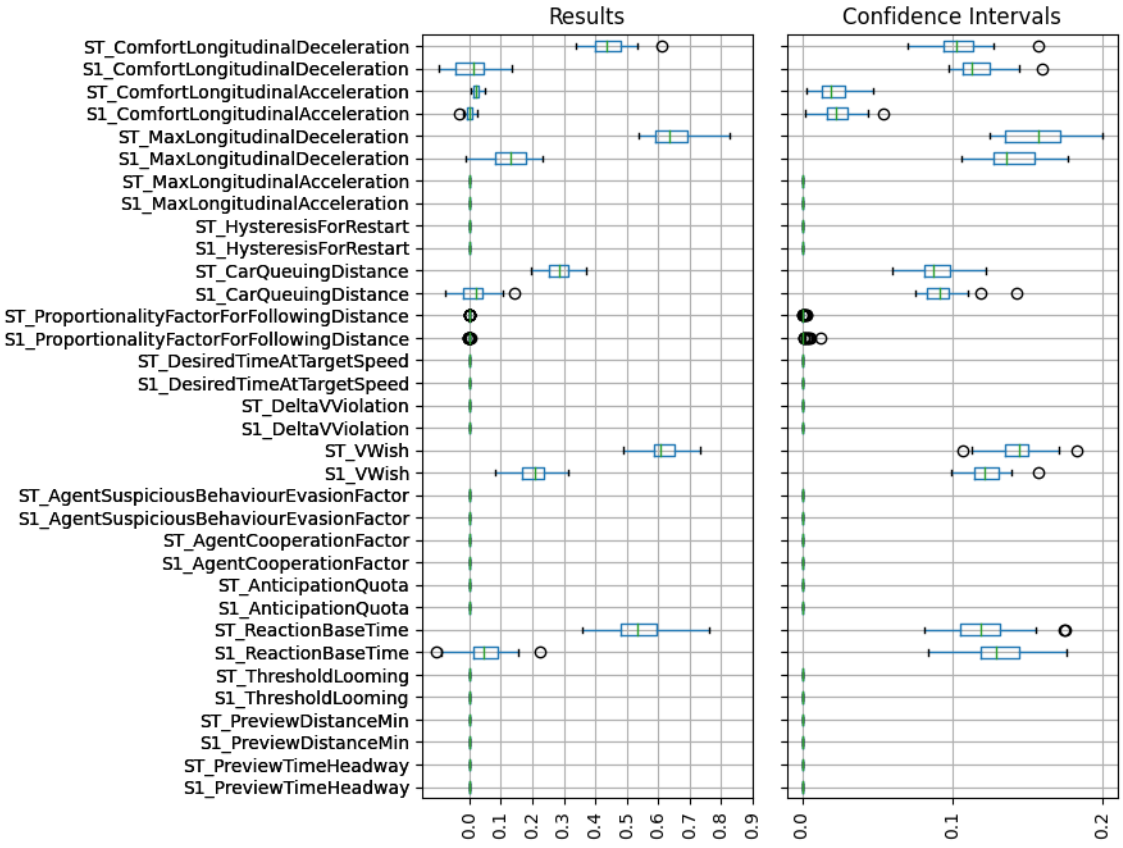

Figure .24.: Complete box plots for velcoity at 17 seconds in emergency brake scenario, Analysis 1.

List of Figures

List of Tables

Bibliography

[1] Eclipse openpass website. URL: `https://openpass.eclipse.org` (Accessed: June 1, 2023).

[2] Bmw openpass documentation: Algorithm scm. URL: `https://openpass-doc.apps.batch.advantagedp.org/advanced_topics/scm_guide/AlgorithmScm/AlgorithmScm.htm` (Accessed: June 1, 2023).

[3] P.E.A.R.S. Iso 21934 technical report: Prospective safety performance assessment of pre-crash technology by virtual simulation - part 1: State-of-the-art and general method overview, 2021.

[4] Martijn van Noort, Taoufik Bakri, Felix Fahrenkrog, and Jan Dobberstein. Simpato - the safety impact assessment tool of interactive. *IEEE Intelligent Transportation Systems Magazine*, 7(1):80–90, 2015.

[5] Ulrich Sander and Nils Lubbe. The potential of clustering methods to define intersection test scenarios: Assessing real-life performance of aeb. *Accident Analysis & Prevention*, 113:1–11, 2018.

[6] Olaf Op den Camp, Sjef van Montfort, Jeroen Uittenbogaard, and Joke Welten. Cats deliverable 6.1: Cats final project summary report. *TNO report*, 2016.

[7] Thomas Helmer. *Development of a methodology for the evaluation of active safety using the example of preventive pedestrian protection.* Springer, 2014.

[8] Alexandra Fries, Felix Fahrenkrog, Katharina Donauer, Marcus Mai, and Florian Raisch. Driver behavior model for the safety assessment of automated driving. volume 2022-June, pages 1669–1674. Institute of Electrical and Electronics Engineers Inc., 2022.

[9] Public openpass documentation. https://www.eclipse.org/openpass/content/html/index.html (Accessed: August 10, 2023).

[10] Robin Jannik Caloudis. Parallelization of a stochastic agent-based simulator on shared memory systems, 2022.

[11] Marcus Mai. *Fahrerverhaltensmodellierung für die prospektive, stochastische Wirksamkeitsbewertung von Fahrerassistenzsystemen der Aktiven Fahrzeugsicherheit*, volume 4. Cuvillier Verlag, 2017.

[12] Manuela Witt, Lei Wang, Felix Fahrenkrog, Klaus Kompaß, and Günther Prokop. Cognitive driver behavior modeling: Influence of personality and driver characteristics on driver behavior. volume 786, pages 751–763. Springer Verlag, 2019.

[13] Alexandra Fries and Felix Fahrenkrog. Validation and verification of the stochastic cognitive driver model. *ACIMobility Summit*, 2021.

[14] Mersenne twister 19937 generator. URL: `https://cplusplus.com/reference/random/mt19937/` (Accessed: June 1, 2023).

[15] Mersenne twister random number engine. URL: `https://cplusplus.com/reference/random/mersenne_twister_engine/` (Accessed: June 1, 2023).

[16] Dr. rer. nat Tobias Neckel. *Lecture Notes Algotithms for Uncertainty Quantification, Chapter 9: Sensitivity Analysis*. 2022.

[17] Daniel Straub. *Lecture Notes Risk Analysis*. 2020.

[18] A. (Andrea) Saltelli. *Global sensitivity analysis : The Primer*. John Wiley, 2008.

[19] Ralph C Smith. *Uncertainty quantification: theory, implementation, and applications*, volume 12. Siam, 2013.

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

[21] George Udny Yule. *An introduction to the theory of statistics*. C. Griffin, 1927.

[22] Salib issue discussion: Negative sobol' indices #135. URL: `https://github.com/SALib/SALib/issues/135` (Accessed: August 10, 2023).

[23] Wojciech Jarosz. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego, September 2008.

[24] Andrew Smith, Robin Lovelace, and Mark Birkin. Population synthesis with quasirandom integer sampling. *Journal of Artificial Societies and Social Simulation*, 20(4), 2017.

[25] Paul Bratley and Bennett L Fox. Algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 14(1):88–100, 1988.

[26] Salib documentation: Sobol' sensitivity analysis. URL: `https://salib.readthedocs.io/en/latest/api.html#sobol-sensitivity-analysis` (Accessed: June 1, 2023).

[27] Discussion on sobol' sequences. URL: `https://math.stackexchange.com/questions/888490/understanding-sobol-sequences` (Accessed: June 1, 2023).

[28] Francesca Campolongo, Andrea Saltelli, and Jessica Cariboni. From screening to quantitative sensitivity analysis. a unified approach. *Computer Physics Communications*, 182(4):978–988, 2011.

[29] Art B. Owen. On dropping the first sobol' point. 8 2020.

[30] Salib documentation: Advanced user guide. URL: `https://salib.readthedocs.io/en/latest/user_guide/advanced.html` (Accessed: June 1, 2023).

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

[32] James E Gentle. *Random number generation and Monte Carlo methods*, volume 381. Springer, 2003.

[33] concurrent.futures — launching parallel tasks. URL: `https://docs.python.org/3/library/concurrent.futures.html#module-concurrent.futures` (Accessed: June 1, 2023).

[34] Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), page 483–485, New York, NY, USA, 1967. Association for Computing Machinery.