Fakultät für Informatik
Technische Universität München

Systematic parameter analysis to reduce uncertainty in crowd simulations

Marion Gödel

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr. Martin Schulz

Prüfende der Dissertation:
1. Prof. Dr. Hans-Joachim Bungartz
2. Prof. Dr. Gerta Köster,
   Hochschule München

Die Dissertation wurde am 10.11.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 25.01.2022 angenommen.

Abstract

Crowd simulations have become an indispensable tool for reviewing evacuation concepts, determining the capacity of venues, and improving the efficiency of built infrastructure such as airports and train stations. However, uncertainties arise during modeling and simulation. For a reliable prediction, these must be considered. In crowd dynamics, this primarily concerns parameter uncertainties. Lack of knowledge about the choice of parameters causes epistemic uncertainty. To date, little to no effort has been spent on considering uncertainties in crowd simulations.

In this thesis, I propose a three-step approach to reduce and quantify epistemic uncertainty in crowd simulations: First, I identify the influential and non-influential parameters. Second, I calibrate the influential parameters against experimental data. The result of the first two steps is a specific model. Finally, I perform uncertainty analysis on the specific model. I select, adapt, and implement suitable methods for each step. The methods are provided in an open-source uncertainty quantification framework tailored to crowd simulations. I apply this approach to a safety-relevant scenario present in all egress situations: a bottleneck. Global sensitivity analysis reveals four influential parameters and two non-influential parameters. While the non-influential parameters can be fixed, the influential parameters need to be calibrated against experimental data. For calibration, I employ Bayesian inference methods that provide full posterior distributions for the parameters. I demonstrate that, in some safety-relevant cases, former methods fail. Subsequent uncertainty analysis reveals that identification and calibration of the influential parameters significantly reduce the output uncertainty compared to the prior parameter distributions. Moreover, I demonstrate how classical parameter estimation can be complemented with new methods for real-time simulations: I perform a feasibility analysis for estimating origin-destination matrices from density images using statistical learning models.

This thesis provides open-source implementations of the uncertainty quantification methods to foster their use in crowd dynamics. By demonstrating how epistemic uncertainty can be handled and how output uncertainty can be reduced, this thesis contributes to increasing the credibility and reliability of crowd simulations.

Zusammenfassung

Fußgängersimulationen sind mittlerweile ein unverzichtbares Werkzeug für die Überprüfung von Evakuierungskonzepten, die Bestimmung der Kapazität von Veranstaltungsorten und die Verbesserung der Effizienz von Infrastrukturgebäuden wie Flughäfen oder Bahnhöfen. Eine zuverlässige Vorhersage erfordert jedoch die Berücksichtigung von Unsicherheiten, die bei der Modellierung und Simulation eingeführt werden. Bei der Fußgängerdynamik betrifft dies vor allem Parameterunsicherheiten. Die Wahl geeigneter Parameterwerte ist oft ungewiss und daher mit epistemischer Unsicherheit behaftet. Bislang wurden kaum Anstrengungen unternommen, um Unsicherheiten bei der Simulation von Menschenmengen zu berücksichtigen.

In dieser Arbeit wird ein dreistufiger Ansatz zur Verringerung und Quantifizierung der epistemischen Unsicherheit in Fußgängersimulationen vorgeschlagen: Zuerst werden die einflussreichen Parameter identifiziert. Danach werden diese anhand experimenteller Daten kalibriert. Das Ergebnis der ersten beiden Schritte ist ein spezifisches Modell. Schließlich wird eine Unsicherheitsanalyse für das spezifische Modell durchgeführt. Für jeden Schritt werden geeignete Methoden ausgewählt, angepasst und implementiert. Diese Methoden werden in einem quelloffenen Software-Framework zur Quantifizierung von Unsicherheiten in Fußgängersimulationen zur Verfügung gestellt. Der vorgeschlagene dreistufige Ansatz wird auf ein sicherheitsrelevantes Szenario angewendet, das in allen Ausgangssituationen vorkommt, einen Engpass. Globale Sensitivitätsanalyse identifiziert vier einflussreiche Parameter und zwei einflusslose Parameter. Während die einflusslosen Parameter festgelegt werden können, müssen die einflussreichen Parameter anhand experimenteller Daten kalibriert werden. Für die Kalibrierung werden Bayes'sche Inferenzmethoden eingesetzt, die vollständige Posterior-Verteilungen für die Parameter liefern. Es wird gezeigt, dass herkömmliche Methoden zur Kalibrierung in einigen sicherheitsrelevanten Fällen versagen. Die anschließende Unsicherheitsanalyse zeigt, dass die Identifizierung und Kalibrierung der einflussreichen Parameter die Unsicherheit in der Vorhersage im Vergleich zu den ursprünglich angenommenen Parameterverteilungen erheblich verringert. Darüber hinaus wird gezeigt, wie die klassische Kalibrierung durch neue Methoden für Echtzeitsimulationen ergänzt werden kann. Hierfür wird eine Machbarkeitsanalyse für die Schätzung von Quelle-Ziel-Matrizen aus Dichtebildern unter Verwendung maschineller Lernmodelle durchgeführt.

Diese Arbeit stellt quelloffene Implementierungen geeigneter Methoden zur Quantifizierung von Unsicherheiten zu Verfügung, um deren Anwendung auf Fußgängersimulationen zu unterstützen. Indem sie aufzeigt, wie mit epistemischer Unsicherheit umgegangen werden kann und wie die Unsicherheit in der Vorhersage reduziert werden kann, trägt diese Arbeit dazu bei, die Glaubwürdigkeit und Zuverlässigkeit von Fußgängersimulationen zu erhöhen.

Acknowledgments

I believe that scientific exchange is a vital part of research. Therefore, I would like to thank everyone with whom I have discussed my work. Each of these conversations has contributed to this thesis and shaped me as a researcher. In particular, I would like to thank:

Prof. Dr. Hans-Joachim Bungartz, who gave me excellent guidance with his experience and foresight. Despite his numerous commitments, he always took the time to discuss my work and to provide constructive feedback.

Prof. Dr. Gerta Köster, who got me interested in studying crowd dynamics during my undergraduate studies. She has been an excellent mentor who always makes time for her students and goes out of her way for them. I learned so much from her during my Ph.D. that I will take with me and pass on.

The pedestrian dynamics research group: Benedikt Zönnchen, Benedikt Kleinmeier, Daniel Lehmberg, Stefan Schuhbäck, Christina Mayr, and Simon Rahn. No matter where we were, in the office, abroad, in the home office, we were always in contact. We supported each other, learned from each other, discussed problems and new ideas, and always got along well. It was great to share this experience with you!

Prof. Dr. Rainer Fischer for the great collaboration, numerous discussions on uncertainty quantification and research in general.

Dr. Mario Teixeira Parente, who inspired me to look into the active subspace method, for discussions on Bayesian inference and uncertainty quantification in general.

Dr. Nikolai Bode, who invited me to Bristol to collaborate on calisbration with Bayesian inference methods. I learned a lot from our collaboration and am glad to have had the opportunity to work with him. While in Bristol, I was also kindly welcomed by the Ph.D. candidates of the Department of Engineering Mathematics.

My former colleagues from the Department of Navigation at German Aerospace Center. I learned so much during my time there and was sent well prepared into the Ph.D. challenge.

My friends who were always there for discussion, who support, inspire me, encourage me, who regularly distracted me from work, and on whom I can always rely.

My family: My parents, who have always supported me and made my studies possible. My siblings for always being there for me. Finally, I would like to thank my husband Michael, who keeps encouraging me to take on new challenges and always supports me. I am grateful to have you in my life!

Contents

List of Figures

List of Tables

1 Introduction

Crowd crushes at events such as Love Parade (2010), Madhya Pradesh temple stampede during the Hindu festival of Navratri (2013), Hajj (2015), Oxford circus (2017), Lag BaOmer pilgrimage in Meron, Israel (2021), stress the importance of pedestrian safety. People in large crowds can be harmed if pedestrian flows do not interact well, space is overcrowded, or infrastructure is flawed. Besides safety aspects, the efficiency of built infrastructure affects our daily lives.

Pedestrian dynamics aims to understand the behavior of individuals and crowds. It is an interdisciplinary field of research that occupies engineers, mathematicians, computer scientists, psychologists, sociologists, and others. Independent of background, the community shares an overall objective: making crowds safer.

The two main approaches in pedestrian dynamics are (1) observation of crowds in real life and in controlled experiments and (2) modeling and predicting behavior. The two approaches are intertwined: Modeling builds on observations, and simulations can indicate where experiments are needed.

Several measures such as continuous crowd monitoring by trained personnel, thoughtful communications to direct people, education on crowd safety for participants and personnel are taken to prevent incidents and increase pedestrian safety. Crowd simulations are an additional important component: They enable us to analyze pedestrian flows for the maximum capacity, identify critical locations in which high densities occur, and evaluate evacuation concepts.

1.1 Motivation

Crowd simulations are based on mathematical models that describe crowd behavior. Several models have been developed over the last decades [Yang et al., 2020] and new models are still being developed. Almost all are aimed at a wide range of applications and situations ranging from capacity estimation at airports to evacuation concepts for schools. They draw on a list of parameters that need to be adapted to the purpose of the study. This is not uncommon. "Mathematical models generally involve parameters that must be 'tuned' so that the model best represents the particular system or phenomenon about which predictions are to be made."[Oden et al., 2010]. Strictly speaking, for each study a specific model needs to be defined by selecting specific parameter values and a simulation scenario has to be set up. However, the model parameters are usually unknown and some even cannot be measured directly. Imperfect knowledge about the parameters introduces uncertainty in the prediction.

Especially in the context of safety, we need to know how good these predictions are. This need has been formulated from inside the community: "Even though calibration

and validation are considered to be essential to determine the reliability and validity of simulation models, researchers currently apply inconsistent procedures or only partially test the simulation tools due to the lack of international standards for verification and validation of pedestrian flow and crowd dynamic simulation tools for general use" [Duives et al., 2016].

One factor that introduces uncertainty into the simulation is the choice of parameters. Parameter values in crowd dynamics are often set by relying on expert knowledge or surveys. As in many other disciplines, "Unfortunately, these model parameters are commonly not known with great precision; they may vary from [...] case to case, or they may not be known at all. In short, they generally involve large uncertainties that can be resolved only with sufficient experimental evidence." [Oden et al., 2010]. Yet experimental evidence is limited in crowd dynamics. Designing experiments with humans is challenging, and carrying them out is expensive, time-consuming, and may even be unethical. Consequently, parameter uncertainty affects the simulations.

The field of uncertainty quantification (UQ) offers a set of standardized methods that aim to improve the accuracy of model predictions. Smith describes UQ as "the science of identifying, quantifying, and reducing uncertainties associated with models, numerical algorithms, experiments, and predicted outcomes or quantities or interest." [Smith, 2014, p. 1]. There are mainly three objectives of UQ: (1) parameter identification or selection to isolate the influential parameters using sensitivity analyses, (2) parameter estimation which is concerned with the choice of the parameter values based on data while quantifying associated uncertainties, (3) uncertainty analysis which propagates input uncertainties through models to analyze the uncertainty in the prediction. Central to all UQ methods is the definition of a quantity of interest. The purpose of the simulation dictates the relevant outputs. For example, the flow measures the capacity of an environment [Seyfried et al., 2009], pedestrian density helps to identify critical locations, and evacuation time is prevalent in safety concepts.

We need to consider uncertainties in the simulation, if possible, reduce them, and quantify their impact on the quantities of interest. In other words, "Today [...] the phenomena and processes we ask computer models to predict are of enormous importance to critical decisions that affect our welfare and security [...]. With such high stakes, we must insist that the predictions include concrete, quantifiable measures of uncertainty." [Oden et al., 2010].

1.2 Scope and overview of this work

When crowd simulations are used to evaluate the safety concepts for buildings or events, central measures such as evacuation time, density, and flow are crucial. Therefore, uncertainties in their prediction should be reduced and quantified. When one relies on a single simulation, a too optimistic prediction at the tails of the distribution of the quantity of interest may be obtained which may compromise pedestrian safety. Instead, the full distribution of the quantity of interest needs to be evaluated e. g. using forward propagation methods. The uncertainty in the simulation results, epitomized by the vari-

ation in the quantities of interest, is directly related to the reliability of the simulation. A high degree of uncertainty in the results complicates the evaluation of security concepts and also the potential redesign of escape routes. Therefore, the uncertainty in the output should be reduced as much as possible. However, even after careful calibration, uncertainty will be present in the prediction due to uncertainty in the data used for calibration as well as parameters that are not perfectly informed by the calibration. This uncertainty needs to be quantified and taken into account. Overall, it is essential to consider and handle uncertainties when performing crowd simulations. The scope of this thesis is to identify a specific model with reduced uncertainty.

In this dissertation, I present a systematic three-step approach for the identification of a specific model and the reduction and quantification of uncertainty in the prediction. All steps are demonstrated on a scenario that reflects key crowd behavior: a bottleneck scenario. The bottleneck is a crucial part of all egress situations because the constriction can lead to high densities. In the following, I describe the steps to identify a specific model with quantified uncertainties in more detail. I state the associated research questions that I answer in the related chapters. Figure 1.1 presents a graphical overview of my work and the outcomes.

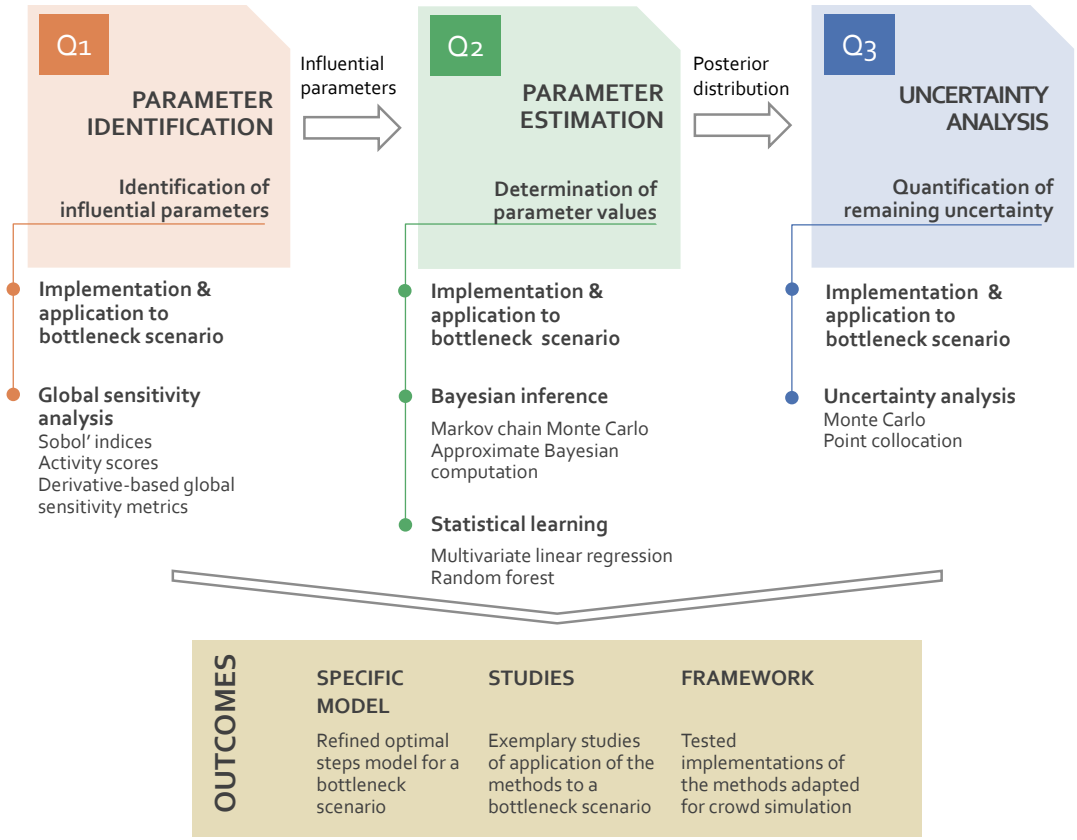


Figure 1.1: Outline of this work.

Uncertainty quantification software First, I deal with the necessary infrastructure for the parameter studies. The studies require a software framework with methods that are specifically selected and adapted for crowd simulations. I derive the requirements and evaluate existing software with respect to them. Since no existing software meets all mandatory requirements, I design and implement my own framework. The overarching requirement addressed in this chapter is:

R1 Choose or design a software for uncertainty quantification for crowd simulations

Subsequent requirements:

R1.1 Derive requirements for a software for uncertainty quantification for crowd simulations

R1.2 Analyze available software regarding the requirements

R1.3 Design and implement a software that fulfills the requirements

Parameter identification: identifying influential parameters Once the infrastructure is set up, I need to identify the most influential parameters of the chosen model family for the bottleneck scenario. The overarching research question is:

Q1 How can we identify influential parameters in the optimal steps model for the bottleneck scenario?

Subsequent research questions:

Q1.1 Which parameter identification methods are suited for crowd dynamics models?

Q1.2 Which parameters are influential and which are non-influential in the bottleneck scenario?

Q1.3 How can we determine if the results are reliable?

For this purpose, I apply global sensitivity analysis methods that assign a sensitivity index to each parameter. As a result, I can distinguish between influential parameters and non-influential parameters that can be fixed to an arbitrary value within their range. The most influential parameters have the largest impact on the uncertainty in the simulation output.

Parameter Estimation: finding values for influential parameters The next step is to estimate the values for influential parameters based on an experimental dataset. The overarching research question is:

Q2 How can we calibrate the influential parameters in the bottleneck scenario?

Subsequent research questions:

Q2.1 Which parameter estimation methods are suited for calibrating crowd dynamics models?

Q2.2 What are the advantages of Bayesian inference methods for calibration compared to established methods, such as point estimates?

Q2.3 What is the posterior distribution for the influential parameters in the bottleneck after calibrating to experimental data?

I employ Bayesian inference to calibrate the influential parameters and estimate the associated uncertainties. By careful calibration, I can reduce the uncertainty in the prediction. For the bottleneck scenario, I use a publicly available dataset to calibrate the free-flow speed. Bayesian inference provides a joint posterior distribution of the uncertain parameters that quantifies the uncertainty associated with the estimation.

Estimation of initial and boundary conditions: finding values for initial and boundary conditions in real-time predictions Parameter estimation is also an essential step towards real-time predictions. In particular, it is required for the live configuration of the simulation scenario. The aim is to recreate the current situation in reality as closely as possible in the simulator. For this purpose, the origins and destinations of the agents need to be defined. Moreover, a number of agents has to be assigned to each origin and their destination has to be defined. Especially the popularity of origin-destination combinations is often unknown. In a live prediction context, a dynamic input is needed. One approach is to integrate sensor data into the simulation. The overarching research question is:

Q2* Can we predict origin-destination matrices for the initialization of origins and destinations in the simulation from live sensor data in form of density heatmaps?

I perform a feasibility study to determine whether the origin-destination relations can be predicted with statistical learning models from abstract sensor input: a series of density heatmaps. This type of sensor data is widely available nowadays.

Uncertainty analysis: measuring the reduction of uncertainty in the simulation output The final step is to quantify the uncertainty in the simulation output. The overarching research question is:

Q3 How can we quantify the uncertainty in the prediction for the bottleneck scenario?

Subsequent research questions:

Q3.1 Which uncertainty analysis methods are suited for crowd simulations?

Q3.2 How large is the uncertainty in the prediction for the bottleneck scenario before and after calibrating influential parameters?

I employ Monte Carlo propagation to quantify the uncertainty in the prediction. Therefore, I take the joint posterior distribution from the parameter estimation and propagate it through the model. As a result, I obtain a distribution for our quantity of interest, the flow through the bottleneck. The shape of the distribution indicates the prediction

uncertainty. For the bottleneck study, I compare the uncertainty of the simulation output for three configurations: first, using the prior parameter distributions, second after fixing the non-influential parameters in the factor fixing setting, and finally, employing the joint posterior distribution obtained with calibration. We observe that the output uncertainty is significantly reduced in the resulting specific model. Both quantification and reduction of the uncertainty enhance the credibility of the simulation outcomes. This allows us to use simulations as a basis when formulating guidelines for the construction of buildings and infrastructures, when preparing and reviewing evacuation guidelines, and for designing safe events.

In short, the important steps for a systematic parameter analysis to quantify and reduce uncertainty in the prediction are:

- Choice of model family for the purpose of the study and the scenario
- Identification of the quantities of interest for the study
- **Construction of a specific model from a model family**
  - **Identification of influential parameters** (Q1)
  - **Calibration of influential parameters** (Q2)
- **Quantification of the uncertainty in the prediction to assess its accuracy** (Q3)

This thesis focuses on the highlighted steps and addresses the related research questions Q1, Q2, and Q3.

1.3 Structure of this work

I assume that readers either have a background in crowd dynamics or uncertainty quantification but not in both. Therefore, Chapter 2 provides an introduction to pedestrian dynamics (Section 2.2) and to uncertainty quantification (Section 2.3). Roughly speaking, pedestrian dynamics can be considered as an application and UQ as a methodology for this work.

In Chapter 3, the locomotion models used in this thesis and the two scenarios which I study are described: A bottleneck scenario which is part of all ingress and egress situations, and a scenario with a multi-directional flow in a train station. I introduce the crowd simulation framework Vadere that offers implementations of several locomotion models. The simulations in this thesis are performed with Vadere. Finally, I describe two effects observed in the simulations, stochasticity and noise, and outline how I handle them.

In Chapter 4, I present the uncertainty quantification framework for Vadere that I developed in this thesis. This chapter addresses the requirement R1: Choose or design a software for uncertainty quantification for crowd simulations.

Chapters 5, 6, 7, and 8 each address one of the research questions presented in the last section: Chapter 5 is concerned with Q1: “How can we identify influential parameters in

the optimal steps model for the bottleneck scenario?". Chapter 6 addresses Q2: "How can we calibrate the influential parameters in the bottleneck scenario?". Chapter 7 deals with Q2*: "Can we predict origin-destination matrices for the initialization of origins and destinations in the simulation from live sensor data in form of density heatmaps?". Chapter 8 is dedicated to Q3: "How can we quantify the uncertainty in the prediction for the bottleneck scenario?".

Each of these chapters is divided into four sections: (1) introduction to the challenge addressed in this chapter (2) state of the art on the topic in crowd simulation to familiarize the readers with the common approaches so that they can put my methodological choices into context, (3) a method section where potential methods for the problem are described, (4) my work performed on this topic, and (5) a summary of the results.

Finally, Chapter 9 summarizes the work performed in this thesis, evaluates the accomplishments, and gives an outlook on future work.

A brief overview of the software tools used for this thesis can be found in Appendix A.

2 Background

In this chapter, we take a look at the basics of modeling and simulation, in particular for pedestrian dynamics. I introduce uncertainty quantification methods and point out how they can be used in modeling and simulation of crowd dynamics.

2.1 Modeling and simulation

We first need to understand the terms "model" and "simulation". I follow the definition provided by [Neelamkavil, 1987] who defines a model as "a representation of a physical system or process intended to enhance our ability to understand, predict, or control its behavior." which is also used in [Oberkampf and Roy, 2010]. It is important to note here that the model is a simplification of the physical process. We need simplifications to make the simulation possible and feasible. However, simplifications introduce a discrepancy between the model and the modeled system. Choosing the best simplifications for the intended purpose of the model is a critical and complex task. Once the model is established, a simulation is "the exercise or use of a model to produce a result" [Oberkampf and Roy, 2010].

The stages of modeling and simulation are often described as a cycle [Schlesinger, 1979, Bungartz et al., 2014], as in Figure 2.1. First, from real-world observations, a simplified, abstract, conceptual model is derived. At the end of this step, we have a formal description of the model, e. g. equations describing a process. Even though models are simplified representations of reality, nowadays, most models themselves are rather complex. As a result, a mathematical formulation is not sufficient, but we need an implementation in order to run simulation experiments with it. Consequently, the next step is the implementation of algorithms often including discretization, to obtain a computer-based model. This model can then be used to simulate the modeled system or process in order to predict its behavior or to control it. While the modeling cycle so far suggests that modeling is a sequential procedure, it often is an iterative process in which the model is refined continuously.

Two crucial tasks in the modeling process are verification and validation. They aim to determine the validity, accuracy, and credibility of a prediction. Both are permanent tasks that are never finalized. Validation addresses the question "do we solve the right equations?" and verification is concerned with "do we solve the equations right?". In other words, verification assures that the code is a correct implementation of the conceptual model. Unit tests that assure that a method correctly performs are a typical tool for code verification. Whenever a mathematical problem needs to be solved in the computer-based model, such as an optimization, verification should also include a comparison to a solver with a high accuracy [Oberkampf and Roy, 2010]. Validation assures

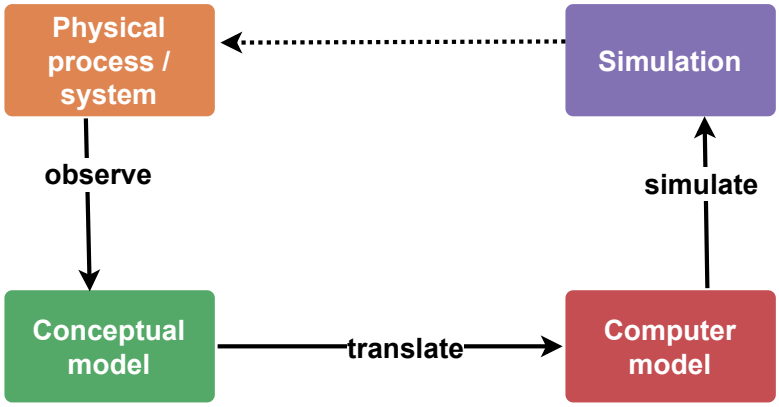


Figure 2.1: The modeling cycle describes the modeling process from observations of a physical system to a simulation of the process.

that the simulations align with the physical process within the application range of the model. Usually, observations of the process such as data from experiments are used for validation. Figure 2.2 integrates model verification and validation in the modeling cycle.

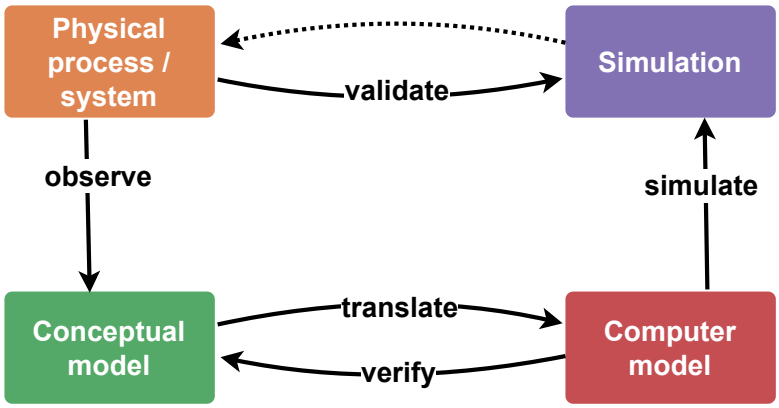
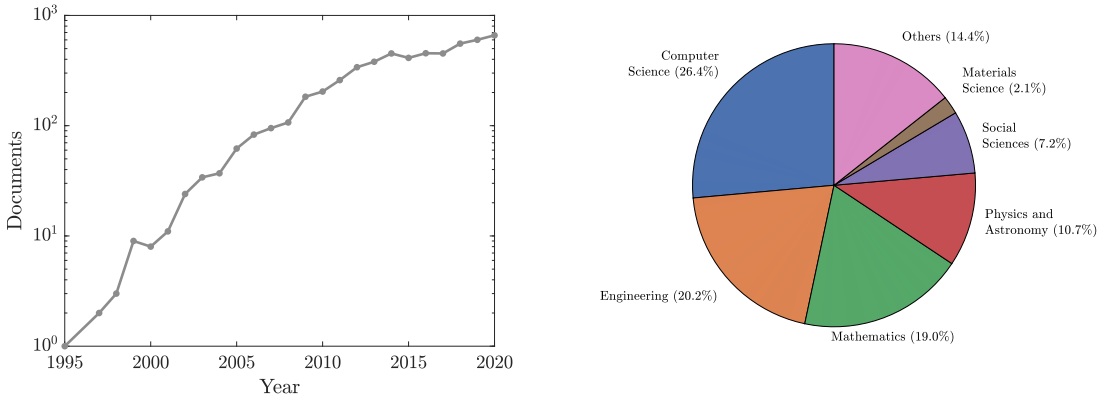


Figure 2.2: The modeling cycle including verification and validation.

2.2 Pedestrian dynamics

Pedestrian dynamics, or crowd dynamics, is concerned with the movements and behavior of individuals and crowds. One of the first publications was Le Bon's study [Le Bon, 1895] from 1895 in which he considered crowds as violent masses, which is an improper description of crowds in general. In 1975 Hirai and Trui published a simulation model based on ordinary differential equations (ODE) for a crowd [Hirai and Tarui, 1975]. It is one of the first publications in our field and is still relevant today. Figure 2.3 shows an increasing research interest in pedestrian dynamics since the 2000s as well as the interdisciplinary nature of the community. Although many modeling decisions are

based on psychological studies of crowd and individual behavior, still the majority of publications stem from natural sciences and engineering. However, lately, there has been an increased effort to consider psychological findings explicitly in pedestrian dynamics [Drury, 2020, Sieben et al., 2017, von Sivers et al., 2016b, Kleinmeier et al., 2020]. In order to overcome communication barriers among the disciplines, a glossary for crowd dynamics has been initiated [Adrian et al., 2019].



(a) Increasing interest in pedestrian dynamics over the last two decades.

(b) Publications in pedestrian dynamics stem from various fields. Disciplines with less than 2% of the documents are summarized in one category (others).

Figure 2.3: Summary of manuscripts in the interdisciplinary field of pedestrian dynamics (data source: Scopus, accessed on 2.7.2021).

The research in pedestrian dynamics can be roughly divided into studies of the behavior, including observing pedestrians in the real world and conducting controlled experiments, and modeling and simulation of pedestrian behavior. Here, I focus on the latter. Nevertheless, both parts are equally important and they even need each other: The first step of modeling is to obtain observations of the physical system. In our case, those are observations of pedestrian behavior. They can be obtained from the real world where there are numerous effects on the pedestrians that cannot be controlled or from (laboratory) experiments in which the behavior in a specific setting is studied. Consequently, observational studies and experiments lay the foundation for modeling pedestrian behavior. Once a computer-based model of crowd behavior has been developed, simulations can reveal potentially critical circumstances for pedestrian safety, as well as maximize flows. Before these insights can be implemented in infrastructure, controlled experiments should be carried out for the configuration. That means simulations can point out which experiments should be performed and can therefore reduce the number of necessary experiments (without simulation) significantly.

Large groups of pedestrians in confined spaces potentially lead to harmful situations, ranging from discomfort up to death. If we understand the dynamics of a crowd, we can

extrapolate to a certain extent how pedestrians will behave in an environment. This can be supported by simulations.

2.2.1 Modeling crowd behavior

By modeling crowd behavior, we can create simulation tools that allow us to analyze pedestrian behavior in different environments. As for pedestrian dynamics in general, the main goal of simulation studies is to ensure pedestrian safety. Two important measures for pedestrian safety are density and flow. Density is the number of pedestrians in a given area. The flow is the number of pedestrians crossing a measurement line over a given time period. Generally speaking, the combination of a high density with a low flow is potentially harmful. However, it depends strongly on the situation. Therefore, these two measures are considered essential in the pedestrian dynamics community. For example, at concerts, the density right in front of a stage is typically quite high. Since audience members often move to this part on purpose and the density decreases with the distance to the stage, the increased density in front of the stage is not problematic. Nevertheless, if the density is high in places where people want to move, and yet there is little flow, the situation is different. This could for example happen when the audience wants to leave the space of the concert, but the exits are too small, blocked, or unhandily placed. These simple examples show that the decision of whether a situation could be harmful is highly complex and requires a lot of experience. Consequently, while crowd simulations are currently able to support personnel that is responsible for crowd safety they are not (yet) capable of deciding automatically if a situation is harmful.

In order to analyze pedestrian safety in a given space, we can simulate the usual or maximum occupancy to identify locations in which high densities occur or flow is reduced. In addition, for events as well as for buildings, evacuation plans have to be created that define how the egress will be organized in case of an emergency. There are statutory guidelines for evacuations of buildings from fire safety to assure occupant safety. Similarly, regulations for events in open spaces exist. Secondary goals of the simulations can be estimation and maximization of capacity, for example in built infrastructure such as train stations or airports. In addition, measures have been defined to evaluate e.g. comfort and efficiency of built infrastructure [Helbing and Johansson, 2011].

2.2.2 Microscopic crowd simulation

One approach for modeling crowd behavior is representing each pedestrian by a virtual pedestrian, a so-called agent. This microscopic representation supports predictions in detailed topographies. Examples are buildings with many obstacles, such as theaters with rows of seats or open space offices, and infrastructures featuring several entrances and exits of regular width, which is about one meter. In addition, microscopic models are necessary if individual measures or microscopic measures are of interest, such as individual egress or waiting times. Most microscopic crowd movement models can be understood as agent-based or individual-based models [Bonabeau, 2002]. In agent-based modeling (ABM), each agent has individual characteristics and rules or heuristics to

make its own decisions. Typical individual attributes are the size of a pedestrian, often modeled by a radius when pedestrians are represented by circular agents, the height, and the free-flow speed, which is the "intrinsic" speed of a pedestrian walking unhindered towards his or her destination.

Microscopic simulation can also be used for larger topographies, but due to the high level of detail, they are computationally expensive. For studies with larger topographies in which macroscopic measures such as densities or flows are to be studied, macroscopic models can be a better choice. Macroscopic models typically consider pedestrians as a continuum [Hughes, 2000, Hoogendoorn et al., 2014, Treuille et al., 2006]. An overview of macroscopic models can be found in [Kormanová, 2013]. In between macroscopic and microscopic models, mesoscopic models are located. They are typically multiscale models, which means models with transition between the microscopic and the macroscopic scale [Borrmann et al., 2012, Biedermann et al., 2016, Bellomo and Bellouquid, 2015, Teknomo and Gerilla, 2008]. In this work, we focus on microscopic models since we study small topographies, mainly a bottleneck scenario.

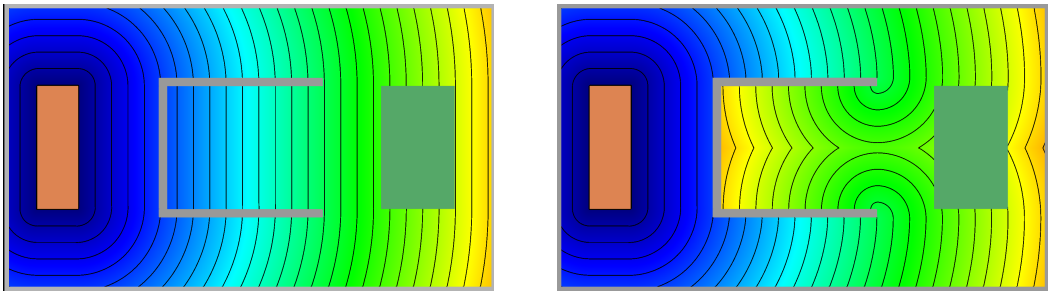
Microscopic crowd simulation software is often divided into three levels [Hoogendoorn and Bovy, 2004]: A strategic level that models the choice of destinations of a pedestrian by activity, a tactical level describing the scheduling of activities and respective destinations and route-choice towards the destinations, and an operational level that describes the locomotion. However, the levels cannot be strictly separated. In this work, I simulate a confined bottleneck scenario with defined origin and destination of the pedestrians. Consequently, activity choice and scheduling play a minor role. Central are route-choice and locomotion.

2.2.3 Wayfinding using navigational fields

Wayfinding, or route-choice, is concerned with the path that a pedestrian takes to reach his or her destination. Navigational fields, also often called floor fields, constitute one approach for modeling wayfinding. Each point in the field indicates the distance to the destination. These fields can be understood as a cognitive map [O'Keefe and Nadel, 1978] that the agent has of its surrounding. A navigational field is based on a mesh that is spanned over the topography. At each node in the mesh, the distance to the destination is calculated. In theory, any distance measure can be used for the floor field. Many topographies, however, include obstacles that need to be circumvented in order for the agent to find its destination.

Figure 2.4 shows an exemplary topography that highlights the importance of a distance measure that considers obstacles. In this scenario, between the origin where the agents are spawned (green rectangle) and their destination (orange rectangle), a U-shaped obstacle is present. If we use a distance measure that does not consider obstacles, agents will get stuck in the obstacle. To resolve this issue, a geodesic distance measure should be used. In general, a navigation field assigns a utility to each position within the simulation area.

The geodesic distance is computed by solving the eikonal equation which models the propagation of a wave. We start the fictional wavefront at the destination and it moves



(a) Euclidean distance measure **(b)** Geodesic distance measure

Figure 2.4: Scenario in which obstacles have to be regarded in the distance function.

around obstacles through the topography. Here comes the mesh into play, the eikonal equation is solved at each node of the mesh in order to obtain the travel time of the wave from the destination to the node. Since the wave is started at the boundary of the destination region, the navigational field depends on the destination. Consequently, we need one navigational field for each destination. If pedestrian density should be considered for wayfinding, e. g. when there is congestion in front of an exit, dynamic navigation fields can be employed [Köster and Zönnchen, 2014].

2.2.4 Locomotion models

Several locomotion models for pedestrians have been developed in the last decades. One of the first model types are the force-based models by Hirai and Tarui [Hirai and Tarui, 1975] and Helbing and Molnár [Helbing and Molnár, 1995]. Nowadays there are numerous extensions to the social force model [Chen et al., 2018]. All of them are based on the idea of force fields in the hodological space by Lewin [Lewin, 1951]. The hodological space represents a psychological space, paths are defined considering psychological factors instead of purely mathematically by the shortest distance. The force fields by Lewin, however, were not intended to be used in a Euclidean space. The idea behind social force models is that agents are subjected to attractive and repulsive forces. Attractive forces stem from destinations while repulsive forces originate from other agents or obstacles. In the social force models, the social force fields are treated as if they were physical forces: The sum of all forces yields the right-hand side of an ordinary differential equation. The equations can be solved using well-known solvers such as Runge-Kutta methods. However, one needs to be careful to avoid numerical instabilities when implementing the social force model [Köster et al., 2013]. The summation of forces can hinder pedestrians from reaching their destinations, and oscillations might occur due to numerical instabilities around the destination region. Nevertheless, social force models can produce smooth trajectories if the step size is chosen sufficiently small.

In between the two publications of social force models, in 1985, Gipps and Markjö introduced a locomotion model based on a cellular automaton [Gipps and Marksjö, 1985]. Cellular automata (CA) divide the space in an even grid where each cell in the grid has a state. Central to the CA are transition rules that define the transition between

states. In the CA proposed by Gipps and Markjö, cells are either free or occupied by an agent or an obstacle. Each agent moves along the cells towards its destination. Often, probabilistic transition rules are defined. Typically, rectangular grids are used for CA which cause artifacts on the agent's trajectories for lateral movements. In addition, the resolution of the grid somewhat limits how topography elements can be placed. For example, door widths can only be a multiple of the resolution. Cellular automata are, however, easy to understand and implement since they do not require any numerical treatment. In addition, they are computationally inexpensive and can therefore also be used for larger crowds.

In addition to social force models and cellular automata, there are several newer locomotion models. They can be grouped in multiple ways, e. g. based on model structure and theory [Teknomo, 2002] or by their handling of time and space. I group the models by the mathematical core problem that they need to solve. That leaves us with models based on ODEs and models that optimize a utility function. The remaining models use heuristic rules that can be easily computed. An overview of locomotion models can be found in Table 2.1.

Table 2.1: Locomotion models grouped by underlying problem: solving ordinary differential equations (ODE), optimizing an utility, or calculating heuristics.

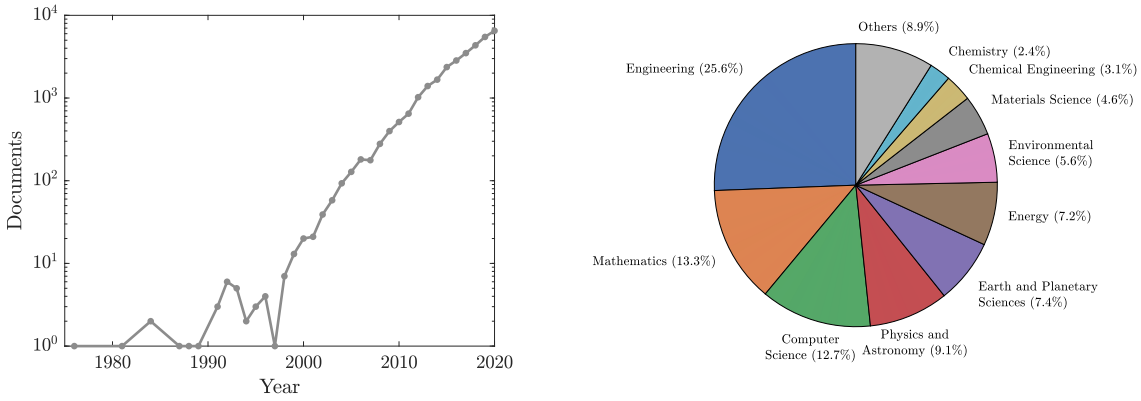
Name of the model	Reference	ODE	Utility	Heuristic
Social force model (SFM)	[Hirai and Tarui, 1975, Helbing and Molnár, 1995]	✔		
Gradient navigation model	[Dietrich and Köster, 2014]	✔		
(Generalized) centrifugal force model	[Chraibi et al., 2010]	✔		
Cellular automaton (CA)	[Gipps and Marksjö, 1985]		✔	
Optimal steps model (OSM)	[Seitz and Köster, 2012, von Sivers and Köster, 2015, Kleinmeier et al., 2019]		✔	
Stochastic headway distance velocity model	[Eilhardt and Schadschneider, 2014]			✔
Velocity obstacles	[van den Berg et al., 2008]		✔	
Discrete choice model	[Antonini et al., 2006]		✔	
Behavioral heuristics model	[Seitz et al., 2016]			✔
Reynolds steering	[Reynolds, 1999]			✔
Cognitive, decision-based model	[von Krüchten and Schadschneider, 2020]			✔

While most locomotion models are aimed for egress situations, they cover a range of situations as the test cases from the RiMEA guidelines [RiMEA, 2016] for microscopic evacuation analysis show: The test cases cover different local topographies such as movement around corners, passing of bottlenecks, congestion in front of stairs. Also, various situations defined by population via age distribution, reaction times, and densities have been considered.

In this dissertation, I mainly work with the optimal steps model (OSM) described in [Kleinmeier et al., 2019]. The core of the model is the optimization of a utility that considers the geodesic distance to the agent's destination as well as other agents and obstacles in the proximity. It is described in more detail in Section 3.1.1. The OSM works with continuous space and quasi-continuous time. Therefore, it does not suffer the same restrictions from spatial discretization as the cellular automaton. Additionally, I use an emulator for the Social Force Model as a use case for calibration using Bayesian inference and a point estimate in Section 6.4.3.

2.3 Uncertainty quantification

Similar to pedestrian dynamics, uncertainty quantification methods have been developed and used only in the last two decades (Figure 2.5a). The number of published manuscripts on uncertainty quantification, or short UQ, is about 10 times of those in pedestrian dynamics. Similar to pedestrian dynamics, we observe a wide interest in the UQ methods, compare Figure 2.5b. However, the researchers are all rooted in science, technology, engineering and mathematics, short STEM, fields. Since uncertainty quantification provides a set of methods, many publications are not focused on UQ itself, but rather on the selection and adaptation of its methods to their application.



(a) Increasing interest in uncertainty quantification over the last two decades.

(b) Publications on uncertainty quantification stem from various fields.

Figure 2.5: Manuscript statistics for "uncertainty quantification" (data source: Scopus, accessed 2.7.2021).

In this work, I select, adapt, and implement methods from uncertainty quantification in order to assess and improve the accuracy, credibility, and reliability of the predictions. This can be understood as a part of model validation [McClarren, 2018].

2.3.1 Definition of uncertainty

Before discussing uncertainty quantification and its purposes, I first need to introduce the term uncertainty. In the literature, I could not find a consistent definition of uncertainty. Often, it is used interchangeably with error. I believe that error and uncertainty are fundamentally different and define them accordingly. I use the definition of error from Tumeo [Tumeo, 1994] who defines it as "the difference between a computed or measured value and a 'correct' value". Tumeo also defines uncertainty as "the concept or condition of being in doubt about a value". We notice that in the definition of uncertainty there is no 'true' reference mentioned. This is the key difference between error and uncertainty. As a consequence, errors are "recognizable deficiencies" [Oberkampf and Roy, 2010] while uncertainties are "potential deficiencies" [Oberkampf and Roy, 2010] meaning that we do not know if they are present. Nevertheless, this definition of uncertainty is still somewhat vague and its use is therefore limited.

2.3.2 Types of uncertainty

I believe that uncertainty can be best understood by looking at where it originates. There are several sources for uncertainties in modeling and simulation. Each model is a simplification of a real world problem or system. Along with the benefits of simplification such as computability and code acceleration come drawbacks: The model is only an approximation to the real system. Thus there is a discrepancy between the model and the system under investigation. This discrepancy is typically studied at specific points where observations are available during validation. If we consider the observations as true reference for simplification, at these points a model error can be calculated. In between (interpolation) as well as outside (extrapolation) of these points, the discrepancy is typically unknown and consequently a source of uncertainty. I refer to this type of uncertainty as model (form) uncertainty. It is also called model discrepancy or inadequacy [Ghanem et al., 2017].

In a real system, observations are always erroneous measurements of the system. We do not have direct access to 'true' reference values from the system itself. Therefore, another type of uncertainty is measurement uncertainty or observational uncertainty. This uncertainty is introduced by the setup and devices in the measurement process. When studying humans, as in pedestrian dynamics, experimental uncertainty also entails behavioral uncertainty [Ronchi et al., 2014] that describes that stochasticity in human behavior leading to different results for the same experiment [Averill, 2011].

All models comes with a set of inputs that need to be chosen carefully for a reliable prediction. Inputs consist of parameters as well as initial and boundary conditions. Imperfect knowledge of the inputs leads to input uncertainty often also described as parameter uncertainty. This type of uncertainty is central for the studies in this manuscript.

Our computerized models are all numerical, not algebraic, models and therefore subject to discretization error and numerical errors. These are errors since we know beforehand that they are present. For example, the navigational field of the agents is only evaluated with a certain discretization, the resolution of the mesh. The optimization of the utility in the optimal steps model is performed with numerical algorithms that include numerical errors. So do the ODE solvers for force-based models. In addition, there can be uncertainties present associated with the discretization, but in this work I do not consider any uncertainties originating from this process.

In the literature, uncertainty is often divided into two main types which are defined more consistently: Aleatory uncertainty "refers to uncertainty about an inherently variable phenomenon"[Sullivan, 2015, p. 4]. It is also referred to as stochastic or irreducible uncertainty while epistemic uncertainty "refers to uncertainty arising from lack of knowledge"[Sullivan, 2015, p. 4]. The sources of uncertainties defined above are all epistemic uncertainties. The distinction between the two types of uncertainty is not always clear because lack of knowledge depends on the point of view [Smith, 2014, p. 8]. Rolling a dice or flipping a coin is usually considered a random process, and therefore subject to aleatory uncertainty. However, if we were to model the process of the coin toss itself, it is no longer random. Instead, the initial conditions of the throw might be unknown and therefore affected by epistemic uncertainty. It is important to note that aleatory uncertainty cannot be further reduced since it is part of the system that is studied. Epistemic uncertainty, however, can be reduced by increasing our knowledge base. In practice, gaining more knowledge is often a complex and costly task. Therefore, it is necessary to identify sources of uncertainty that have a large impact on the simulation to invest our resources efficiently.

2.3.3 Overview of uncertainty quantification methods

Uncertainty quantification methods are often divided into three groups: uncertainty analysis or forward propagation, sensitivity analysis, and methods for inverse problems. Central to any study is the quantity of interest, a predefined simulation outcome. For crowd dynamics, this can be the flow or the density in a certain region, or the number of agents still in the building for an egress simulation as in [von Sivers et al., 2016b]. Sensitivity analysis methods aim to attribute the uncertainty in the simulation output to the input parameters. The result is usually a sensitivity index, a metric, for each studied parameter which quantifies the impact of this parameter on the uncertainty in the simulation output. Local sensitivity analysis studies the behavior of the model around a fixed reference point, while global sensitivity analysis allows to study the impact of the parameter over a certain range or under a given distribution. A typical inverse problem is to find a parameter vector for which the model outcomes resemble given observed data. This task is often called parameter estimation. The result can be a single parameter vector, as for classical vector, or a so-called informed posterior distribution of the uncertain parameters that reflects the uncertainty in the data relative to the model. Uncertainty analysis propagates uncertain parameters and their associated distributions

through the system to quantify their impact on the simulation outcomes. Typical outputs are statistical moments or even the full distribution of the quantity of interest.

For each of the three main methodical groups, I provide an overview of methods in the respective chapter: Sensitivity analysis methods can be found in Section 5.3, methods for parameter estimation in Section 6.3, and methods for uncertainty analysis in Section 8.3.

2.4 Summary

In this chapter, I gave a brief introduction to the concepts of modeling and simulation by presenting the modeling cycle. The cycle includes four steps: From observations of the physical system, in our case the behavior of a crowd, a conceptual model is formulated which is then implemented to obtain a computer model. The computer model can then be used to predict crowd behavior. Two essential steps of modeling are verification and validation that assure the validity of the model.

Building on the modeling cycle, I discussed modeling of crowd behavior. I focused on microscopic crowd simulation that relies on agent-based models in which each agent has individual properties. Out of the three layers of models for pedestrian behavior - strategic, tactical, and operational - I explained how navigational fields can be used for wayfinding which is part of the tactical layer and I described common locomotion models for the operational layer.

I then introduced uncertainty quantification, a research field that is concerned with quantifying uncertainties in all steps of a simulation in order to assess and improve the reliability of predictions. I follow the broad definition of Tumeo of uncertainty as "the concept or condition of being in doubt about a value" [Tumeo, 1994] and described different types of uncertainty, in particular parameter uncertainty, to explain how uncertainties arise. Finally, I gave a brief overview of typical approaches in uncertainty quantification.

3 Modeling choices: locomotion model, scenario, and stochasticity

In this chapter, we learn about two locomotion models that are used for simulations in this thesis, the optimal steps model and an emulator for a social force model. Any locomotion model is strictly speaking a family of models since it has a set of parameters that can be adjusted in order to reconstruct different scenarios. For a concrete purpose, a specific model needs to be derived from the family. This is achieved by adapting the parameters to the situation under investigation. In this thesis as a whole, I demonstrate how methods from uncertainty quantification can be employed to obtain a specific model. The selected methods can be applied to any locomotion model. I illustrate this on the optimal steps model for the bottleneck scenario. The social force model emulator is designed specifically for a bottleneck scenario as well and it is employed for one use case for parameter estimation in Section 6.3. I choose this model because it exhibits a faster-is-slower dynamic, which causes ambiguity in the input-output relationship.

We take a look at the two scenarios on which I focus in this thesis. First, a bottleneck scenario, which is central for pedestrian safety in all egress scenarios since the constriction leads to increased density. I identify influential and non-influential parameters in the bottleneck and calibrate the influential parameters using experimental data in order to obtain a specific model with reduced uncertainty. Second, a multi-directional flow through an overpass at a train station for which I continuously predict the occupancy of origins-destination combinations based on trajectory data. The prediction is an example of how we can initialize a real-time simulation for which we cannot use static information. The overpass is well-suited for this application since there are several origins and destinations. The passenger flow changes throughout the day due to the train schedule and commuting. Consequently, so do the origins and destinations of passenger paths.

The simulations of the bottleneck scenario are performed with the optimal steps model implemented in the Vadere crowd simulation framework. Hence, I provide a general overview of the framework. Additionally, I briefly introduce the simulation loop which is the core of the framework and needs to be understood in order to extend the framework. Moreover, I describe which steps are necessary to build a scenario for simulations.

Most uncertainty quantification methods are designed for deterministic systems. Thus, I also discuss how I deal with stochasticity and noise in the simulation output. We take a closer look at two effects observed when simulating the bottleneck scenario. I distinguish stochasticity from noise. While the former refers to variation at a fixed parameter vector, the latter denotes a high sensitivity of the model response to small changes in the input around a fixed value. I explain why and where stochastic terms are introduced in the simulator and discuss their effects.

3.1 Locomotion models

In Section 2.2.4, we learned about locomotion models for the operational layer from the pedestrian dynamics community. In particular, we got to know the functionality of two types of models: social force models and cellular automata. In this chapter, we take a closer look at the optimal steps model and the emulator for the social force model, which are both used in this work.

3.1.1 Optimal steps model

The optimal steps models (OSM) [Seitz and Köster, 2012, von Sivers and Köster, 2015, Kleinmeier et al., 2019] is an agent-based model for locomotion of pedestrians. Each virtual pedestrian, or agent, has a number of individual attributes such as its individual free-flow walking speed, step length, and step frequency. The main idea of this model is that each agent decides on the position for its next step by maximizing a utility that balances several goals: Reaching the destination while keeping a distance from obstacles as well as from other agents. Maximization of utility implies perfect rationality of the economic man [Ingram, 1888]. There are several enhancements of the OSM for complex scenarios to describe queueing behavior [Köster and Zönnchen, 2014], movement on stairs [Köster et al., 2019], groups [Seitz, 2016, p. 155ff][Seitz et al., 2014], helping behavior [von Sivers et al., 2016b], and cooperative behavior [Kleinmeier et al., 2020]. Here, I describe the OSM without extensions.

The temporal discretization in the optimal steps model is linked to the discrete steps of the pedestrian. In reality, the movement during walking is continuous, whereas in the model the movement takes place at discrete points in time. The stepping events for the agents are determined by the update scheme. The optimal steps model was originally proposed with a sequential update scheme [Seitz and Köster, 2012]. That means, at a discrete time step all agents are moved in a sequential fashion according to a predefined order. However, in a later study, the authors recommend an event-driven update scheme [Seitz et al., 2014]. In this scheme, a queue handles all events, and anytime an event is due, actions are triggered. As mentioned already, each agent is assigned an individual stepping frequency. From the moment when the agent is "born", or spawned, in the simulation, its regular stepping events are registered in the event queue. The OSM can also be combined with other update schemes if necessary, but it should be kept in mind that the update schemes have an impact on the simulation results [Seitz and Köster, 2014]. The event-driven update scheme allows for natural motion, as agents have individual step sequences, just like pedestrians in reality. The agents adhere to an intrinsic individual stepping frequency. Consequently, they move at different time steps as opposed to a serial or parallel update scheme in which all agents move at the same time. However, the naturality of this approach comes at a cost: Event-driven update schemes are not ideal for parallelization and are therefore more costly. A parallelized version of the optimal steps models event-driven update scheme that aims to mitigate this issue has just recently been published [Zönnchen and Köster, 2020]. It is mainly

effective in large-scale scenarios. Since my scenario is rather small, I employ the regular event-driven update scheme of the OSM.

We take a look at the utility optimization in more detail: At every stepping event, the current agent checks its direct surroundings for the best possible next position. For this step, the utility is evaluated within a disc around the current position of the agent. The disc's radius is the individual step length of the agent. Within the disc, three utilities are evaluated: the destination utility or navigational field, U_n, the obstacle utility U_o, and the agent utility U_p.

The destination utility can be understood as a cognitive map that the agent has of its surrounding. It encodes the distance of each point in the topography towards the (next) destination of the agent. The navigational field is calculated by solving the eikonal equation on a regular grid using the fast marching method [Sethian, 1996]:

$$\begin{aligned} ||\nabla u(x)||f(x) &= 1 \quad \text{for } x \in \Omega \subset \mathbb{R}^2 \\ u(x) &= 0 \quad \text{if } x \in \Gamma. \end{aligned} \tag{3.1}$$

Figure 3.1 shows the destination utility for a bottleneck scenario in which agents need to move through a constriction to their destination. Recently, a more efficient algorithm that solves the eikonal equation on an unstructured triangular mesh that adapts to the topography has been developed [Zönnchen and Köster, 2018]. Solving the eikonal equation models the propagation of a wavefront that starts at the destination and disseminates all over the topography around obstacles. For $f(x) = 1$, it is a geodesic distance measure. In order to include pedestrian densities in the navigational field, dynamic floor fields can be employed [Köster et al., 2014]. They encode the density information into the travel speed function $f(x)$ of the eikonal equation.

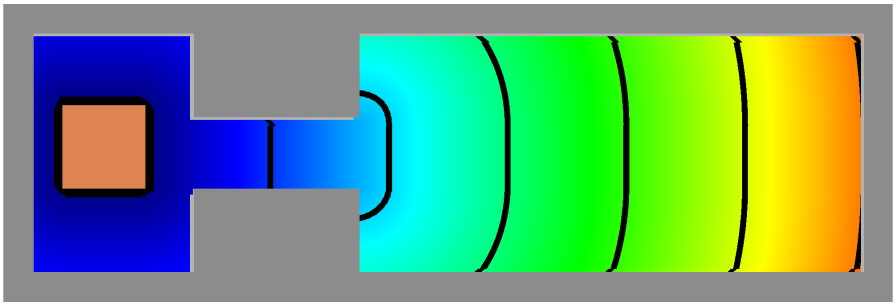


Figure 3.1: Navigational map for agents to find their destination (orange) in the bottleneck scenario.

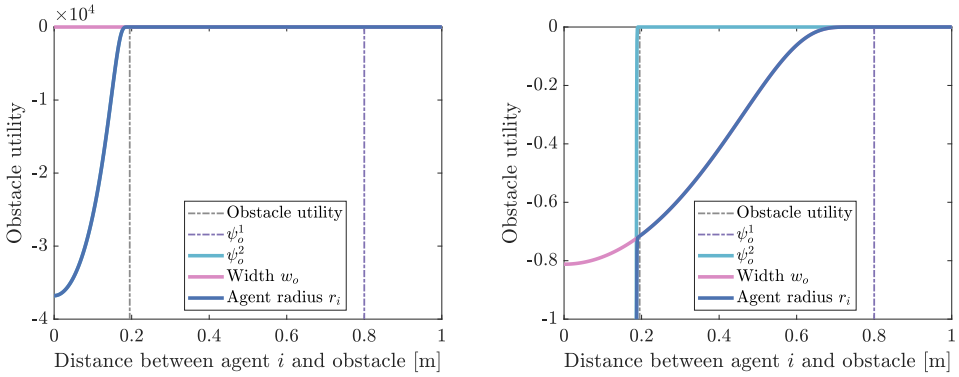
By taking obstacles into account, the navigational field ensures that agents do not step on obstacles in the simulation. This is, however, not sufficient for a natural movement around obstacles. In reality, pedestrians keep a certain distance to walls that depends on the height and characteristics of the wall [Moussaïd et al., 2009]. This behavior is modeled by an obstacle utility that introduces dips in the utility around obstacles. It is visualized in Figure 3.2). The obstacle utility is described by

$$U_{o,j}(x) = \begin{cases} \psi_o^2(x) + \psi_o^1(x) & \text{if } d_o(x) < r_i \\ \psi_o^1(x) & \text{if } r_i \le d_{o,j}(x) < w_o \\ 0 & \text{else.} \end{cases} \tag{3.2}$$

with

$$\psi_o^1(x) = \quad h_o \cdot \exp\left(\frac{2}{\left(\frac{d_o(x)}{w_o}\right)^2 - 1} \right), \tag{3.3}$$

$$\psi_o^2(x) = \quad 10^5 \cdot \exp\left(\frac{1}{\left(\frac{d_o(x)}{r}\right)^2 - 1} \right). \tag{3.4}$$



(a) Full x-axis **(b)** Zoom on x-axis

Figure 3.2: The obstacle utility $U_{o,j}(x)$ determines the behavior of an agent close to an obstacle.

The agent utility assures that the agents do not collide and keep a certain distance from each other. This models the behavior of pedestrians towards out-group members. Again, dips are introduced in the utility at and around the positions of other agents by

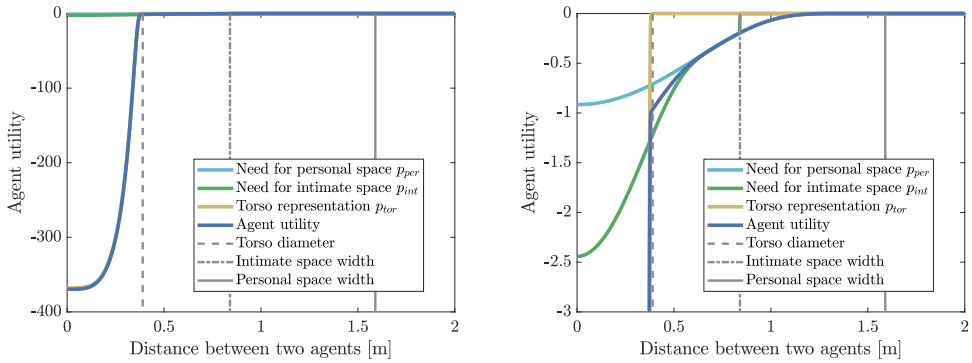
$$U_{p,l,i}(x) = -\begin{cases} p_{l,i}^{per}(x) + p_{l,i}^{int}(x) + p_{l,i}^{tor}(x) & \text{if } d_i(x) < (r_i + r_l) \\ p_{l,i}^{per}(x) + p_{l,i}^{int}(x) & \text{if } (r_i + r_l) \le d_i(x) < \delta_l^{int} + r_l + r_i \\ p_{l,i}^{per}(x) & \text{if } \delta_l^{int} + r_l + r_i \le d_i(x) < \delta_l^{per} + r_l + r_i \\ 0 & \text{else} \end{cases} \tag{3.5}$$

with

$$p_{l,i}^{tor}(x) = h_p \cdot \exp\left(\frac{4}{\left(\frac{d_i(x)}{\delta_l^{per}+r_i+r_l}\right)^{2\cdot c_p}-1}\right) \tag{3.6}$$

$$p_{l,i}^{int}(x) = \frac{h_p}{a_p} \cdot \exp\left(\frac{4}{\left(\frac{d_i(x)}{\delta_l^{int}+r_i+r_l}\right)^{2\cdot b_p}-1}\right) \tag{3.7}$$

$$p_{l,i}^{per}(x) = 10^3 + \exp\left(\frac{1}{\left(\frac{d_i(x)}{r_i+r_l}\right)^{4}-1}\right). \tag{3.8}$$



(a) Full x-axis **(b)** Zoom on x-axis

Figure 3.3: Agent utility $U_{p,l,i}$ represents interaction between pedestrians. Pedestrians naturally keep a certain distance among them. In the agent utility, this is modeled by introducing Hall's interpersonal distances [Hall, 1966].

The agent utility is visualized in Figure 3.3. Its three parts refer to Hall's interpersonal distances [Hall, 1966]. He divided them into intimate distance (close phase $p_{l,i}^{tor}(x)$ and far phase $p_{l,i}^{int}(x)$), personal distance $p_{l,i}^{per}(x)$ for interactions with close friends or family, social distance for acquaintances, and public distance.

The utility of a position x is then the sum of the navigational field $U_n(x)$, the utility for interactions among agents $U_{p,l,i}$, and the largest absolute obstacle utility $U_{o,j}(x)$:

$$U_l(x) = U_n(x) + \sum_{i=1, i\neq l}^{n} U_{p,l,i}(x) + \min_{j\in\{1,\dots,m\}} U_{o,j}(x). \tag{3.9}$$

In Figure 3.4, the utility $U_l(x)$ is shown exemplarily for one agent in a bottleneck scenario. Agent i finds its next position $x_i(t_{k+1})$ by maximizing the utility within the agent's step circle around its current position $P_i(x_i(t_k))$ [Kleinmeier et al., 2019]:

$$x_i(t_{k+1}) = \arg\max_{y\in P_i(x_i(t_k))} U_l(y). \tag{3.10}$$

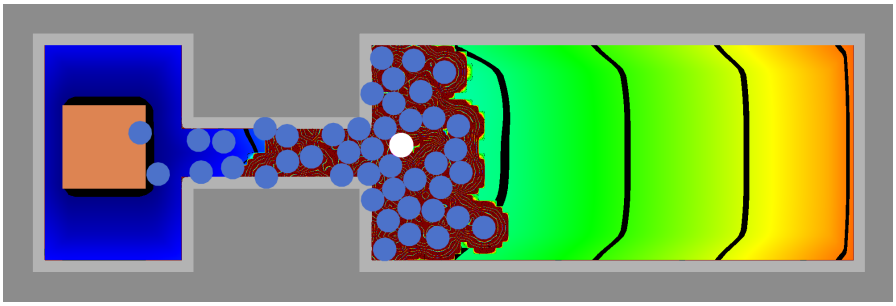


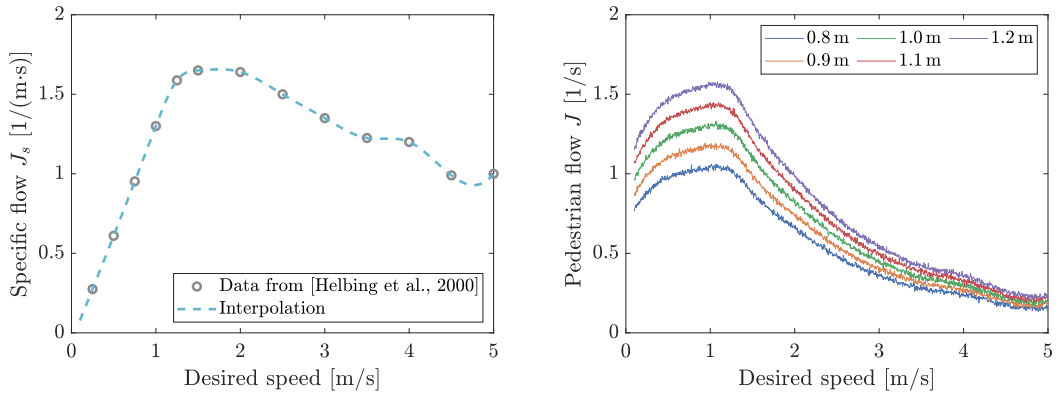
Figure 3.4: The utility $U_l(x)$ for the selected (white) agent balances the navigational field from origin on the right to destination (orange), obstacle repulsion (light gray) and utility dips (dark red) close to other agents (blue).

3.1.2 Social force model emulator for a bottleneck scenario

The majority of the simulations in this thesis are performed with the optimal steps model. However, I use an emulator of the social force model (SFM) for an egress scenario in which I calibrate the free-flow speed to point out an issue with calibration. In this scenario, 200 agents are leaving a room with one door. Thus, the scenario also shows a bottleneck dynamic. As explained in Section 2.2.4, the social force model is based on an ordinary differential equation that considers attracting forces towards the agent's destination and repulsive forces from other agents and obstacles. Solving the ODE provides the position for the agents at consecutive time steps.

The social force model presented by Helbing et al. [Helbing et al., 2000] exhibits a faster-is-slower dynamic in a bottleneck scenario, a much-discussed hypothesis in pedestrian dynamics stating that lower speeds can lead to a faster egress than higher speeds. Since I only use the model to demonstrate an issue with classical calibration, I refrain from implementing it. Unfortunately, I was not provided access to the original model. Instead, I build an emulator based on data presented in [Helbing et al., 2000]. I use cubic interpolation on the data for the relationship between the desired speed and specific flow. The desired speed is equivalent to the free-flow speed in the optimal steps model. The specific flow J_s denotes the flow per unit width. The interpolation is then used as a basis for the emulator: I derive the flow for the five different bottleneck widths from the specific flow. In order to resemble the randomness due to the initialization in the social force model, I add a Gaussian noise, $\mathcal{N}(0, 0.01)$, to the model. The resulting relationship between free-flow speed and flow can be seen in Figure 3.5.

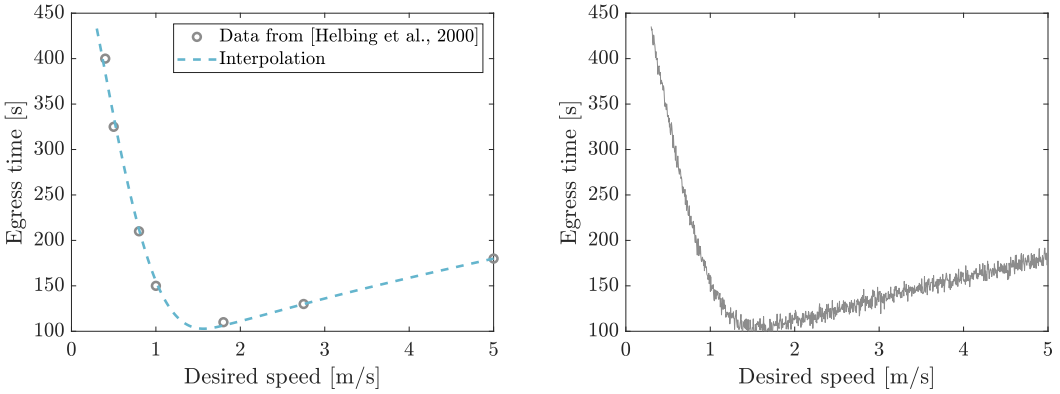
Analogously, I create an emulator for the relationship between desired speed and egress time, named leaving times in [Helbing et al., 2000]. Using the data presented in [Helbing et al., 2000], I perform cubic interpolation and add a Gaussian noise, $\mathcal{N}(0, \sigma^2 = 25)$. The resulting emulator is presented in Figure 3.6.



(a) Interpolation of specific flow data from [Helbing et al., 2000]

(b) Emulator for widths from 0.8 m to 1.2 m (including noise)

Figure 3.5: Social force model emulator for five different bottleneck widths, constructed with flow data from [Helbing et al., 2000] including an additive zero-mean Gaussian noise with variance $\sigma^2 = 10^{-4}$.



(a) Interpolation of egress times from [Helbing et al., 2000]

(b) Emulator including Gaussian noise

Figure 3.6: Social force model emulator for leaving times or egress times for a room with 200 people, constructed with egress times from [Helbing et al., 2000] including an additive zero-mean Gaussian noise with variance $\sigma^2 = 25$.

3.2 Studied scenarios

In this work, I examine two scenarios: a bottleneck scenario for which I investigate the parameters as well as an overpass in a train station for which I derive dynamic initialization.

3.2.1 Bottleneck scenario: crucial for improving safety

Bottleneck scenarios are widely investigated in experiments [Kretz et al., 2006, Liddle et al., 2009, Seyfried et al., 2009, Rupprecht et al., 2011, Liao et al., 2014] and simulations [Nishinari et al., 2004, Martinez-Gil et al., 2015, Gao et al., 2014] within the collective dynamics community. They are a part of all egress situations. Since constrictions lead to increased densities and delays in evacuations, bottlenecks are crucial for pedestrian safety. Additionally, flow can be used for capacity estimation, which has been taken up by guidelines for safety.

I reconstruct the bottleneck experiments performed by Seyfried et al. in the Vadere simulator [Seyfried et al., 2009]. The flow of pedestrians through bottlenecks was investigated in 18 experiments while the width of the bottleneck was varied between 0.8 and 1.2 meters (in 0.1 meter increments). Each experiment was carried out with 20, 40, and 60 participants. As the quantity of interest, I measure the flow at the end of the bottleneck,

$$J = \frac{N}{\Delta t} = \frac{N}{t_N - t_1}, \tag{3.11}$$

where N is the number of agents in the scenario and Δt is the difference between the time when the first agent crosses the measurement line t_1 and the time that the last agent crosses it t_N. The experiment was designed to measure the flow. Thus I choose the flow as the quantity of interest.

Figure 3.7 shows a snapshot of the simulation at 12 seconds. In this setup, 60 agents (blue) are moving from their origin (green) through the bottleneck created with obstacles (gray) to their destination (orange).

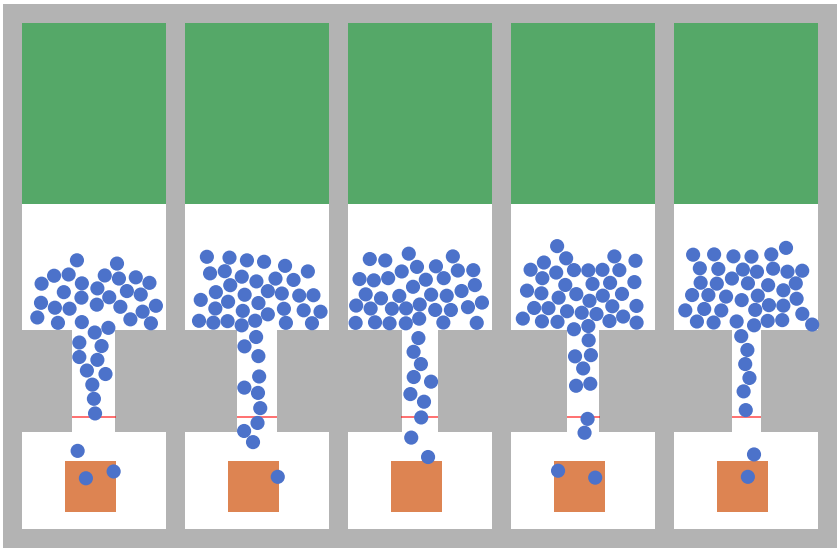


Figure 3.7: Snapshot of a simulation of the five bottlenecks after 12 s.

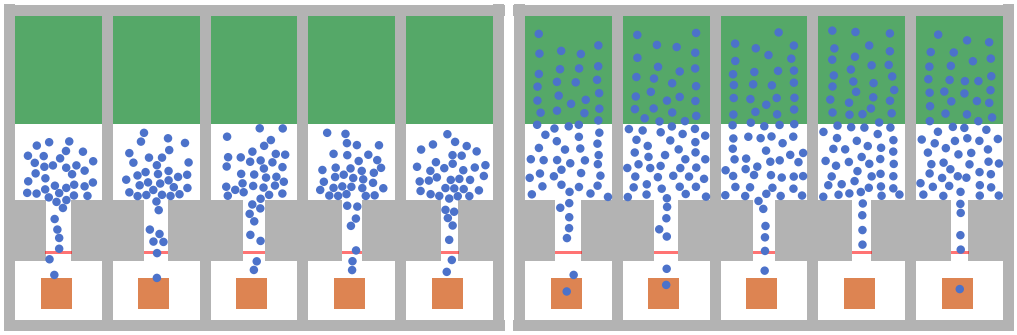
Uncertain input parameters in the bottleneck scenario Typically, bottleneck experiments in experimental pedestrian dynamics study the effect of door widths on the density in front of the opening. Thus, I study five bottlenecks with different widths ranging from 0.8 m to 1.2 m.

In order to obtain a specific model, I examine seven parameters of the optimal steps model. The first two parameters are the mean and the standard deviation of the free-flow speed. They define the Gaussian distribution from which the individual speeds are drawn. In addition, I also vary the number of agents in the simulation between 40 and 80 since the experiments were carried out with 40, 60, and 80 participants. When varying the number of agents, it is important to make sure that the dynamics remain the same. If the number of agents gets too low, there is no more crowding in front of the bottleneck, instead, agents would be able to move freely through the constriction. For the selected parameter range, the dynamics are maintained. Furthermore, I vary two parameters that define the obstacle repulsion and the personal space strength. They are the heights of the respective utilities, h_o, and h_p. The utilities are described in more detail in Section 3.1.1. Moreover, I modify the minimum step length for the agents. If a minimum step length larger than zero is chosen, the next position is discarded if its distance to the current position is smaller than the minimum step length. That implies the pedestrians are a bit more patient. Finally, I add a control parameter that has by design no effect on the simulation in this scenario. It serves for verification and as a reference to determine which parameters are influential. Uncertainty quantification methods typically require a distribution of the examined parameters reflecting all initial knowledge on the parameters. These prior parameter distributions for the parameters are summarized in Table 3.1.

Table 3.1: Uncertain parameters and their distribution used for the sensitivity analysis.

Index	Parameter	Unit	Range
1	Control parameter		$\mathcal{U}(1.0, 5.0)$
2	Free-flow speed mean	m/s	$\mathcal{U}(0.5, 2.2)$
3	Free-flow speed std	m/s	$\mathcal{U}(0.1, 1.0)$
4	Number of agents		$\mathcal{U}(40, 80)$
5	Obstacle repulsion h_o		$\mathcal{U}(2.0, 10.0)$
6	Personal space strength h_p		$\mathcal{U}(5.0, 50.0)$
7	Minimum step length	m	$\mathcal{U}(0, 0.15)$

Figure 3.8 shows snapshots of the simulation of the bottleneck scenario. On the left, the lower limits of all parameters are chosen; on the right, the upper limits of all parameters are chosen. We observe a large difference between the two extremes of the parameter set.



(a) Lower limits are chosen for all parameters

(b) Upper limits are chosen for all parameters

Figure 3.8: Effect of the uncertain parameters on the distribution of agents in front of and within the bottleneck (still taken after 12 s simulation time).

3.2.2 Train station overpass: multi-directional flow

In order to complement the parameter estimation, I estimate origin-destination matrices that are necessary for the dynamic initialization of a live simulation. For the origin-destination analysis, I study the passenger flow in an overpass of the Basel train station in Switzerland. Besides events and schools, public transport buildings (traffic hubs) such as train stations or airports are among the most studied locations in pedestrian dynamics.

Our scenario is an overpass over all platforms that leads pedestrians from the station hall to the platforms and connects both entrances of the stations, compare Figure 3.9. Stereo sensors that record pedestrian trajectories throughout the overpass monitor a section of this overpass. Within this area, there are stairs, escalators, and elevators leading from the overpass to the platforms and the station hall, as well as, ticket machines, benches, and shops. Swiss Federal Railways (SBB) provided trajectory data obtained by the stereo sensors. The topography is shown in Figure 3.10 together with 1% of the positions recorded throughout one day. It is similar to a long corridor, but it features several exits on both sides allowing for a multi-directional flow.

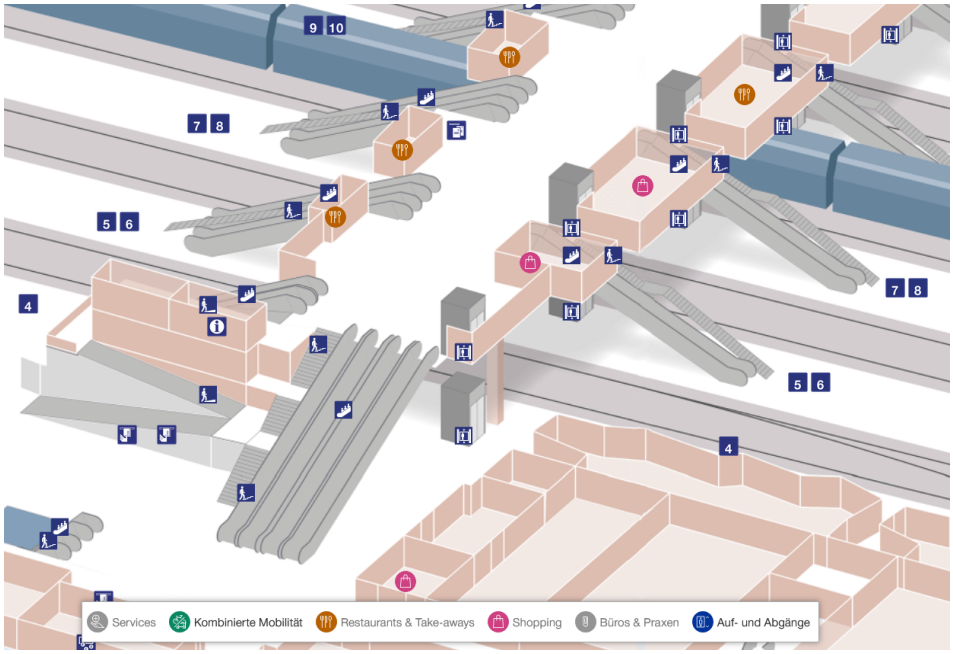


Figure 3.9: Plan of the train station overpass[1].

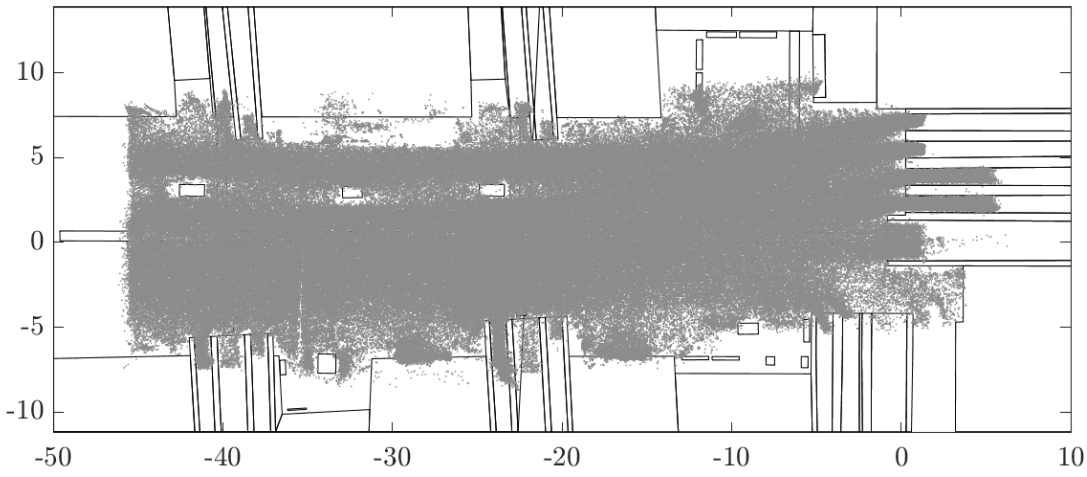


Figure 3.10: Recorded pedestrian positions (1%) in the train station overpass over one day. Data was provided by Swiss Federal Railways (SBB). All data in meters. On the right, escalators lead to the station hall whereas, on the left, the overpass continues. In the upper part, escalators and elevators lead to two platforms (at $x \approx -40\,\mathrm{m}$ and $x \approx -20\,\mathrm{m}$). Similarly, in the lower part, there are connections to three platforms at about -40,-20, and $0\,\mathrm{m}$.

[1] `https://plans.trafimage.ch/basel-sbb?lang=de&layer=basel_innenplan`, accessed at 20.08.2021, ©SBB/CFF/FSS, Geodaten: ©OpenStreetMap contributors, ©swisstopo (5704003351).

3.3 Vadere crowd simulation framework

The simulations of the bottleneck scenario are carried out using the open-source crowd simulation framework, Vadere [Kleinmeier et al., 2019] built by the pedestrian dynamics research group at Munich University of Applied Sciences. The framework includes implementations of several models: optimal steps model, social force model, cellular automaton, gradient navigation model [Dietrich and Köster, 2014], behavioral heuristics model [Seitz et al., 2016]. Vadere includes a graphical user interface (GUI) which can be used to set up a simulation, but also to evaluate simulations by displaying agents' trajectories, movement directions, densities, and many more. The implementations are verified by unit tests in a continuous integration setup. This setup also contains test scenarios defined in the RiMEA guidelines [RiMEA, 2016] i.e. standardized test cases for crowd simulations for the validation of the models.

Vadere has a modular structure so that it can be easily extended to integrate new functionality. The software follows a model view controller pattern [Gamma et al., 1994]. Following this structure, the code is separated in three packages: `VadereState` (model), `VadereGui` (view), and `VadereSimulator` (controller). For more information on the architecture, I refer to [Kleinmeier et al., 2019].

3.3.1 Core of the simulation: simulation loop

Independent of the model, the core of the implementation is the so-called simulation loop. The models use different update schemata for the agents, from parallel updates in which all agents are updated at the same time, via sequential updates, to the event-driven update scheme for the OSM. However, for all update schemes, we use an event queue in which all events, in particular, stepping events, of the agents are registered. In some models, that might be achieved by equidistant events such as parallel and sequential update schemes. The simulation loop iterates over all registered events. In every iteration, the update method of the locomotion model is called in order to calculate the next position(s). Consequently, all locomotion models need to implement an update method. For the simulation loop, it does not matter which locomotion model is used.

In every iteration of the simulation loop, the scenario elements, a psychology layer [Kleinmeier et al., 2020], and the locomotion are updated. Updating the topography includes, for example, spawning new agents as well as removing agents from the simulation that have reached their destinations. There are different stimuli that may, over time, become present in the simulation. The psychology layer evaluates whether the agent perceives the stimulus and if so, whether it adapts its behavior. As stated in the last paragraph, the update of the locomotion model triggers the calculation of the agent's next position.

After these three steps have been executed, a new state with new positions and stimuli is created and sealed. Based on the state, so-called output processors are evaluated. The processors calculate potential quantities of interest based on the current state. A standard output processor saves the current position of each agent to file. This is necessary for post-visualization. In addition, processors are available to evaluate speeds, density,

flow, egress time, and many more. Moreover, new processors can be implemented to evaluate custom measures.

3.3.2 Building a scenario

I now briefly describe, from a user perspective, what needs to be done in order to carry out a simulation. Central is the creation of a new scenario. The scenario holds all information necessary for the simulation, from the topography to the parameters of the models.

Part of the scenario is the topography, for example, a bottleneck. The topography includes sources where the agents are spawned and destinations to which they are headed. This information is usually clear in confined scenarios, but it can get complex in larger scenarios. Typical approaches here are to integrate data from surveys, observational data, as well as, experience of experts such as rail operators for train stations or event managers for regular festivities. In Chapter 7, I present a novel approach to learning entries of origin-destination matrices based on real-time monitoring data. Topographies can also integrate obstacles that represent any area on which the agent cannot step such as walls, non-solid ground like water, or streets with traffic. When real topographies are to be recreated in Vadere, it is important to think about how certain properties can be represented in the simulation. That might include obstacles in places where there are no walls because it is clear that pedestrians cannot pass e. g. because it is not allowed due to social norms or local conditions. For larger topographies such as streetscapes, an OpenStreetMap converter[2] can be utilized.

In addition, to the topography, the scenario also contains the models and submodels that are used for the simulation and the parameter set. In Vadere, the following models are implemented: optimal steps model, the gradient navigation model, a social force model, the optimal velocity model, Reynold's steering model, the behavioral heuristics model, and a biomechanics model. In addition, a cellular automaton can be obtained by configuring the optimal steps model accordingly [Kleinmeier et al., 2019]. For each of these models, a default configuration is available. The optimal steps model implementation includes the optimal stairs model [Köster et al., 2019] that models locomotion on stairs. The user needs to choose the appropriate model for the scenario at hand. In addition, submodels such as the centroid group model for small pedestrians groups can be added to the simulation. Recently, a psychology layer has been added to Vadere [Kleinmeier et al., 2020] that is independent of the locomotion model. It can be used to simulate cooperative behavior, such as a crowd that allows a single agent to pass through.

Finally, the scenario also holds the parameter choice for the chosen model and potentially submodels. Since the parameters vary among the models, I focus on the model-independent parameters here. In order to model a certain population, typically the free-flow speed of the agents is adapted. The free-flow speeds in the simulation are drawn

[2]The necessary steps to integrate data from OpenStreetMap into Vadere are described in `https://gitlab.lrz.de/vadere/vadere/tree/master/Tools/Converters/osm2vadere(extended)/osm_converter.py`.

from a truncated normal distribution. Consequently, there are four parameters: `speedDistributionMean`, `speedDistributionStandardDeviation`, `minimumSpeed`, and `maximumSpeed`. A high free-flow speed mean indicates a young, fit, target-oriented population while a low free-flow speed mean rather models an older population or kids. Similarly, a small standard deviation suggests a homogeneous population and a larger standard deviation a more heterogeneous population. In Section 3.2.1, I described all parameters of the optimal steps model that are studied in this work.

Once the scenario is defined, simulations can be carried out either using the GUI, or the command-line interface. The suq-controller[3] allows to run simulations with several sets of parameters. It is an interface for the communication between Vadere and the uncertainty quantification framework (see Section 4.5).

3.4 Stochasticity and noise

"As ecologists and biologists, we try to find the laws that govern the functioning and the interactions among nature's living organisms. Nature, however, seldom presents itself to us as a deterministic system." [Hartig et al., 2011]

Most models for crowd simulation include stochastic terms that reflect lack of knowledge. Ronchi refers to this as behavioral uncertainty [Ronchi et al., 2014]. Typically, stochasticity is introduced in the simulation when starting positions of the agents or individual characteristics such as free-flow speed, radius, or height, are assigned. Decisions made during the simulation, such as resolving conflicts in a parallel update scheme, constitute another source of randomness. Since attributes, such as speeds and starting positions, are indeed unknown and do vary, fixing them would unduly simplify the model. Stochastic terms lead to non-deterministic model output. Most methods for uncertainty quantification, however, are designed for deterministic models. In this section, we take a closer look at how and why stochastic terms are introduced in the optimal steps model, how they affect the simulation, and how we can handle them in order to run uncertainty analyses.

3.4.1 Stochastic terms in crowd simulations

It is common that locomotion models introduce stochastic terms [Duives, 2016, p. 146]. However, how and where stochasticity is introduced varies from model to model and even from implementation to implementation. This section refers to the optimal steps model implementation in Vadere.

I distinguish between direct and indirect introduction of stochastic terms. Direct introduction of stochastic terms includes the random assignment of an initial position for each agent within the source area. In Vadere, this can be deactivated by `spawnAtRandomPosition` and then the agents are spawned next to each other, starting in the lower-left corner of the source area. Without random spawning, the agents are not distributed equally over the source area and therefore local densities within the source may

[3] `https://gitlab.lrz.de/vadere/suq-controller`

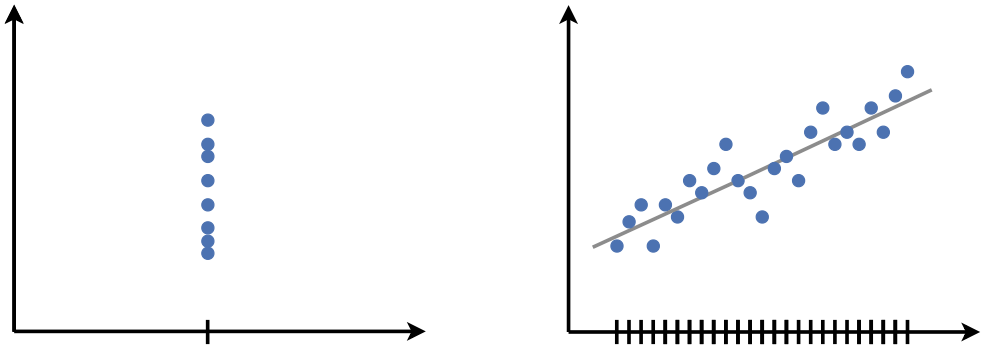
vary. As a consequence, at the beginning of the simulation, the agents may diverge in all directions. In addition, each agent in the simulation is assigned an individual free-flow speed. In Vadere, as in many other simulation frameworks, the speeds are drawn from a normal distribution. If the standard deviation of the distribution, `speedDistributionStandardDeviation` is larger than zero, the assigned free-flow speeds are stochastic. Moreover, the step length of each agent in the simulation is derived from its free-flow speed according to the experiments conducted and evaluated in [Seitz and Köster, 2012]. The relationship between free-flow speed and step length includes a Gaussian term too. By modifying `stepLengthSD` in Vadere, the size of this effect can be varied. With `stepLengthSD` equals zero, the relationship between free-flow speed and step length is deterministic. The free-flow speed is a central parameter in the simulation since it affects the simulation in several ways: As stated before, the step length is also derived from the free-flow speed. That means, the size of the disc on which the best next position is found. Free-flow speed and step length likewise determine the step frequency and therefore the event queue, for event-based models such as the optimal steps model.

All terms explained above refer to the initialization of the simulation. In addition, there are terms introduced in the simulation loop. In the default parameter setting for the optimal steps model, `varyStepDirection` is activated. The effect of this parameter depends on the optimizer chosen for the utility optimization. For the Nelder-Mead algorithm, `varyStepDirection` means that the initial positions of the optimizer in each iteration of the simulation loop are chosen differently. In each iteration, a random offset is drawn from $\mathcal{U}[0, 1]$.

I refer to all of the above as direct stochastic terms. Besides, there are indirect stochastic components. As described above, both free-flow speed and starting position are randomly assigned during the initialization phase. The mapping between both factors contains an indirect stochastic term that depends strongly on the implementation. If we want to exclude any stochasticity, it is not enough to make sure that the same speeds and positions are drawn, they also need to be assigned to the same agents. In Vadere, we have no direct access to this mapping. This is one example of how stochastic terms can be related and I refer to it as an indirect stochastic term.

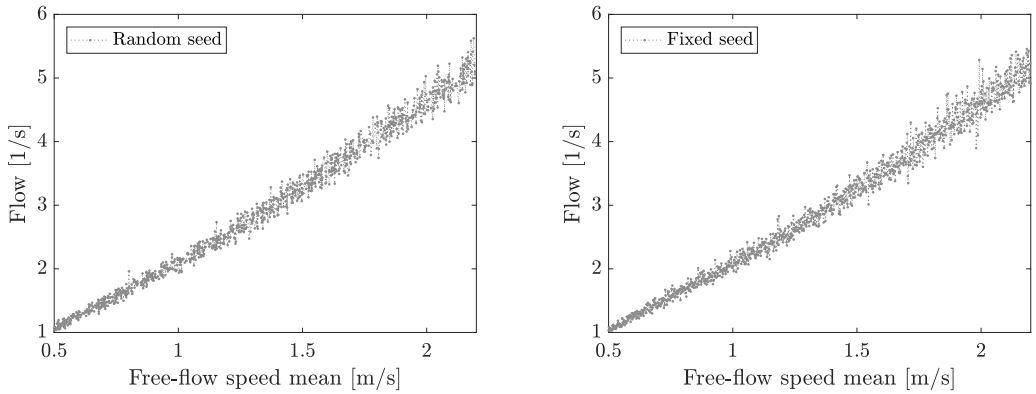
3.4.2 Effects of stochastic terms

We observe two effects due to randomness in the simulation, and I refer to them as stochasticity and noise: I state that stochasticity is present when the model output varies even though the model is evaluated at a fixed set of parameter values (compare Figure 3.11a). Additionally, I argue that noise is present if we observe large variations in the model output for similar parameter values even though all stochastic terms are fixed (compare Figure 3.11b) e. g., by fixing a seed for the pseudo-random number generator. While the first one, stochasticity, is an effect that I would intuitively expect, the latter may not be anticipated. However, we observe this effect in Vadere e. g. when simulating the flow through the bottleneck scenario and varying the free-flow speed while holding the seed of the simulation constant, as in Figure 3.12.



(a) Stochasticity **(b)** Noise

Figure 3.11: Stochasticity vs. noise: While stochasticity is an effect at a fixed parameter value or vector, noise is epitomized by non-monotonous variations when the parameter is varied around the fixed value.



(a) Random seed **(b)** Fixed seed

Figure 3.12: Noise in simulations of the bottleneck scenario with the optimal steps model: Even though, the seed of the pseudo-random number generator is fixed, the model output is sensitive to small changes in the variable, here the free-flow speed.

In order to understand where this noise stems from, we need to take a closer look at the random number generation. In Vadere, we draw all random numbers from one central seed. This allows us to fix this seed in order to reproduce the results. There is, however, still room for improvement. Ideally, from this central seed, we would generate a set of subkeys for different stochastic terms to decouple the stochastic terms. The current setting might introduce a higher sensitivity to small changes in the sequence of the simulation: The results change whether we first draw the free-flow speeds or first draw the starting positions. While this sounds like a mild side effect, it can have far-reaching consequences in the simulation. Let's say that we minimally vary the free-flow speed mean from its reference value. In this case, we draw slightly different free-flow speeds, even with the same seed. Based on that, also the step length and the stepping

events change. Consequently, other agents might be at different positions at the stepping events. This might also interfere with the spawning of agents.

Sometimes not all agents can be spawned directly at initialization because the source area is too small. In this case, during the simulation, random numbers are drawn for the positioning of the remaining agents. Small changes in the free-flow speed may lead to different positions, which may affect also the spawn time and starting position of further agents. In order to avoid issues due to spawning, it is important to note that for random spawning a significantly larger source is necessary than for equidistant placing since random spawning prevents optimal packing.

A similar situation can also arise if, for example, the number of agents is slightly varied from N to $N + 1$. For each term such as free-flow speeds, an additional random number is then drawn from the probability distribution. Therefore, starting positions and free-flow speeds might differ even though the same seed is used for N agents. This effect is further enhanced when random numbers are drawn during the simulation. Thereby, a small variation in the free-flow speed can have a larger effect on the simulation than expected.

While introducing subkeys could mitigate this issue to some degree, it does not resolve the issue completely. The described effect, that a small change in the speed might have a larger impact on the simulation by leading to a small change in the next position and stepping times which again lead to different positions of other agents, would not be mitigated. Also, the groups for the subkeys need to be carefully selected and might depend on the chosen model or even the update scheme. For example, with a parallel update scheme, conflicts typically arise and they are often solved using a random system so that different agents "win" over time. If this is the case, a separate subkey is necessary.

3.4.3 Handling stochasticity and noise

There are several ways how to deal with stochasticity and noise in the simulations. Ideally, I would only use methods that can deal with stochastic models. However, since uncertainty quantification methods are typically designed for deterministic systems, I am limited by the available methods. In Section 6.3.3, I use approximate Bayesian computation for Bayesian inference. This method is specifically designed for stochastic simulators. Another approach is to remove the stochasticity and noise from the simulation, or at least to reduce it. In order to remove both effects, I build a (deterministic) surrogate model in Section 6.4.1.1. However, choosing a suitable surrogate model is a complex and computationally demanding task, especially in higher dimensions. Therefore, I only use this approach in a one-dimensional setting. Instead, I can also average several simulations with the same set of parameter values to reduce the stochasticity at this fixed set. This also reduces the noise. In my investigations, averaging 10 repetitions at each parameter set turned out to be a good balance between accuracy and computational effort. The noise is reduced as a by-product of the averaging. Averaging is my default approach if not indicated otherwise.

3.5 Summary

In this chapter, I introduced the basic building blocks for my goal to find a specific model and quantify its uncertainties. I described the two scenarios that I examine in this work, the bottleneck scenario and the multi-directional flow in the train station. While the bottleneck scenario is modeled from a controlled experiment, Swiss Federal Railways (SBB) provided real-world trajectories for the overpass scenario. I use this data for the dynamic estimation of parameters necessary for real-time prediction.

After describing the scenarios, I elaborated on two locomotion models, the optimal steps model and an emulator for the social force model. Both are used in the thesis to simulate the flow through a bottleneck scenario.

The simulations in this manuscript are carried out with the crowd simulation framework Vadere. I outlined the main component of the simulator, the simulation loop that is driven by an event queue. In every stepping event, the locomotion model calculates the next position of an agent. Moreover, I explained which steps are necessary to build a simulation scenario from a use case such as the bottleneck experiment.

Finally, I described two effects observed in the model outcomes, stochasticity and noise. I explained why and where stochastic terms are introduced in the simulation and outlined their consequences. In this work, I handle stochasticity and noise in three ways: Where possible, I choose methods that are designed for stochastic simulators. When this is not possible, I either create a surrogate model that eliminates stochasticity and noise or I average a fixed number of simulations to reduce stochasticity, which also diminishes the noise.

4 Uncertainty quantification framework

In this chapter, I describe the framework for the systematic analysis of parameters in crowd simulations which I designed and implemented as part of this thesis. I perform all analyses throughout this manuscript with it. I defined the requirements for the framework (Section 4.1) and the state-of-the-art software for uncertainty quantification (Section 4.2). Based on both, I argue why I developed a new framework and describe its architecture in Section 4.3. In Section 4.4, I present core algorithms. I outline the interface with Vadere in Section 4.5. In Section 4.6, we take a look at how the routines are tested. Finally, I summarize the key points of the design and implementation of the framework.

The reader should be familiar with uncertainty quantification terminology. I refer to Section 2.3 for a brush-up and introduction to uncertainty quantification. Descriptions of the methods implemented in this chapter as well as information on the choice of methods can be found in Section 5.3 for sensitivity analysis, Section 6.3 for parameter estimation, and Section 8.3 for uncertainty analysis.

Requirements addressed in this chapter

R1 Choose or design a software for uncertainty quantification for crowd simulations

- R1.1 Derive requirements for a software for uncertainty quantification for crowd simulations
- R1.2 Analyze available software regarding the requirements
- R1.3 Design and implement a software that fulfills the requirements

4.1 Requirement analysis

Figure 4.1 shows the stages of agile software development: Based on requirements, a design for the software is set up. Next, the software is developed and tested. Development can also be performed in a test-driven manner which means that first the tests are written and then the functional code is implemented and needs to pass the tests. If the tests are successful, the software is deployed which means the software is provided to the users, for example, through a new release. Subsequently, the software is reviewed. In classical software engineering, these stages are planned and executed once for the whole project. In agile development, the stages are performed for each sprint. A sprint is a fixed time during which a set of requirements is implemented. After a sprint, requirements are again assessed and evaluated.

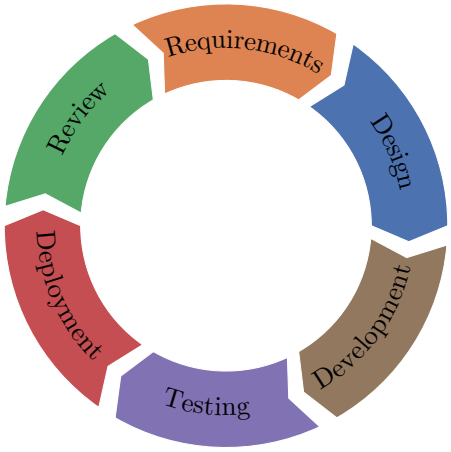


Figure 4.1: Agile software cycle.

The collection of requirements is a typical task at the beginning of every software project. When describing the requirements for the uncertainty quantification software, I distinguish functional from non-functional requirements. The former can be thought of as features of the software, the latter as properties. The requirements are prioritized by importance and analyzed according to their feasibility. Due to limited resources, usually, not all requirements can be fulfilled. Table 4.1 lists the functional and non-functional requirements for the uncertainty quantification framework. Non-functional requirements are typically less measurable than functional requirements. Therefore, it should be examined whether they can be converted into functional requirements through specification. As an example, NFR4 can be turned into a functional requirement by specifying a code coverage.

Table 4.1: Functional and non-functional requirements for the uncertainty quantification framework developed in this thesis.

Functional requirements (FR)	
FR1	Support existing interfaces to Vadere: Provide an interface to the suq-controller which can be used for communication with Vadere.
FR2	Support crowd dynamics models in Vadere: The methods provided in the framework should be suitable for crowd simulation and work with all models provided in Vadere.
FR3	Provide routines for parameter identification: Implement methods for global sensitivity analysis to identify influential parameters in crow simulations.

FR4 Provide routines for parameter estimation:
Implement methods for Bayesian inference to estimate parameter distributions from experimental data.

FR5 Provide routines for uncertainty analysis:
Implement methods to quantify the uncertainty in the simulation output by propagating uncertain input parameters.

FR6 Ensure reproducibility of the results:
Assure that the results are reproducible especially for sampling methods that employ pseudo-random number generators such as Monte Carlo sampling.

FR7 Make results persistent:
Store results in a text file with a simple format so that they can be post-processed with third-party software.

FR8 Provide a graphical user interface:
Optional requirement. Especially for external Vadere users, a GUI simplifies handling of the software. Alternatively, predefined run scripts should be provided.

Non-functional requirements (NFR)

NFR1 Modular design:
Separate the algorithms from the infrastructure to ensure that the framework is easily expandable with additional methods.

NFR2 Adhere to coding style:
Follow established coding styles to increase readability.

NFR3 Open-source:
Provide the framework open-source and therefore only integrate open-source solutions

NFR4 Well-tested:
Carefully verify and validate the provided algorithms.

4.2 State of the art on uncertainty quantification software

There are several open-source frameworks for uncertainty quantification, compare Table 4.2. Based on the requirements, I analyze which frameworks are suitable for our purposes. Differences from the requirements are stated as "disqualifiers". However, I am not aware of any framework that fulfills all of the mandatory requirements discussed in the last section.

Table 4.2: Open-source frameworks for uncertainty quantification as of September 3rd, 2021, alphabetically sorted.

Name	**Chaospy**
Institution	University of Oslo
Language	Python

License	MIT
Routines for	Uncertainty analysis with advanced Monte Carlo methods or polynomial chaos expansions and sensitivity analysis with Sobol' indices
Reference	[Feinberg and Langtangen, 2015]
Website	`https://chaospy.readthedocs.io/en/master/`
Repository	`https://github.com/jonathf/chaospy`
Latest commit	August 2021
Disqualifier	No routines for parameter estimation

Name	**Dakota**
Institution	Sandia National Laboratories
Language	`C++`
License	LPGLv3 (version > 5.0),
Routines for	Parameter estimation with nonlinear least squares or Bayesian inference, sensitivity analysis, uncertainty analysis
Reference	[Adams et al., 2014]
Latest release	August 2021 (v6.14)
Website	`https://dakota.sandia.gov/`
Disqualifier	Not implemented in Python, no Python interface available

Name	**MUQ** (**M**IT **u**ncertainty **q**uantification)
Institution	Uncertainty quantification group at Massachusetts Institute of Technology
Language	`C++`/ Python library
License	BSD-3-Clause
Routines for	Bayesian inference with Markov chain Monte Carlo approach, polynomial chaos expansion, uncertainty analysis, sensitivity analysis
Website	`https://mituq.bitbucket.io/source/_site/index.html`
Repository	`https://bitbucket.org/mituq/muq2/src/master/`
Latest commit	July 2021
Disqualifier	Global sensitivity analysis only via generalized polynomial chaos expansion.

Name	**OpenCOSSAN**
Institution	University of Liverpool
Language	Matlab toolbox
License	Toolbox: LGPLv3, Matlab: proprietary
Routines for	Sensitivity analysis, Monte Carlo sampling, reliability analysis, surrogate models including polynomial chaos expansion
Reference	[Patelli, 2016]
Website	`http://cossan.co.uk/wiki/index.php/Main_Page`
Repository	`https://github.com/cossan-working-group/OpenCossan`
Latest commit	October 2020
Disqualifier	Not implemented in Python, no Python interface available, requires Matlab license

Name	**OpenTURNS** (**open** source initiative for the **t**reatment of **u**ncertainties, **r**isks'**n** **s**tatistics)
Institution	Airbus Group, EDF Research and Development, IMACS, ONERA, Phimeca Engineering
Language	`C++` / Python library
License	LGPLv3
Routines for	Sensitivity analysis, Bayesian inference
Reference	[Baudin et al., 2017]
Website	`http://openturns.github.io/openturns/master/index.html`
Repository	`https://github.com/openturns/openturns`

Latest commit	July 2021
Disqualifier	No routines for uncertainty analysis provided
Name	Π**4U**
Institution	ETH Zürich, Chair of Computational Science
Language	C
License	GPL2.0
Routines for	Bayesian inference, uncertainty propagation
Reference	[Hadjidoukas et al., 2015]
Website	`http://www.cse-lab.ethz.ch/research/projects/pi4u/`
Repository	`https://github.com/cselab/pi4u`
Latest commit	April 2018
Disqualifier	Not developed in Python, no Python interface, no active development, no routines for sensitivity analysis provided
Name	**PROMETHEE**
Institution	Institut de Radioprotection et de Sûreté Nucléaire
Language	R
License	Apache License 2
Routines for	Sensitivity analysis, uncertainty propagation
Website	`http://promethee.irsn.fr/`
Latest release	March 2017
Disqualifier	Not implemented in Python, no Python interface available, no active development, no routines for parameter estimation
Name	**PSUADE** (**p**roblem **s**olving environment for **u**ncertainty **a**nalysis and **d**esign **e**xploration)
Institution	Lawrence Livermore National Laboratory
Language	`C++`
License	LGPLv2.1
Routines for	Sampling, sensitivity analysis, parameter estimation with Markov chain Monte Carlo approach
Reference	[Tong, 2017]
Website	`https://computing.llnl.gov/projects/psuade/`
Repository	`https://github.com/LLNL/psuade`
Latest commit	February 2018
Disqualifier	Not implemented in Python, no Python interface available, no active development
Name	**Queso** (**q**uantification of **u**ncertainty for **e**stimation, **s**imulation and **o**ptimization)
Institution	Center for Predictive Engineering and Computational Sciences at the University of Texas at Austin
Language	`C++`
License	LGPLv2.1
Routines for	Statistical inverse problems, uncertainty analysis
Reference	[Prudencio and Schulz, 2012]
Website	`http://libqueso.github.io/queso/html/index.html`
Repository	`https://github.com/libqueso/queso`
Latest commit	August 2019
Disqualifier	Not implemented in Python, no Python interface available, no active development, no routines for sensitivity analysis

Name	**Tasmanian** (**t**oolkit for **a**daptive **s**tochastic **m**odeling **a**nd **n**on-**i**ntrusive **a**pproximatio**n**)
Institution	Oak Ridge National Laboratory
Language	Python / `C++` library
License	BSD-3-Clause
Routines for	Parameter estimation with Bayesian inference with an adaptive Metropolis algorithm
Reference	[Stoyanov et al., 2013]
Website	`https://tasmanian.ornl.gov/`
Repository	`https://github.com/ORNL/Tasmanian`
Latest commit	September 2021
Disqualifier	Framework is focused on sparse grids, for parameter estimation only one method is provided, no routines for sensitivity analysis and propagation

Name	**UQEF** (**u**ncertainty **q**uantification **e**xecution **f**ramework)
Institution	Technical University of Munich
Language	Python
Routines for	Uncertainty analysis with Monte Carlo methods or polynomial chaos
Reference	[Künzner, 2021, p. 75ff]
Disqualifier	Not yet publicly available, provides only routines for uncertainty analysis

Name	**UQLAB** (**u**ncertainty **q**uantification in Mat**lab**)
Institution	ETH Zürich, Chair of Risk, Safety and Uncertainty Quantification
Language	Matlab [1]
License	UQLabCore: proprietary, UQLabModules: BSD-3-Clause, Matlab: proprietary
Routines for	Sensitivity analysis, Bayesian inference, surrogate models including polynomial chaos expansion
Reference	[Marelli and Sudret, 2014]
Website	`www.uqlab.com`
Latest release	February 2021
Disqualifier	Not developed in Python, no Python interface, requires Matlab license

Name	**UQ-PyL** (**u**ncertainty **q**uantification **Py**thon **l**aboratory)
Institution	GCESS, China
Language	Python
License	GPL
Routines for	Sensitivity analysis, reliability analysis, surrogate modeling, uncertainty propagation
Reference	[Wang et al., 2016]
Website	`http://www.uq-pyl.com/`, `http://www.uq-pyl.com/file/Wang_EMS_2015.pdf`
Latest release	October 2015
Disqualifier	No active development

Name	**Uranie**
Institution	French Atomic Energy Commission (CEA)
Language	`C++`, based on the ROOT framework, interfaces to Python and `C++`
License	LGPLv3
Routines for	Sensitivity analysis, inverse uncertainty quantification, polynomial chaos expansion, uncertainty analysis, surrogate models

[1] A beta version of a Python-based version, UQ[py]Lab, which is currently developed is available at `https://uqpylab.uq-cloud.io/`.

Reference	[Blanchard et al., 2019]
Website	`https://sourceforge.net/projects/uranie/`
Latest release	December 2020, v4.5.0
Disqualifier	Several external dependencies, e. g., ROOT framework
Name	**UQTk** (**UQ T**ool**k**it)
Institution	Sandia National Labs
Language	`C++`/Python (PyUQTk) library
License	BSD-3-Clause
Routines for	Uncertainty analysis, sensitivity analysis, surrogate construction, Bayesian inference
Reference	[Debusschere et al., 2017]
Website	`https://www.sandia.gov/uqtoolkit/`
Repository	`https://github.com/sandialabs/UQTk`
Latest commit	July 2021
Disqualifier	Not well-tested under Windows.
Name	**SALib** (**S**ensitivity **A**nalysis **Lib**rary)
Institution	University of California and University of Oxford
Language	Python
License	MIT
Routines for	Sensitivity analysis
Reference	[Herman and Usher, 2017]
Website	`https://salib.readthedocs.io/en/latest/`
Repository	`https://github.com/SALib/SALib`
Latest commit	September 2021
Disqualifier	Provides only routines for sensitivity analysis

4.3 Architecture of the framework

Even though there are several frameworks, none of them fulfills all of the mandatory requirements (FR1-7). Therefore, I decided to develop a new framework. This comes with several advantages: First, in my framework, I have high flexibility. I can tailor the software to Vadere, e. g. by using Python for which there are already Vadere add-ons (FR1, FR2). Additionally, the framework can be lightweight. There is no unnecessary overhead due to supporting multiple applications. While in principle any crowd dynamics model can be analyzed with the methods provided in the framework, the choice of methods is tailored to Vadere (FR2). Moreover, I maintain independence. In Vadere, we aim to integrate as few external packages as possible. In the design of the framework, I adhere to this principle. As a consequence, the framework is not dependent on external interfaces that might change with new releases. However, I have integrated some packages that fit the requirements. That includes the suq-controller[2] as an interface to Vadere, SALib [Herman and Usher, 2017] for global sensitivity analysis with Sobol' indices using the Sobol' sequence, and chaospy for polynomial chaos expansions [Feinberg and Langtangen, 2015]. Finally, my framework serves as a learning tool. Implementing as well as testing the core algorithms improves the understanding of the methods. In

[2] `https://gitlab.lrz.de/vadere/suq-controller`

order to fulfill NFR3, the framework is open-source and published under the GNU Lesser Public License v3.0, like Vadere, on Gitlab[3].

I developed the framework in an object-oriented fashion which is natively supported by Python. Object-oriented software benefits from reusability through inheritance and flexibility by polymorphism. I designed the framework in a modular way so that methods can be exchanged, combined, and extended easily (NFR1) as the architecture shows, compare Figure 4.2. The software is structured as follows: The core algorithms can be found in the implementations of the `Calculator`. Each `Calculator` holds a `Parameter` object with the configuration of the study and a `Result` object that is empty at start and filled by `Calculator`'s routines. A `Parameter` object contains a `Model`, in this work, mainly the `VadereModel`, and a parameter distribution, `Prior`. These two ingredients are necessary for all uncertainty quantification approaches. The term "prior" comes from Bayesian statistics. Strictly speaking, in this software, this term is only accurate for the Bayesian inference routines. For the other routines, it refers to the distribution of the uncertain parameters. The `Prior` object is generated from a user-supplied dictionary with the attributes of the distribution by a `PriorFactory`, following the factory pattern. In order to speed up the computations, the model can be replaced by a surrogate model that is cheaper to evaluate. Based on the requested surrogate type, `SurrogateFactory` generates a `Surrogate`, also following the factory pattern. Once the calculation is performed and the `Result` object is filled, the results can be written to file by a `FileWriter` and visualized by a `Plotter`. The implementation of the `FileWriter` fulfills the persistency requirement (FR7). `FileWriter` uses a `DataSaver` that holds a file path and provides some helper function to save variables as well as summaries of the results to file.

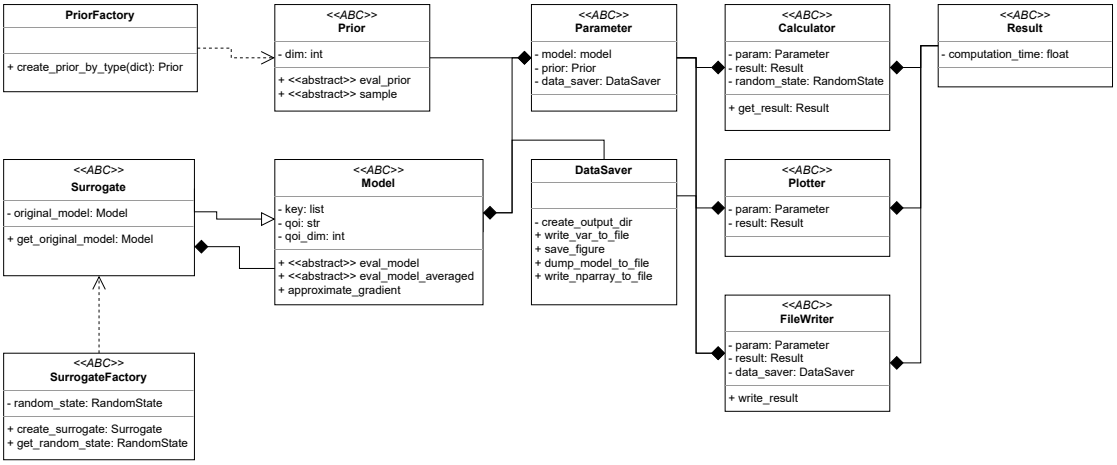


Figure 4.2: UML diagram for the uncertainty quantification framework.

The uncertainty quantification framework consists of three parts: (1) parameter identification, (2) parameter estimation, and (3) uncertainty analysis. The main components

[3] `https://gitlab.lrz.de/vadere/uncertainty-quantification`

for each part are presented in a UML diagram of the `Calculator` implementations. Figure 4.3 provides a schematic overview of the methods implemented in the framework. For parameter identification, Sobol' indices, activity scores, and derivative-based global sensitivity metrics (DGSM) are implemented. Activity scores and DGSM are derived from gradients or gradient approximations while Sobol' indices only require model evaluations. A detailed description of the indices can be found in Section 5.3. In order to estimate parameters, the framework provides a likelihood-based sampling approach, a Markov chain Monte Carlo algorithm, and a likelihood-free sampling approach, an algorithm for approximate Bayesian computation, described in Section 6.3. For uncertainty analysis, both a sampling-based method, Monte Carlo sampling, and propagation based on generalized polynomial chaos expansion can be employed. An extensive description is given in Section 8.3.

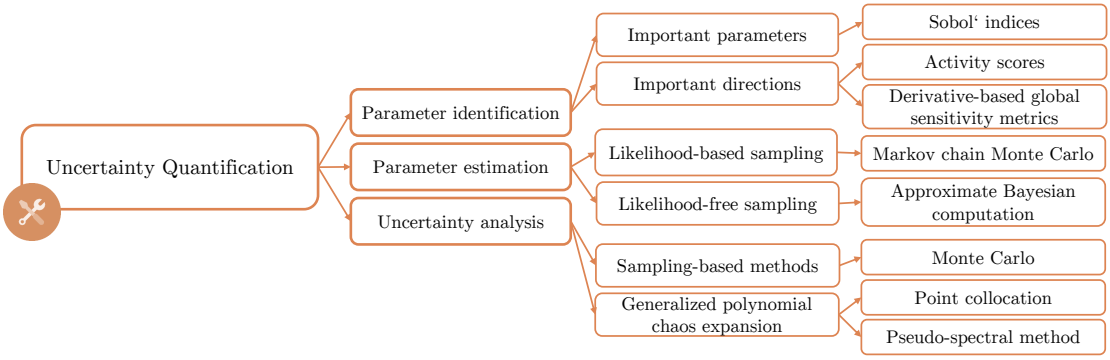


Figure 4.3: Schematic description of the uncertainty quantification framework, presented in [Gödel et al., 2019a].

4.3.1 Parameter identification

Figure 4.4 provides an overview of the implemented routines for parameter identification. FR3 requires the implementation of global sensitivity analysis methods. I provide two approaches as implementations of the `Calculator` base class: The `SobolIndexCalculator` for total and first-order Sobol' indices and the `ActiveSubspaceCalculator` for derivative-based global sensitivity metrics and activity scores. For each type, I give two specific calculation methods: Sobol' indices can either be calculated through the software package SALib [Herman and Usher, 2017] (`SobolIndexCalculatorSALib`) that uses a Sobol' sequence for the samples or with the implementation in `SobolCalculatorMC` based on Monte Carlo samples.

The central part of determining activity scores, first eigenvector metric, or derivative-based global sensitivity metrics, is the uncentered covariance matrix of the model gradients C. This matrix can either be calculated from gradients or gradient approximations using finite differences (`ActiveSubspaceCalculatorGradient`) or based on a local linear model (`ActiveSubspaceCalculatorLLM`).

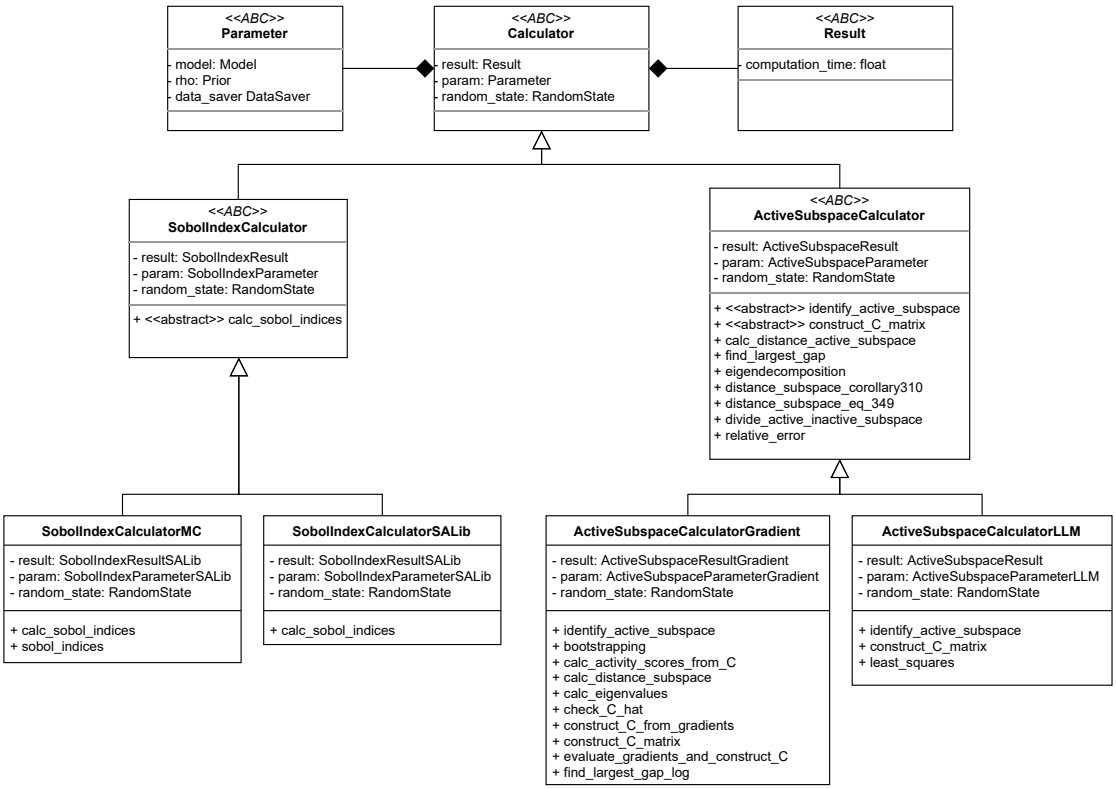


Figure 4.4: Routines for parameter identification implemented in the uncertainty quantification framework.

4.3.2 Parameter estimation

Following the outlined overall structure of the framework, I introduce the `InversionCalculator` as the central element for parameter estimation. Depending on the parameterization, the `InversionCalculator` can show different behavior. To map the parameterization, I choose the strategy pattern [Gamma et al., 1994]. It has the advantage that the implementation of the `InversionCalculator` is independent of the strategies. Only the strategies depend on concrete implementations. A code snippet that shows how the strategy is invoked is presented in Listing 4.1. Figure 4.5 shows the UML diagram featuring the `InversionCalculator` and the `SamplingStrategy`. Using the strategy pattern ensures the modularity of the framework (NFR1). I provide two sampling strategies: rejection sampling, a method from likelihood-free Bayesian inference, and random walk Metropolis algorithm, which is a Markov chain Monte Carlo approach for likelihood-based Bayesian inference. Both methods are suitable for the crowd dynamics models implemented in Vadere (FR2, FR4).

```
1    inv_calc = InversionCalculator(param, result)
2    rejection_strategy = RejectionSampling(param)
```

```
3       inv_calc.set_sampling_strategy(rejection_strategy)
4
5       # implicitly invokes concrete strategy implementation
6       inv_calc.inversion()
7
8       inv_results = inv_calc.get_result()
```

Listing 4.1: Invoking a sampling strategy in the `InversionCalculator`.

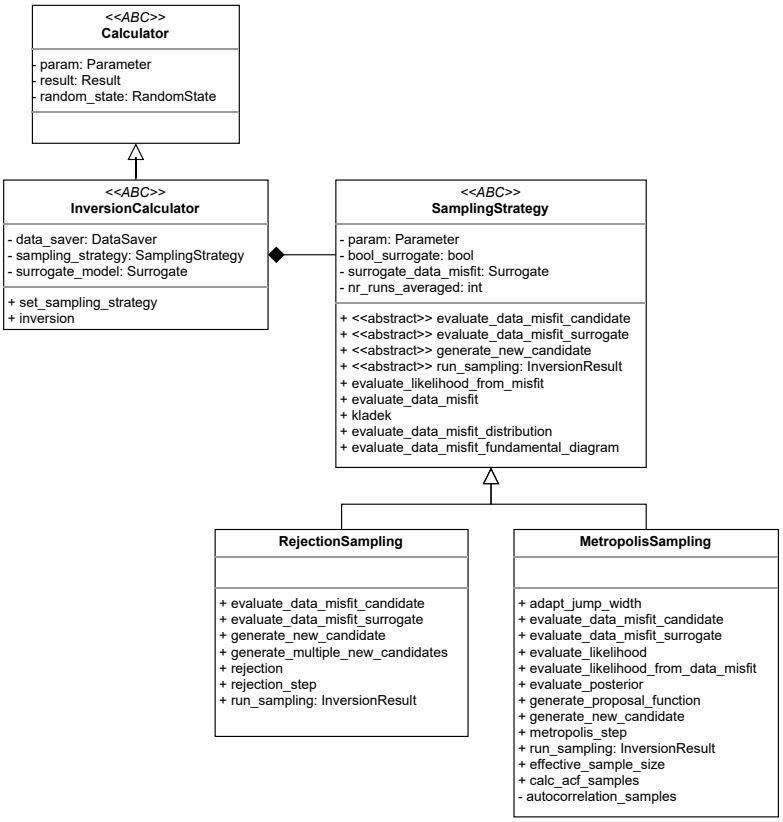


Figure 4.5: Routines for parameter estimation implemented in the uncertainty quantification framework.

4.3.3 Uncertainty analysis

For uncertainty analysis, the framework provides routines for Monte Carlo sampling and generalized polynomial chaos, fulfilling FR5. In Figure 4.6, a UML diagram of the uncertainty analysis part of the framework is illustrated. I introduce the `PropagationCalculator` which can directly be used for the Monte Carlo approach. Random samples are generated and propagated through the model. Optionally, the response can also be averaged at each parameter vector over several repetitions. When using chaospy [Feinberg and Langtangen, 2015], low-discrepancy schemes as Halton or Sobol' sequences,

and Latin Hypercube sampling become readily available[4] for sampling. For generalized polynomial chaos expansion, the `PropagationCalculator` is extended by the `PropagationCalculatorGPCE` which creates the nodes, and the in case of pseudo-spectral projection also weights, according to either the interpolation/regression or quadrature scheme. Then this calculator performs the regular propagation for the nodes and afterward calculates the expansion by interpolation/regression, or quadrature. Besides generalized polynomial chaos expansion (gPCe), there are two more specific implementations of the `PropagationCalculator`: `PropagationCalculatorPosterior` which redefines the `generate_samples` method in order to use samples of a known posterior distribution, e.g. obtained by Bayesian inference and `PropagationCalculatorPointEstimate` for propagation of a point estimate used for parameter estimation. Both implementations are used for the comparison between Bayesian inference and point estimation in Section 6.4.3.

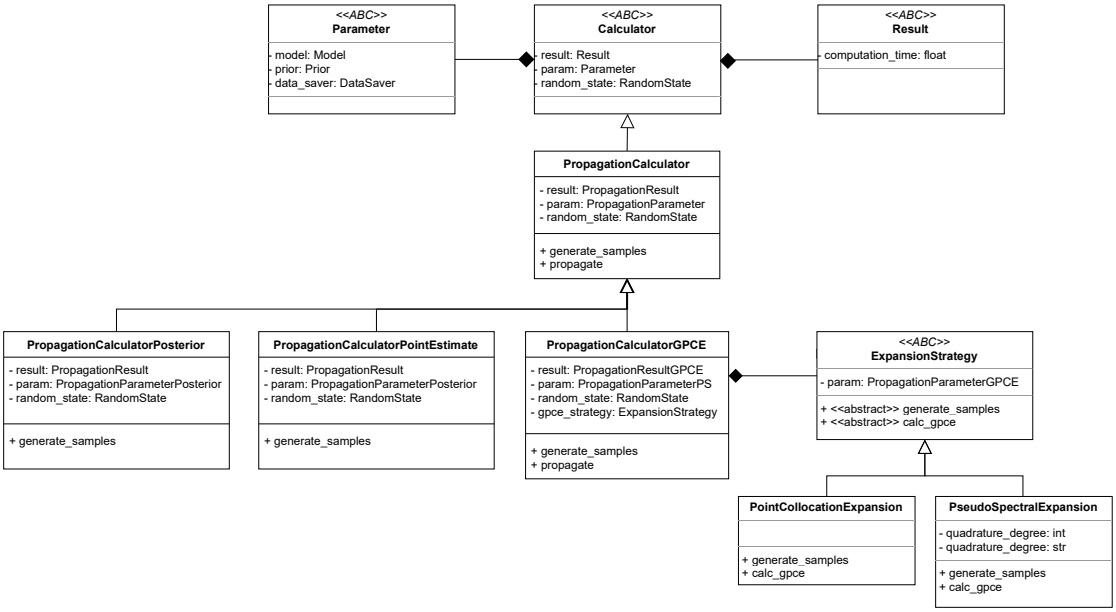


Figure 4.6: Routines for uncertainty analysis implemented in the uncertainty quantification framework.

4.4 Algorithms

In this section, the core algorithms for the three parts of the framework, parameter identification, parameter estimation, and uncertainty analysis, are presented. The implementation of the mathematical methods is shown in pseudocode listings.

[4] `chaospy.Distribution.sample()` creates pseudo-random generated samples, more details can be found at `https://chaospy.readthedocs.io/en/master/api/chaospy.Distribution.sample.html`.

4.4.1 Parameter identification: Sobol' indices and activity scores

Listing 4.2 shows the implementation of the calculation of Sobol' indices described in Section 5.3.1. I implemented Jansen's method [Jansen, 1999, Saltelli et al., 2010] for the approximation of the Sobol' indices. It generates two matrices, `A` and `B` filled with samples. Then, it iteratively generates the matrix `ABi`, which is matrix `A` except for row `i` which stems from matrix `B`.

```
1  def sobol_indices(model, N, rho):
2
3      # generate random samples
4      samples = rho.sample(2*N)
5
6      # split samples in two parts: A and B
7      A = samples[:, 0:N]
8      eval_A = model.eval_model(A)
9      variance_A = variance(eval_A)
10     B = samples[:, N:2*N]
11
12     # for first-order indices, we need evaluations of B
13     eval_B = model.eval_model(B)
14     variance_AB = variance(concatenate(eval_A, eval_B))
15
16     for i in range(0, m):
17         ABi = A
18         ABi[i, :] = B[i, :]
19         eval_ABi, _, _ = model.eval_model(ABi)
20
21         eval_first_jansen = eval_B - eval_ABi
22         first_order_jansen[i] = 1 - (dot_product(eval_first_jansen,
               eval_first_jansen) / (2*N) /variance_AB)
23
24         eval_total_jansen = eval_A - eval_ABi
25         total_order_jansen[i] = dot_product(eval_total_jansen,
               eval_total_jansen) / (2*N*variance_AB)
26
27     return first_order_jansen, total_order_jansen
```

Listing 4.2: Calculation of Sobol' indices according to Jansen's formula [Jansen, 1999, Saltelli et al., 2010].

In Listing 4.3, the implementation of the calculation of the activity scores and the identification of the active subspace as explained in Section 5.3.2 are shown. First, the parameter density for the p input parameters is transformed to the hypercube $[0,1]^p$. Then, we draw samples from the density. We evaluate the gradients or approximate them at each sample to construct the uncentered covariance matrix `C_hat` of the model gradients. From `C_hat`, we calculate the activity scores, which are one of my sensitivity metrics.

```
def identify_active_subspace(lower_transformed, upper_transformed, dim
    , alpha, k):

    # transform parameter density to [0,1] hypercube
    rho = UniformGenMult(lower_transformed, upper_transformed, dim)

    # calculate number of samples
    n_samples = ceil(alpha * k * log(dim)) # number of samples

    # generate samples according to the density of the uncertain
        parameters
    samples = rho.sample(n_samples)
    samples_transformed = transform_coordinates_from_unit(samples)

    # construct C matrix approximation with gradient approximations
    C_hat = evaluate_gradients_and_construct_C(samples_transformed,
        n_samples)

    # calculate activity scores (includes separation of active and
        inactive subspace)
    activity_scores = calc_activity_scores_from_C(C_hat, rho)

    return activity_scores
```

Listing 4.3: Identifying an active subspace to calculate activity scores according to Algorithm 1.1 from Constantine [Constantine, 2015, p. 4].

4.4.2 Parameter estimation: Metropolis algorithm and rejection sampling

For parameter estimation, the framework provides two strategies: The Markov chain Monte Carlo based Metropolis algorithm is implemented in `MetropolisSampling` (Listing 4.5) and the likelihood-free rejection sampler in `RejectionSampling` (Listing 4.4). Each `SamplingStrategy` needs to define a method to generate new candidates, `generate_new_candidate` as well as a method that runs the sampling scheme, `run_sampling`. The framework also provides the option to create a surrogate model for the distance measure function and run the sampling on the surrogate.

In this work, I employ the Metropolis algorithm and the ABC rejection algorithm. The framework can easily be extended with other sampling strategies, such as likelihood-free Markov chain Monte Carlo with a global comparison to a threshold for ABC [Marjoram et al., 2003].

```
def rejection(n_candiates, tolerance):

    # generate n_candidates new candidate
    candidates = generate_multiple_new_candidates(n_candidates)

    # evaluate distance measure at candidates
    distance_measure = evaluate_distance_measure(candidates)

```

```
9       # find candidates which fulfill the criterion
10      samples = candidates[distance_measure <= tolerance]
11
12      # calculate acceptance rate
13      acceptance_rate = len(candidates_accepted) / n_candidates
14
15      return samples, acceptance_rate
```

Listing 4.4: Rejection sampler for approximate Bayesian computation.

```
1   def metropolis_step(last_sample, posterior_last_sample, jump_width):
2
3       # generate new candidate
4       candidate = generate_new_candidate(last_sample, jump_width)
5
6       # evaluate model to calculate likelihood
7       data_misfit = evaluate_data_misfit_candidate(candidate)
8
9       # evaluate posterior
10      likelihood_candidate = evaluate_likelihood_from_data_misfit(
            data_misfit)
11      posterior_candidate = evaluate_posterior(prior_candidate,
            likelihood_candidate)
12
13      if posterior_candidate >= posterior_last_sample: # accept
14          sample = candidate
15          posterior_sample = posterior_candidate
16      else
17          u = uniform(1)
18          acceptance_criterion = posterior_candidate /
                posterior_last_sample
19          if u <= acceptance_criterion
20              sample = candidate
21              posterior_sample = posterior_candidate
22          else
23          sample = last_sample
24          posterior_sample = posterior_last_sample
25
26      return sample, posterior_sample
```

Listing 4.5: One iteration of the random walk Metroplis algorithm.

4.4.3 Uncertainty analysis: Monte Carlo sampling and generalized polynomial chaos expansion

For uncertainty analysis, the framework provides routines for Monte Carlo sampling and propagation based on generalized polynomial chaos expansion. Listing 4.6 shows the essential steps of propagation using generalized polynomial chaos. For calculating the coefficients, different strategies can be used: point collocation or the pseudospectral approach. The expansion is calculated by an `ExpansionStrategy` which can either be

`PointCollocationExpansion` which works with regression or `PseudoSpectralExpansion` based on quadrature. The latter requires a `quadrature_rule` in addition to the `expansion_order`, typically Gaussian quadrature, and a `quadrature_degree` which is the order of the quadrature. I implement a strategy pattern so that the choice of strategy is independent of `PropagationCalculatorGPCE`.

```
1  def propagate(rho, expansion_order):
2
3      # generate samples from prior
4      distribution = translate_prior_to_chaospy_distribution(rho)
5
6      # invokes concrete strategy implementation
7      nodes, weights = gpce_strategy.generate_samples(distribution)
8
9      # propagation of prior samples
10     qoi_averaged, n_samples_propagation, prior_samples = super().
           propagate(nodes)
11
12     # generate expansion
13     expansion = chaospy.generate_expansion(expansion_order,
           distribution)
14     qoi_gpce = gpce_strategy.calc_gpce(expansion, qoi_averaged, nodes,
15     weights) # invokes concrete strategy implementation
16
17     # modes of response distribution
18     expected = chaospy.E(qoi_gpce, distribution)
19     std = chaospy.Std(qoi_gpce, distribution)
20
21     return expected, std
```

Listing 4.6: Uncertainty analysis with generalized polynomial chaos in `PropagationCalculatorGPCE`.

Listings 4.7 and 4.8 show how the samples are generated and the expansion is calculated for point collocation and the pseudo-spectral approach, respectively. While for point collation a regression is used, the pseudo-spectral approach employs a quadrature rule.

```
1  def generate_samples(n_samples):
2
3      # create samples from prior distribution
4      nodes = prior.sample(n_samples)
5      weights = None
6      return nodes, weights
7
8  def calc_gpce(expansion, qoi_averaged, nodes, weights):
9
10     # fit regression
11     qoi_gpce = chaospy.fit_regression(expansion, nodes, qoi_averaged)
```

```
12        return qoi_gpce
```

Listing 4.7: Generation of samples and calculation of generalized polynomial chaos expansion with point collocation in `PointCollocationExpansion`.

```
1  def generate_samples(distribution, quadrature_order, quadrature_rule):
2      # generate quadrature points
3      nodes, weights = chaospy.generate_quadrature(quadrature_order,
           distribution,
4      quadrature_rule)
5      return nodes, weights
6
7      def calc_gpce(expansion, qoi_averaged, nodes, weights):
8      # generate the general polynomial chaos expansion polynomial
9      qoi_gpce = chaospy.fit_quadrature(expansion, nodes, weights,
           qoi_averaged)
10     return qoi_gpce
```

Listing 4.8: Generation of samples and calculation of generalized polynomial chaos expansion with the pseudospectral approach in `PseudoSpectralExpansion`.

4.5 Interface with Vadere crowd simulation

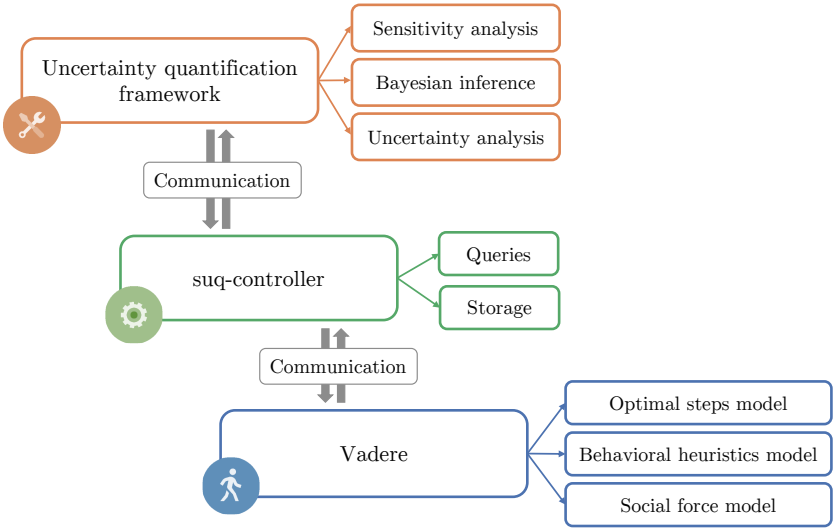


Figure 4.7: Schematic description of the uncertainty quantification framework, presented in [Gödel et al., 2019a].

The uncertainty quantification framework provides routines adapted for Vadere. Figure 4.7 gives a coarse overview of the framework and its relation to Vadere. Communication between uncertainty quantification and Vadere is facilitated by the suq-controller[5]. This fulfills FR1 which requires to support existing interfaces to Vadere. The suq-controller enables evaluating multiple runs of Vadere at different parameter sets.

The output processors in Vadere store the measures of the defined quantities of interest. If new quantities of interest are to be evaluated, new processors need to be implemented. For this thesis, I introduced a few new processors: The `ParadeLengthProcessor`, which evaluates the length of a demonstration march used in a time-dependent sensitivity analysis [Rahn et al., 2021], and the `FlowProcessor`, which calculates the flow according to [Seyfried et al., 2009] which I use throughout this thesis.

4.6 Code verification

For code verification, I set up a continuous integration pipeline in Gitlab. The tests are split into two groups, unit tests with test models and functional tests with Vadere. The code coverage is 60%. The latest test and coverage reports can be found on Gitlab[6].

I implemented tests to ensure that the reproducibility of the results (FR6) is maintained by the algorithms. They are provided in the package `test.seed`. In general, it is tested whether the results are identical if the analysis is run twice with the same seed and if the results differ if the analysis is run with different seeds.

For parameter identification, I provide routines for Sobol' indices and activity scores. I verified the routines by checking the Sobol' indices for the Ishigami test function [Ishigami and Homma, 1990] and comparing the activity scores for piston and circuit test function to [Constantine and Diaz, 2017]. In addition, I made sure that both indices converge with factor $N^{-1/2}$ where N is the number of samples. Moreover, I assert that the Sobol' indices are bounded by the activity scores as established by [Constantine and Diaz, 2017, Sobol' and Kucherenko, 2009]. All of these tests are implemented in test classes and are checked for every commit by the continuous integration pipeline.

In order to test the parameter estimation routines, the Metropolis algorithm, and rejection sampling, I employ a linear test model and check whether the true parameter value can be recovered by inference with artificial data. The same is tested for one- and two-dimensional problems with Vadere. These tests are similar to the proof-of-concept carried out in [Gödel et al., 2019a].

In addition, I implemented unit tests to check the approximation of the gradients and the calculation of the uncentered covariance matrix C of the model gradients for activity scores based on test models.

[5] `https://gitlab.lrz.de/vadere/suq-controller`
[6] `https://gitlab.lrz.de/vadere/uncertainty-quantification/pipelines`

4.7 Summary

This chapter focused on the requirement *R1: Choose or design a software for uncertainty quantification for crowd simulations*. My findings yield the following results regarding the three sub-requirements:

R1.1: Derive requirements for a software for uncertainty quantification for crowd simulations

I defined functional and non-functional requirements on an uncertainty quantification software: It needs to support existing interfaces to the Vadere crowd simulation framework (FR1) and support all crowd dynamics models implemented in Vadere (FR2). Also, for my studies, routines for parameter identification (FR3), parameter estimation (FR4), and uncertainty analysis (FR5) are necessary. The results need to be reproducible (FR6) and stored for post-processing (FR7). A graphical user interface would be ideal for external users (FR8), however, this is an optional requirement. As non-functional requirements, a modular design is envisioned (NFR1), the software should adhere to established coding style (NFR2), it should be published open-source (NFR3) and be well-tested (NFR4).

R1.2: Analyze available software regarding the requirements

There are several software frameworks for uncertainty quantification available. However, none of them fulfills all mandatory requirements (FR1-7). Then, I discussed the state of the art on software for uncertainty quantification and their shortcomings regarding the requirements. As a result, I decided to design and implement my own framework that I will use throughout the thesis. The analysis of the state of the art software identified two useful and well-tested Python software packages, chaospy and SALib, that I integrate into my framework. Their shortcoming is that they only provide routines for sensitivity analysis and uncertainty analysis, and sensitivity analysis, respectively.

R1.3: Design and implement a software that fulfills the requirements

The framework that I designed and implemented fulfills all mandatory requirements: Its architecture assures modularity (NFR1). The strategy pattern is employed to separate algorithms from infrastructure so that it is easily expandable. It provides routines for parameter identification, parameter estimation, and uncertainty analysis (FR3, FR4, FR5). The methods are adapted to crowd simulations and can be applied to all models implemented in Vadere (FR2). The results of the analysis can be stored to file (FR7). I implemented several tests to assure that the results are reproducible (FR6). In general, the algorithms are tested using unit tests as well as functional tests with Vadere (NFR4). I established a continuous integration pipeline that conducts all tests at every commit. The framework supports an existing interface to Vadere, the suq-controller (FR1). It is implemented in Python and therefore compatible with the suq-controller as well as other Python-based Vadere add-ons. The code adheres to the Python style guide PEP8[7] (NFR2). The uncertainty quantification framework is provided open-source, available on Gitlab[8] (NFR3).

[7] `https://www.python.org/dev/peps/pep-0008/`

[8] `https://gitlab.lrz.de/vadere/uncertainty-quantification`

5 Parameter identification: identifying influential parameters

A parameter is influential if changing it has a considerable impact on the model response. In this chapter, I describe how such parameters have been identified for crowd simulations so far (Section 5.2) to enable the reader to put my methodical choices into context. In Section 5.3, I propose and describe systematic methods for local and global sensitivity analysis that are tailored to crowd simulations. In particular, I elaborate on a variance-based index, Sobol' indices, and derivative-based indices which are based on an active subspace (Section 5.3). Then, I use all of them to identify influential parameters in the bottleneck scenario (Section 5.4). Finally, I summarize the findings.

5.1 Introduction

There is no standardized approach for sensitivity analysis in crowd simulation. Often, sensitivity information is derived from a forward propagation for several parameters. The parameters, for which the variance in the quantity of interest is large after propagation, are then deemed influential. In addition, in some publications, individual methods for sensitivity analysis have been developed. While they are customized to the problem at hand, the results are not comparable to other studies.

Uncertainty quantification includes methods for sensitivity analysis to identify the influential parameters in a simulation. Sensitivity analysis rates the parameters according to the amount of uncertainty they contribute to the output. In order to reduce the uncertainty in the output, sensitivity analysis helps with prioritization: Non-influential parameters can be fixed to an arbitrary value within their range. This setting is called factor fixing. Influential parameters should be studied in more detail (factor prioritization). Reducing uncertainty is often expensive; therefore, the effort should be allocated where the largest reduction is possible.

In this chapter, I present methods for sensitivity analysis and select two of them that are suited specifically for crowd simulation. I adapt them for our models and implement them in my uncertainty quantification framework. Then, I apply both methods to the bottleneck scenario and compare the results.

The results of the sensitivity analysis are a starting point for reducing the uncertainty in the simulation output. One option is parameter calibration. Calibration is expensive, so often only the (most) influential parameters can be calibrated. Looking at the problem from the other side, calibration is only meaningful for influential parameters. Calibration of the influential parameters is performed in the next chapter. Another option is to gather more data through observations or experiments. This is also costly and must therefore

be aimed at the most promising spots. The sensitivity analysis reveals these promising spots, that is, the most influential parameters.

Research questions addressed in this chapter

Q1 How can we identify influential parameters in the optimal steps model for the bottleneck scenario?

- Q1.1 Which parameter identification methods are suited for crowd dynamics models?
- Q1.2 Which parameters are influential and which are non-influential in the bottleneck scenario?
- Q1.3 How can we determine if the results are reliable?

In [Gödel et al., 2020a], I presented a sensitivity analysis with Sobol' indices and activity scores that focuses on the density in front of the bottleneck instead of the flow through the bottleneck. In [Rahn et al., 2021], we obtained Sobol' indices through a polynomial chaos based surrogate for a case study of a protest march. I refer to the papers in the text.

5.2 State of the art on parameter identification in crowd simulation

In pedestrian dynamics, several publications which claim to study "sensitivity" of parameters present forward propagations rather than sensitivity analysis according to the understanding of uncertainty quantification. I describe and perform forward propagation in Chapter 8. Between forward propagation and sensitivity analysis, primarily the target spaces differ: For propagation, the result lies in the space of simulation results, whereas for sensitivity analysis, it lies in the parameter space. The result of the sensitivity analysis is a sensitivity value for each parameter, which measures the contribution of that parameter to the uncertainty in the simulation result. Then, the parameters can be ranked according to their influence. That means sensitivity analysis requires a sensitivity metric. The common approaches in pedestrian dynamics often lack this metric. Instead, the variation in the simulation output is compared visually. In addition, most of these approaches vary the parameters without a defined sampling scheme and in a one-factor-at-a-time (OAT) fashion. That means, only one parameter is varied while all others are fixed. OAT approaches fail by design to identify parameter interactions of any order. In this work, I use the term sensitivity analysis whenever a metric is defined that quantifies the individual parameter contribution towards the output uncertainty.

Table B.1 summarizes publications from the pedestrian dynamics community in which a forward propagation was performed. The sensitivity of the parameters is derived from the results of the propagation without defining a specific metric. As in our analysis, [Sparnaaij et al., 2019, Colombi et al., 2017, Collins et al., 2014, Shi et al., 2021, 2018, Duives et al., 2016] analyze a bottleneck scenario. A few of these publications also aim

to study the free-flow speed or parameter related to the free-flow speed [Sparnaaij et al., 2019, Kouskoulis et al., 2018, Shi et al., 2021, Davidich and Köster, 2013, Teknomo and Gerilla, 2005] as well as obstacle repulsion [Kouskoulis et al., 2018, Colombi et al., 2017], and personal space strength [Sparnaaij et al., 2019, Kouskoulis et al., 2018, Colombi et al., 2017]. Several of the authors point out that they are working with stochastic simulators. Their approach is to average several repetitions at each parameter set. Only Sparnaaij et al. [Sparnaaij et al., 2019] discuss how to find the necessary number of repetitions. I decide not to follow their approach since its additional model evaluations are costly.

Next, we focus on contributions that define a sensitivity metric. Table B.2 summarizes the sensitivity analysis conducted in pedestrian dynamics so far. I am only aware of three publications. Compared to Table B.1, the metric is an additional criterion. In all three manuscripts, the sensitivity of the model response regarding speed parameters is studied [Zhong and Cai, 2015, Chen et al., 2019, Kurtc et al., 2021]. Additionally, obstacle repulsion [Zhong and Cai, 2015, Chen et al., 2019] and personal space strength [Zhong and Cai, 2015] are of interest. Zhong et al. and Kurtc et al. also handle their stochastic simulators by averaging over a fixed number of repetitions. Zhong et al. compute the sensitivity metrics in a bottleneck scenario [Zhong and Cai, 2015]. Only Kurtc et al. use an approach that considers parameter interactions instead of performing only an OAT analysis. Here, I study the impact of a larger number of parameters on flow through a bottleneck scenario instead of density in a corridor. Consequently, I examine a different dynamic.

Finally, we take a short look at an adjacent community that performs traffic simulations. Models and methods are often similar to crowd dynamics. In traffic simulation, however, sensitivity analysis is already a common tool. Table B.3 summarizes the works of Sfeir et al. and Punzo et al.. Both use Sobol' indices to analyze the importance of individual parameters. Punzo et al. study sensitivities in a social force model, similar to the social force models popular in crowd dynamics. A more comprehensive list of sensitivity analyzes carried out for traffic simulations can be found in [Sfeir et al., 2018].

Parameters associated with origin-destination popularity, or destination choice are intuitively and demonstrably influential for different quantities of interest [Haghani et al., 2018, Davidich and Köster, 2013]. Since this information is typically dynamic, the estimation differs from the methods presented in this chapter. We discuss an approach to learn dynamic origin-destination matrices based on density heatmaps in Chapter 7.

5.3 Overview of methods for parameter identification

Sensitivity analysis methods from the field of uncertainty quantification aim to identify influential and non-influential parameters. Figure 5.1 provides an overview of the sensitivity methods described in the following. Local sensitivity analysis evaluates the impact of parameters on a simulation outcome, the predefined quantity of interest, around a fixed reference value. Common approaches are screening methods, which sample the uncertain parameter and observe the change in the response. Often, one-factor-at-a-time

methods are used which vary only one parameter while the others are fixed. They can only identify contributions by individual parameters; interactions are neglected. Another approach is sequential bifurcation [Bettonvil and Kleijnen, 1997]. A natural basis for local sensitivity analysis is differentiation. Besides the common partial derivatives, Borgonovo and Apostolakis propose the differential importance measure [Borgonovo and Apostolakis, 2001].

Often we are not only interested in the sensitivity of a model to a parameter around a reference value but over the parameter's range. In this case, global sensitivity analysis is suitable. Regression-based methods create a linear surrogate based on which the sensitivity is assessed. Whenever the assumption of a linear relationship between parameter and quantity of interest is too strict, rank-based methods [Saltelli and Sobol', 1995] such as the Spearman correlation coefficient [Spearman, 1904] can be employed. Screening methods can be globalized to evaluate parameter ranges instead of the behavior around a fixed parameter. Here, Morris screening [Morris, 1991] is a well-known representative. For Morris screening, local derivatives are first approximated and then averaged to provide a globalized measure. Other screening methods are screening by groups [Kleijnen, 1987], sequential bifurcation method [Bettonvil and Kleijnen, 1997] a globalized version of sequential bifurcation, and factorial fractional design [Montgomery, 2013, p. 320ff]. Besides regression-based approaches and globalized screening methods, variance-based methods can be utilized for global sensitivity analysis. For this approach, the variance in the quantity of interest is the basis for the sensitivity metric. An established metric are Sobol' indices [Sobol', 1993], which use an ANOVA-HDMR[1] representation of the variance of the model response. Sobol' indices can either be obtained from the model itself or from a polynomial chaos based surrogate [Sudret, 2008]. Other variance-based measures are Pearson's correlation ratio [Pearson and Galton, 1895] as well as Wagner's variance-based sensitivity measures [Wagner, 1995]. Instead of focusing on a single moment of the distribution of the quantity of interest, such as the variance, moment-independent or density-based methods consider the full distribution of the model response [Borgonovo, 2007]. Finally, also for global sensitivity analysis, some approaches are based on derivatives. This includes metrics based on active subspaces [Constantine and Diaz, 2017] such as activity scores and the first eigenvector metric. The active subspace here is a lower-dimensional representation of the input parameter space based on important directions in the parameter space. In addition, Sobol' and Kucherenko propose the derivative-based global sensitivity metrics (DGSM) [Sobol' and Kucherenko, 2009].

Decision trees presented by de Rocquigny and Tarantola [de Rocquigny et al., 2008] can support selecting an appropriate method. More detailed overviews of the methods can be found in [Borgonovo and Plischke, 2016, Iooss and Lemaître, 2015].

From the collection of methods, I choose Sobol' indices since they only require function evaluations and can therefore be calculated for any model. This is certainly one of the reasons why they are widely established and used in many applications [Saltelli et al., 2008, p. 237ff]. In addition, I decided to use the active subspace method with the goal to identify a lower-dimensional subspace in our input parameter space. The active

[1] **An**alysis **o**f **variance** (ANOVA), **h**igh-**d**imensional **m**odel **r**epresentation (HDMR)

subspace method detects important parameter directions, instead of focusing solely on the importance of single parameters.

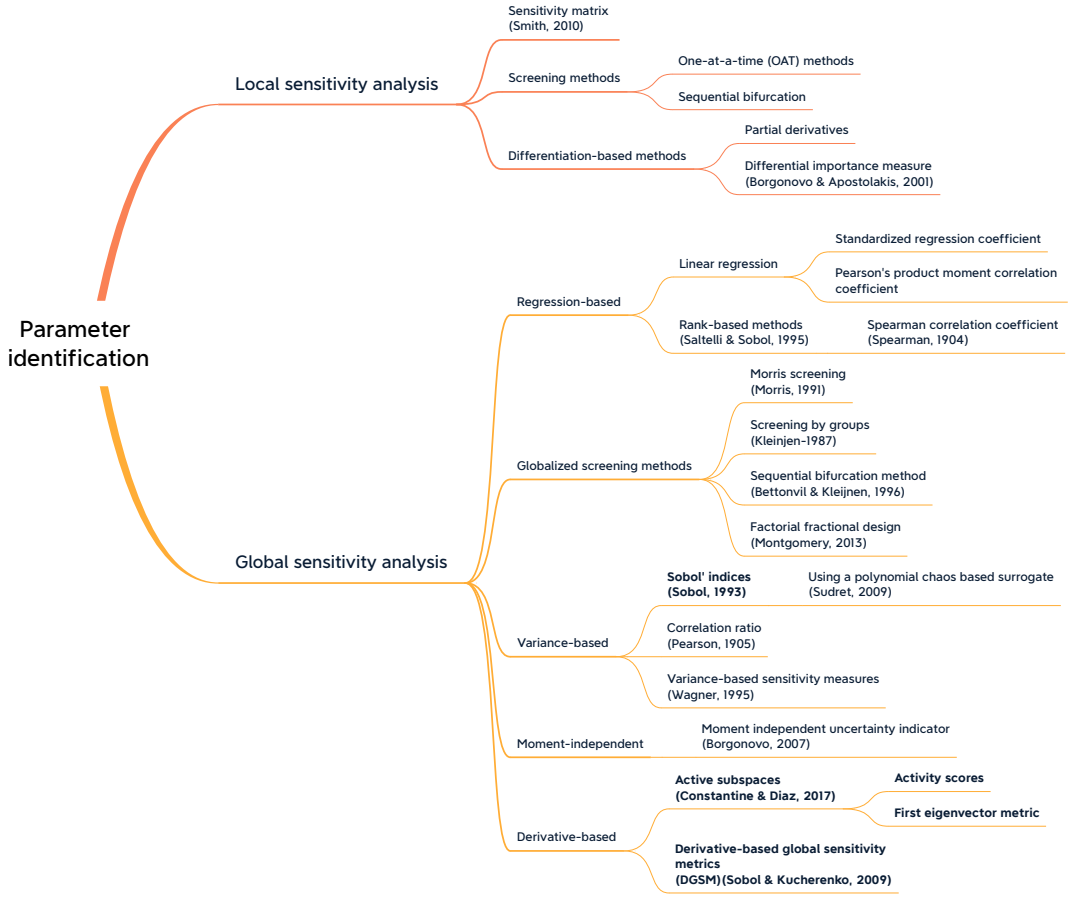


Figure 5.1: Overview of methods for parameter identification. Methods that were selected for crowd simulations are highlighted.

5.3.1 Sobol' indices

In the following, we assume that our model f is a square-integrable function defined over a k-dimensional unit hypercube Ω. Any model that meets this condition can be represented by an analysis of variance, or ANOVA, decomposition. ANOVA decomposition splits our model f into a set of orthogonal functions depending on the factors $x_i, i = 1, \ldots, n$ and their interactions [Sobol', 2001]:

$$f(x) = f_0 + \sum_{s=1}^{n} \sum_{i_1<\ldots<i_n}^{n} f_{i_1\cdots i_s}(x_{i_1}, \ldots, x_{i_s}) \tag{5.1}$$

$$= f_0 + \sum_i f_i(x_i) + \sum_{i<j} f_{ij}(x_i, x_j) + \cdots + f_{12\cdots n}(x_1, x_2, \ldots, x_n) \tag{5.2}$$

where $1 \leq i_1 < \cdots < i_s \leq n$. The representation is an exact representation of f when we use all 2^n terms. Contributions of single parameters are represented by the first-order univariate functions f_i while the interaction terms $f_{i_1\cdots i_s}, s > 1$ quantify the interactions of several input parameters x_i on the model response y [Smith, 2014, p. 323f].

Based on the decomposition of the model response (eq. 5.2), we can also split its variance $V[y]$ into the variance of the orthogonal functions $f_{i_1\cdots i_s}$. With the partial variances $D_{i_1\cdots i_s}$ of $f_{i_1\cdots i_s}$ defined as $D_{i_1\cdots i_s} = \int f^2_{i_1\cdots i_s} dx_{i_1} \ldots dx_{i_s}$, we can write the total variance D as

$$D = V[y] = \int_\Omega f^2(x)dx - f_0^2 = \sum_{s=1}^{n} \sum_{i_1<\ldots<i_k}^{n} D_{i_1\cdots i_s} \tag{5.3}$$

with $f_0 = E[Y]$ [Sobol' and Kucherenko, 2005]. By square integrating each term of the decomposition 5.2, we obtain the so-called ANOVA-HDMR decomposition for $V(Y)$:

$$V(Y) = \sum_i V_{f_i} + \sum_i \sum_{j>i} V_{f_{ij}} + \ldots + V_{f_{12\ldots k}}. \tag{5.4}$$

We need the total variance D and the partial variances $D_{i_1\cdots i_s}$ to define the first-order indices S_j and the higher-order indices, $S_{i_1\cdots i_s}$ of order i, as the ratio between partial variance and total variance. The total effect index S_{T_j} is defined as the sum of the indices of all orders that include parameter j:

$$S_j = S_{i_1} = \frac{D_i}{D} \tag{5.5}$$

$$S_{i_1\cdots i_s} = \frac{D_{i_1\cdots i_s}}{D} \tag{5.6}$$

$$S_{T_i} = S_i + \sum_{j>i} S_{ij} + \sum_{j>i} \sum_{k>j} S_{ijk} + \ldots + S_{123\ldots n} \tag{5.7}$$

Sobol' indices in practice In practice, we often cannot analytically evaluate these indices because they require solving high-dimensional integrals. Instead, they are approximated by a finite sum. We use Jansen's method to approximate the indices [Jansen, 1999]. For a given sampling factor M, the number of Monte Carlo samples generated is $N = \lfloor M/(m+1) \rfloor$. We generate two matrices, A and B, with N samples each according to the input parameter density. The variation of all input factors except one is modeled by constructing matrices $A_B^{(i)}$ that are identical to matrix A except for column i which originates from matrix B. Then, variances of model evaluations at row j of A and B to $A_B^{(i)}$, respectively, are approximated. The first-order indices S_i are approximated by

$$\hat{S}_i = 1 - \frac{1}{2Nv_y} \sum_{j=1}^{N} \left(f(B)_j - f(A_B^{(i)}{}_j) \right)^2, \tag{5.8}$$

where the variance of the model response $V[y]$ is approximated by the sample variance v_y of the evaluations $f(A)$ and the subscript j stands for row j of the matrix. The total order indices S_{T_i} are approximated by

$$\hat{S}_{T_i} = \frac{1}{2Nv_y} \sum_{j=1}^{N} \left(f(A)_j - f(A_B^{(i)}{}_j) \right)^2. \tag{5.9}$$

From 5.7 we can see that the total effects are larger than or equal to the first-order indices S_i. Potential differences between total effects and first-order indices in each parameter indicate higher order contributions i. e. interaction effects among the parameters. The difference between first-order and total effect index $S_{T_i} - S_i$ is a measure for the interaction of factor x_i with other factors. Consequently, if the total order effect S_{T_i} is identical to the first-order index S_i for parameter i, then this parameter is not involved in any interactions toward our quantity of interest V. If we divide both sides of 5.4 by $V(Y)$, we obtain

$$\sum_i S_i + \sum_i \sum_{j>1} S_{ij} + \ldots + S_{12\ldots k} = 1. \tag{5.10}$$

The consequence of this relation is that the sum of the first-order indices has to be equal to one if there are no interactions present. In this case, the model under study is purely additive.

5.3.2 Activity scores, first eigenvector, and derivative-based global sensitivity metrics

Constantine and Diaz define two metrics based on the active subspace theory: activity scores and the first eigenvector metric [Constantine and Diaz, 2017]. Active subspace methodology aims to identify important directions in the input parameter space to construct a lower-dimensional parameter space. Evaluating both metrics requires gradients of the model f, or at least approximations of the gradients. Constantine [Constantine, 2015] proposes a sampling-based approach to identify active subspaces. Based on his work, we describe the methodology to find an active subspace of dimension $k < m$ within a m-dimensional parameter space.

At first, the number of samples needs to be defined. A rule of thumb to achieve a sufficient accuracy is $M = \alpha \cdot k \cdot \ln(m)$ with oversampling factor $\alpha \in [2, 10]$ [Constantine, 2015, p. 35]. The samples are drawn from the input parameter distribution ρ. At the sample points $x_i, i = 1, \ldots, M$, the gradient $\nabla f(x_i)$ is evaluated and used to construct the matrix

$$C = \int \nabla f(x_i) \nabla f(x_i)^T \rho \, dx. \tag{5.11}$$

Here, the diagonal entries $C_{ii}, i = 1, \dots m$ of C are the derivative-based global sensitivity metricsA defined by Sobol' and Kucherenko [Sobol', 1993]:

$$\nu_i = \int_{H^n} \left(\frac{\partial f}{\partial_{x_i}} \right)^2 dx. \tag{5.12}$$

Constantine [Constantine, 2015, p. 22] describes C as the "(uncentered) covariance matrix of the gradient". In order to identify the important directions in the parameter space, we calculate the eigenvalues and eigenvectors of C by a singular value decomposition:

$$C = W \Lambda W^T. \tag{5.13}$$

Now, each column of $W = [w_1 \dots w_n]$ is a eigenvector of C and the diagonal elements $\lambda_i, i = 1, \dots, m$ of $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_m)$ are the eigenvalues of C. The eigenvalues are sorted descending by their size. W accordingly holds the eigenvector corresponding to the ith eigenvector in column i.

The first eigenvector w_1 can already be used as a sensitivity metric: entry $w_{1,j}, j = 1, \dots, m$ is a metric for parameter index j. Since the eigenvectors are orthonormal due to the symmetry of C, there is only ambiguity in the direction (positive or negative). If we know the direction of the correlation (positive or negative) between one parameter and the quantity of interest, we can rotate the eigenvector accordingly. Then, the index $w_{1,j}$ shows not only the size but also the direction of the contribution of parameter i on the quantity of interest. In order to obtain comparable results between the first eigenvector metric and the activity scores, the first eigenvector metric needs to be normalized by its length.

Based on the eigenvalues, we can now construct subspaces of arbitrary dimensions. An ideal subspace is as low dimensional as possible while capturing most of the information. The eigenvalues portray the amount of information of the corresponding direction in the parameter space. Now, similar to the principal component analysis, the user needs to choose how many eigenvalues to keep to form the subspace. Unless computational constraints prevent it, the best choice is to look for a spectral gap in the eigenvalues and use the eigenvectors corresponding to eigenvalues above the gap to form a subspace [Constantine, 2015, p. 37]. Often, log-scale plots are used to identify the spectral gap.

We assume that the spectral gap appears between λ_k and λ_{k+1}. Then, the eigenvectors $w_1, \dots, w_k$ form the active subspace $W_1 = [w_1 \cdots w_k]$. Consequently, the remaining eigenvectors of W form the inactive subspace $W_2 = [w_{k+1} \cdots w_m]$. Constantine and Diaz [Constantine and Diaz, 2017] define the activity scores α_i as

$$\alpha_i = \alpha_i(k) = \sum_{j=1}^{k} \lambda_j w_{i,j}^2, i = 1, \dots, n. \tag{5.14}$$

The calculation depends on the size k of the chosen subspace: only the first k eigenvalues and eigenvectors are used.

Activity scores, first eigenvector metric, and DGSM in practice In the case of crowd simulations, gradients of the quantity of interest are typically not available. For models based on differential equations such as the social force model or the gradient navigation model, gradients are theoretically available. However, this is only the case if the simulation output is a direct output of the models. The output of microscopic models such as social force based models is a set of trajectories. When other quantities of interest are considered, in addition to derivations for the model equations, we also need derivations of the post-processing algorithms. As a consequence, the more universal approach to the gradients for the quantity of interest are approximations.

Furthermore, it depends on the model whether meaningful gradients can be obtained. Especially for rule-based models such as cellular automata or the behavioral heuristics model, the output is discontinuous and the gradients are not defined. The fewer discontinuities a model has, the more likely it is that the gradient approximations are expedient. Especially for computer models in which there are no explicit formulas for the simulation outcomes available, it is not clear whether we can evaluate gradients.

We choose to use central differences to approximate the gradients. If the gradients are approximated, we obtain the approximation $\hat{C}$ instead of C:

$$\hat{C} = \int \nabla \hat{f}(x_i) \nabla \hat{f}(x_i)^T \rho dx \approx C \tag{5.15}$$

$$\hat{C} = \hat{W} \hat{\Lambda} \hat{W} \approx C. \tag{5.16}$$

Consequently, the eigenvectors $\hat{w}$ and eigenvalues $\hat{\lambda}$ of $\hat{C}$ are only an approximation to w and λ, respectively. Thus the spectral gap $\hat{k}$ in the eigenvalues $\hat{\lambda}$ is not necessarily identical to the spectral gap in λ. In the case of central differences, the accuracy of the approximation depends on the step size h. The step size must be smaller than the first eigenvalue after the gap $\hat{\lambda}_{k+1}$ to avoid the appearance of a so-called phantom eigenvalue [Constantine, 2015] that dissimulate the actual spectral gap. Finally, also the activity scores $\hat{\alpha}_i$ and first eigenvector metric $\hat{w}$ are only an approximation to α_i and $w_{1,i}$:

$$\hat{\alpha}_i = \hat{\alpha}_i(k) = \sum_{j=1}^{\hat{k}} \hat{\lambda}_j \hat{w}_{i,j}^2, i = 1, \ldots, n. \tag{5.17}$$

Due to the need for gradients, the activity scores are not as universally applicable as Sobol' indices. This is one reason why I evaluate Sobol' indices and activity scores: If they align, we gain trust in the activity scores and therefore we can use the additional information on the parameter directions that are provided.

5.3.3 Links between indices

In practical applications, the indices rank the parameters often give comparable parameter rankings [Constantine and Diaz, 2017]. However, we can construct models for which the indices rank the parameters differently [Sobol' and Kucherenko, 2009]. Even though the indices measure the sensitivity in different ways, there are links between them. Activity scores can be understood as truncated derivative-based indices nu_i. If there is

no eigenvalue gap, so $k = m$, and therefore all eigenvalues are used, activity scores and derivative-based indices are identical:

$$\alpha_i(k = m) = \sum_{i=1}^{m} \lambda_i w_{i,j}^2 = C_{ii}. \tag{5.18}$$

Derivative-based metrics are then the diagonal entries of C in eq. 5.13.

A link between activity scores and Sobol' indices is presented in Theorem 4.2 in [Constantine and Diaz, 2017] and Theorem 2 from [Sobol' and Kucherenko, 2009]. The total Sobol' effects are bounded by the activity scores:

$$S_{T_i} \leq \frac{p_i}{V[Y]}(\alpha_i(n) + \lambda_{n+1}) \tag{5.19}$$

with Poincaré constant $\sqrt{p_i} = \frac{d}{\pi}$ of diameter d for the region Ω from [Bebendorf, 2003] according to uniform distribution of factor x_i. In our case $d = 2$ for $x_i \in [-1, 1]$ due to the scaling of domain and partial derivatives.

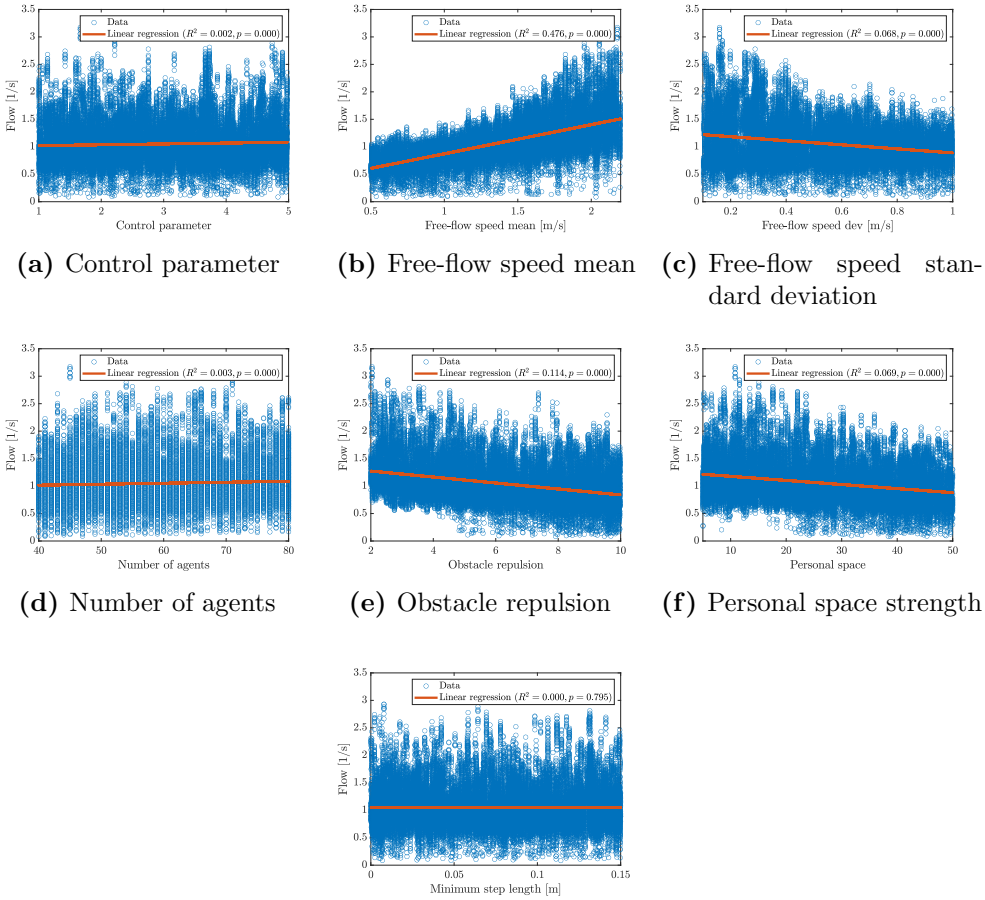
5.4 Studying parameter sensitivities in a bottleneck scenario

In this chapter, I apply Sobol' indices, activity scores, and derivative-based global sensitivity metrics on the bottleneck scenario to identify influential and non-influential parameters. An extensive description of the studied bottleneck can be found in Section 3.2.1. Finally, I discuss sensitivity analysis for time-dependent quantities of interest and efficient methods for computationally expensive scenarios.

5.4.1 Relationship between parameters and quantity of interest

At first, I analyze the relationship between each uncertain parameter and our quantity of interest, the flow through the bottleneck. The parameters are explained in more detail in Section 3.2.1. For the scatter plots in Figure 5.2, I use data generated when calculating the activity scores and derivative-based scores. Therefore, all parameters were varied simultaneously which leads to a strong noise.

From the scatter plots, we can see that the free-flow speed mean has the largest impact on the flow. We observe a strong trend in the linear fit and a large R^2 which implicates that the data is represented well by the fit. The free-flow speed mean is expected to be influential due to the relationship $J = \rho v w$ from [Rupprecht et al., 2007] where J is the flow, ρ is the density, v is the speed, and w is the bottleneck width. From this formula, as well as from the scatter plots we see a strong positive correlation that means the flow increases with the free-flow speed mean. Besides the free-flow speed mean, other parameters seem to have an impact on the flow as well. These are free-flow speed standard deviation, obstacle repulsion, and personal space strength. Different from the mean speed, all of these parameters have a negative correlation with the flow. Consequently, if we increase the parameters, the flow is reduced. We take a closer look at how these parameters and the flow are related: The free-flow speed standard deviation



(a) Control parameter **(b)** Free-flow speed mean **(c)** Free-flow speed standard deviation

(d) Number of agents **(e)** Obstacle repulsion **(f)** Personal space strength

(g) Minimum step length

Figure 5.2: Scatter plot of input-output relation for each uncertain input parameter in the bottleneck scenario using the model evaluations from the derivative-based method. Each run is shown, that is, 10 data points for each configuration. In total, 81 gradient samples are evaluated which leads to a total of 9720 runs.

determines the variation among the individual velocities. A larger variation among the free-flow speeds increases the time span between when the first agent crosses the measurement line and when the last agent crosses the measurement line, which is larger Δt and therefore, with $J = \Delta N/\Delta t$, a lower flow. Higher demand for personal space reduces the density and therefore, at a constant speed, the flow according to $J = \rho v w$. Finally, obstacle repulsion has an impact on the width of the bottleneck that is actually used; it is similar to varying the width of the bottleneck. Looking again at $J = \rho v w$, a reduced flow with increased obstacle repulsion or decreased effective w is plausible. Three parameters have no influence on our quantity of interest, the flow: the number of agents, the minimum step length, and the control parameter. The formula for the flow is independent of the actual number of agents; it only evaluates the ratio of differences

in agents ΔN and duration Δt. If doubling the number of agents leads to doubling the duration for crossing the bottleneck, we measure the same flow. Consequently, the number of agents should have no impact. The minimum step length leads to small steps being discarded. In this case, the agent remains in its position. Therefore, the minimum step length only leads to minor changes in the trajectory. From the results, I conclude that these do not have an impact on the overall dynamics. For the control parameter, I choose a parameter that has no impact on the simulation of the bottleneck scenario to test the sensitivity indices: a parameter of the queueing model which is not activated for the bottleneck. While we could use the scatter plots to classify parameters into influential and non-influential, it is risky since they are a simplified measure of first-order effects. We need to look at the total effects, in addition, to reliably rank the parameters before we can fix non-influential parameters.

5.4.2 Sobol' first-order and total indices

I evaluate the first-order and total Sobol' indices for different sampling factors M in Figures 5.3 and 5.4, respectively. Table 5.1 translates the sampling factor M to the number of model evaluations.

Table 5.1: Number of model evaluations for calculating Sobol' indices based on the sampling factor M.

Sampling factor M	50	100	500	1000	5000
Number of model evaluations	562	1125	5625	11250	56250

The first-order index measures solely the impact of the parameter itself, while the total order index measures the impact including all interaction effects in which this parameter is present. In both plots, the free-flow speed mean is the most influential parameter, agreeing with our analysis of the scatter plots. Besides this parameter, the free-flow speed standard deviation, the obstacle repulsion, and the personal space strength are influential when measuring the flow through the bottleneck. The other three parameters, the control parameter, the number of agents, and the minimum step length, are close to zero and therefore deemed as non-influential. In the case of the control parameter, it confirms that the method appropriately evaluates parameters that have no influence. We also observe that with increasing sampling factor M, the variation decreases as expected. For a low sampling factor, negative values for indices can arise due to numerical errors.

When we compare first-order and total Sobol' indices in Figure 5.5, we notice that they are similar. Also, the total Sobol' indices are all larger or equal than the first-order indices, which always has to be the case. Only for the free-flow speed mean and standard deviation, the first-order indices are lower than the total indices. That means interaction is present between those two parameters. Saltelli states that "two factors [...] interact when their effect on Y cannot be expressed as a sum of their single effects on Y" [Saltelli et al., 2008]. If we add up the first-order indices we obtain $\sum S_i = 0.8148 < 1$. Since a purely additive model would yield a sum of 1, we can conclude that the model is mainly additive except for the second-order effect.

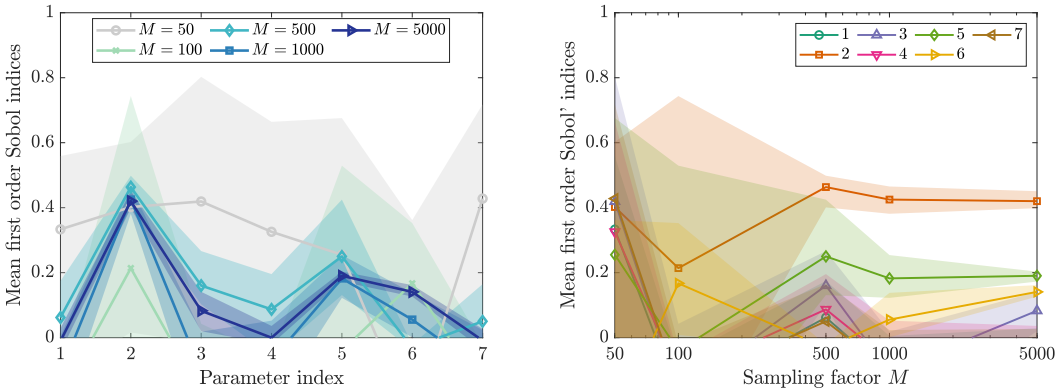


Figure 5.3: Sobol' first-order indices for the bottleneck scenario calculated with Monte Carlo approach using Jansen's method [Jansen, 1999, Saltelli et al., 2010]. For each sampling factor M, three runs were performed. Parameters under study: control parameter (1), free-flow speed mean (2) and standard deviation (3), number of agents (4), obstacle repulsion (5), personal space (6), minimum step length (7).

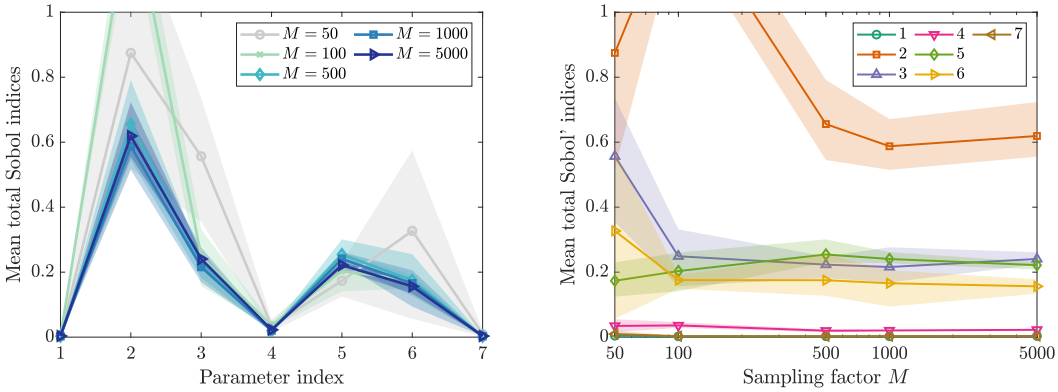


Figure 5.4: Sobol' total effect indices for bottleneck scenario calculated with Monte Carlo approach using Jansen's method [Jansen, 1999]. For each sampling factor M, three runs were performed. Parameters under study: control parameter (1), free-flow speed mean (2) and standard deviation (3), number of agents (4), obstacle repulsion (5), personal space (6), minimum step length (7).

5.4.3 Derivative-based global sensitivity metrics, first eigenvector metric, and activity scores

I calculate the first eigenvector metric and activity scores by identifying an active subspace in the input parameter space. Based on gradient approximations using central differences, I evaluate the uncentered covariance matrix C. Figure 5.6 depicts the eigenvalues and the first eigenvector of C. In the eigenvalues in Figure 5.6a, there is a spectral gap between the first and second eigenvalue. Consequently, there exists a one-dimensional subspace. That means a single linear combination of the input parameters

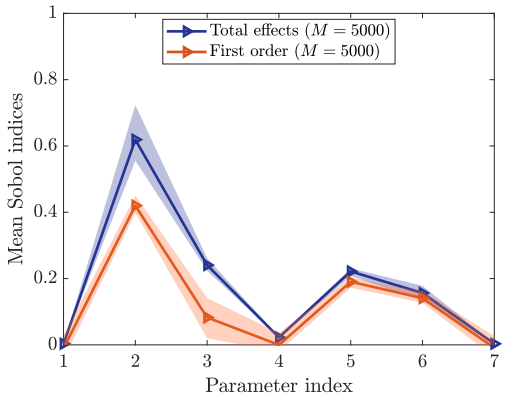
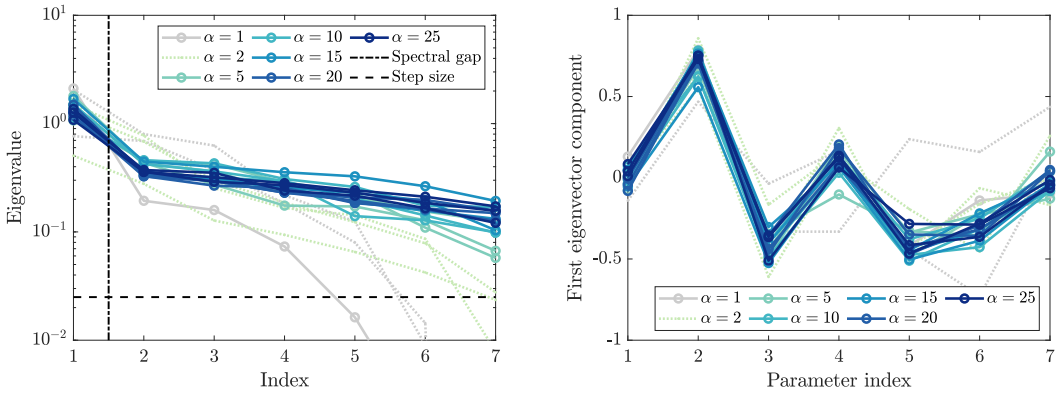


Figure 5.5: Total and first-order Sobol' indices for the bottleneck scenario, calculated using Jansen's method with sampling factor $M = 5000$. Parameters under study: control parameter (1), free-flow speed mean (2) and standard deviation (3), number of agents (4), obstacle repulsion (5), personal space (6), minimum step length (7).

forms the basis for the active subspace. The first eigenvector in Figure 5.6b presents this important direction in the input parameter space. The direction of eigenvectors is ambiguous. However, since we know from the scatter plot that the free-flow speed mean, parameter 2, has a positive correlation with the flow, we rotate the eigenvector accordingly.



(a) Eigenvalues of matrix C **(b)** First eigenvector of matrix C

Figure 5.6: Eigenvalues and first eigenvector components of the matrix C. The spectral gap between the first and second eigenvalue reveals a one-dimensional subspace. Parameters under study: control parameter (1), free-flow speed mean (2) and standard deviation (3), number of agents (4), obstacle repulsion (5), personal space (6), minimum step length (7).

The first sensitivity metric that we analyze is the first eigenvector metric. Figure 5.7 depicts the normalized first eigenvector. In agreement with the Sobol' indices, we find

that the free-flow speed mean is the most influential parameter, followed by the free-flow speed standard deviation, the obstacle repulsion, and the personal space strength. The number of necessary gradient samples for calculating the active subspace is typically determined by $M = \alpha k \log(m)$ [Constantine, 2015, p. 35], where k is the size of the desired subspace, m is the dimension of the input parameter size, and α is the oversampling factor. The latter needs to be chosen by the user. Oversampling factors in $[5, 10]$ are common [Constantine, 2015]. Compared to the Sobol' indices, we observe that the variation in the first eigenvector metric is significantly lower for a small number of samples. Table 5.2 lists the number of model evaluations required for each oversampling factor. For small oversampling factors α, the identified subspace differs from a one-dimensional subspace (dotted lines).

Table 5.2: Number of model evaluations for calculating activity scores based on the oversampling factor α.

Oversampling factor α	1	2	5	10	15	20	25
Number of model evaluations	840	1680	4200	8260	12320	16380	20440

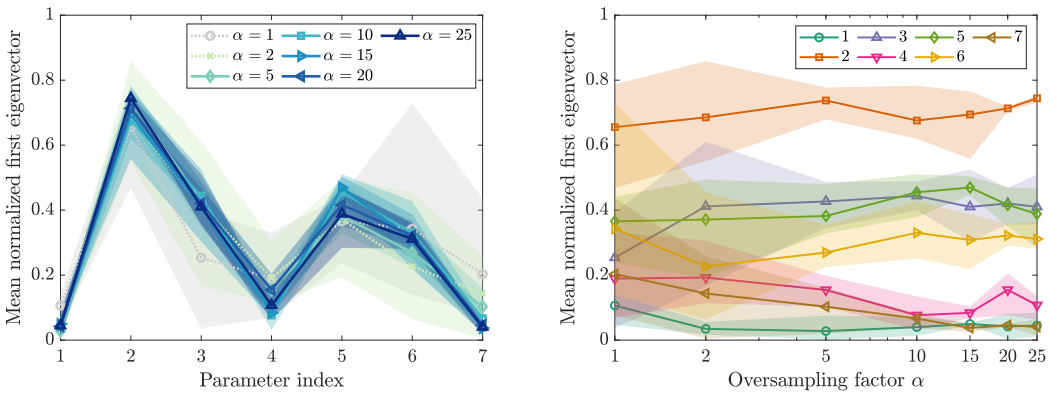


Figure 5.7: Parameter sensitivities measured by normalized first eigenvector metric for the bottleneck scenario. The mean eigenvector entries obtained from three runs (solid lines) are shown together with the limits of the scores (shaded areas). Parameters under study: control parameter (1), free-flow speed mean (2) and standard deviation (3), number of agents (4), obstacle repulsion (5), personal space (6), minimum step length (7).

Next, we take a look at the activity scores in Figure 5.8. While they agree with the first eigenvector metric qualitatively, there are some differences in the size of the sensitivity indices, especially for the number of agents, parameter 4. Finally, we evaluate the derivative-based global sensitivity metrics. The indices presented in Figure 5.9, again qualitatively align with the Sobol' indices and the activity scores. The DGSM vary less than the other indices. They assign a larger value to the parameters with the lowest impact and a smaller value to the parameter with the largest impact. However, if we compare the lowest sensitivity values to those of the control parameter, we still deem the

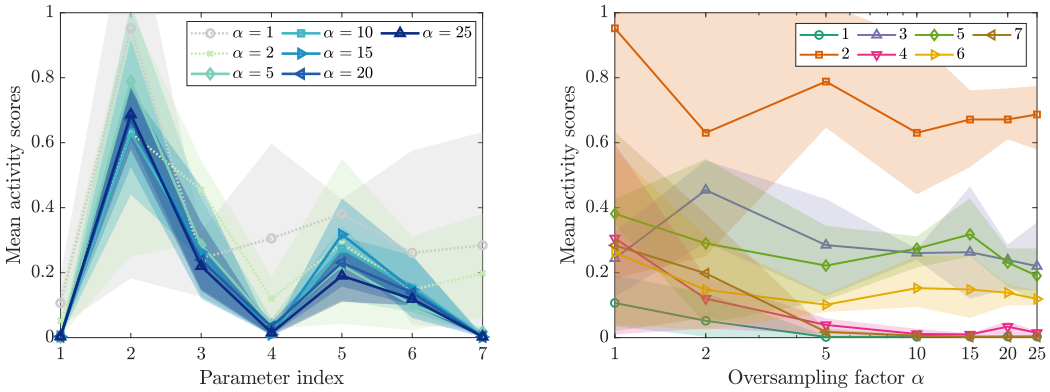


Figure 5.8: Parameter sensitivities measured by activity scores from active subspace method for the bottleneck scenario. For each value of the oversampling factor α, three runs are performed. The mean activity scores obtained for the runs (solid lines) are shown together with the limits of the scores (shaded areas). Parameters under study: control parameter (1), free-flow speed mean (2) and standard deviation (3), number of agents (4), obstacle repulsion (5), personal space (6), minimum step length (7).

number of agents, 4, and minimum step size, 7, as non-influential. Also, the free-flow speed mean is still the most influential parameter. Since each metric measures differently, we cannot expect the indices to be identical. While the derivative-based global sensitivity indices measure the average response to small perturbations in the input parameters, the activity scores for each parameter are calculated inversely from the joint important parameter direction that spans the active subspace, and the total sensitivity indices measure the variance that can be attributed to each parameter [Constantine and Diaz, 2017].

5.4.3.1 Active variable

Now, we take a closer look at the active subspace. For our seven-dimensional input parameter space, we were able to identify a one-dimensional active subspace. The important parameter direction formed by the first eigenvector can be used to transform any parameter vector x to the active subspace. The transformed parameter vector $w_1^T x$ is the so-called active variable. In Figure 5.10 shows one- and two-dimensional sufficient summary plots for the active variable. A scatter plot that contains all available regression information is referred to as a sufficient summary plot. Figure 5.10a shows the flow through the bottleneck against the active variable $w_1^T x$ whereas Figure 5.10b depicts the flow against both the active variable $w_1^T x$ and the inactive variable $w_2^T x$,. The large magnitude of the coefficient of determination $R^2 = 0.773$ suggests that a large portion of the output variance can be described using the active variable $w_1^T x$. In the two-dimensional sufficient summary plot, however, we can barely see any change in the direction of $w_2^T x$, the second most important direction in the parameter space.

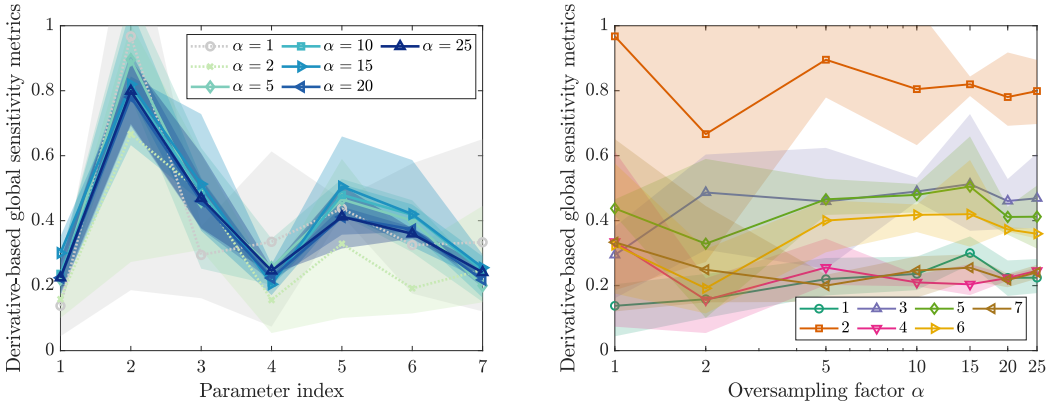
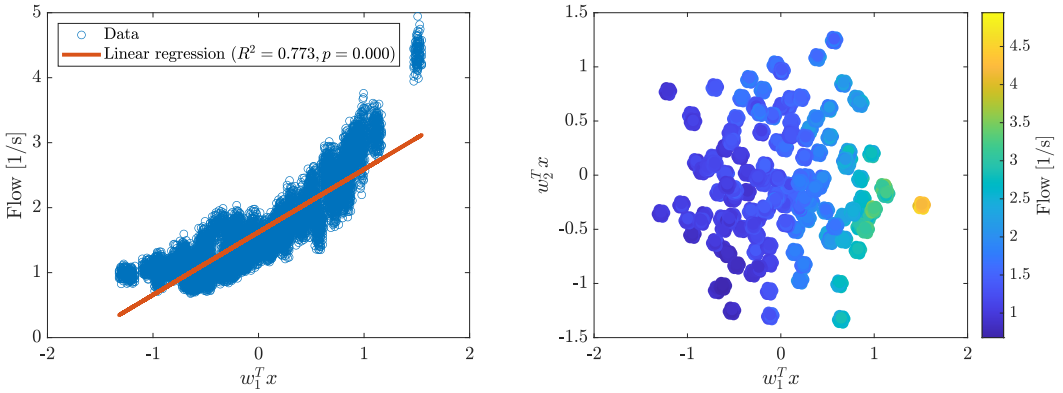


Figure 5.9: Parameter sensitivities measured by derivative-based global sensitivity metrics ν_i for the bottleneck scenario. For each value of the oversampling factor α, three runs are performed. The mean derivative-based global sensitivity metrics obtained for the runs (solid lines) are shown together with the limits of the scores (shaded areas). Parameters under study: Control parameter (1), free-flow speed mean (2) and standard deviation (3), number of agents (4), obstacle repulsion (5), personal space (6), minimum step length (7).

Both plots support choosing a one-dimensional subspace. The active subspace can be exploited for dimension reduction that allows for parameters studies that may not be feasible otherwise.



(a) One-dimensional summary plot

(b) Two-dimensional summary plot

Figure 5.10: Sufficient summary plots of the flow through the bottleneck against the active variable $w_1^T x$ and the first inactive variable $w_2^T x$ using the gradient samples for 81 sample points ($\alpha = 25$). The active variable $w_1^T x$ is the most important direction in the input parameter space. It is the base of the identified active subspace. The first inactive variable $w_2^T x$ is the second most important parameter direction.

5.4.3.2 Confidence intervals for the eigenvalues and the subspace distance

We perform a bootstrapping according to [Constantine, 2015, p. 43] to approximate the confidence intervals of the eigenvalues and the subspace distance because the eigenvalues are only approximated. Figure 5.11a shows the bootstrapping interval

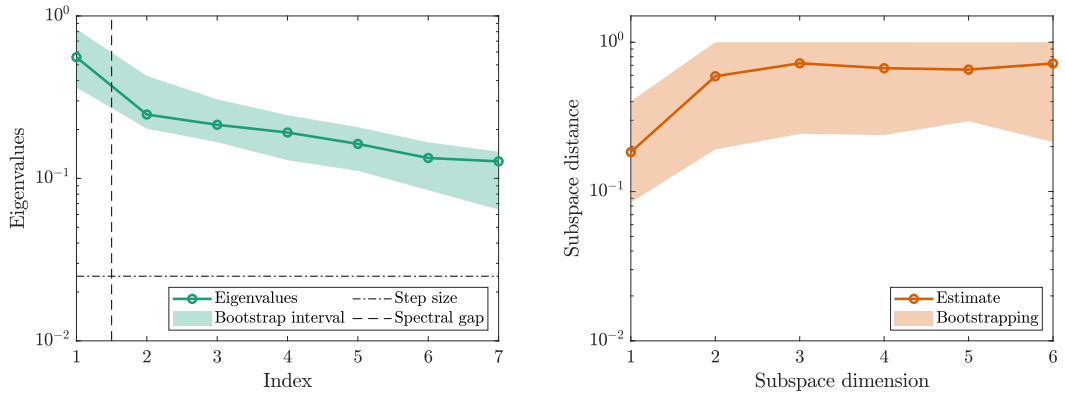
$$[\min_i(\lambda_i^*), \max_i(\lambda_i^*)] \tag{5.20}$$

for a run with $\alpha = 25$ as we obtained the best results with this oversampling factor. The bootstrapping confirms the spectral gap between first and second eigenvalue observed in Figure 5.6a. In [Constantine, 2015, p. 32], the distance between two subspaces is defined as

$$\text{dist}(\text{ran}(W_1), \text{ran}(\hat{W}_1)) = \| W_1^T \hat{W}_2 \|, \tag{5.21}$$

that is the distance between true active subspace W_1 and its approximation $\hat{W}_1$. In the case of bootstrapping, we calculate the distance between the obtained active subspace $\hat{W}_1$ of a single run to each inactive subspace $\hat{W}_1^{i_B}$ for each bootstrap replicate $i_B = 1, \ldots, M_{boot}$.

This distance is shown with its bootstrap interval in Figure 5.11b. We observe the smallest subspace error for a one-dimensional subspace. This result also strongly supports the choice of the one-dimensional subspace.



(a) Eigenvalues of matrix C

(b) Subspace distance according to (5.21) from [Constantine, 2015, p. 32]

Figure 5.11: Estimation of confidence intervals for eigenvalues and subspace distance by bootstrapping ($M_{boot} = 1000$). Bootstrapping for the eigenvalues confirms spectral gap between first and second eigenvalue. Smallest subspace distance is found for one-dimensional subspace.

5.4.4 Sensitivity ranking

When we compare the normalized sensitivity metrics, the parameters are ranked consistently for all metrics (compare Figure 5.12). The agreement between activity scores and

Sobol' indices supports the applicability of activity scores using gradient approximation to our model. Consistently, the free-flow speed mean is assessed as the most influential parameter. That means this parameter is responsible for most of the uncertainty in the model response. Consequently, the uncertainty associated with the parameter should be reduced by subsequent studies. This focus on the largest contributor is often referred to as the factor prioritization setting [Saltelli et al., 2008]. Also, all metrics attribute little to no contribution to the quantity of interest to the following parameters: number of agents and minimum step length. The indices for both are in the same range as the control parameter for which we know that it has no impact on the result. These parameters can be fixed to any value within their range for the subsequent studies. This reduction of the input parameter space is known as factor fixing setting [Saltelli et al., 2008].

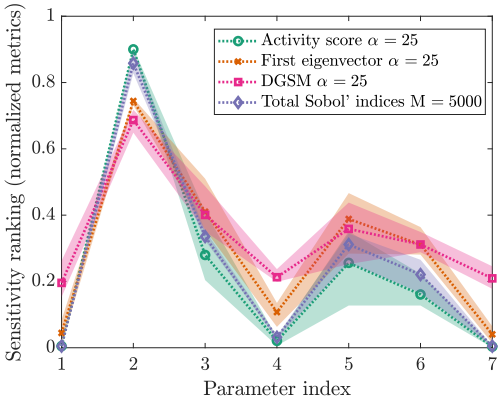


Figure 5.12: Normalized activity scores, first eigenvector entries, derivative-based global sensitivity metrics (DGSM), and Sobol' indices for the bottleneck scenario. Parameters under study: control parameter (1), free-flow speed mean (2) and standard deviation (3), number of agents (4), obstacle repulsion (5), personal space (6), minimum step length (7).

5.4.5 Time-dependent analysis of sensitivities

In the bottleneck scenario, we focus our analysis on a scalar definition of the flow as the quantity of interest. The flow through the bottleneck can also be evaluated at several times in the simulation, using e. g. the flow over time $J(t) = \rho(t)v(t)w$ in [Rupprecht et al., 2011]. Here, density $\rho(t)$ and velocity $v(t)$ are evaluated continuously throughout the simulation, and from those measurements, we obtain a time series of flow measurements. This time series can be understood as an observable of a dynamical system. Dynamical systems are omnipresent since they can describe many physical processes. Accordingly, many methods have been developed or adapted specifically for these systems. Crowd simulations can also be viewed as a dynamic system. The state is described by the agent's current position and speed. The sensitivity analysis methods presented in Section 5.3 are designed for scalar-valued quantities of interest. If we are

interested in the dynamics of a system, a continuously measured quantity of interest might be of interest.

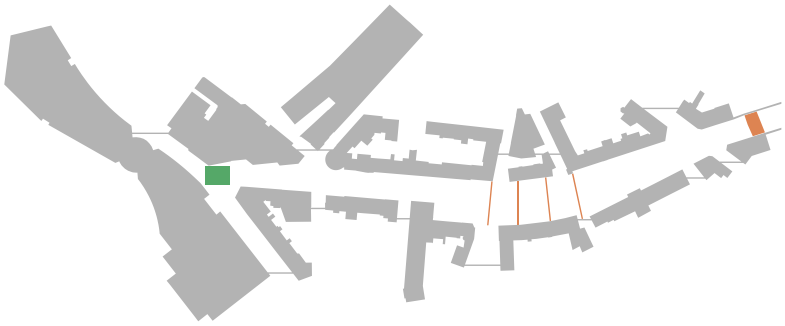


Figure 5.13: Fictional protest march scenario along the Richard-Wagner-Straße in Kaiserslautern. The topography is a simplified version of the actual environment. The agents start from the source (green), pass four intermediate destinations (orange lines) before they reach their final destination (orange polygon).

In [Rahn et al., 2021], we studied the length of a protest march in the city center of Kaiserslautern. The length of the march is measured using the geodesic distance which is also employed for the floor field. It is of great practical interest for the authorities because the route planning depends on it. Figure 5.13 depicts the topography. We used the most direct approach and calculated the sensitivity indices at every time step. While this requires extra effort as the calculation of the indices at each time step, it does not require additional model evaluations. Then, we analyzed the time series of the sensitivity indices. We studied the impact of the number of agents and the free-flow speed standard deviation on the length of the protest march. Table 5.3 holds details on the parameters. The number of agents was chosen as an uncertain parameter because the number of participants at protest marches is often not known beforehand. It depends on several factors such as weather and competing events. Both parameters are expected to have a considerable impact on the length of the protest march.

Table 5.3: Uncertain input parameters and their distribution used for the sensitivity analysis of the protest march.

Parameter	Unit	Range
Number of agents		$\mathcal{U}(400, 1200)$
Standard deviation of free-flow speed	m/s	$\mathcal{U}(0.05, 0.10)$

In the protest march scenario, we observed a shift from the number of agents to the free-flow speed standard deviation as the governing parameter with respect to the protest march length, as shown in Figure 5.14.

The results scientifically confirmed the usefulness of measures to reduce the variation in the walking speeds of the participants such as floats and banners that aim at synchronizing participants' speeds. Our results highlight that it is essential to run through

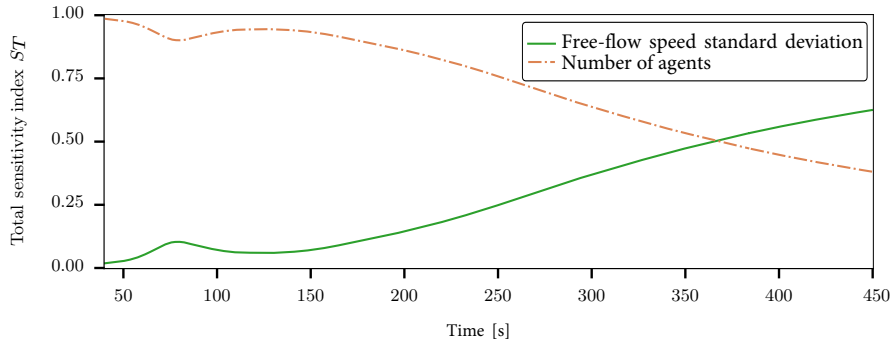


Figure 5.14: Parameter sensitivities measured by total Sobol' indices for the free-flow speed standard deviation and the number of agents with respect to the length of the protest march.

several what-if scenarios with varying numbers of participants so that routes of appropriate lengths can be assigned and contingency plans can be developed. Our findings demonstrate how uncertainty quantification can become a cornerstone of computer-based decision support systems for the planning of large events.

5.4.6 Efficient sensitivity analysis for computationally expensive scenarios

The investigated bottleneck scenario is comparably small and contains only a few dozens of agents. Simulating a whole event may include a large topography such as parts of a city for an urban event and a large number of agents, e. g. Oktoberfest in Munich with up to 350000 visitors at a time. In situations like these, it becomes computationally too expensive to evaluate the model at a larger number of sample points.

There are enhanced methods that can be applied in order to reduce computation time. For example, the calculation of Sobol' indices can be sped up using a low-discrepancy quasi-Monte Carlo sampling with the Sobol' sequence to reduce the number of model evaluations [Kucherenko et al., 2015]. This configuration is implemented in the SALib package [Herman and Usher, 2017] which is also employed in the uncertainty quantification framework.

Instead of changing the sampling method, we can also build a substitute for the model, which is then cheaper to evaluate than the original model. For Sobol' indices, generalized polynomial chaos expansion (gPCe) can be a basis for constructing a surrogate model. We describe the gPCe approach in more detail in chapter 8. These surrogates are constructed such that Sobol' indices can be cheaply derived [Sudret, 2008]. In [Rahn et al., 2021], we used two types of polynomial chaos expansions, point collocation, and the pseudo-spectral approach, as a basis for the sensitivity analysis. In order to benefit from this approach, less effort must be spent on the model evaluations for the construction of the surrogate than for the sensitivity analysis without a surrogate. This is typically the case, when the topography is large or when we have a large number of agents.

5.5 Summary

The work that I presented in this chapter aims to answer the research question *Q1: "How can we identify influential parameters in the optimal steps model for the bottleneck scenario?"* which is split into three sub-questions:

Q1.1: Which methods from sensitivity analysis are suited for crowd dynamics simulations?

I presented methods from uncertainty quantification to identify influential parameters and ranked them by their impact on the simulation outcome. I selected two metrics, Sobol' indices and activity scores, for the bottleneck scenario. Sobol' indices are a universal method for sensitivity analysis that can be used for all crowd simulation models. Activity scores identify lower-dimensional subspaces in the input parameter space, which can be employed for subsequent analysis. However, they require gradients or at least gradient approximations, which may not be well-defined in rule-based models such as cellular automata. In general, I recommend starting with Sobol' indices for studies of parameter sensitivity in crowd simulation. This method is widely known and accepted. In addition, there are multiple, and even open-source, implementations for the calculation of the indices available. Since the method purely relies on model evaluations, it can be applied to any crowd dynamics model.

Q1.2: Which parameters are influential and which are non-influential in the bottleneck scenario?

I calculated both activity scores and Sobol' indices for the bottleneck scenario. As a byproduct of the active subspace calculation, I also obtained the first eigenvector metric and the derivative-based global sensitivity metric. All metrics rank the parameters in the bottleneck scenario consistently: The most influential parameter is the free-flow speed mean, followed by the free-flow speed standard deviation, the obstacle repulsion, and the personal space strength. The analysis also showed that interactions are only present between free-flow speed mean and standard deviation. Two parameters, the minimum step length, and the number of agents were deemed non-influential. For subsequent studies, these parameters can be fixed to an arbitrary value within the range.

Moreover, when calculating the activity scores, the routines identified a one-dimensional active subspace in the seven-dimensional input parameter space. That means there is one linear combination of the input parameter that has a strong impact on the quantity of interest, the flow. This information about the system's structure obtained by the active subspace method can be exploited to construct a cheap, lower-dimensional global surrogate that makes parameter studies for expensive models feasible. In the next chapter, I calibrate the influential parameters to reduce the uncertainty in the simulation output.

In addition, we used Sobol' indices in [Rahn et al., 2021] to study the impact of the free-flow speed standard deviation and the number of agents on the length of a protest march in the city of Kaiserslautern over the course of the simulation. We observed a switch in the governing parameter from the number of agents to the free-flow speed standard deviation. In this case study, the time-dependent analysis of the sensitivity revealed the dynamics of the system.

Q1.3: How can we determine if the results are reliable?
Since the parameters were ranked identically by all four metrics, the results confirm each other. Additionally, they are in agreement with our knowledge of crowd dynamics: The flow is a product of speed and density. A high impact of the free-flow speed on the flow through the bottleneck is natural because the free-flow speed impacts the actual speed. An increased obstacle repulsion decreases the actual width of the bottleneck. Consequently, it reduces the flow. The total number of virtual pedestrians in the simulation, however, does not impact the flow. All of these points confirm the reliability of the ranking of the parameters.

6 Parameter Estimation: finding values for influential parameters

In the last chapter, I distinguished the influential from the non-influential parameters in the bottleneck scenario using sensitivity analysis. Now, I search for optimal values for these influential parameters that adjust the simulation to a given dataset. In Section 6.2, I remind the readers of the state of the art on calibration in crowd simulation so that they can evaluate my choice of methods for parameter estimation. I introduce such methods for parameter estimation from uncertainty quantification in Section 6.3. I select two methods for Bayesian inference and demonstrate their suitability on the bottleneck scenario in Section 6.4.

6.1 Introduction

I have identified the free-flow speed mean and standard deviation, the obstacle repulsion, and the personal space strength as influential parameters in the bottleneck scenario. In order to reduce the uncertainty in the simulation output, I will now calibrate these parameters. Calibration in crowd simulation is often performed manually or visually, which means, the parameter is adapted by hand until the simulation output matches a given data, e. g., an experimental dataset, qualitatively or quantitatively. For automated calibration, point estimates such as maximum likelihood estimates for parameter calibration have become more and more popular in the last years. While they constitute a systematic approach, point estimates typically only provide the best fit for the parameter value. They do not reflect the uncertainty in the parameter value after the calibration. Bayesian inference methods, however, integrate information about the uncertainty in the data, and in the misfit between data and model, in the outcome. Instead of providing just a value for the parameter, Bayesian inference methods provide a so-called posterior distribution of the unknown parameter. The posterior is a probability density function, which reflects the uncertainty after calibration. I present Bayesian inference methods from uncertainty quantification and choose two sampling methods that fit the needs of crowd simulation, Markov chain Monte Carlo (MCMC) and approximate Bayesian computation (ABC). Both are established methods that provide a full posterior distribution. I calibrate the influential parameters in the bottleneck scenario. I demonstrate the workings of MCMC and ABC in this scenario, I explain the differences of the methods, and I discuss in which situation which method is preferable. Then, I compare ABC as representative for Bayesian inference methods with a common point estimate. Here, the posterior mode is used as a point estimate. Based on three case studies, I discuss

in which cases Bayesian inference methods are beneficial and when a point estimate is sufficient.

Research questions addressed in this chapter

Q2 How can we calibrate the influential parameters in the bottleneck scenario?

- Q2.1 Which parameter estimation methods are suited for calibrating crowd dynamics models?
- Q2.2 What are the advantages of Bayesian inference methods for calibration compared to established methods, such as point estimates?
- Q2.3 What is the posterior distribution for the influential parameters in the bottleneck scenario after calibrating to experimental data?

Most of the findings in this chapter were published as journal article [Gödel et al., 2022] and in conference proceedings [Gödel et al., 2019a,b]. I refer to the respective papers in the text.

6.2 State of the art on parameter estimation in crowd simulation

In crowd dynamics, parameter estimation is not a common term, instead typically "calibration" is used. Calibration traditionally refers to finding a single value [Constantine, 2015, p. 65] for physical parameters [Oberkampf and Roy, 2010, p. 44] using experimental data [American Society of Mechanical Engineers, 2006]. However, I use calibration in a wider sense, including all parameter types, not restricted to finding a single value or using only experimental data. This setting is often referred to as parameter estimation. In short, I use calibration and parameter estimation interchangeably.

I define calibration as the process of adjusting model parameters such that a predefined simulation outcome resembles given data. However, the scientific community has not agreed upon a standardized approach for calibration in crowd simulation [Lovreglio et al., 2015]. I describe common approaches to calibration in three categories: based on the dataset, the parameters, and the methodology.

First, let us take a look at the data for calibration. I distinguish between microscopic data like individual trajectories and macroscopic data, which are aggregated measures like density or flow [Schadschneider, 2001]. For microscopic calibration, the first challenge is to obtain individual trajectories. They can be extracted from video footage by manual annotation [Antonini et al., 2006, Berrou et al., 2007, Ko et al., 2013, Robin et al., 2009, Tang and Jia, 2011], semi-automated using software [Dias and Lovreglio, 2018, Hoogendoorn and Daamen, 2007, Zeng et al., 2017], or from other sensors [Seer et al., 2014a,b]. The second challenge is the comparison of the observed trajectories with the simulated ones. Typical approaches include simulating one individual while moving all others according to observed trajectories [Zeng et al., 2017] as well as placing all agents at positions observed at a fixed time step and then simulating them in only one

simulation step [Wolinski et al., 2014]. Related to this, a suitable distance measure must also be defined [Guy et al., 2012, Wolinski et al., 2014]. It should be noted that errors in the trajectories can have a large impact on the result of calibration [Rudloff et al., 2014]. Even though individual agents are used in agent-based models, my goal is not to represent one specific individual, but rather to find representative types in order to portray a population. Therefore, I concentrate on macroscopic calibration as in [Berrou et al., 2007, Chu, 2009, Steiner et al., 2007, Wolinski et al., 2014]. The macroscopic quantity of interest in the bottleneck scenario is the flow. The data set for the calibration stem from observations in the real world. From video data, in addition to trajectories, macroscopic measures such as density or flow can be extracted. Alternatively, data collected in experiments can be used. I calibrate against flow measurements from a controlled experiment.

Second, let us look at the calibrated parameters. Most models contain both physical and non-physical parameters. Physical parameters can directly be measured, such as the torso size of a pedestrian, while non-physical parameters cannot be measured. Typical examples are parameters associated with the interaction among pedestrians as well as with obstacles such as the personal space strength, which expresses a person's need to keep a certain distance to others, and the obstacle repulsion in the optimal steps model. Some physical parameters can even be directly be measured from trajectories without requiring model evaluations. This so-called direct calibration has been used for speeds in social force models [Hussein and Sayed, 2018, Tang and Jia, 2011, Zeng et al., 2014]. Non-physical parameters, however, require indirect calibration. That means model evaluations are compared to data. This is necessary because the parameters exist only in the model, but not in reality. Calibration of parameters associated with the interaction forces in social force models has been studied extensively [Daamen et al., 2013, Dias et al., 2018, Hoogendoorn and Daamen, 2006, 2007, Johansson et al., 2007, Seer et al., 2014a, Steiner et al., 2007, Taherifar et al., 2019, Tang and Jia, 2011, Voloshin et al., 2015, Zeng et al., 2017]. These are similar to obstacle repulsion and the personal space concept in the optimal steps model. I argue that the free-flow speed or desired speed, which I calibrate in this study, lies in between the two classes. On the one hand, the speed of an individual, in general, can be measured and, in some experiments, participants are asked to move through a topography in order to estimate their free-flow speed. On the other hand, subjects are observed during experiments and may therefore adapt their behavior. In addition, fitness, time of day, and mood may also have an impact. Consequently, an intrinsic free-flow speed cannot be measured without bias. I demonstrate the methods for the calibration of the most influential parameter, the free-flow speed mean. Then, I calibrate all four influential parameters, the free-flow speed mean and standard deviation, obstacle repulsion, and personal space strength.

Third, we look at approaches for calibration. Qualitative calibration involves visually comparing simulation results to video footage, either to compare motion and behavior or to compare quantities of interest such as density-flow relationships. However, visual comparisons are often subjective and difficult to standardize. To obtain comparable results, it would be necessary to formalize the comparison. Qualitative calibration implies that the parameters are varied by hand which often means one parameter is adapted at

a time. I believe that this approach is common in practical application, but literature is scarce. Theoretically, qualitative calibration could be formalized and computerized. The work presented by Steiner et al. who present a criterion for the smoothness of the trajectories is a step in this direction [Steiner et al., 2007]. Besides qualitative calibration, there is quantitative calibration in which the quantity of interest is compared numerically. This, too, can be done manually. If a one-factor-at-a-time approach is used for calibrating multiple parameters, the global optimum of the distance measure, might not be found. For automated quantitative calibration, regression with least squares [Guo et al., 2012, Johansson et al., 2007, Tang and Jia, 2011, Seer et al., 2014b] and maximum likelihood estimation [Antonini et al., 2006, Campanella et al., 2011, Daamen and Hoogendoorn, 2012, Hoogendoorn and Daamen, 2006, 2007, Ko et al., 2013, Lovreglio et al., 2015, Robin et al., 2009, Zeng et al., 2014] are popular. For the comparison between simulated and observed data, a distance measure needs to be defined. Both regression and maximum likelihood estimation aim to minimize the distance. I will use Bayesian inference methods for quantitative calibration which provides a full posterior distribution for the uncertain parameters.

Finally, one must address the challenges caused by the fact the crowd simulators, as a rule, are stochastic simulators. There are very few publications on this issue. They either remove the noise [Daamen et al., 2013] or average a fixed number of repetitions in order to remove or at least reduce the stochasticity in the output [Chu, 2009, Taherifar et al., 2019].

6.3 Methods for parameter estimation

In this section, I describe methods for parameter estimation from an uncertainty quantification perspective. The mathematical problem formulation is the inverse problem in which one aims to find p, the parameter vector which gives a model response $f(X, p)$ close to the observed data d such that

$$y = f(X, p) \tag{6.1}$$

where X are independent variables and y denotes the quantity of interest. That means f is not only the simulation that gives us positions of the agents, but it includes the output processors that calculate the quantity of interest, such as speed, density, or flow. In practice, when we have observations that the measurement introduces measurement noise ϵ:

$$v = f(X, p) + \epsilon$$

The statistical inverse problem is to estimate p and quantify its uncertainties given the noisy measurements. Since the dependence on X is of secondary importance, I omit X in the following within the notation $f(p)$.

Parameter estimation aims to solve the statistical inverse problem. The methods for this can be divided into frequentist estimators and Bayesian estimators. In the frequentist approach, parameter estimation means that there is a fixed true parameter

value, which we need to find. In this work, I employ Bayesian techniques for the inverse problem. In the Bayesian view, the uncertain parameter is considered a random variable. Therefore, the solution of the inverse problem is a posterior density for the uncertain parameter. Figure 6.1 illustrates how Bayesian inference works. Central to Bayesian inference is Bayes' theorem

$$\rho_{pos}(x \mid d) = \frac{\rho_{prior}(x) \cdot \rho_{like}(d \mid x)}{\rho(d)} \equiv \rho_{prior}(x) \cdot \rho_{like}(d \mid x) \tag{6.2}$$

which relates the posterior density $\rho_{pos}(x \mid d)$ to the prior distribution ρ_{prior} and the likelihood $\rho_{like}(d \mid x)$ using the evidence $\rho(d)$. The likelihood function represents the probability of the data given the parameter set x.

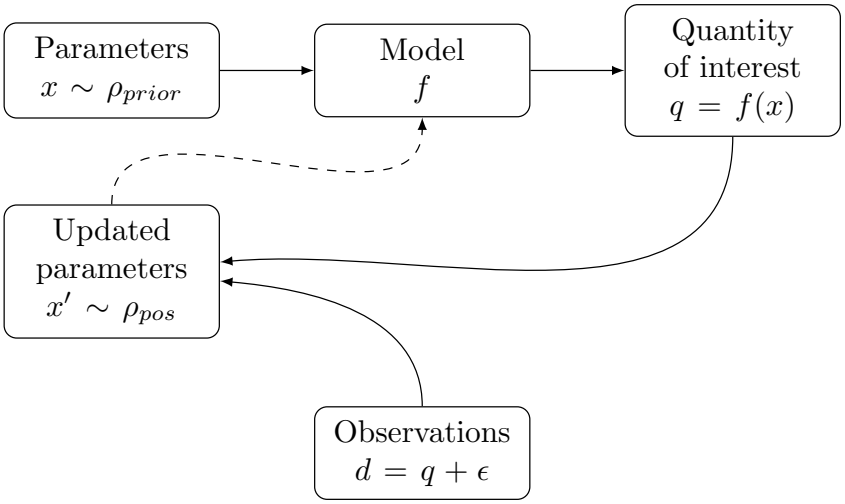


Figure 6.1: Scheme for Bayesian inference: Based on a prior distribution of the uncertain parameters, the model is evaluated and the model output is compared to the observations. The same quantity of interest needs to be measured in both simulation and observation. The result is an informed posterior distribution of the uncertain parameters that can be used for subsequent studies.

According to Hadamard [Hadamard, 1902], a problem is well-posed if three criteria are met: 1) existence, i. e., a solution exists, 2) uniqueness, i. e., the solution is unique, 3) stability, i. e., the solution continuously depends on the data. Inverse problems are often ill-posed because measurement noise is present. In this case, the solution is not unique. In practical applications, the continuity of the solution on the parameters is also often violated. Regularization can help to transform an ill-posed problem into a well-posed problem. While the frequentist approach works on the ill-posed inverse problem, the Bayesian inverse problem is well-posed [Stuart, 2010].

Figure 6.2 summarizes frequentist and Bayesian approaches for parameter estimation. As already mentioned, regression and maximum likelihood estimation techniques for optimizing the distance function are commonly used in pedestrian dynamics. Both methods

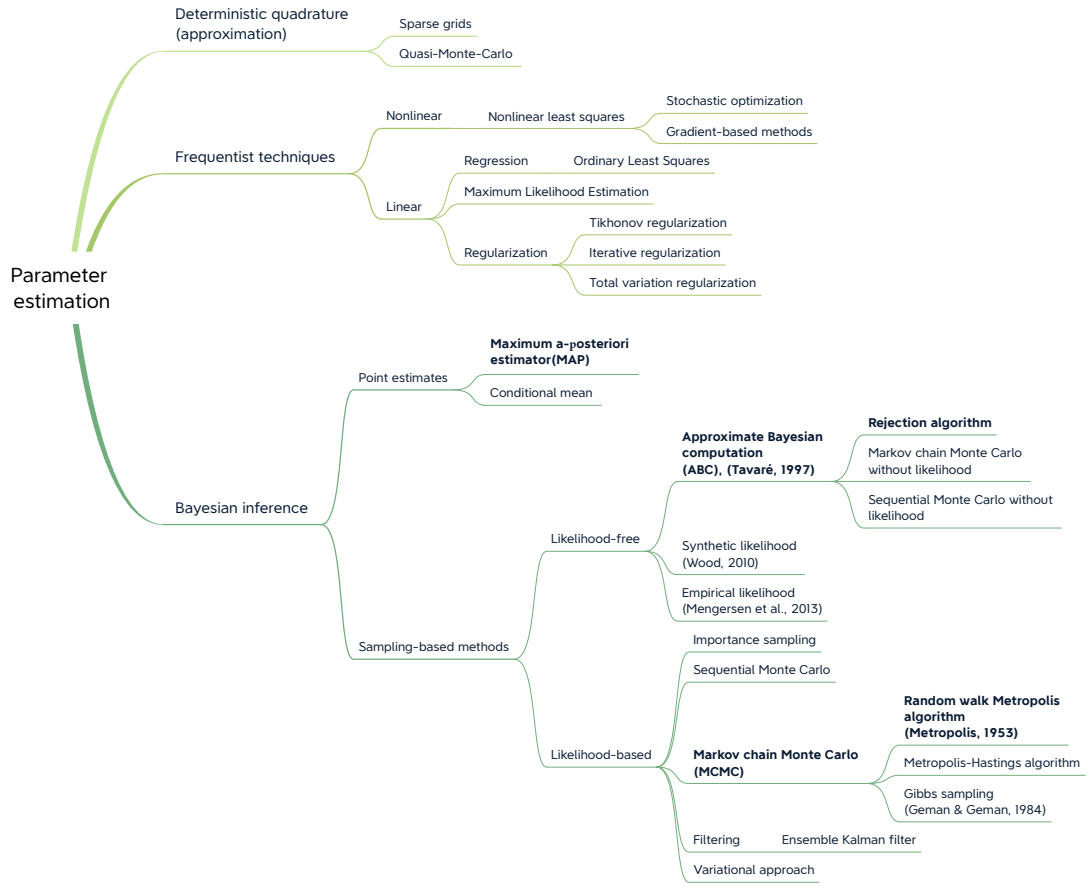


Figure 6.2: Overview of methods for parameter estimation. Methods that were selected for crowd simulations are highlighted.

assume a linear model. For nonlinear models, such as crowd dynamics models, equivalently, nonlinear least squares can be employed. On the other hand, there are Bayesian inference methods. They include point estimation techniques such as the maximum a-posteriori estimator, which solves an optimization problem, and the conditional mean, which solves an integration problem. Most common are sampling-based approaches for Bayesian inference. Here I distinguish between likelihood-free and likelihood-based approaches. I choose an established likelihood-based approach, namely the random walk Metropolis algorithm [Metropolis et al., 1953], a Markov chain Monte Carlo approach. For many problems, the likelihood function is either not known or computationally expensive. For this reason, likelihood-free approaches such as the synthetic likelihood [Wood, 2010] and empirical likelihood [Mengersen et al., 2013] have been developed. In this work, I additionally employ approximate Bayesian computation [Tavaré et al., 1997, Beaumont et al., 2002] as a likelihood-free technique because it can deal with stochastic simulators [Toni et al., 2009]. Please note that likelihood-free methods also use a likeli-

hood but the likelihood is computed implicitly within the method and not passed to it explicitly. Both methods are explained in more detail in the following sections.

6.3.1 Posterior mode as point estimate

For the comparison of Bayesian inference, represented by approximate Bayesian computation, with point estimation in Section 6.4.3, I define the posterior mode as the point estimate. I derive the point estimate from the model evaluations obtained during ABC rejection: I choose the parameter value closest to the data in terms of the distance measure f_d, the posterior mode $x_{pe} = \arg\min_{x_c} f_d(x_c)$. This is equivalent to the maximum a-posteriori estimate.

6.3.2 Bayesian inference with likelihood: Markov chain Monte Carlo method

In all Bayesian approaches, a prior distribution that reflects the initial knowledge on the uncertain parameters is updated using observational data and as a result, we obtain an informed posterior distribution for the uncertain parameters. The idea behind the Markov chain Monte Carlo approach is to construct an ergodic Markov chain whose stationary distribution is the posterior distribution. The MCMC approach is an iterative algorithm: The chain starts at an initial point and moves through the parameter space according to the proposal distribution which is the basis for generating new proposals. Both the initial point and the proposal distribution need to be defined by the user. Each position in the chain is a candidate. Candidates which meet a criterion based on a distance measure become samples of the posterior distribution. In this fashion, running the chain generates samples from the posterior distribution.

There are several MCMC algorithms. The random walk Metropolis algorithm [Metropolis et al., 1953] is suited for symmetric proposal distributions. If the proposal distribution is asymmetric, an extension, the Metropolis-Hasting algorithm [Hastings, 1970] is required. The Markov chain, unfortunately, suffers from poor mixing behavior in high dimensions. In these cases, Gibb's sampling [Geman and Geman, 1984, Gelfand, 2000] can be used instead.

Metropolis Algorithm

I employ the random walk Metropolis algorithm. It requires a model $f(x)$ that can be evaluated at parameter set x, a symmetric proposal function $g(x)$, and a number of iterations N. Typically, for the proposal distribution a Gaussian distribution centered at the current position x_t of the Markov chain, $\mathcal{N}(x_t, \Delta\tau)$, is used. The covariance of the proposal distribution is jump width $\Delta\tau$. The iterative algorithm consists of three steps. Algorithm 1 requires the evaluation of the likelihood in the initialization and in step 2 of every iteration to approximate the posterior distribution.

These steps are performed iteratively until the specified number of iterations N is reached. The Markov chain needs a certain time until it converges. Therefore, the samples obtained before the chain has settled are often removed to avoid distortions of

Algorithm 1 Random walk Metropolis algorithm

Initialization: Evaluate $p(x_0) = \rho_{prior}(x_0) \cdot \rho_{like}(d \mid x_0)$, given data d and initial parameter set x_0.

1. Create new candidate $x' = x_t + \Delta\tau \cdot z$ where z is a sample from the proposal distribution $g(x)$.
2. Evaluate $p(x') = \rho_{prior}(x) \cdot \rho_{like}(d \mid x)$ at candidate position x' which is equivalent to the posterior.
3. If $p(x') \geq p(x_t)$, accept the candidate x' as new sample $x_{t+1} = x'$
   Otherwise, calculate acceptance ratio $r = \frac{p(x')}{p(x_t)}$.
   a) Draw $u \sim \mathcal{U}(0, 1)$.
   b) If $u < r$, accept the candidate x' as new sample $x_{t+1} = x'$.
   c) Otherwise, reject the candidate. In this case, the sample is the last candidate, $x_{t+1} = x_t$.

the posterior distribution. The number of samples discarded is referred to as burn-in N_b.

Common measures of the performance of the chain are the acceptance rate, which is the ratio between the number of accepted candidates and the number of iterations N. In addition, the effective sample size (ESS) estimates the number of independent posterior samples obtained by the chain.

Since the jump width is an essential parameter to the MCMC algorithm and its optimal size depends on various factors such as the initial point, I decided to add an adaptive jump width regulation. This is a common approach [Haario et al., 1999]. However, it needs to be kept in mind that the resulting posterior distribution might differ from the true posterior. The adaptive jump width is regulated according to the acceptance rate. For a one-dimensional problem, the optimal acceptance rate is at 44% [Gelman et al., 1996]. I evaluate the acceptance rate in batches and compare it to a given interval of the desired acceptance rate. If the acceptance rate is outside of the interval, the jump width is adjusted accordingly.

6.3.3 Bayesian inference without likelihood: approximate Bayesian computation

For models in which the likelihood function is intractable or difficult to calculate, likelihood-free methods can be employed. They refrain from likelihood evaluations. Instead, the likelihood is only calculated implicitly.

Approximate Bayesian computation is commonly used in biological sciences such as population genetics, ecology, and epidemiology. ABC can deal with stochastic models which is the case for most biological models. This also makes them a good fit for

models in crowd dynamics. Therefore, I choose approximate Bayesian computation as a likelihood-free method.

ABC can be carried out with different algorithms. A straightforward choice is the rejection algorithm. In addition, ABC can also be combined with likelihood-free versions of sequential Monte Carlo or Markov chain Monte Carlo sampling strategies [Marjoram et al., 2003]. For all algorithms, a central step is the comparison of an acceptance criterion to a user-defined tolerance. This global comparison is the main difference of likelihood-free MCMC compared to classical MCMC which relies on a local comparison. In the sequential Monte Carlo approach, a series of decreasing tolerances is used instead of a single tolerance. The ABC rejection sampler follows three main steps outlined in Algorithm 2.

Algorithm 2 Rejection sampler for approximate Bayesian computation

1. A large number of candidates x_c is generated from the prior.
2. At each candidate, the model $f(x)$ is evaluated.
3. The distance measure $\rho(\cdot, \cdot)$ is compared to predefined tolerance ϵ. If $\rho(f(x_c), d) < \epsilon$, the candidate is accepted and therefore becomes a sample of the posterior approximate, otherwise x_c is rejected.

The distance measure is defined by the user. It serves as a metric for the distance between the model evaluation at the candidate $f(x_c)$ and the data d used for Bayesian inference. The choice of tolerance ϵ can be understood as a trade-off between computability and accuracy. For $\epsilon = 0$, the method is exact, i. e. the accepted candidates are from the true posterior. When using a large tolerance, Bayesian inference finds the prior since all candidates are accepted. One can argue that for $\epsilon = 0$ the rejection sampler does not constitute an ABC algorithm since the rejection is exact. For most practical applications, however, $\epsilon = 0$ is not feasible anyway. Strictly speaking, for $\epsilon > 0$, the result of the rejection algorithm is an approximation to the posterior $\pi(\delta \mid \rho(d, X) \leq \epsilon)$ instead of the true posterior $\pi(\delta \mid d)$. In the limit, $\epsilon \to 0$ and $N \to \infty$, the approximation approaches the true posterior, $\pi(\delta \mid \rho(d, X) \leq \epsilon) \to \pi(\delta \mid d)$.

Since the rejection scheme does not gain information about the posterior from rejected candidates as MCMC does, a large number of candidates is necessary to approximate the posterior well. When the data is high-dimensional, often summary statistics $S(x)$ are applied to reduce the dimension. Simple examples are mean or standard deviation of the data. Then, the distance is calculated between the summary statistics of data $S(d)$ and the model evaluation $S(X)$ at parameter X is used for the decision about acceptance or rejection, compare Algorithm 3.

The summary statistics need to be sufficient, i. e. $S(X)$ holds all the information about X, in order to find the true posterior. In practice, however, finding sufficient summary statistics is not a trivial task and requires a likelihood. Using a non-sufficient summary statistic in ABC rejection adds a second layer of approximation. Several manuscript are

Algorithm 3 Rejection sampling with summary statistics

3. If $\rho(S(d), S(X)) \leq \epsilon$: accept, otherwise: reject.

concerned with finding a summary statistic for ABC [Jung and Marjoram, 2011, Blum, 2010, Barnes et al., 2012, Fearnhead and Prangle, 2012, Burr and Skurikhin, 2013].

6.4 Studying calibration of crowd simulation

In the following, I apply both a likelihood-based and a likelihood-free Bayesian inference method to solve the Bayesian inverse problem. Both methods provide a full posterior distribution for the uncertain parameters. Point estimates, which are commonly used for calibration in crowd dynamics, only provide a single estimate. Posterior distributions can be employed for subsequent studies to consider residual uncertainty in the parameters after calibration. Likelihood-free and likelihood-based approaches are suitable in different settings which I will discuss. For the decision, the stochasticity of the model and the likelihood function itself should be considered. These arguments can be understood as motivation for the different analyses performed in this chapter.

Stochastic models Likelihood-based inference in the form of Markov chain Monte Carlo approaches requires a deterministic model. If a likelihood-based approach is to be applied to a stochastic model, the model can be replaced by a deterministic surrogate. Whenever the generation of a surrogate model is too complex, averaging multiple evaluations at the same parameter value can be an approximation to a deterministic average model lying underneath. Nevertheless, in this case, the magnitude of the stochastic effects is not considered for the inference. The likelihood-free algorithms that I employ in this chapter can deal with stochastic models. Models for crowd dynamics are stochastic due to their initialization. If the stochastic effects need to be taken into account for calibration, approximate Bayesian computation is preferable.

Likelihood function Likelihood-based inference requires, as indicated by its name, a likelihood. The likelihood is the probability of the observed data given a parameter set. It is often represented by a zero-mean Gaussian distribution whose covariance is the measurement noise from the observations. In most practical applications, the true likelihood is unknown and the Gaussian likelihood is an assumption. However, this is a strong assumption that is not always justified or suitable. If the likelihood is unknown and it is not possible to make assumptions about it without restricting the analysis, it is intractable, or it is computationally too demanding to evaluate, likelihood-free inference can be used.

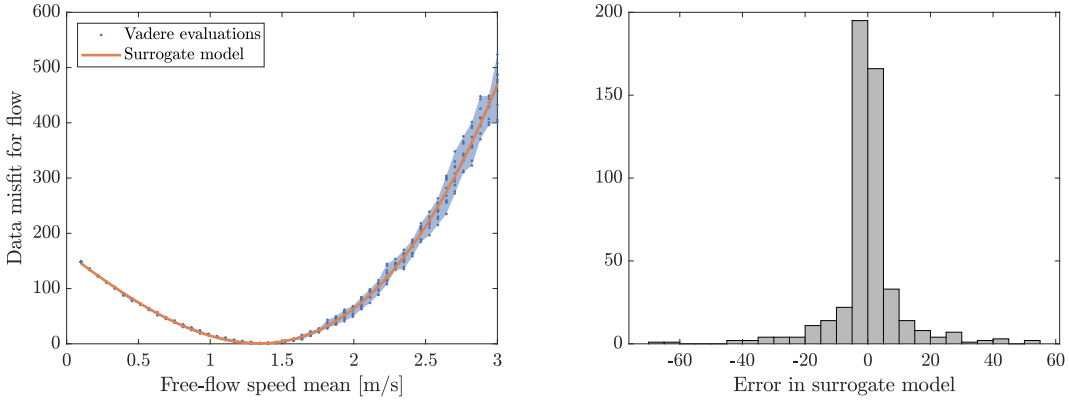
6.4.1 Likelihood-based inference with Markov chain Monte Carlo

I study four configurations to demonstrate Bayesian inference with a Markov chain Monte Carlo approach: First, a proof-of-concept with artificial data using a deterministic surrogate for the data misfit. Second, I perform calibration with experimental data and a deterministic surrogate for the data misfit function. Third, I give a proof-of-concept with artificial data with averaging of several repetitions is carried out. Finally, I use experimental data for Bayesian inference with experimental data while averaging several repetitions.

6.4.1.1 Proof-of-concept: artificial data, surrogate for data misfit

At first, I carry out a proof-of-concept using artificial data, that is, the data used for inference is obtained from simulation. Consequently, we know the true parameter value and can compare the posterior obtained with Bayesian inference to it. I calibrate the free-flow speed mean in the widest bottleneck presented in Section 3.2.1. The data point for Bayesian inference is the model output at the true parameter $\theta^* = 1.34\,\mathrm{m/s}$.

In order to remove the stochasticity in the simulator, I build a surrogate model for the data misfit function $g_d(x) \approx f_d(x)$. I fit a polynomial surrogate of order 4 by regression. The resulting surrogate for the data misfit function is shown in Figure 6.3 together with a histogram of the residuals to the actual model.

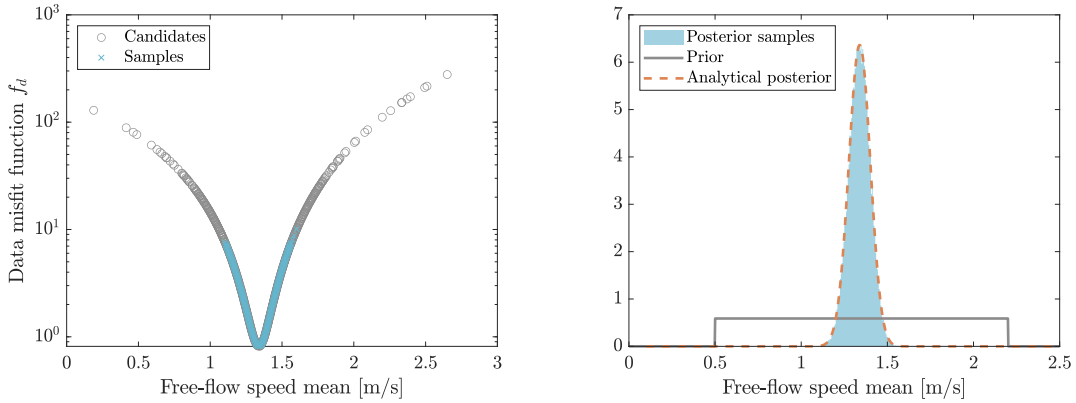


(a) Polynomial surrogate **(b)** Residuals

Figure 6.3: Surrogate model for data misfit function $f_d(x)$ using the model response at the free-flow speed mean θ^* as artificial data point, $d = f(\theta^*)$. The surrogate is build for a bottleneck scenario in which the flow through the bottleneck is observed.

Figure 6.4a shows the data misfit function for all candidates generated by the Markov chain. The posterior samples are color-coded. As expected, the minimum of the data misfit is found for the true parameter value. Figure 6.4b shows the prior distribution of the uncertain parameter together with its posterior. We observe that the posterior shape differs significantly from the prior indicating that the posterior was informed by the data and that the inferred parameter is identifiable. The analytical posterior is obtained by

calculating the product of likelihood and prior at equidistant samples. The alignment between the histogram of the samples and this analytical posterior verifies the sampling. The resulting posterior has a mean of 1.3383 m/s, mode of 1.3258 m/s, and a standard deviation of 0.0622 m/s. Both mean and mode are about the same as the true parameter value, which demonstrates that Bayesian inference is able to recover the true parameter value. The width of the posterior is mainly driven by the measurement noise in the relationship between data and model evaluation.



(a) Data misfit function **(b)** Histogram of posterior samples

Figure 6.4: Posterior of the free-flow speed mean in the scenario with five bottlenecks of increasing widths obtained with Metropolis algorithm.

In the next step, we take a look at the evolution of the Markov chain. Figure 6.5 plots the evolution of candidates and samples over a logarithmic x-axis to focus on the starting period of the chain and settled period. The Markov chain starts at the user-defined starting point, in this case, a free-flow speed mean of 0.75 m/s. The starting point is always a sample since there is no local comparison possible yet. We observe that the chain needs a certain period to settle in the region of the true posterior. The samples generated before the chain has settled are often referred to as burn-in. I choose a burn-in of 100 iterations of the chain. After the burn-in period, the variation in the candidates is still larger than the variation in the samples. While intuitively appealing, it would not be ideal for candidates and samples to be identical after burn-in since this implicates an acceptance rate of 100%. With a large acceptance rate, the samples are highly correlated.

I use an adaptive jump width regulation that keeps the acceptance rate between 0.4 and 0.6. Figure 6.6a shows the evolution of the jump width and the acceptance rate over the number of iterations. In [Gödel et al., 2019a], I empirically showed that this adaptive regulation obtains the largest effective sample size compared to several runs with fixed jump widths. Figure 6.6b shows the correlation measured by the autocorrelation function (ACF) between the samples. Since the candidates are drawn iteratively from a Markov chain, they are highly dependent. Consequently, also the samples are dependent. Ideally, we would obtain independent posterior samples. The correlation among samples depends

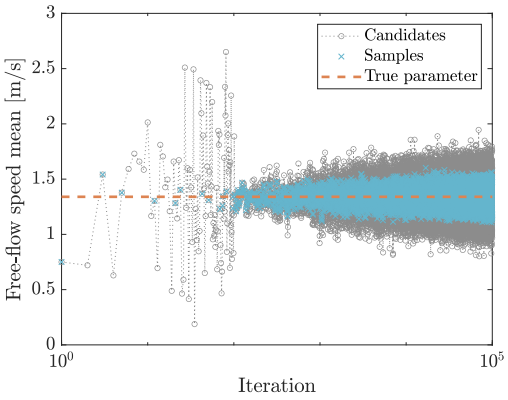


Figure 6.5: Evolution of the Markov chain: After a settling phase, the candidates and samples concentrate around the true parameter value $\theta^\star$ of the artificial data point $d = f(\theta^\star)$.

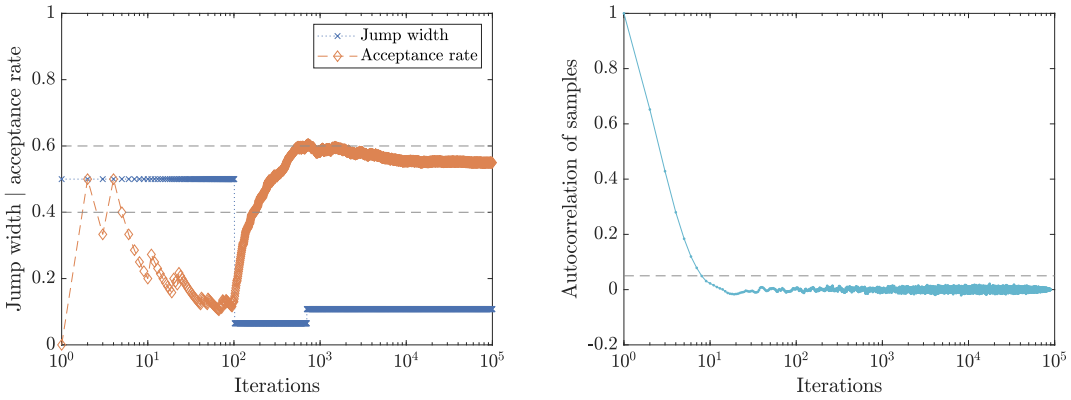
on the chosen jump width for the proposal function. I calculate the effective sample size as an estimate for the number of independent posterior samples obtained by the chain according to [Kruschke, 2015, p. 184]:

$$\mathrm{ESS} = N/(1 + 2\sum_{k=1}^{\infty} \mathrm{ACF}(k) \tag{6.3}$$

where N is the number of iterations of the chain, and $\mathrm{ACF}(k)$ is the autocorrelation function of the chain at lag k. According to [Kruschke, 2015, p. 184], I neglect autocorrelations below 0.05 for when calculating the effective sample size. The ESS of this chain is 15187.06, which is about 15% of the number of candidates generated. This can also be seen from the ACF in Figure 6.6b: Until a lag of 7, the autocorrelation is higher than 0.05. If we choose every seventh sample, $1/7 \approx 15\%$ remain as uncorrelated samples. The overall acceptance rate is 48.3%, close to the optimal acceptance rate for a one-dimensional inference of 43% [Gelman et al., 1996].

6.4.1.2 Bayesian inference in the bottleneck scenario: experimental data, surrogate for data misfit

After I demonstrated that Bayesian inference with an MCMC method, the Metropolis algorithm, can recover the true parameter value in a proof-of-concept setup, I now integrate experimental data in my setup. Instead of an artificial data point, I use the flow values measured by Seyfried et al. in [Seyfried et al., 2009]. Otherwise, the scenario is the same. Again, I construct a polynomial surrogate for the data misfit. We cannot employ the surrogate from the previous inference since the data misfit function depends on the data. Since I use real data now, we do not know the underlying true parameter value and, consequently, we can only judge whether the posterior is plausible or not.



(a) Jump width and acceptance rate

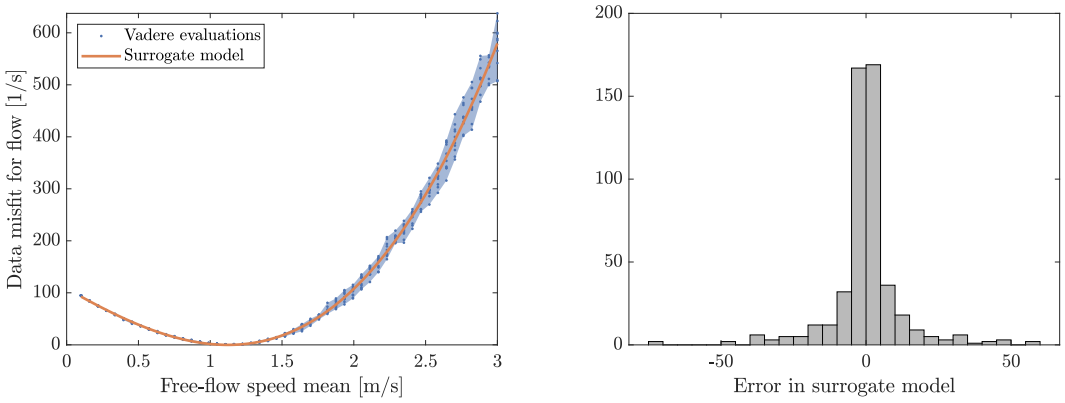
(b) Autocorrelation of samples

Figure 6.6: Performance criteria of Markov chain Monte Carlo sampling: a) the adaptive jump width regulation is effective in keeping the acceptance rate between 0.4 and 0.6 after about 200 iterations, b) autocorrelation of samples points with lag ≤ 7 are considerably correlated.

The polynomial surrogate of order four is shown in Figure 6.7. It has two roots in the parameter interval of $[0, 3]$, at 1.0876 and 1.1590, and is therefore not a non-negative polynomial. This is not ideal since the data misfit cannot reach negative values for the normal likelihood. A few methods, e. g. radial basis function interpolation, provide non-negative surrogates if the data for the fit is non-negative. Another option is to interpolate the mean of the model responses with a spline. This option might, however, add local extrema in which the chain could get stuck. I decided to keep the polynomial surrogate because the negative values only occur for a small interval, $[1.0876, 1.1590]$, the differences to the data are small, and the likelihood is still well-defined. It is, however, important to keep these issues in mind since they might have larger implications in other models and are not as easy to detect when multiple parameters are inferred.

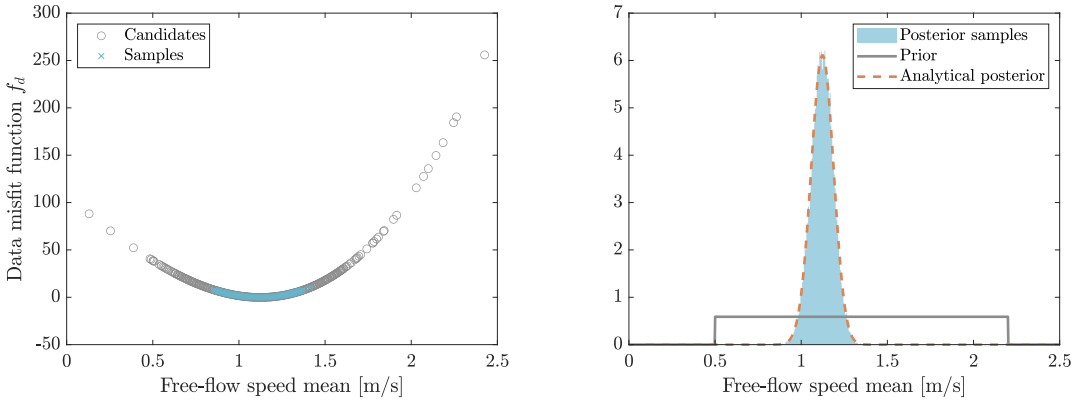
The resulting posterior is shown in Figure 6.8. As for the proof-of-concept, the histogram of posterior samples coincides with the theoretical posterior and thus confirms the sampling. It has a mean of 1.122 m/s, mode of 1.132 m/s, and standard deviation of 0.0644 m/s. Compared to the artificial dataset, the mode is a bit smaller, indicating a lower free-flow speed across all bottleneck widths. This posterior can be used for subsequent studies. When propagating it through the model, we obtain the remaining uncertainty in the output.

The evaluation of the performance criteria in Figure 6.9 yields similar results as the inference with artificial data. The overall acceptance is 52.6%, well within the targeted interval of $[40\%, 60\%]$ and the effective sample size is 13911.90, about 15% of the posterior samples generated by the chain.



(a) Polynomial surrogate **(b)** Residuals

Figure 6.7: Surrogate model for the data misfit function $f_d(x)$ for the five bottleneck scenario. Data misfit function measures the distance between model evaluations at different values for the free-flow speed mean and an experimental data set from [Seyfried et al., 2009].

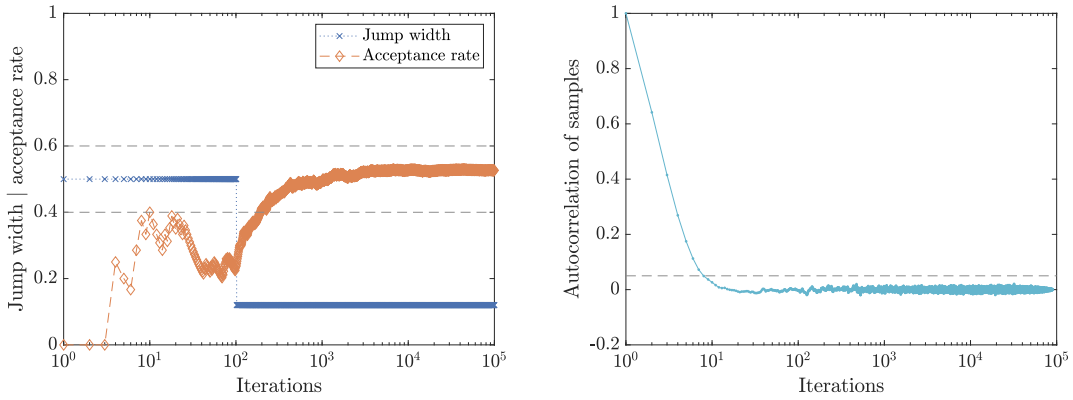


(a) Data misfit function **(b)** Histogram of posterior samples

Figure 6.8: Posterior samples for free-flow speed mean obtained with Metropolis algorithm for the five bottleneck scenario. For the calibration, the experimental flow values measured by [Seyfried et al., 2009] were used.

6.4.1.3 Proof-of-concept: artificial data, averaging of model runs

Instead of using a surrogate, I now average several repetitions at each parameter value to remove or at least reduce the stochasticity in the model. Averaging repetitions is computationally demanding, but it can be performed for any model without the complexity of having to choose a function type for the surrogate. In the one-dimensional setup, this option is feasible, for a higher-dimensional inference, the iterative algorithm might be too slow. Bayesian inference without a surrogate in this one-dimensional case



(a) Jump width and acceptance rate

(b) Autocorrelation of samples

Figure 6.9: Performance criteria of Markov chain Monte Carlo sampling: a) the adaptive jump width regulation successfully maintains an acceptance rate between 0.4 and 0.6 once the chain has settled, b) autocorrelation among the samples shows a high correlation within about 7 iterations.

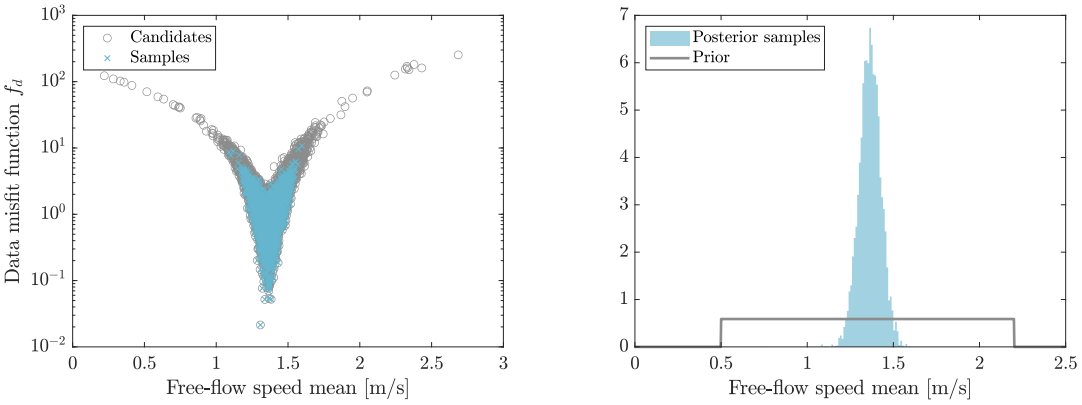
is substantially slower than the construction of the surrogate and inference with the surrogate in the last section. Therefore, I use only 10^4 iterations for the inference. From Figure 6.10a we can see that averaging of 10 data points is not sufficient to remove the stochasticity. Instead, stochasticity and noise are only reduced.

The mean of the posterior samples is 1.3638, the mode is 1.3399, and the standard deviation is 0.0630 (Fig. 6.10). Even though there is still some stochasticity and noise in the averaged model response, likelihood-based inference successfully recovers the true parameter value of 1.34 m/s. Looking at the performance metrics of the chain, the acceptance rate is at 54.6%. The effective samples size is 1468.74 and therefore comparably lower than for the previously studied cases with the surrogate. Figure 6.11 shows the evolution of the jump width over the chain iterations as well as the autocorrelation among samples. Both are similar to the results with the surrogate model.

6.4.1.4 Bayesian inference in the bottleneck scenario: experimental data, averaging of model runs

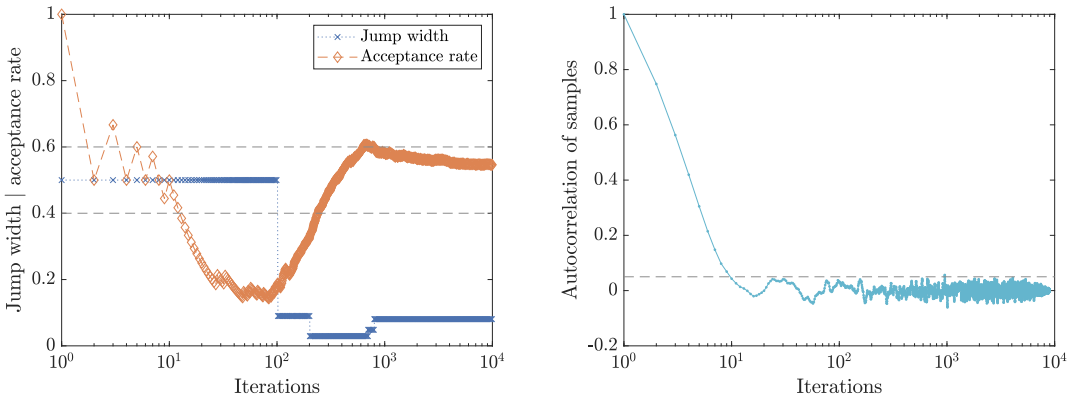
Since the proof-of-concept inference using an artificial data point with averaging 10 repetitions was successful in recovering the true parameter value, I now utilize the same setup for the experimental dataset. I increase the number of repetitions to 25 because we still observe variation in the data misfit when averaging 10 repetitions. However, there is no significant reduction of the variation in the data misfit function with an increased number of repetitions as we can see from Figure 6.12.

The posterior has a mean of 1.1436 m/s, mode of 1.1616 m/s, and standard deviation of 0.0714 m/s. In terms of the performance of the Markov chain, I obtain an overall acceptance rate of 0.4471 and an effective sample size of 1399.60. Both are similar to the



(a) Data misfit function **(b)** Histogram of posterior samples

Figure 6.10: Posterior samples obtained with the Metropolis algorithm for calibration against artificial data while averaging repeated model evaluations at each candidate.



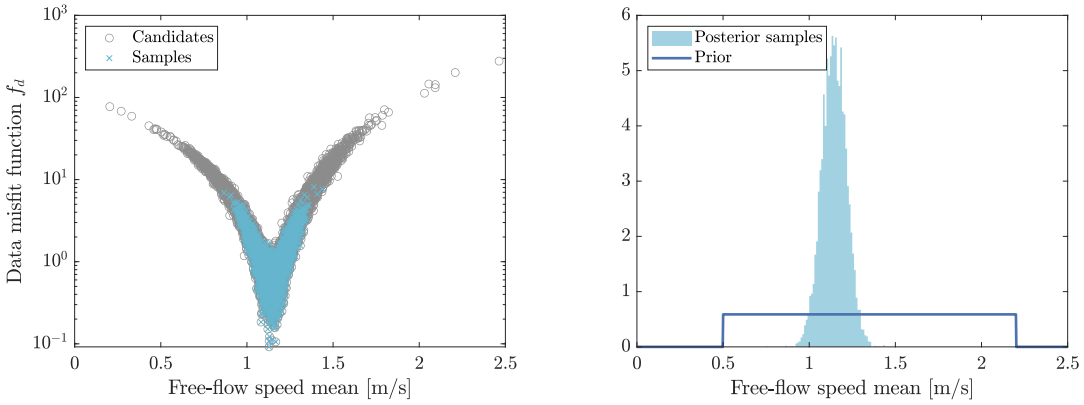
(a) Jump width and acceptance rate **(b)** Autocorrelation of samples

Figure 6.11: Performance criteria for Markov chain Monte Carlo sampling for calibration against artificial data while averaging repeated model evaluations at each candidate.

case study with the surrogate. Figure 6.13 shows that the adaptive jump with regulation maintains the acceptance rate between 0.4 and 0.6 for the most part once the chain has settled after about 200 iterations. Autocorrelation is high among posterior samples with a small lag, below 8, which means only about 15% of the samples can be considered uncorrelated posterior samples.

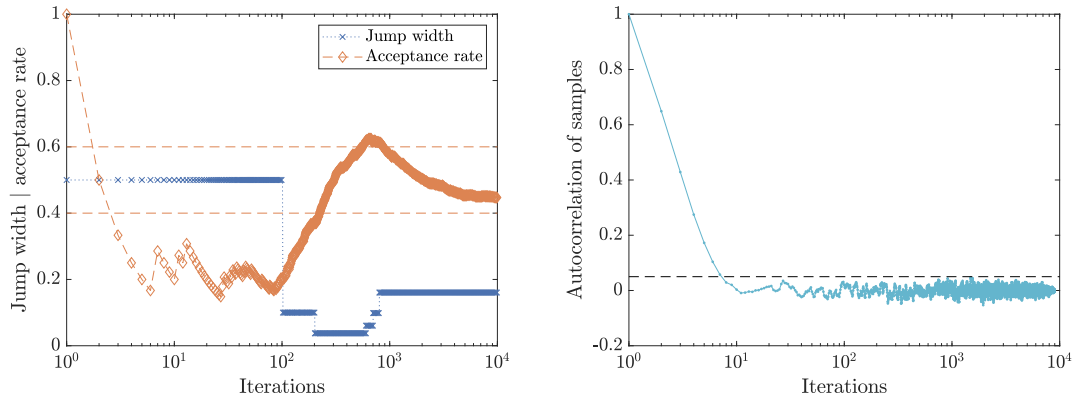
6.4.1.5 Evaluation

In this section, I calibrated the free-flow speed mean in the bottleneck scenario from Section 3.2.1 against flow data. For the calibration, the random walk Metropolis algorithm,



(a) Data misfit function **(b)** Histogram of posterior samples

Figure 6.12: Posterior samples for free-flow speed obtained with Metropolis algorithm for calibrating against experimental data. In every iteration of the Metropolis algorithm, 25 repetitions are averaged for the model response.



(a) Jump width and acceptance rate **(b)** Autocorrelation of samples

Figure 6.13: Performance criteria of Markov chain Monte Carlo sampling for calibrating against experimental data while averaging repeated model evaluations at each candidate: a) adaptive jump width regulation needs about 200 iterations to keep the acceptance rate between 0.4 and 0.6, b) autocorrelation is considerable for posterior samples with a lag of less than 8.

a likelihood-based Markov chain Monte Carlo approach, was employed. I looked at two configurations: First, using a surrogate model for the data misfit function, and second, averaging several repetitions at each parameter value. For each one, I first performed a proof-of-concept with artificial data to check whether the inference finds the true parameter value. Then, I calibrated the free-flow speed mean parameter against experimental flow data from [Seyfried et al., 2009]. Both the proof-of-concept with the surrogate model and averaging repetitions obtained a posterior distribution centered around the

true parameter value. The results of calibration against the experimental data set were similar for the surrogate model and for averaging repeated model evaluations.

6.4.2 Likelihood-free inference with approximate Bayesian computation

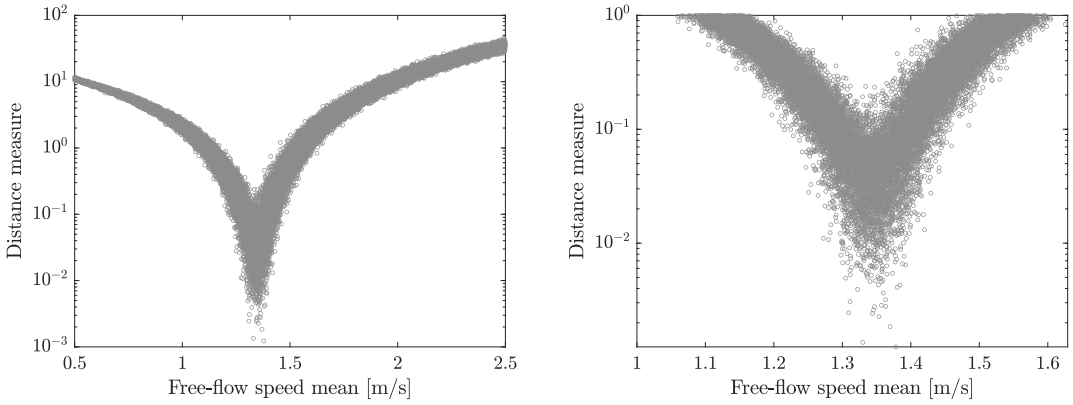
When averaging repeated model evaluations at the same parameter value in the last section, we observed a considerable noise in between different parameter values. Likelihood-based inference is designed for deterministic models. Therefore, we need to remove or at least reduce the variation. One drawback of this approach is that the stochasticity in the model is not considered for inference. The resulting posterior may contain significantly fewer parameter values that yield responses close to the observed data. In addition, the likelihood may not always be known or computable. Therefore, I now use an inference method that can deal with stochastic models, approximate Bayesian computation, and does not require a likelihood. Again, the bottleneck scenario described in Section 3.2.1 is investigated. I calibrate the free-flow speed mean against an artificial flow value obtained from simulation and against flow data measured in the experiments by [Seyfried et al., 2009].

6.4.2.1 Proof-of-concept: artificial data

Similar to the data misfit function for the Metropolis algorithm, ABC evaluates a distance measure. This measure is defined by the user. I define a Euclidean distance measure, $f_d(x_c) = \| d - f(x_c) \|_2^2$. Figure 6.14 shows the distance measure evaluated at all candidates. We observe that the measure holds a substantial variation because I take no measures to limit the randomness in the model responses. The minimum of the data misfit occurs in the interval $[1.3, 1.4]\,\mathrm{m/s}$ in which the true parameter value of $1.34\,\mathrm{m/s}$ lies.

Even though I use artificial data, the magnitude of the distance measure is rather high. In this example, I try to calibrate five scenarios at once. We cannot expect to obtain the same magnitude as for a single bottleneck. Instead, the difference between observations and model response is larger. For illustration, I show the distance measure for the calibration of a single bottleneck, the widest bottleneck of 1.2 m, in Figure 6.15. The distance measure reaches values up to 10^{-10} while for the five bottlenecks the minimal distance measure is at about 10^{-3}. While individually, we observe small values of about 10^{-7} to 10^{-8}, for individual bottlenecks, we also see a large variation in the distance measure especially in the regions of the best fit. That makes it unlikely that we obtain a good fit between observations and model response for all bottlenecks with only one realization, that is one seed for the random number generator. As a result, the smallest values for the distance measure that compares all five bottlenecks at once, are substantially larger.

In Figure 6.16, I evaluate the posterior for different tolerances ϵ. The tolerance needs to be defined by the user. Beaumont et al. choose the tolerance so that 1% of the candidates are accepted [Beaumont et al., 2002]. For the artificial data, I choose the tolerance accordingly, which yields a tolerance of $\epsilon = 3 \cdot 10^{-2}$ (acceptance rate 1.30%).



(a) Full x-axis **(b)** Zoom on x-axis

Figure 6.14: Distance measure $f_d(x_c)$ for approximate Bayesian computation evaluated at all candidates generated from the prior distribution for the free-flow speed. $f_d(x_c)$ quantifies the disagreement between model evaluation at candidate x_c for the free-flow speed mean and an artificial flow measurement for $x^\star = 1.34\,\mathrm{m/s}$ in the bottleneck scenario.

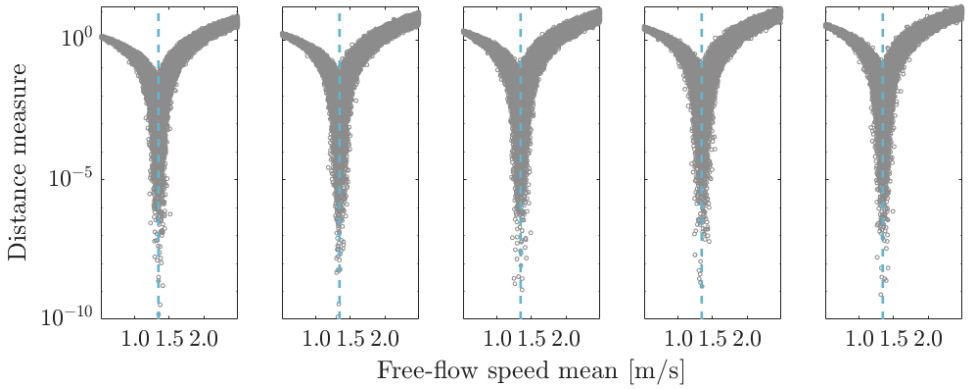
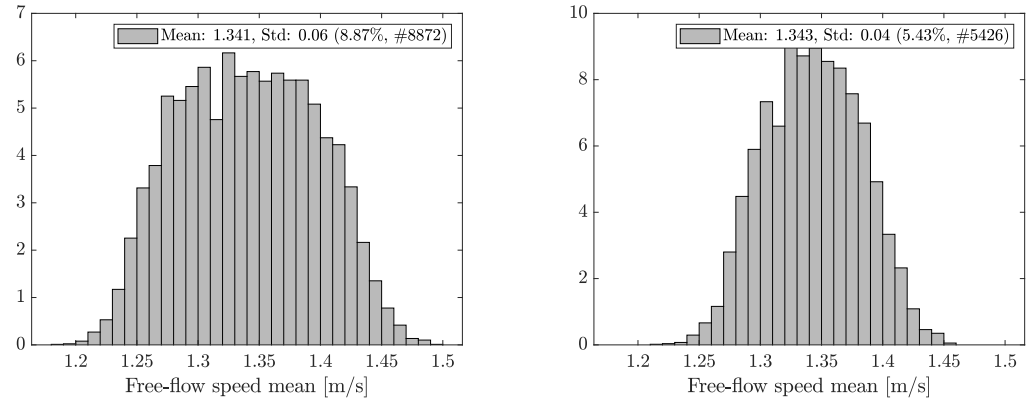


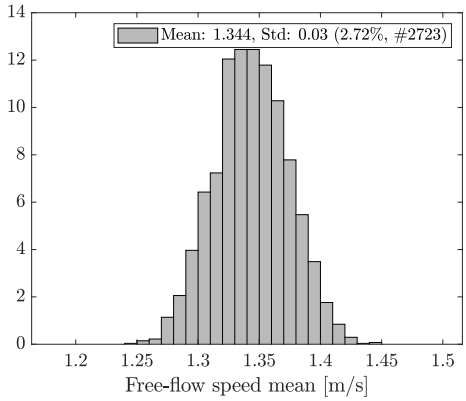
Figure 6.15: Distance measures for approximate Bayesian computation when calibrating five individual bottlenecks (bottleneck width increases from 0.8 m (left) to 1.2 m (right)) present considerably lower values for the true parameter than the distance measure for the scenario with five bottlenecks in Figure 6.14.

For all evaluated tolerances, the posterior is centered at the true parameter value of 1.34 m/s indicating that the Bayesian inference is successful. We also observe that the width of the posterior depends on the choice of the tolerance. Remember that the rejection sampler is asymptotically exact for $\epsilon \to 0$ and $N \to \infty$. In practice, however, the computational effort limits the number of candidates N and numerical constraints limit the choice of the tolerance.

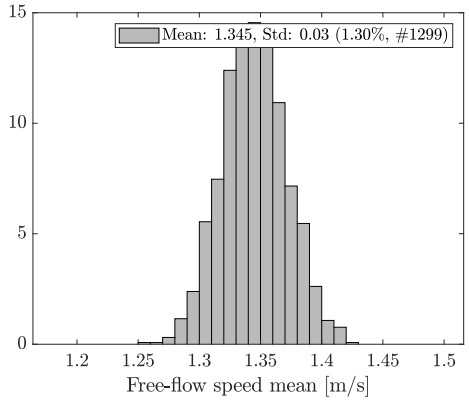


(a) $\epsilon = 2 \cdot 10^{-1}$

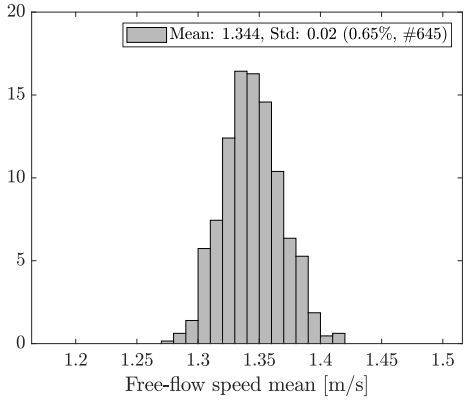
(b) $\epsilon = 10^{-1}$



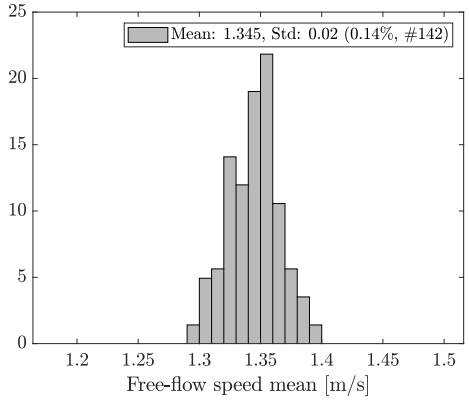
(c) $\epsilon = 5 \cdot 10^{-2}$



(d) $\epsilon = 3 \cdot 10^{-2}$



(e) $\epsilon = 2 \cdot 10^{-2}$

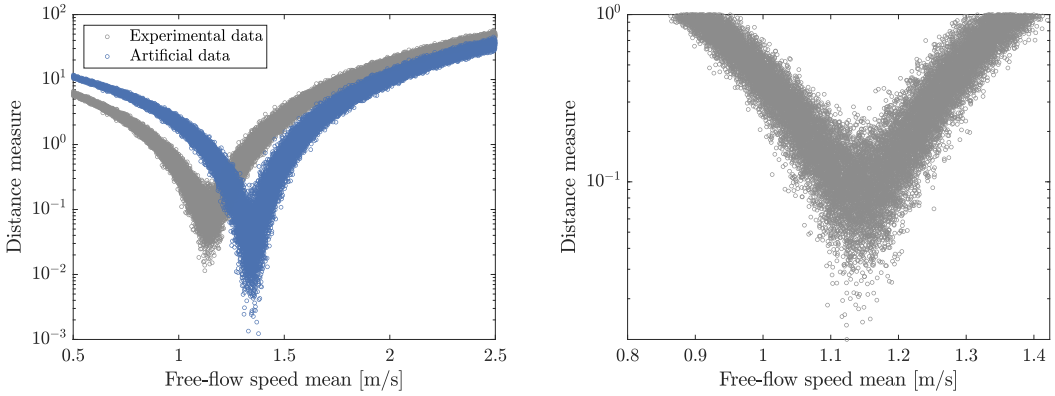


(f) $\epsilon = 10^{-2}$

Figure 6.16: Histogram of posterior samples for the free-flow speed mean obtained with different tolerances ϵ for approximate Bayesian computation. Posterior mean is not affected by the tolerance, but the posterior width depends strongly on the choice of ϵ.

6.4.2.2 Bayesian inference in the bottleneck scenario: experimental data

After the proof-of-concept with artificial data successfully recovered the true parameter value, I infer the free-flow speed based on the experimental data set. The distance to the real data in Figure 6.17 has qualitatively the same shape as for artificial data. There is one region for the free-flow speed mean parameter in which the simulation results approach the data. Quantitatively, the distance measure reaches lower values for the artificial data point. As one would expect, data generated from the same model fits better than external data. As a consequence, the variation in the distance measure is lower for the experimental data at the region of the best fit. The parameter values that lead to the minimal distance measure are lower for the experimental data than for the artificial data. That means pedestrians in the experiments had slower free-flow speeds than 1.34 m/s.

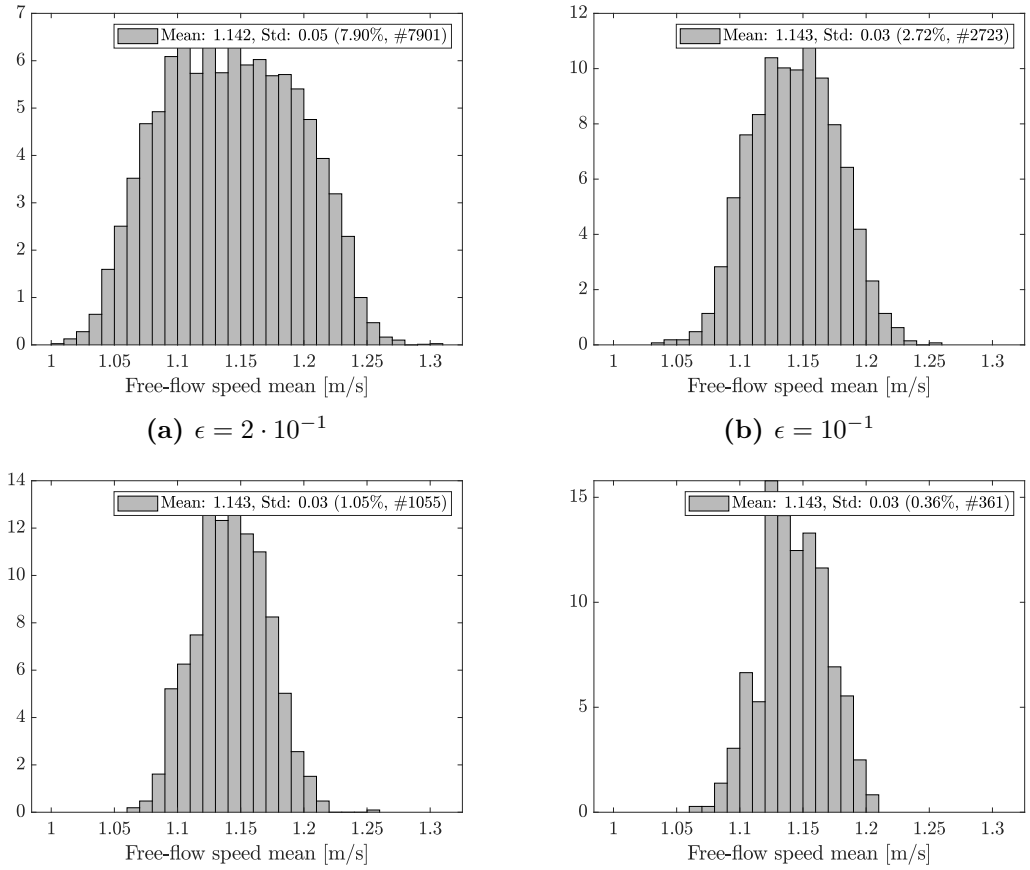


(a) Comparison of distance measure with artificial and experiment data

(b) Zoom for data misfit with experimental data

Figure 6.17: Distance measure obtained by approximate Bayesian computation when calibrating the free-flow speed mean in the five bottleneck scenario against flow measurements.

I choose a tolerance of $7 \cdot 10^{-2}$ that corresponds to an acceptance rate of about 1%. This results in the posterior distribution for the free-flow speed presented in Figure 6.18c. Histograms for different tolerances in Figure 6.18 show that the mean is not affected by the choice of the tolerance, but the width of the posterior strongly depends on it. Consequently, the choice of tolerance is crucial. The mean of the posterior samples is then at 1.143 m/s and the standard deviation at 0.03 m/s. These results with ABC are similar to the MCMC results: For MCMC, I obtain a mean of 1.122 m/s and standard deviation of 0.064 m/s using the surrogate and 1.143 m/s and 0.063 m/s for mean and standard deviation when averaging repetitions. The standard deviation of the ABC posterior is lower than that obtained with MCMC.



(a) $\epsilon = 2 \cdot 10^{-1}$ **(b)** $\epsilon = 10^{-1}$

(c) $\epsilon = 7 \cdot 10^{-2}$ **(d)** $\epsilon = 5 \cdot 10^{-2}$

Figure 6.18: Histogram of posterior samples for the free-flow speed mean obtained with different tolerances ϵ for approximate Bayesian computation. The mean of the posterior distributions is not affected by the choice of tolerance. However, the width of the posterior is.

6.4.2.3 Discussion of tolerance

ABC theory says that rejection sampling is asymptotically exact, meaning that for an infinitely small ϵ, $\epsilon \to 0$, and large N, $N \to \infty$, the result is the true posterior distribution from Bayes' theorem. Nevertheless, in the presence of measurement error or model error, $\epsilon \to 0$ may not provide the true posterior but the parameter value that yields a minimum of the distance measure. Wilkinson et al. show that ABC is exact in case of model error and propose a method how to deal with measurement and model error e. However, the algorithm requires the density of the error π_e [Wilkinson, 2013]. Van der Vaart et al. present a way to define the density π_e if there are multiple measurements of the same quantity of interest present for a stochastic model [van der Vaart et al., 2018]. Unfortunately, in the bottleneck setup in which I calibrate the parameters, multiple

measurements are not available. In general, they are difficult to obtain in pedestrian dynamics since repetition of experiments with the same participants may lead to different results. This is often referred to as learning effect [Kleinmeier and Köster, 2020, Kretz et al., 2006, Kemloh Wagoum et al., 2017]. Alhamadi et al. point out difficulties when choosing a tolerance in the presence of model or measurement error focusing on deterministic problems [Alahmadi et al., 2020].

I am not aware of any literature on how to choose the tolerance ϵ when performing ABC on a stochastic model in the presence of measurement or model error. In this case, it is not sufficient to choose a single small ϵ or a set of small tolerances as in sequential Monte Carlo approaches for ABC. In the following, I set ϵ so that I keep 1% of the candidates as in [Beaumont et al., 2002].

6.4.2.4 Evaluation

In this section, I calibrated the free-flow speed mean using approximate Bayesian computation, a likelihood-free Bayesian inference that can deal with stochastic simulators. In a proof-of-concept setting with an artificial data point, the method successfully recovered the true parameter value. When calibrating the model against the experimental data obtained by [Seyfried et al., 2009], the results of ABC are in line with the MCMC results obtained in Section 6.4.1. The advantages of ABC over MCMC are that it does not require a likelihood or assumption on the likelihood and that it can be applied directly on stochastic simulators. However, the choice of the tolerance, a central parameter of ABC, is complex in practical applications, especially when measurement noise is present. The decision for one method must be made depending on the specific situation: If a likelihood can be assumed and either a surrogate model or sufficient resources to average multiple repetitions is available, I recommend MCMC. Otherwise, ABC is the better choice.

6.4.3 Comparison of Bayesian inference to a point estimate

I compare Bayesian inference, represented by the rejection sampler of approximate Bayesian computation, to a point estimate in three case studies. Point estimation is to date a common technique for parameter calibration while Bayesian inference is not established yet. The three case studies serve to demonstrate the benefits and limitations of both approaches for calibration. This section follows [Gödel et al., 2022]. First, I calibrate the free-flow speed in the optimal steps model. Second, I calibrate the desired speed in a social force model which exhibits a faster-is-slower dynamic. Finally, I compare the point estimate to the full posterior for a multidimensional calibration of all influential parameters in the bottleneck scenario.

6.4.3.1 Unimodal posterior

In the first use case, I simulate the bottleneck scenario with the optimal steps model. In Figure 6.19, we observe a monotonically strictly increasing trend between free-flow speed mean and flow. As expected, the flow increases with the speed and also with the

width of the bottleneck. Since I use only one repetition at each parameter value, there are stochasticity and noise present; both increase with the speed.

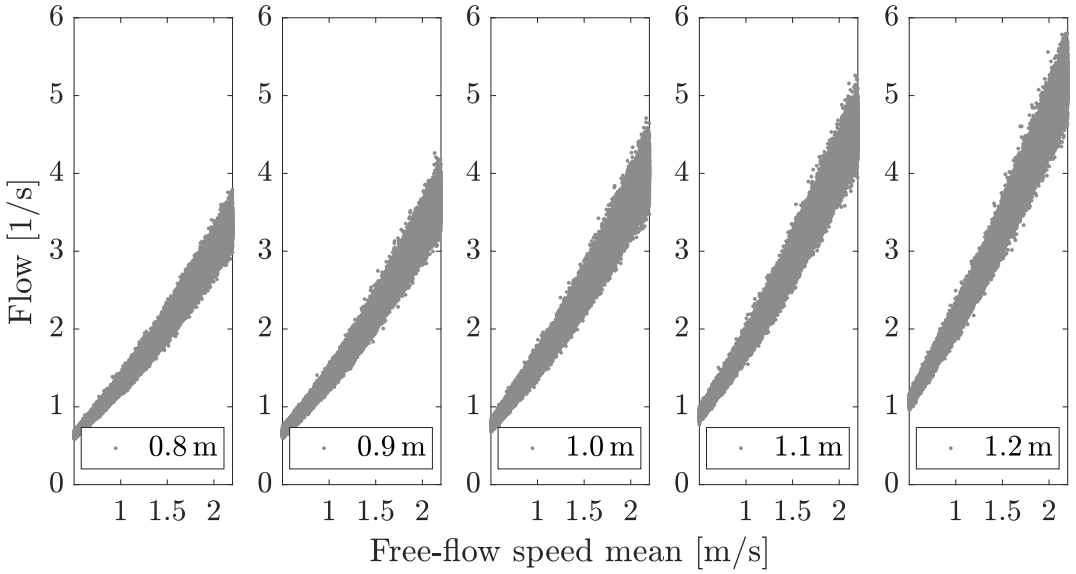
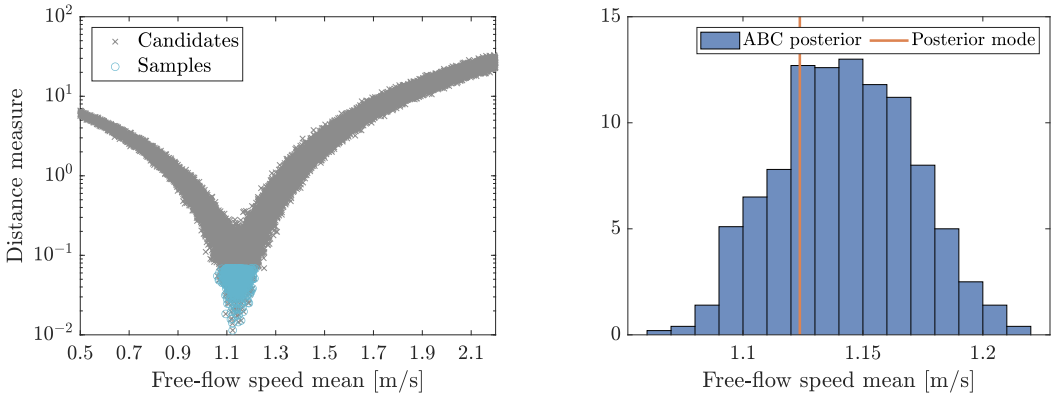


Figure 6.19: Relationship between free-flow speed mean and flow for the five bottlenecks of different widths. Bottleneck width increases from left to right (0.8 m to 1.2 m).

Since the trend of the input-output relationship is strictly monotonically increasing for each bottleneck width, we expect that the distance measure for calibrating a single bottleneck has a single minimum. It is difficult to draw conclusions when comparing all five bottlenecks. In Figure 6.20a, the distance between the flow in all bottlenecks and data is shown. We observe the same shape as expected for comparing individual bottlenecks to data. I choose a tolerance of $\epsilon = 0.0685$ in order to keep 1% of the candidates as samples of the posterior according to Beaumont [Beaumont et al., 2002]. The number of samples is denoted N_ϵ. The histogram of the posterior samples in Figure 6.20b shows a symmetric, unimodal posterior. Mean and mode of posterior are both at about 1.14 m/s and the standard deviation is 0.027. In addition, I also evaluate the posterior mode. It is slightly lower than the mean and mode of the posterior histogram.

In the next step, I separately propagate the posterior and its mode through the model to study the variation in the model response. From the posterior distribution, all N_ϵ samples are propagated. For comparability, the point estimate is propagated also N_ϵ times through the system. Due to the stochasticity in the initialization, the propagations lead to different results, even though the parameter value is fixed. The histogram of the flow values obtained from propagation in Figure 6.21 shows a similar width and shape for both the full posterior and the point estimate. This suggests that the variation due to random initialization is equally large as the variation due to the posterior distribution. The randomness in the model has a significant impact.



(a) Distance measure **(b)** Histogram of posterior samples

Figure 6.20: Results of Bayesian inference for calibration of the free-flow speed in the bottleneck scenario with approximate Bayesian computation (ABC), tolerance $\epsilon = 0.068$ (acceptance rate = 1%).

We also observe that the distribution of flow values deviates from the observed data set for both point estimate and full posterior. One reason could be the simultaneous adjustment to five different bottleneck experiments. As shown in Section 6.4.2.1, individual comparison yields a better fit. In addition, model and experiment might differ due to underlying effects in the experiments such as learning effects that are not modeled in the simulation. Another factor could be the measurement error on the flow measurements, which is not quantified in the experimental results [Seyfried et al., 2009].

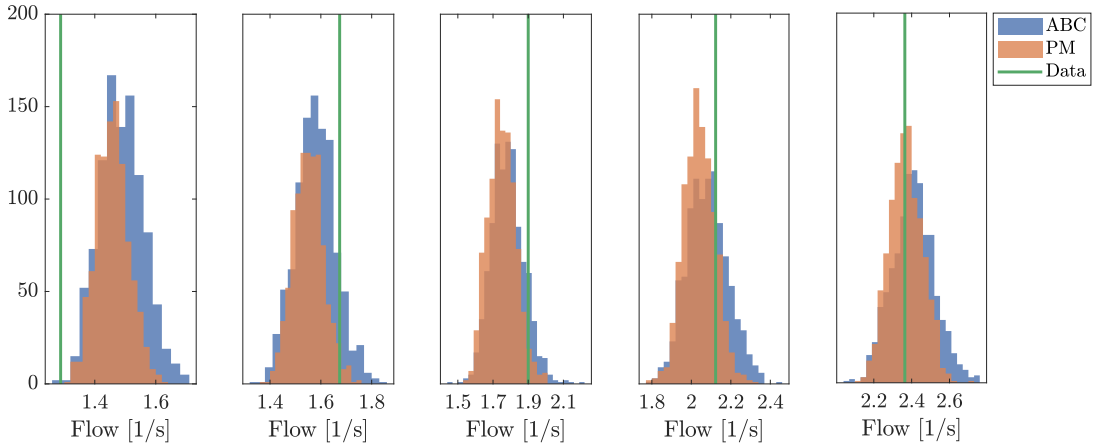


Figure 6.21: Comparison of flow values obtained from propagation of posterior mode (PM) obtained from N_ϵ repetitions and full posterior obtained with approximate Bayesian computation (ABC) with tolerance $\epsilon = 0.0685$. Bottleneck width increases from left (0.8 m) to right (1.2 m).

My goal now is to quantitatively evaluate the variation in the flow for both point estimate and full posterior. Therefore, I run $M = 200$ propagations and perform a linear regression over five flow values for the different bottleneck widths. As a metric for the variation of the flow, I choose the size of the confidence interval of the slope of the linear fit. Figure 6.22 illustrates the regression for the propagation of the full ABC posterior. The box plots indicate the variation at each bottleneck width. Across all regressions, I find the average confidence interval and its size.

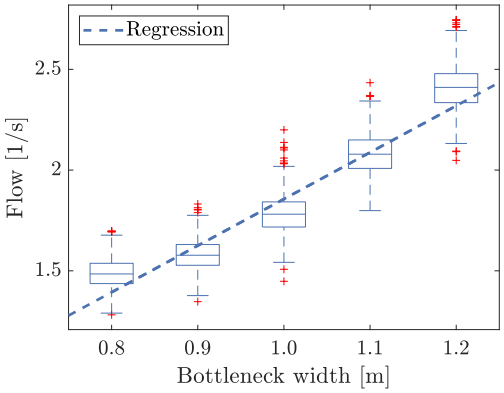


Figure 6.22: Regression for propagated posterior obtained with approximate Bayesian computation ($\epsilon = 0.0685$). A box plot of the flow samples resulting from propagation is shown together with the linear regression.

As discussed in Section 6.4.2.3, the optimal tolerance for stochastic models when measurement error or model error is present, is not known. In theory, we obtain the true posterior in the limit when approaching a tolerance of zero, $\epsilon \to 0$, while the sample size approaches infinity, $N \to \infty$. In practice, however, due to limited numerical accuracy and limited computational power, both approximations are limited. I evaluate the size of the confidence interval over possible tolerances in Figure 6.23. The figure shows that the results strongly depend on the tolerance. Nevertheless, when considering tolerances $\epsilon \geq 0.0685$ there is a difference between posterior mode and full posterior that grows with increasing tolerance.

For a fixed tolerance of 0.0685, Table 6.1 summarizes the confidence interval for the observed data, as well as the flow values obtained from propagating the point estimate and the ABC posterior. It is important to note that the confidence intervals for the propagation are larger than that obtained from the data. This should always be the case. Otherwise, the uncertainty after calibration and propagation would be smaller or equal to the uncertainty in the data which would indicate stronger reliability in the results of the prediction. Since the reliability of the prediction is limited by the reliability of the data used for calibration, this obscures that the data contains uncertainty.

In summary, for the monotonically strictly increasing trend between uncertain parameter, free-flow speed, and the quantity of interest, the flow, I obtain a symmetric unimodal posterior. In this case, there is no large difference between using a full poste-

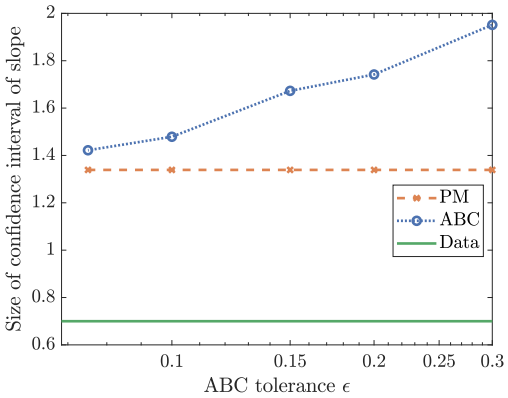


Figure 6.23: Size of confidence interval of slope (linear fit) for different tolerances ϵ for approximate Bayesian computation (ABC) posterior compared to posterior mode (PM) and experimental data.

Table 6.1: Confidence intervals for slope of linear regression for observations as well as propagated posterior mode and full posterior obtained by approximate Bayesian (ABC) computation.

	Data	Posterior mode	ABC posterior
Confidence interval	$[2.251, 2.951]$	$[1.651, 2.990]$	$[1.595, 3.080]$
Size of confidence interval	0.699	1.338	1.485

rior obtained with Bayesian inference methods to using a point estimate for propagation. However, this is only the case for a simulator with a stochastic component. For practical applications this is good news: If the relationship between uncertain parameter and quantity of interest is monotonically strictly increasing, a point estimate is sufficient. The advantage is that the point estimate is cheaper to evaluate. In the next two subsections, I point out two cases in which it is problematic to rely on point estimates only.

6.4.3.2 Bimodal posterior

For the second case study, I examine a setup of five bottlenecks simulated with a social force model emulator. As described in Section 3.1.2, I build a surrogate model for the egress of 200 agents leaving the room simulated with Helbing's social force model [Helbing et al., 2000]. The emulator is shown together with the experimental dataset used for calibration in Figure 6.24.

The relationship between speed and flow exhibits a faster-is-slower dynamic, which means, it is not just monotonically increasing but displays an extremum. The largest flow is obtained for the desired speed around 1.3 m/s, lower and higher speeds lead to smaller flows. Consequently, the function is not bijective, meaning that every parameter value leads to exactly one function value but only surjective, a function value can be

reached by several parameter values. That means the parameter is not identifiable and can therefore not be unique estimated.

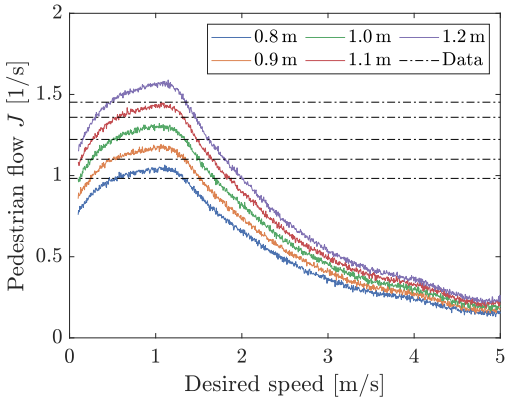
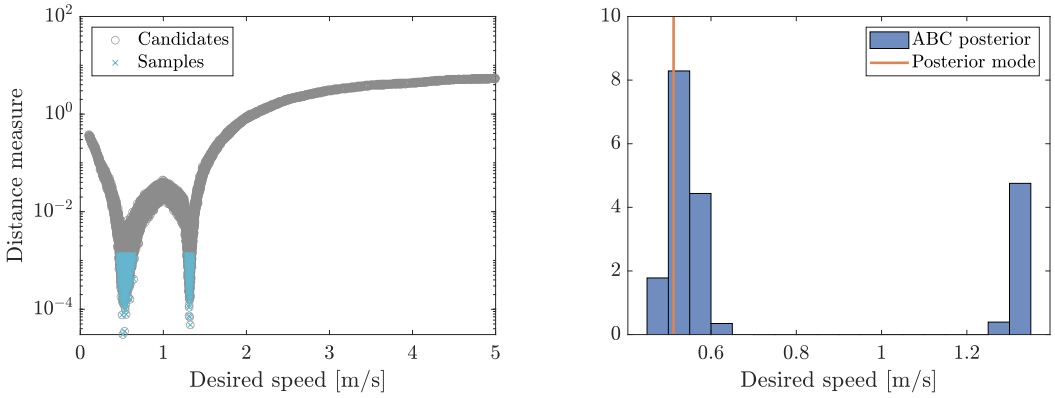


Figure 6.24: Social force model emulator for five different bottleneck widths, constructed with flow data from [Helbing et al., 2000] including an additive zero-mean Gaussian noise with variance $\sigma^2 = 10^{-4}$ with experimental data used for inference.

The distance measure for the social force model emulator in Figure 6.25 displays two minima, as expected due to the faster-is-slower dynamic. I choose a tolerance of $\epsilon = 0.00128$ to keep 1% of the candidates. Thereby, I obtain a bimodal posterior distribution around the two minima at about 0.5 m/s and 1.2 m/s, compare Figure 6.25b. By design, a point estimate can only catch one of the two minima. Either one can be the posterior mode.



(a) Distance measure (sum of squares)

(b) Posterior distribution

Figure 6.25: Results of calibrating the free-flow speed in the social force model emulator with approximate Bayesian computation, tolerance $\epsilon = 0.00128$.

Consequently, when propagating the speeds for which the minima occur, I obtain the same flow. Even though posterior and point estimate differ greatly, the flow values after

propagation are similar. However, if we focus the propagation on another quantity of interest, such as the egress time, the results between full posterior and posterior differ drastically. In the same way that I created the emulator for the speed-flow relationship, I create a second emulator for the egress times based on the so-called leaving times reported in [Helbing et al., 2000], see Figure 6.26. Again, I introduce an additive zero-mean Gaussian noise with covariance 0.01. The emulator is built for an egress scenario in which 200 pedestrians leave a room through a door.

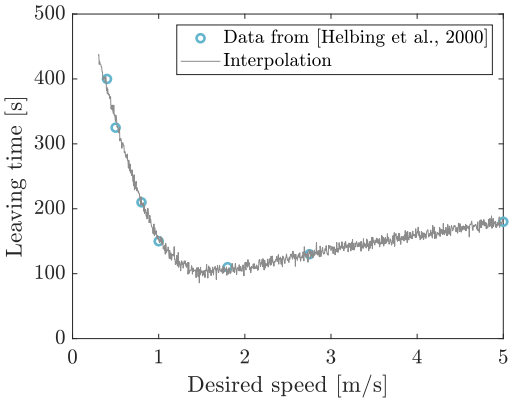


Figure 6.26: Social force model emulator for leaving times or egress times over desired speed, obtained by interpolating data from Helbing et al. [Helbing et al., 2000].

In Figure 6.27, we can see that propagating the posterior mode leads to egress times about 330 s, while the bimodal posterior finds egress times between 110 s and 360 s. Since the point estimate found the lower speed of the two speeds that lead to the flow most similar to the observed flow, the point estimate is just conservative in this case. If however, the point estimate finds the faster speed at about 1.2 m/s it would lead to an egress time of about 110 s. That result would indicate significantly more efficient egress. Based on the data, we cannot decide which egress time is "true". Relying on the faster egress by using the point estimate would compromise pedestrian safety.

Using Bayesian inference methods, one can capture any posterior distribution instead of relying on symmetric unimodal posterior distributions as we implicitly do when employing point estimates. If those posterior shapes are ignored by using only a point estimate for propagation, egress times obtained from calibrated simulation can be too optimistic and therefore may put pedestrians at risk. My example of calibrating the speed by comparing to flow measurements is similar to a calibration towards a speed-flow relationship which is common in crowd dynamics [Steiner et al., 2007, Taherifar et al., 2019, Wolinski et al., 2014, Chu, 2009, Davidich and Köster, 2012]. Especially, when using only point estimates, calibration towards a fundamental diagram is not sufficient for studying other quantities of interest after propagation

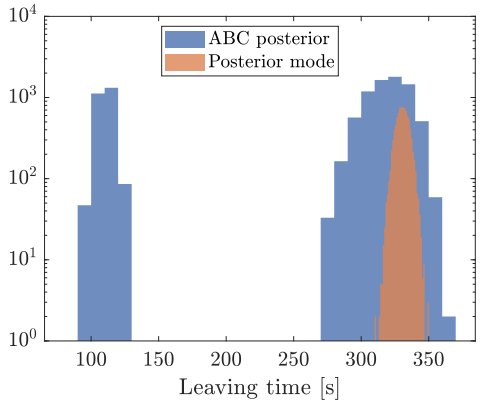


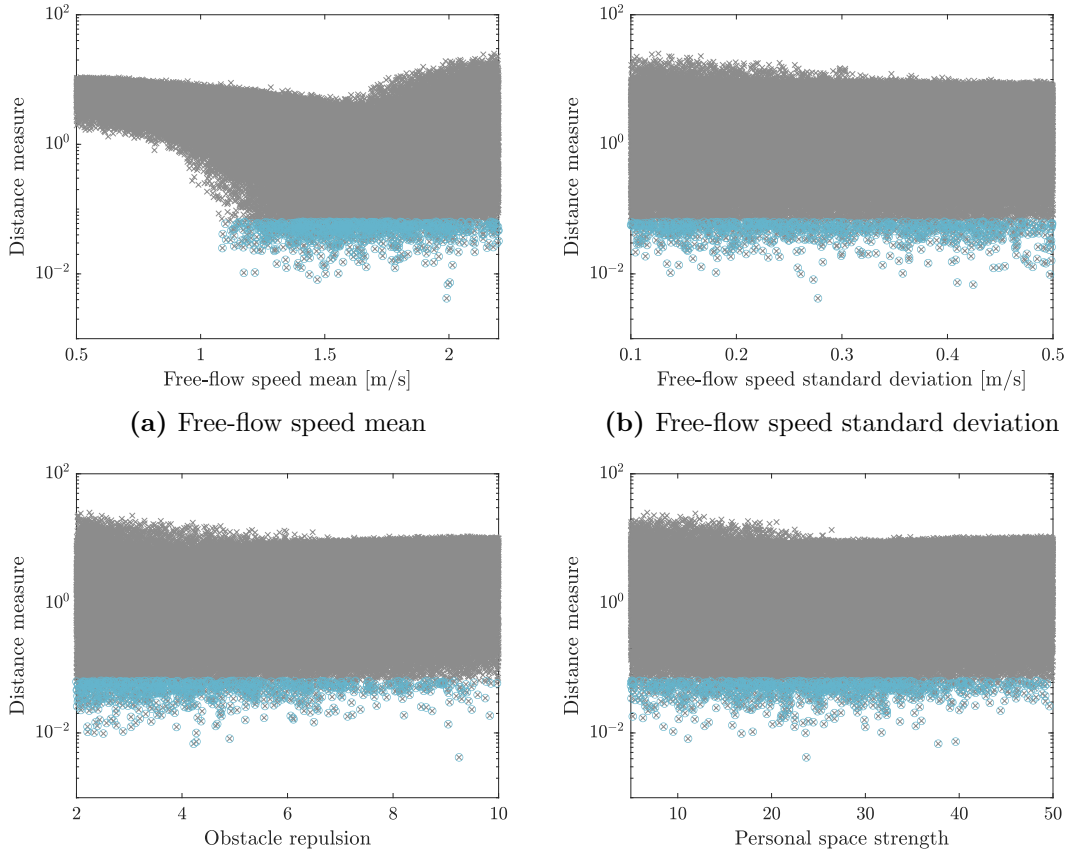
Figure 6.27: Leaving time, or egress time, when propagating the point estimate and the full posterior obtained for calibration of the free-flow speed in the social force model emulator.

6.4.3.3 Multivariate posterior

After studying the one-dimensional cases that focus on the most influential parameter, I now infer all influential parameters in the bottleneck scenario identified in Chapter 5: free-flow speed mean, free-flow speed standard deviation, personal space strength, and obstacle repulsion. As priors for Bayesian inference, I use the flat priors from the sensitivity analysis. I only reduce the interval for the free-flow speed standard deviation from $[0, 1]$ to $[0, 0.5]$ since we know that the participants in the experiment were all students. That means they have a similar age and fitness from which I conclude that their free-flow speeds are rather uniform. For larger free-flow speed standard deviations, the correlation with the free-flow speed mean becomes larger due to the truncated normal distribution. Consequently, a smaller interval also reduces the correlation.

For this calibration, the distance measure, $f_d : \mathbb{R}^4 \to \mathbb{R}$, maps the four-dimensional parameter vector to a scalar. Consequently, I cannot plot it in all dimensions. Instead, I plot the distance measure over each parameter separately. For Figure 6.28, the model evaluations obtained during the calibration are utilized. Consequently, all parameters are varied at once leading to a large variation over each parameter. We observe a strong trend when plotting the distance measure with respect to the free-flow speed. For the other parameters, the relationship is dominated by noise.

I set the tolerance to $\epsilon = 0.06378155$ in order to keep 1% of the candidates as samples. The result of Bayesian inference is a four-dimensional joint posterior distribution. I evaluate the univariate posterior distribution in Figure 6.29. That is the marginal posterior distribution for a single parameter. In addition, the posterior mode for the parameter is shown. The univariate posteriors for free-flow speed mean (Fig. 6.29a), obstacle repulsion (Fig. 6.29c), and personal space strength (Fig. 6.29d) vary significantly from their flat prior indicating that these parameters are well informed by the data. The posterior of the free-flow speed standard deviation in Figure 6.29b, however, is similar



(a) Free-flow speed mean

(b) Free-flow speed standard deviation

(c) Obstacle repulsion

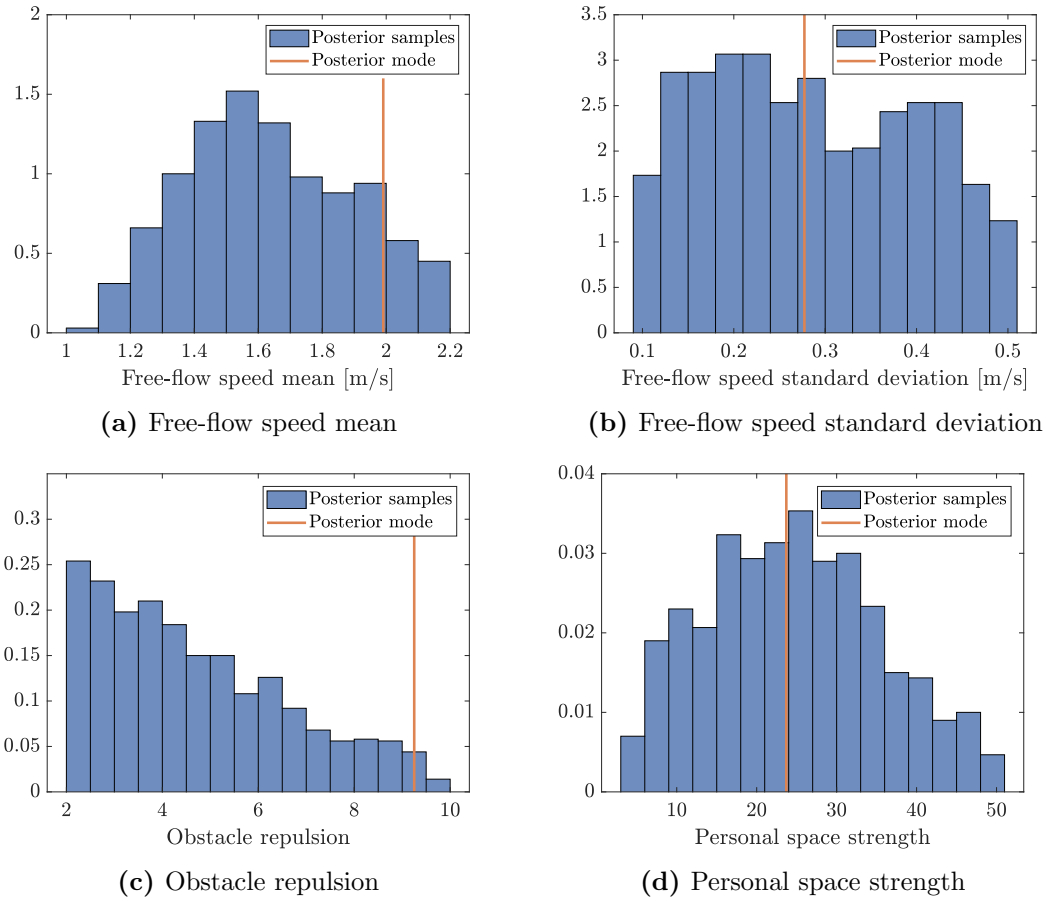
(d) Personal space strength

Figure 6.28: Distance measure evaluated at candidates ($\epsilon = 0.06378155$) generated by the rejection sampler for approximate Bayesian computation when calibrating all influential parameters in the bottleneck scenario.

to the flat prior. That means this parameter is not informed to the same degree as the other parameters.

Note that the univariate posterior distribution for the obstacle repulsion has its mode at the edge of the interval. This indicates that the most suitable parameter value might lay outside the prior. Ideally, I would increase the interval since the results suggest that there might be an even better fit with a lower obstacle repulsion. However, I know that a certain repulsion from the obstacle is necessary to represent the motion realistically. This contradiction may indicate discrepancies between model and experiment. In practice, we need not only a good agreement between model evaluation and data, but also a meaningful parameter value. For this reason, I stick to the specified prior.

In addition to the univariate posteriors, we take a look at the bivariate posterior distributions in order to evaluate the correlations among the parameters. For the four parameters, there are six couples. Both univariate and bivariate posteriors are summa-



(a) Free-flow speed mean

(b) Free-flow speed standard deviation

(c) Obstacle repulsion

(d) Personal space strength

Figure 6.29: Univariate posterior distributions for influential parameters in the bottleneck scenario. Calibration is carried out through approximate Bayesian computation with a tolerance of 0.06378155 (acceptance rate 1%).

rized in the triangle plot in Figure 6.30. The triangle plot shows the univariate posterior densities on the diagonal and the bivariate distributions on the lower diagonal. The upper diagonal holds Pearson's correlation coefficient for the bivariate combination. We observe a correlation between the free-flow speed mean and each of the other parameters. All combinations with a correlation coefficient above 0.4 are shown in Figure 6.31. The correlation between the parameters emphasizes that the data is best represented when the parameter value for the free-flow speed mean is not chosen independently from the other parameters. In other words, the parameters' values should be sampled from the joint posterior, not from the univariate posterior distributions. Interpretation of this correlation is complex since it is not with respect to the quantity of interest, the flow, but to the distance measure. The distance measure combines the comparison between observation and simulation for five bottlenecks at once. The distance measure function

f_d also depends on the data chosen for the calibration, which means, the correlation does not necessarily hold on general, but only for this data set.

If we recall the sensitivity analysis in Chapter 5, we do not recall any strong correlations among parameters. That might at first seem inconsistent. It is, however, important to note that the correlations in the sensitivity analysis are with respect to the quantity of interest, the flow, for a single bottleneck. A correlation between two parameters observed in the calibration means that their inferred posterior distributions are correlated toward the distance measure including observational data from [Seyfried et al., 2009].

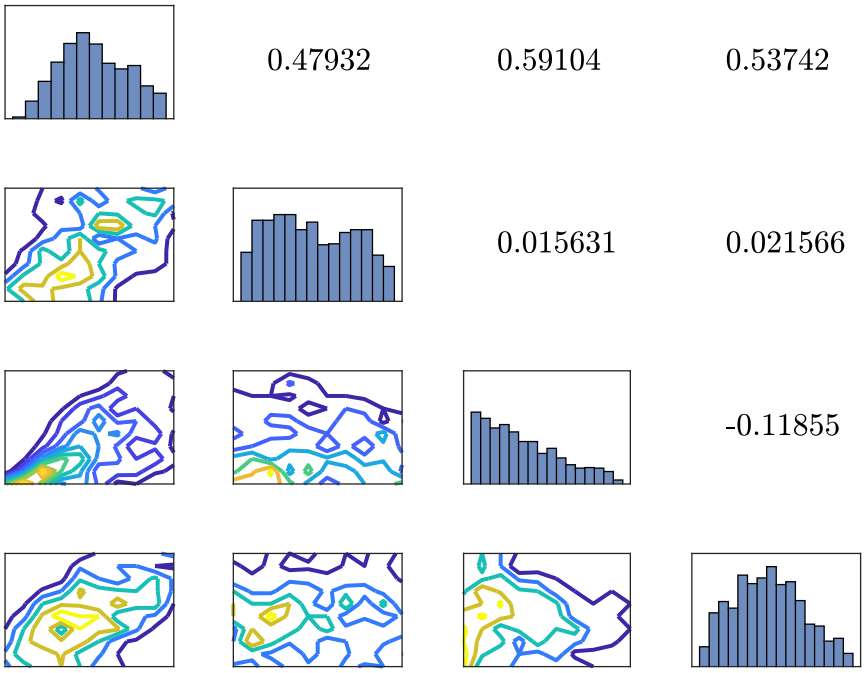
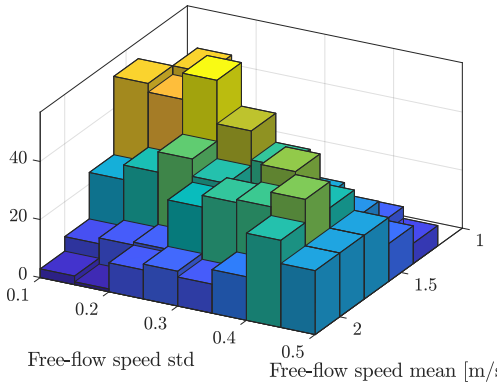
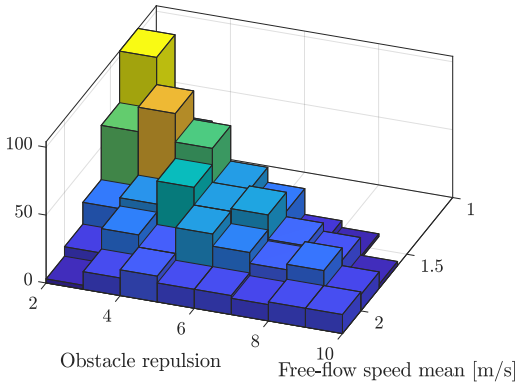


Figure 6.30: Triangle plot for posterior density for the influential parameters in the bottleneck scenario: On the diagonal, the univariate posterior density is shown for each parameter. The lower diagonal holds the bivariate posterior densities. The upper diagonal shows the Pearson correlation coefficient for the bivariate posterior densities.

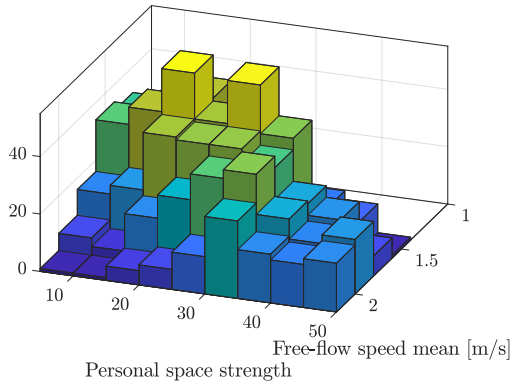
The four-dimensional calibration highlights that Bayesian inference provides additional information compared to point estimates. When carrying out propagations based on the posterior, it is important to choose the values for free-flow speed and obstacle repulsion not independently, but based on their joint posterior distribution. In addition, we also learned that free-flow speed mean, obstacle repulsion, and personal space strength are well informed by the data, while the free-flow speed standard deviation is



(a) Bivariate posterior for free-flow speed mean and free-flow speed standard deviation (Pearson correlation coefficient: 0.47932)



(b) Bivariate posterior for free-flow speed and obstacle repulsion (Pearson correlation coefficient: 0.59104)



(c) Bivariate posterior for free-flow speed mean and personal space strength (Pearson correlation coefficient: 0.53742)

Figure 6.31: Bivariate posterior distributions obtained with approximate Bayesian computation ($\epsilon = 0.06378155$) for all parameter combinations with correlation larger than 0.4.

not. The posterior mode, on the other hand, cannot provide insights on the correlations among the inferred parameters as well as the quality of the calibration.

This example is intended as a demonstration of principle. In this example, I use five measures of the flow to calibrate four parameters. I recommend using more data in practical applications for a reliable calibration.

6.4.3.4 Evaluation

In this section, I compared calibration with a point estimate to a Bayesian inference method, that is approximate Bayesian computation, in three case studies. First, I calibrated the free-flow speed mean in the optimal steps model. When propagating both the

point estimate and the full posterior obtained through calibration, the variation in the quantity of interest depends mainly on the user-defined tolerance ϵ of ABC. Qualitatively, there was no significant difference between point estimation and Bayesian inference and therefore a point estimate appeared sufficient.

Second, I calibrated the desired speed in a social force emulator that exhibits a faster-is-slower dynamic for an egress scenario. Here, Bayesian inference revealed a bimodal posterior distribution that the point estimate could not recover. Consequently, a part of the posterior distribution is lost when relying on point estimation. I evaluated the egress times for the ABC posterior and the point estimate to emphasize the importance of considering the full posterior distribution for subsequent studies. For the point estimate, the egress times only varied around a single value instead of displaying the correct bimodal shape. This means that the egress times are miscalculated endangering the pedestrians' safety. This exposes a need for caution when transferring calibrated parameters from one safety scenario to the next.

Finally, I performed a multidimensional calibration of the four influential parameters of the optimal steps model in the bottleneck scenario: the free-flow speed mean and standard deviation the personal space strength and the obstacle repulsion. The uni- and bivariate posterior distributions obtained with ABC showed the free-flow speed mean, the personal space strength and the obstacle repulsion were well informed by the data. However, the free-flow speed standard deviation was not informed to the same degree. In addition, the multivariate posterior revealed a strong correlation between free-flow speed mean and each other parameter.

6.4.4 Higher-dimensional Bayesian inference

Inferring a large number of parameters is computationally expensive, especially when iterative methods such as the Metropolis algorithm are used. In addition, Markov chains exhibit a bad mixing behavior in higher dimensions. One way to overcome both issues is to reduce the size of the input parameter space. The idea of active subspaces [Constantine et al., 2016], as presented in Section 5.3.2, is to identify important parameter directions in the input parameter space and to build a lower-dimensional representation. The directions are identified based on the gradients of the forward model. In the case of computer models, the gradients can be approximated by e. g. finite differences. The directions are then used to generate a surrogate for the forward model, the so-called ridge approximation. Once the ridge approximation is generated, the Bayesian inference method e.g. the Markov chain is run only in the active subspace instead of the complete input parameter space [Constantine et al., 2016]. Figure 6.32 shows how a Markov chain Monte Carlo method can be combined with the active subspace.

In the uncertainty quantification framework, routines are implemented to identify an active subspace. They can be employed for this approach. The only missing link is the construction of the ridge approximation. A simple yet effective choice for the approximation is polynomial regression. Polynomial regression is readily available in Python's scikit-learn package [Pedregosa et al., 2011]. For the regression, one can reuse the model evaluations generated to calculate the gradients. A more detailed description

for the construction of a response surface as an approximation can be found e. g. in [Teixeira Parente et al., 2019].

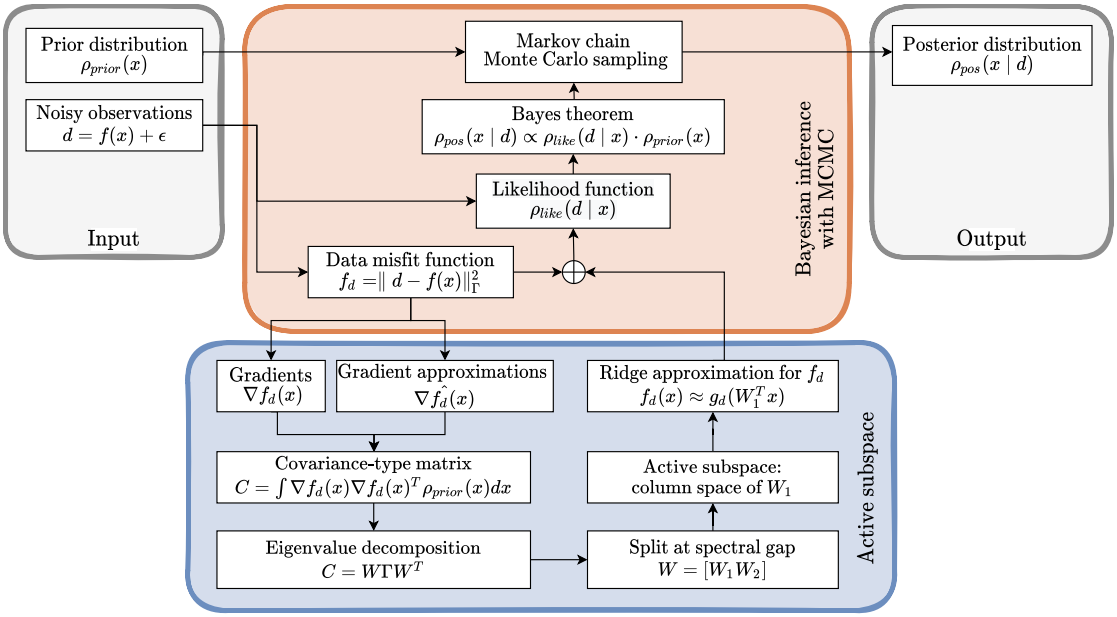


Figure 6.32: Concept of Bayesian inference with active subspaces.

6.5 Summary

This chapter focused on the calibration of the influential parameters in the bottleneck scenario to answer the research question *Q2: "How can we calibrate the influential parameters in the bottleneck scenario?"*

Q2.1: Which parameter estimation methods are suited for calibrating crowd dynamics models?

So far, calibration in crowd dynamics has mainly been performed using point estimates such as maximum likelihood estimates. Bayesian inference methods provide full posterior distributions which consider residual uncertainty after calibration when employed for subsequent studies. I identified two approaches for Bayesian inference that are suitable for crowd simulation: the likelihood-based random walk Metropolis algorithm and the likelihood-free rejection sampler for approximate Bayesian computation. I discussed how to decide whether a likelihood-based or a likelihood-free approach is suitable. Likelihood-based inference is preferable when a likelihood function is available or an assumption can be taken about its distribution and the model is deterministic. For a stochastic model, one can either generate a deterministic surrogate model or average responses of repeated model evaluations. Otherwise, likelihood-free inference with ABC is advantageous.

I applied both approaches: I first carried out a proof-of-concept by using artificial data. Both methods obtain a posterior distribution that was centered around the true parameter value. That means Bayesian inference was able to capture the true parameter value through the data.

Q2.2: What are the advantages of Bayesian inference methods for calibration compared to established methods, such as point estimates?
I compared ABC as a representative for Bayesian inference to a point estimate, the posterior mean, in three case studies. In contrast to point estimation, Bayesian inference methods provide a full posterior distribution. In the first case study, a symmetric, unimodal posterior distribution was revealed. In this case, a point estimate may be sufficient. In the second case study, I calibrated a social force based model emulator that exhibits a faster-is-slower dynamic. ABC revealed a bimodal posterior. The point estimate, by design, finds only one mode. When it is used for subsequent propagation for another quantity of interest, such as the evacuation time, the results are either too conservative or too optimistic endangering the safety of pedestrians. In the third case study, a multi-dimensional calibration, the joint posterior distribution contained additional information about the parameters and the calibration process which need to be considered for subsequent studies. In general, Bayesian inference methods are advantageous when the posterior is not symmetric or has multiple modes as well as when multiple parameters are calibrated. In these cases, the full posterior is necessary for subsequent studies.

Q2.3: What is the posterior distribution for the influential parameters in the bottleneck scenario after calibrating to experimental data?
I demonstrated how the Metropolis algorithm, an MCMC method, and the rejection sampler for ABC consistently estimate the most influential parameter, the free-flow speed mean, using experimental data. When calibrating it in the scenario with five bottlenecks, both methods obtained a comparable posterior distribution. The resulting posterior is centered around 1.12 m/s (for the Metropolis algorithm) and 1.14 m/s (for the rejection sampler) which is a plausible free-flow speed for the participants in the experiments which were students and staff of the Central Institute for Applied Mathematics in Jülich. In addition, in Section 6.4.3.3, I calibrated all four influential parameters in the bottleneck scenario using ABC. The resulting multivariate joint posterior will be used to quantify the uncertainty in the quantity of interest, the flow, after calibration in the next chapter.

7 Estimation of initial and boundary conditions: finding values for initial and boundary conditions in real-time predictions

The previous chapters dealt with forming a specific model from a model family for the bottleneck scenario. I demonstrated how methods from uncertainty quantification address this central task for classical crowd dynamics prediction. This chapter focuses on a crucial step towards real-time predictions at events, the online initialization of the simulation. In Section 7.2, I familiarize the readers with the state of the art on online parameter learning in pedestrian dynamics focusing on origins and destinations of agents so that they can put my methodological choices into context. I propose two methods for statistical learning that I employ for initialization in Section 7.3: multivariate linear regression and random forest. Both models are simple, robust, and easy to understand. Moreover, they are explainable. With these, I conduct a feasibility study on whether origin-destination (OD) matrices can be estimated from density heatmaps (Section 7.4). These heatmaps can be gained from several sensor types. In this case study, I generate them from real-world trajectory data. Finally, I summarize the findings.

7.1 Introduction

In addition to classical predictions performed before an event or during the design phases for buildings, an emerging challenge is a real-time prediction. The goal is to use predictions live at events to support those responsible with additional information about the anticipated behavior or even to provide decision support. Numerous challenges need to be tackled before crowd simulation can foster these goals. A central task is the continuous adjustment of parameters as well as initial and boundary conditions to the current situation. Choosing suitable values for these inputs is essential for a reliable prediction. The increasing monitoring of pedestrians and locations through various sensors, such as cameras and GPS sensors, comes in handy. Sensor data is the central source for the adaptation of parameters. Nevertheless, the available measurements are usually not direct measurements of a simulation parameter. Translating sensor data to inputs for the simulation is an important step in this process. In addition, there are simulation inputs that cannot be measured directly by sensors. Central information for the setup of a scenario are locations for destinations and origins of the agents and popularity of origin-destination combinations. The latter is encoded in an origin-destination matrix.

In this chapter, I use trajectory data from stereo sensors mounted in a train station to predict dynamic origin-destination matrices using linear regression and random forest. The data was provided by Swiss Federal Railways (SBB). I calculate density heatmaps because they are easy to handle and can, in general, be derived more easily from video footage than trajectory data. I train both a linear regression and a random forest regressor on a training set. Then, I evaluate the performance of both algorithms on a validation set. In addition, I study the impact of several parameters in the setup. I discuss the benefits and limitations of both methods. The resulting origin-destination matrices can be used as dynamic input for a live prediction.

At this point, I do not aspire to achieve the highest possible accuracy. Instead, I conduct a feasibility study on whether we can learn origin-destination matrices from density heatmaps.

Research question addressed in this chapter

Q2* Can we predict origin-destination matrices for the initialization of origins and destinations in the simulation from live sensor data in form of density heatmaps?

Preliminary work was presented at the Pedestrian and Evacuation Dynamics Conference 2018 in Lund and published in the proceedings of the conference [Gödel et al., 2020b]. I refer to the paper in the text.

7.2 State of the art on online parameter learning in crowd simulation

Real-time predictions at events constitute an emerging challenge in the pedestrian dynamics community. There are several issues to tackle. The largest ones are code acceleration and initialization. Algorithms must be at least real-time capable. In order to enhance pedestrian safety, computing the predictions must be sufficiently fast that enough time remains to take action. Static initialization does not suffice anymore. Instead, the current status needs to be translated to simulation inputs and fed to the simulation. The issues associated with real-time capability, as well as, approaches to solving them, are described in [Zönnchen, 2021, p. 1ff]. Here, I focus on the initialization while bearing the computational burden in mind.

At first, we take a look at studies aiming to set up systems for real-time predictions. Kemloh Wagoum et al. propose a real-time evacuation assistant for a stadium evacuation of 50000 pedestrians [Kemloh Wagoum et al., 2013]. The simulation is fed with an automated count of the pedestrians. The focus of the contribution, however, lies on the parallelization and acceleration of the algorithm. Baqui and Löhner presented their framework that evaluates crowd density from video footage at the Hajj at the Pedestrian Evacuation Dynamics conference in 2018 [Baqui and Löhner, 2020]. The density is derived from head counting by a machine learning model. The estimation aims at the

initialization of a microscopic crowd simulation. In the research project S^{2}UCRE[1], density information was extracted from video footage and used to initialize the simulation. Image processing based on neural networks was employed to extract pedestrian counts and speeds from video data. A grid-based pedestrian count was used to spawn agents in the simulation. Within a control cycle, the spawn rate was continuously adapted in order to find an agreement between the predicted and observed density using Bayesian optimization [Kneidl, 2021].

Next, I review work on the estimation of OD matrices. Like this chapter, these publications focus on the estimation of the matrices without providing a framework for real-time predictions. The estimation of OD matrix is an established research topic in traffic simulations [Cascetta, 1984, Bell, 1991]. A review can be found in [Pitombeira Neto et al., 2017]. For car traffic, static OD matrices are often sufficient. For crowd simulation, however, dynamic OD matrices are necessary because we observe more fluctuations and pedestrians have more degrees of freedom in their movement. The first step for origin-destination matrix estimation is defining the origins and destinations of the agents. In the overpass scenario, I define them based on the trajectory data. This might, however, not always be possible. Khan et al. use short tracks, tracklets, automatically extracted from video footage to identify origins and destinations with an unsupervised hierarchical clustering algorithm [Khan et al., 2016]. Their approach is applied to six scenarios: An airport, the Hajj, a train station, an escalator, a university campus, and a gallery in Milan's city center.

For Pouke et al. dynamic origin-destination matrices are just an intermediate result to validate a crowd model based on points of interest [Pouke et al., 2016]. They compute OD matrices on a city level directly from wifi mobility traces obtained in downtown Oulu, Finland. Li et al. estimate OD matrices from RGB-D images for emergencies with low visibility using a joint convolutional neural network (CNN) [Li et al., 2019]. RGB-D images are a combination of an RGB image and its corresponding depth image. A CNN is trained directly on images. Ground truth is established by manual annotation of pedestrians. Their study is carried out with RGB-D cameras installed in the hallways of a university building. Chan et al. employ a bi-level programming model, a commonly used approach for OD matrix estimation in car traffic simulation [Chan et al., 2007]. The inputs to the method are pedestrian counts. The bi-level programming model is demonstrated on an abstract, larger-scale network.

The pedestrian traffic in train stations or even a city-wide transit system is of increased interest because these areas feature a larger set of potential origins and destinations. Wong and Tong use observed passenger counts from Hong Kong to estimate dynamic OD matrices for the whole transit system [Wong and Tong, 1998]. They employ an entropy-based approach for estimation. Ahn et al. estimate a static OD matrix for the Takatsuki train station in Japan aiming to forecast pedestrian flow patterns within the

[1]Research project S^{2}UCRE, Safety and security in urban environments: crowd monitoring, prediction, and decision support, 2017-2021, sponsored by the German Federal Ministry of Education and Research `www.s2ucre.de`.

station [Ahn et al., 2017]. They use a so-called gravity model by Kawakami[2] for the OD matrix estimation. Input for the estimation are traffic counts manually observed at the station. Bauer estimates daily averaged OD matrices from pedestrian counts obtained from sensors at the entrances and exits [Bauer, 2012]. The data set stems from an Austrian shopping center. Data is separated into a training and a validation set; the performance is evaluated using the R^2 score. The OD matrices are estimated with a generalized method of moments. Hänseler et al. estimate dynamic OD matrices in the Lausanne train station based on tracking data, travel surveys, and train timetables [Hänseler et al., 2017]. Yet, while their goal is similar, I aim to estimate OD matrices solely based on relatively simple density information instead of combining different types of data. This limitation of information makes it much easier to collect the data reliably and continuously in a complex scenario such as a train station. Thus it improves applicability.

My work on estimating OD matrices consists of three parts: First, I analyzed how the distribution of a pedestrian stream to three destinations can be predicted [Gödel et al., 2020b]. Substitute input data from which I gain the heatmaps are simulated with Vadere. Based on this, I took experimental trajectory data from the Jülich research center to predict the origin distribution of three pedestrian streams that merge into a single destination [Gödel et al., 2020c]. In both setups, the distribution of the agents, or pedestrians, to the destinations and origins is estimated with high accuracy. This positively answers my preliminary research question, whether density heatmaps contain enough information to predict destinations. However, the studies were based on either artificial data or controlled experiments, both with unidirectional flows and a limited variation in behaviors. Thus, while the results are encouraging, their informative value is limited, and one hesitates to generalize to real-world data. These previous studies motivate using the proposed approach on real-world trajectory data. I investigate the pedestrian flow through an overpass with stairs leading to platforms and the station hall in a train station. Since the flow is multi-directional, it no longer suffices to predict only origin or destination distributions. Instead, the method predicts full origin-destination matrices. This is a much more complex task because it entails many features.

7.3 Statistical learning models for online parameter learning

In statistical learning, the algorithm for prediction is typically called a model. I employ two models, a multivariate linear regression, and a random forest. There is a multitude of models for statistical learning, I pick models that are simple yet robust, easy to understand, and explainable: multivariate linear regression and random forest. Common other models are neural networks, such as CNNs, which are trained on images, as well as recurrent neural networks, which incorporate previous information. The disadvantage of neural networks for this case study is that their performance depends highly on the setup and the parameters. The goal is a feasibility analysis to see if a learning model

[2]The reference is only available in Japanese: Statistical Aspects of Gravity Models and Entropy Models, Shogo Kawakami, 1978 doi:10.2208/jscej1969.1978.272_135.

can estimate the OD matrices from density heatmaps rather than a performance analysis aiming to find the model that provides the highest accuracy.

In supervised learning, each model is first trained on so-called training data. The training data set consists of samples, which are the input to the model, and targets, which are the correct response for the samples. My samples are density heatmaps, and the targets are OD matrices. Each sample consists of features. In this case, each entry of the OD matrix is a feature. Some features might be more important than others. Choosing the influential features is a complex task called feature selection. The goal of training is not only that the model can resemble the training data well but also that it generalizes well, which means it adapts properly to unseen data. Therefore, the model should not fit the training data too closely. Otherwise, overfitting occurs, which implicates that the model performs only well on the training data set, but not on a different data set. Once training is completed, the performance of the trained model is evaluated on a test set. As for the training set, the test set consists of samples and targets. Training and test sets must be disjoint.

7.3.1 Multivariate linear regression

As the first model, I introduce multivariate linear regression. The model is well understood and fast. It is a linear model which means that it assumes a linear relationship between input variables and output variables. Linear regression has no user-defined parameters. While it is a classical method from statistics, it also can be used for statistical learning and follows the same structure. This problem needs multivariate linear regression since the output is not scalar, but a set of matrix entries. The general formulation for multivariate linear regression is

$$Y = XB + E$$

where the targets Y are represented by the samples X multiplied with the regression coefficients B plus the approximation error E. In this case, each row of the targets Y is one OD matrix, each row in X is a sequence of density heatmaps.

7.3.2 Random forest

Random forest is a nonlinear supervised statistical learning model. It is fast, explainable, robust, interpretable, and depends only on a few parameters. The three main parameters to tune the performance of random forest are the number of estimators or trees that are trained to the data, the maximum number of splits from the root node, or maximum depth, and the number of features used in each split. Random forests were introduced by Leo Breiman [Breiman et al., 1984]. A random forest is an ensemble of decision trees. Decision trees tend to overfit data. This effect is reduced when relying on a set of trees and consolidating the results. Each tree is trained with a random subset of the data, a so-called bootstrap sample, and for each tree node, a random subset of the features is used for training. Consequently, each tree is trained differently and independently of the others. Hence, the correlation between individual trees is low. In order to make

a prediction, a sample is handed to all trees and their average response is returned as the prediction. This averaging of responses ensures the high robustness of the forest. Figure 7.1 illustrates the workings of random forest. Each tree grows by splitting a subset of the data. At each split, from a subset of all features, the feature is selected that leads to the best split. The split criterion is the mean squared error (MSE). Hence, the best split minimizes the MSE. The data set is split at that threshold that minimizes the MSE. The algorithm continues iteratively until either a stopping criterion, such as a maximum number of splits or a minimum number of samples is each terminal node, is met or terminal nodes are reached.

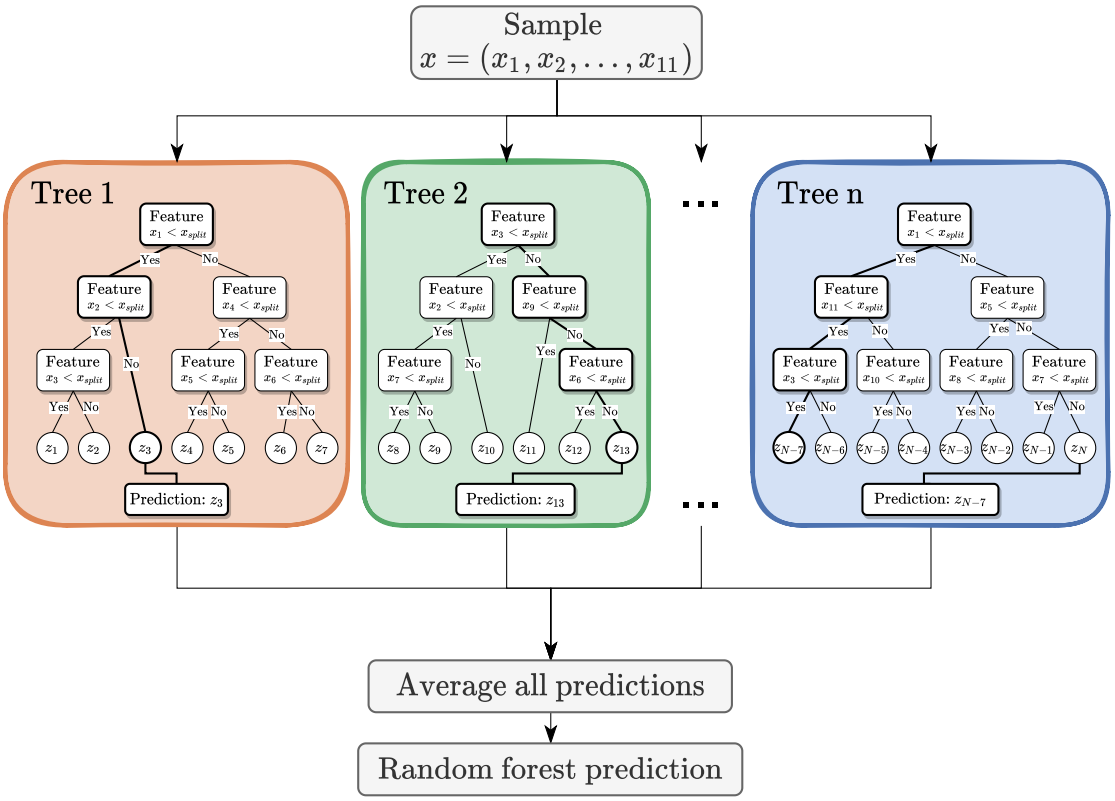


Figure 7.1: Schematic working of random forest regression.

7.4 Studying statistical learning for origins and destinations of pedestrians

In this section, I analyze a data set provided by Swiss Federal Railways (SBB). I evaluate the distribution of the speeds as well as the number of pedestrians in the overpass throughout the day. Then, I describe how the trajectories are preprocessed to obtain the density heatmaps as well as the OD matrices. Afterward, I evaluate the performance of both models.

7.4.1 Analysis of trajectory data

The data set consists of pedestrian trajectories for the overpass in a train station which allows access to shops as well as to the platforms, as described in Section 3.2.2. The trajectories stem from stereo sensors by Xovis mounted at the ceiling. They record three-dimensional trajectories (x, y, h) with a resolution of 0.1 seconds. I use only the two-dimensional positions (x, y). The data contains imperfections such as time gaps up to three seconds or broken trajectories that occur e. g. when a pedestrian enters a shop and cannot be captured by the sensor.

7.4.1.1 Speeds

One can easily derive the speed of the pedestrians from the trajectories and I analyze it in order to gain some insights into the population. Since no personal information such as age, fitness, and type or purpose such as commuter, traveler, or shopper is available, I hope to learn about the population from the speed distribution. Figure 7.2 shows the distribution of the walking speeds. For each pedestrian, the speed is calculated in each time step based on the positions. That means stationary periods are included. The observed walking speeds are asymmetrically distributed, their mean is around 1.13 m/s and the standard deviation around 0.37 m/s. The observed speeds match up with those observed by Davidich et al. during rush hour in a German train station with mode 1.04 m/s and standard deviation 0.51 m/s [Davidich and Köster, 2013]. The results are also in accordance with earlier measurements [Weidmann, 1993, Bosina and Weidmann, 2017, Bosina, 2018]. They indicate that we observe a typical situation at a train station.

7.4.1.2 Pedestrian count

When we take a look at the number of pedestrians over the course of the day for the three days in the data set presented in Figure 7.3, we observe a strong variation. Qualitatively, the patterns are similar for all three days. We observe two rush hours around 5-9 a.m. and 4-7 p.m.. Before the morning rush hour and after the evening rush hour, traffic dwindles almost to zero. In addition to this trend, we observe a fluctuation with a higher frequency. These peaks and tails are most likely caused by pedestrians onboarding and de-boarding trains. These large variations demonstrate that we need a method that dynamically estimates the OD matrices for short time intervals.

7.4.2 Preprocessing of raw data

The density heatmaps and the "ground truth" origin-destination matrices, which serve as samples and targets for the estimation, are generated from the raw trajectory data. I work with density heatmaps because they are non-personal, their size is constant, they can be gained from different types of sensors, even directly, or extracted from video footage like in the research project S^2UCRE. I calculate the density from trajectory snippets within a time interval of a fixed length of ten seconds, $\Delta t = 10\,\mathrm{s}$. This decision is based on the following rough calculation: Within ten seconds, pedestrians move an

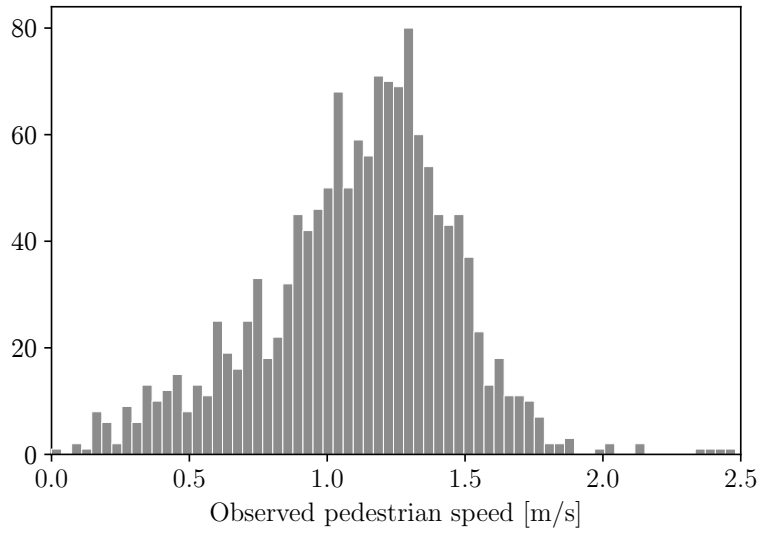
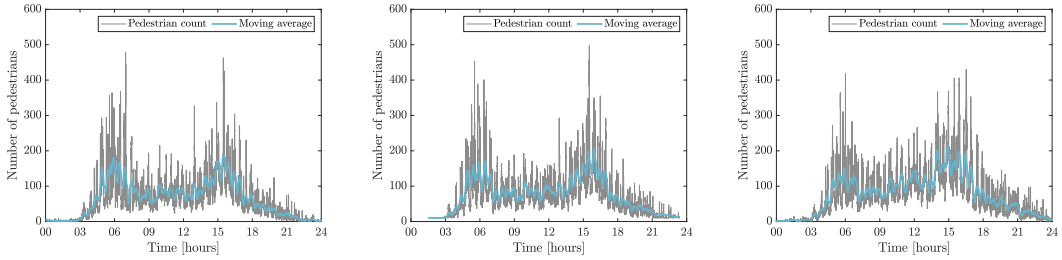


Figure 7.2: Observed speeds of all pedestrians within the overpass. The speed is measured over the complete time a pedestrian remains inside the measurement area, including stationary periods.



(a) Day 20190902 (Mon) **(b)** Day 20190905 (Thu) **(c)** Day 20190906 (Fri)

Figure 7.3: Number of pedestrians in the overpass over the course of the three separate days of the data set, Monday, Thursday, and Friday. The pedestrian count is evaluated every ten seconds. In addition, a moving average filter with a span of 120 (= 20 min) is shown.

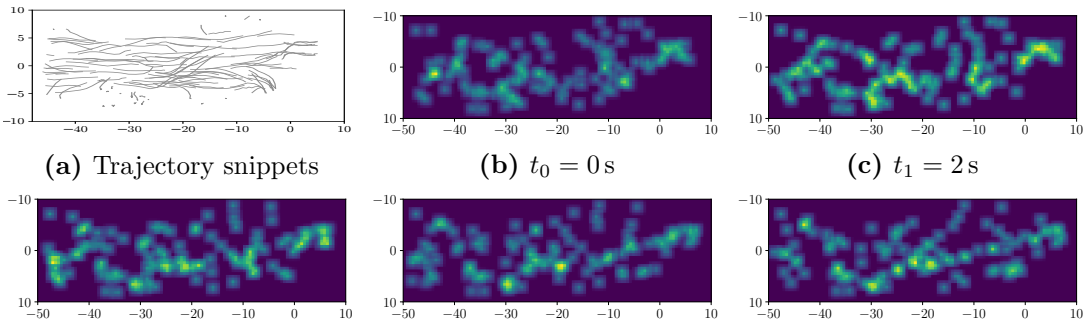
average of 11 meters, which is a substantial distance in the overpass but shorter than the origin-destination links. Time intervals in which less than ten pedestrians are present in the overpass are not considered. The trip counts for the "true" origin-destination relations are also derived directly from the trajectory data. I ignore measurement errors in the trajectories. If the density maps are obtained from video footage additional errors are introduced that I cannot account for in this work. In this case, the raw data stems from stereo sensors installed on an indoor ceiling, hence their setup is optimized for

pedestrian tracking. In most applications, this is not the case and measurement error may have a stronger impact on the proposed process.

7.4.2.1 From trajectories to density heatmaps

For a time interval of $\Delta t = 10\,\mathrm{s}$, I create a series of five density heatmaps at equidistant time steps ($t_0 = 0\,\mathrm{s}$, $t_1 = 2\,\mathrm{s}$, …, $t_4 = 8\,\mathrm{s}$). At each time step, the density at each position in the overpass is computed with a Gaussian filter [Seitz and Köster, 2012]. That means, every pedestrian within the observation area and within the time interval is represented by a Gaussian bell. The observation area is divided into a regular grid of 0.5 m width per cell. In a cell, the contributions of the Gaussian representations of all nearby pedestrians are added. As a result, I obtain five density heatmaps which consist of 108×38 values each.

In Figure 7.4 an exemplary series of density heatmaps for a ten-second time interval is shown. Figure 7.4a displays the trajectory snippets for which the heatmaps were calculated. From the evolution of the density, we observe that there is a multi-directional movement within the observation area. In addition, we notice that the density matrices are sparse.



(a) Trajectory snippets **(b)** $t_0 = 0\,\mathrm{s}$ **(c)** $t_1 = 2\,\mathrm{s}$

(d) $t_2 = 4\,\mathrm{s}$ **(e)** $t_3 = 6\,\mathrm{s}$ **(f)** $t_4 = 8\,\mathrm{s}$

Figure 7.4: From trajectory snippets (a) to a series of five density heatmaps (b-f), both for a time interval of ten seconds.

7.4.2.2 Decomposition of input samples

I form input samples of five density heatmaps to estimate the corresponding OD matrix. In total, this amounts to 20520 entries, or features, per sample. When we use several of these input samples, their size will slow down the computations or even make them infeasible. On the other hand, we need a substantially larger number of input samples than features to avoid overfitting. Fortunately, the matrices are sparse as can be seen in Figure 7.4. Thus, I reduce the dimension of the input samples by a principal component analysis (PCA). As an additional benefit, the PCA helps to reduce the noise in the data. The PCA decomposes the input samples into

$$X = W Z^T \approx W_{\hat{n}} Z_{\hat{n}}^T \tag{7.1}$$

where $X \in \mathbb{R}^{m \times n}$ contains m centered density heatmap samples with n features, W contains the principal components of X, or eigenvalues of $X^T X$, and Z contains the right singular vectors of X, or eigenvectors of $X^T X$. The singular vectors in Z are sorted in descending order by the magnitude of the singular values of X. By only keeping the first $\hat{n}$ components, the reduced matrix has dimensions $m \times \hat{n}$. The matrix Z is stored by the routines. It is necessary for the reconstruction of X or its approximation $\hat{X}$. X can only be fully recovered if all components are kept, $\hat{n} = n$. For the case that only components are kept, $\hat{n} < n$, the reconstruction approximates X.

Figure 7.5 shows the explained variance of the reconstructed data over the number of components. It indicates how well the components explain the original data. I decide to use 321 components to reach an explained variance of 75%. Note that with this, one cannot expect the estimated OD matrices to perfectly fit the true OD matrices. However, my goal is not to obtain the highest accuracy but a proof of concept that OD matrices can indeed be learned.

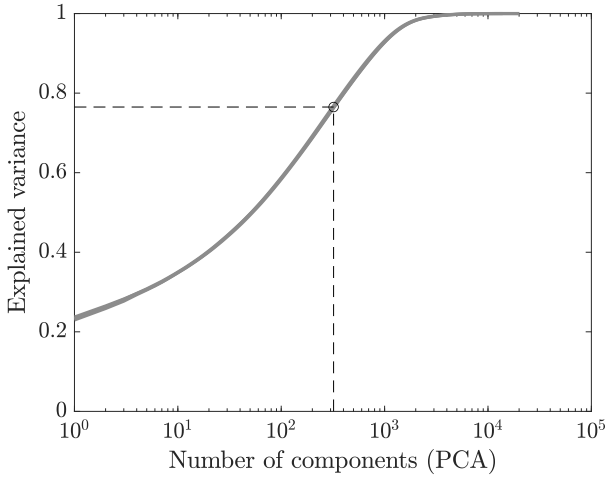


Figure 7.5: Explained variance of the principal component analysis (PCA) for the input samples. Each sample contains five density maps computed at equidistant steps within the time interval ($\Delta t = 10\,s$).

7.4.2.3 Ground truth: defining origins and destinations

By plotting the trajectories of a complete day (see Figure 7.6), I identify nine relevant areas in which most of the trips start or end. They correspond to escalators, stairs, and elevators that lead to the platforms (ids 1,2,7,8) as well as to the station hall (ids 3,4,5) and to the remaining part of the overpass that is not covered by the sensor (id 0) as we can see from Figure 7.7. I define these areas as origins and destinations with the ids $0 - 8$.

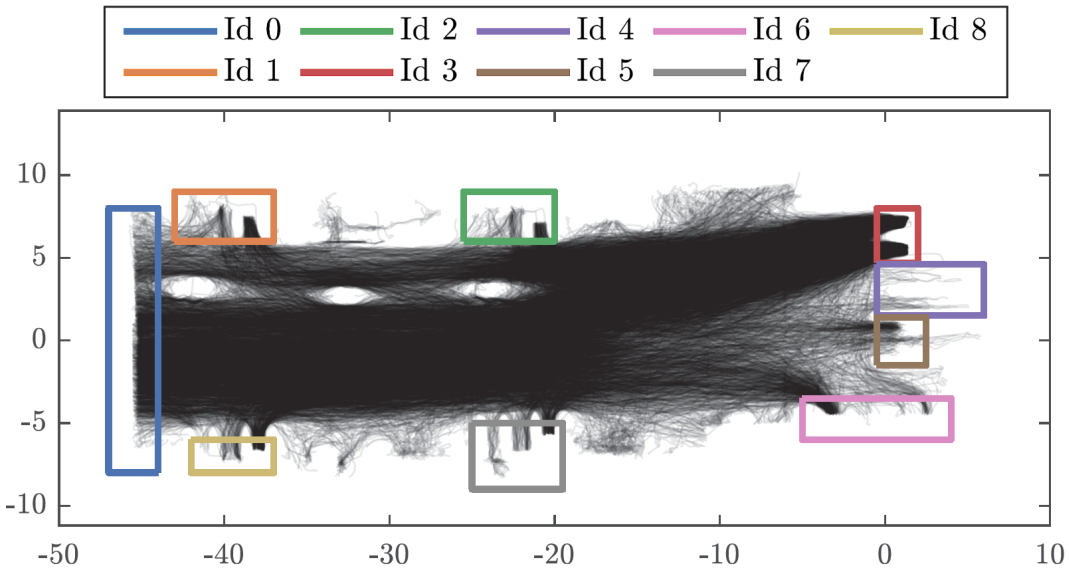


Figure 7.6: Pedestrian trajectories of a single day show which areas are strongly frequented. Nine areas (ids 0-8) are identified as relevant origins and destinations.

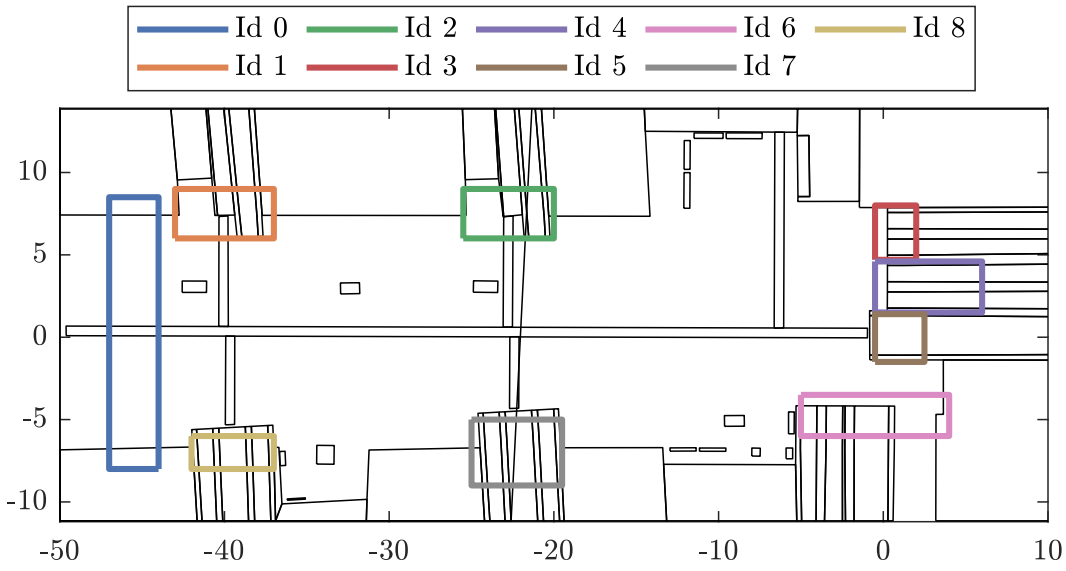


Figure 7.7: Section of the overpass with escalators, stairs, and elevators to the platforms (ids 1,2,6,7,8) and the station hall (ids 3,4,5) and the remaining part of the overpass (id 0) as possible origins and destinations.

In order to determine the OD matrices for a time interval Δt, each pedestrian within the observation area is mapped to one origin and one destination. If the start or endpoint

of the full trajectory is outside of the identified areas, I assign an artificial origin or destination with index -1. In this way, I map about 65% of the pedestrians to both an origin and a destination. Please note that this means that the ground truth is not ideal and is likely to impact the results.

My definition of the OD matrix is based on trip counts. Each entry $a_{i,j}$ of the OD matrix $A = (a_{i,j})_{i,j\in[1,10]}$ contains the number of pedestrians who traveled from origin i to destination j. This leads to a 10×10 origin-destination matrix. Exemplary OD matrices can be seen in Figure 7.8. We observe that the OD matrices vary, as expected.

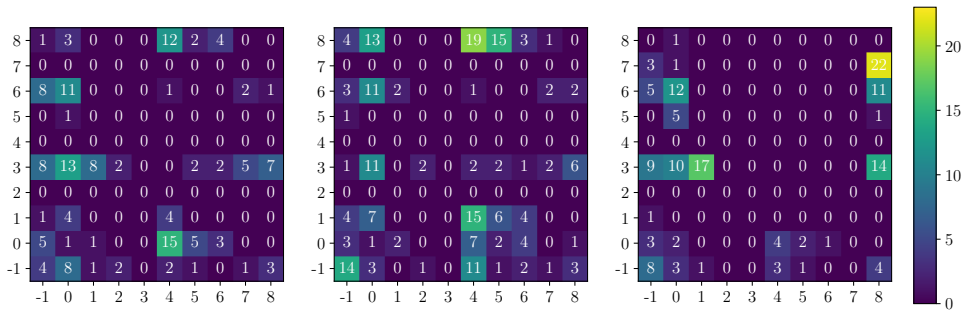


Figure 7.8: Exemplary origin-destination matrices for a time intervals of ten seconds. Pedestrians whose trajectories could not be matched with an origin or destination, are assigned -1 as origin and/or destination index.

7.4.2.4 Setup of the learning models

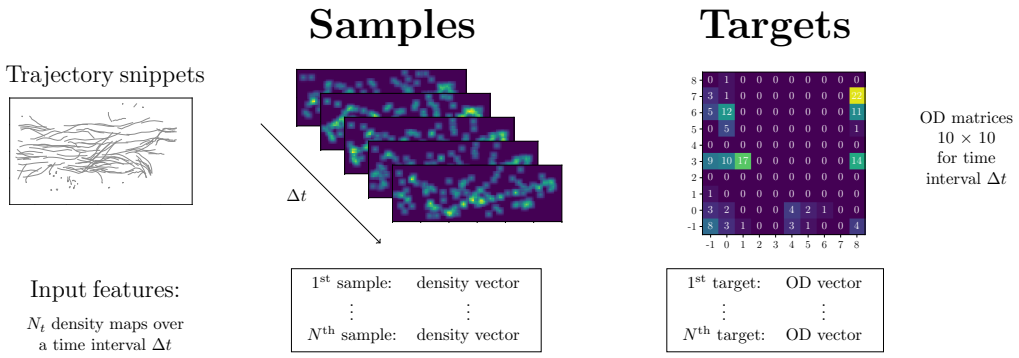


Figure 7.9: Processing scheme for the estimation of origin-destination (OD) matrices from a series of density heatmaps.

Figure 7.9 visualizes the processing of the trajectory data to samples and targets. For the input, the time interval Δt is sampled with N_t density maps. After decomposition, each sample contains 321 entries. These are then flattened in row-major and concatenated to an input sample of size $N \times 321$. Each input sample is paired with a target

sample that contains the OD matrix for the same observation time interval Δt, as a row-major flatted vector with ten target entries. Since the observation period for each input and output sample is $\Delta t = 10\,s$, the resulting data set for a single day contains approximately 4000 samples. The time intervals Δt for two consecutive samples are disjoint.

7.4.3 Performance of the models

I measure the performance of both multivariate linear regression and random forest through the R^2 score. In addition, I assess how well the models generalize to unseen data using cross-validation.

7.4.3.1 Performance metric: R^2 score

I employ the coefficient of determination R^2 as a metric for the performance of the algorithm. The score is defined as

$$R^2(y, \hat{y}) = 1 - \frac{\sum_n^{i=1}(y_i - \hat{y}_i)^2}{\sum_n^{i=1}(y_i - \bar{y}_i)^2} \tag{7.2}$$

where y_i is the ground truth result with mean $\bar{y}_i$ and $\hat{y}_i$ is the prediction. The coefficient of determination indicates how well the predicted values fit the ground truth values. There is no lower limit for the score; the upper limit is 1. For a multidimensional output, the scores are averaged, weighted by the variance of the ground truth samples[3].

When conducting this study, I noticed that the results strongly depend on the chosen metric. The R^2 score is a common metric of the scikit-learn library that puts the results into the statistical context. However, a custom performance measure for origin-destination matrices that includes the impact of the initialization on the simulation results might be useful.

7.4.3.2 Cross-validation

One way to assess the generalization error and the robustness of a machine learning model for a set of data is to perform k-fold cross-validation. It means to separate the data into k disjoint parts, so-called folds, and then use $k - 1$ folds for training and one fold for validation. Instead of evaluating the performance for one split of the data (into training and test set), the performance is analyzed for each combination.

I perform 3-fold cross-validation with a data set consisting of three different weekdays. For every fold, two days are used as the training set and one day is used as the test set. The folds are not of exactly equal length since there is some variation on how many intervals are disregarded, because there are too few pedestrians. With this setup, we can analyze if the model's performance depends on the day.

[3]Default calculation method for the score in Python's scikit-learn package up to version 0.23.0.

7.4.3.3 Linear regression

The output of the prediction, the OD matrix, has 100 entries. Most machine learning models perform better with a scalar output or at least a low dimensional output. Therefore, I decided to apply a PCA decomposition to the 100-dimensional target space to reduce the output dimension. That means I predict the heatmaps on the reduced basis of the PCA which can also affect the accuracy.

As a performance metric, I calculate two scores. First, the R^2 score in the component domain, and second, the R^2 score in the OD matrix domain. For the latter, I reconstruct the full OD matrix before computing the score.

We observe that the scores in the component domain are superior for a low dimensional output (Figure 7.10). This is the result of the trade-off between a low dimensional output that improves the regression's performance and the decomposition error that is large for a low number of components. In addition, in Figure 7.10 the scores are shown for each fold (markers). We observe similar R^2 scores for all folds demonstrating that the model is robust against the variation between days. I argue that the scores, while not ideal, indicate that the linear regression model captures a major part of the OD matrix variation over the day. This is supported by the evolution of the components over time in Figure 7.11.

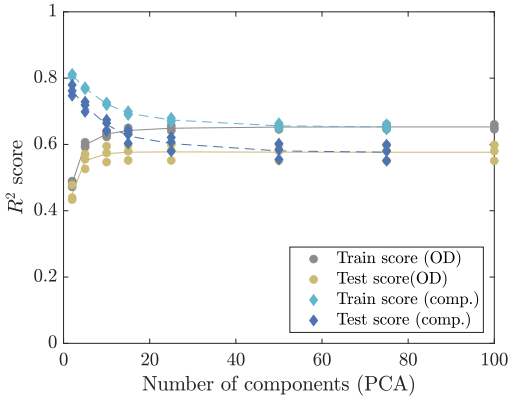
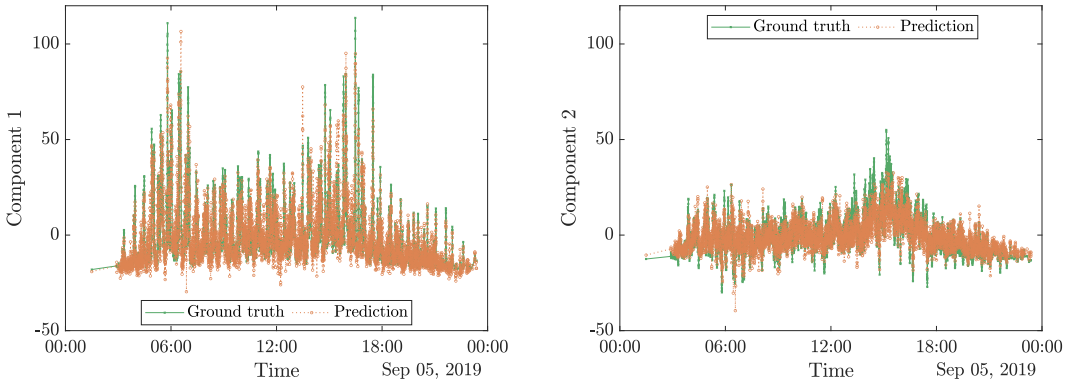


Figure 7.10: Performance for multivariate linear regression with reduced output dimension after principal component analysis (PCA). R^2 scores are calculated in the component domain as well as for the reconstructed origin-destination (OD) matrices.

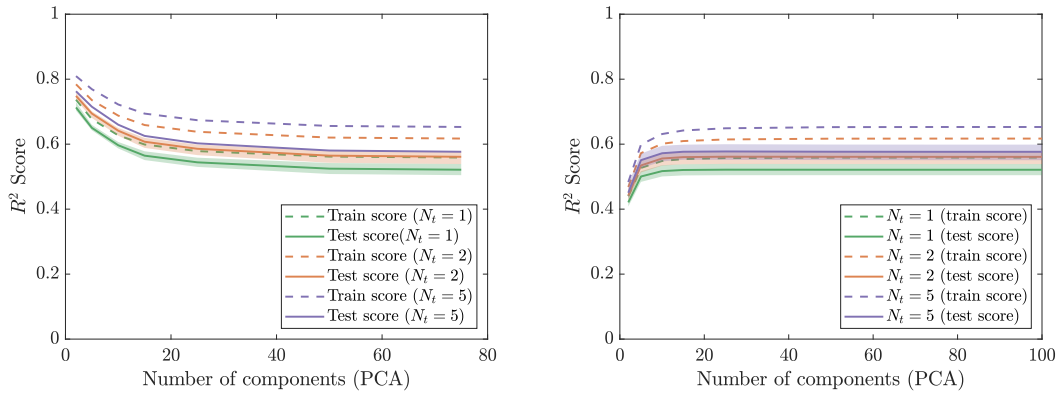
Initially, I chose $N_t = 5$ heatmaps in each sample to capture the movement of the crowd with the density heatmaps. However, it is not clear whether the linear regression model profits from the series of density heatmaps. I clarify this by comparing the performance with a single density heatmap, $N_t = 1$, to a series of $N_t = 2$ and $N_t = 5$ density heatmaps per samples, see Figure 7.12. Again, I use the PCA decomposition for the input samples with 75% explained variance. That corresponds to 99 components for a single heatmap in each sample and 182 components for a series of two density heatmaps, respectively. I expect that using a series of density maps to predict the OD matrices improves the performance. In fact, there is an increase in the R^2-score when



(a) First PCA component **(b)** Second PCA component

Figure 7.11: Predicted and ground truth components of principal component analysis (PCA) of origin-destination matrices over the course of one day.

comparing a single heatmap, $N_t = 1$, to a series of two heatmaps, $N_t = 2$. Using $N_t = 5$ heatmaps brings only a negligible improvement over $N_t = 2$.



(a) Component scores **(b)** Origin-destination matrix scores

Figure 7.12: Average performance of multivariate linear regression with different input samples: Each sample contains either a single density heatmap or a series of two or five density heatmaps.

7.4.3.4 Analysis of predicted components

Let us take a look at why the performance of the regression remains so far from an ideal R^2 value of 1. In the previous sections, we saw that keeping two components of the OD matrix decomposition results in the best performance for the component scores. Plotting the two components of the prediction and ground truth visualizes how well the predictions fit the targets in the component domain. See Figure 7.13. The predictions

of the linear regression overlap with part of the ground truth data. Nevertheless, the shapes of the data clouds differ substantially. I suspect that the overlap reflects the linear part of the relationship between input samples and target samples. This indicates that a nonlinear model is necessary to obtain a better fit.

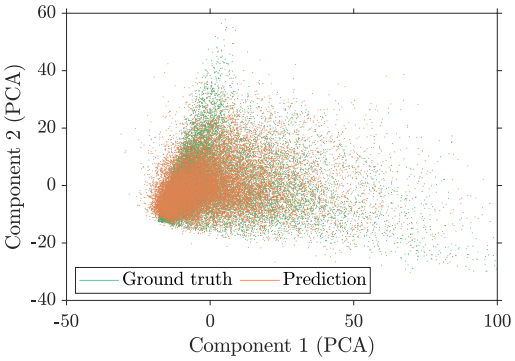


Figure 7.13: Scatter plot of the first and second component of the principal component analysis of the origin-destination matrix. The multivariate linear regression predictions of the components are shown together with the ground truth components.

To briefly summarize: The results of the regression model show that OD matrices can be learned from density heatmaps. However, we observed that a linear model does not suffice to obtain high accuracy.

7.4.3.5 Nonlinear model: random forest

I decided to use random forest as a nonlinear model. Random forest is a well-understood and relatively simple model with a low number of parameters. It is also a model that can be analyzed. One can apply random forest straight away by replacing the linear regression in the scikit-learn setup. In this way, we can find out whether a nonlinear model is better suited to learning the OD matrices. I analyzed the impact of the main parameters, the number of trees, the number of features at each split, and the maximum depth, (see C.1) and use the best configuration[4]. Figure 7.14 shows the results. The performance of random forest in this configuration is slightly worse compared to linear regression when considering the test scores. In order to rule out overfitting due to a low number of samples, I created a larger data set by using overlapping time intervals, see C.2. However, the performance of random forest does not change with the larger data set.

[4]Number of trees, `n_estimators`: 100, maximum depth of a tree, `max_depth`: 10, maximum number of features considered for each split, `max_features`: `n_features` (321)

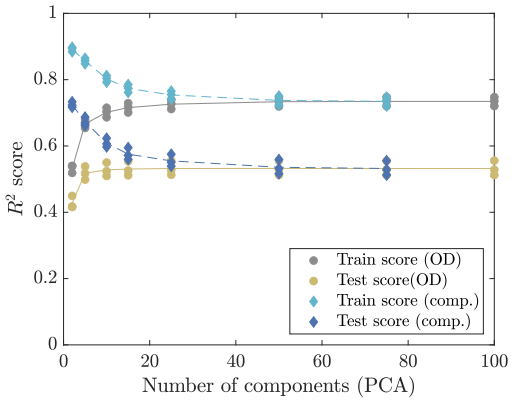


Figure 7.14: Performance of random forest model measured by the R^2 score. Instead of the full origin-destination (OD) matrix, the components of its principal component analysis (PCA) are predicted.

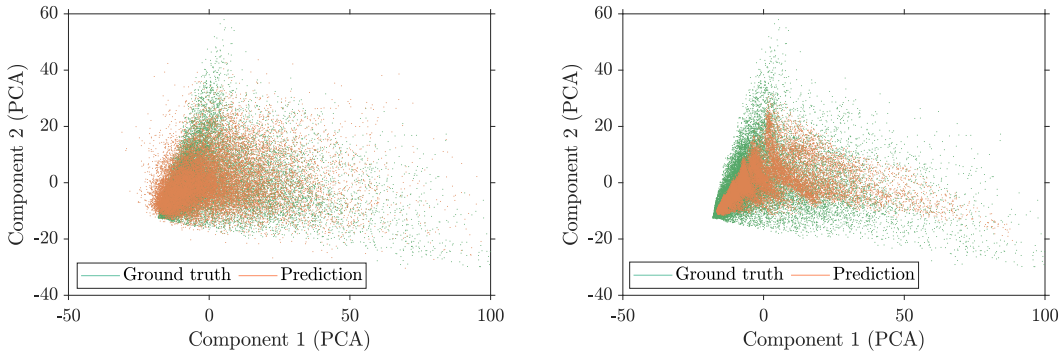
7.4.3.6 Component and performance analysis

Figure 7.15 compares the first two ground truth components and the first two predicted components of the OD matrix for linear regression and random forest. The predictions of the random forest model seem to fit the ground truth components much better despite the mediocre scores.

Even if the ground truth and predicted components still do not overlap perfectly. This means that some part of the data set is not captured by the random forest model. I consider this encouraging news since linear regression still outperforms random forest even though the shape is captured better with random forest, see Figure 7.14. The point cloud that is formed by the first two components of the ground truth forms a triangular point cloud. Linear regression, however, creates a circular point cloud. The regression predicts values outside of the ground truth. The random forest predictions, on the other hand, lie within the ground truth. Therefore, I believe that in order to improve the accuracy, a nonlinear model is necessary. However, finding the optimal nonlinear model is a time-consuming task that is out of scope for this work.

7.5 Summary

In this chapter, I described the dynamic initialization of real-time simulations of crowds during events as a current challenge in crowd dynamics. The origins and destinations of the agents are central information for the initialization. They are vital to running a simulation and have a large impact on the results. Improper input values can even compromise the prediction. Origin-destination matrices describe the popularity of combinations of origins and destinations. In a train station, origin-destination matrices indicate for example how many pedestrians head from the main entrance towards a certain platform and how many pedestrians head towards the different shops. This chapter



(a) Linear regression **(b)** Random forest

Figure 7.15: Component analysis for multivariate linear regression and random forest. The scatter plot of the first and second component of the principal component analysis (PCA) of the origin-destination matrix is shown for each method compared to the ground truth components.

addressed the research question *Q2*: "Can we predict origin-destination matrices for the initialization of origins and destinations in the simulation from live sensor data in form of density heatmaps?"*.

I demonstrated how to estimate dynamic origin-destination matrices from density heatmaps with statistical learning models. This is one concrete example to show how new methods complement classical parameter estimation. Density heatmaps are relatively easy to obtain in practice from sensor data, in this case, trajectory data. Swiss Federal Railways provided trajectories for the overpass in the Basel train station. I used a series of five heatmaps in a time interval of ten seconds as input for the models. The goal was to predict the OD matrix for the same time interval. In order to reduce the dimensionality of the input data, I performed a PCA on the input space, maintaining an explained variance of 75% of the data.

I compared two statistical learning models for the estimation: a multivariate linear regression model to see whether a simple linear model might suffice and random forest to detect possible nonlinearity. We observed that the linear regression successfully predicts the OD matrices. Further analysis pointed out that the linear relationship mapped in the linear regression explains only a part of the data. I conclude that a nonlinear model is necessary to capture the full relationship. Consequently, I applied a random forest model to the problem which captured the shape of the output space better but overall performed slightly worse in terms of R^2 score.

Overall, my results show that learning models such as multivariate regression and random forest both successfully predict origin-destination matrices from density heatmaps which can nowadays be automatically extracted from sensor data. However, there is still room for improvement in terms of accuracy. These results encourage further efforts to find nonlinear models to predict the OD matrices.

For real-time crowd predictions, real-time estimated OD matrices are necessary to initialize the simulation. More broadly, this case study shows how dynamic initialization based on live data can be addressed with new methods.

8 Uncertainty analysis: measuring the reduction of uncertainty in the simulation output

In order to reduce the uncertainty when simulating the bottleneck scenario, I followed a two-step approach: First, I identified influential and non-influential input parameters. Second, I calibrated the influential parameters using experimental data. Now, I assess the uncertainty in the prediction. This task is called uncertainty propagation, forward propagation, or uncertainty analysis. To put my choice of methods into context, I present the state of the art on uncertainty analysis in crowd simulation in Section 8.2 and give an overview of methods for forward propagation from uncertainty quantification in Section 8.3. I choose Monte Carlo sampling for forward propagation on the bottleneck scenario and analyze the uncertainty in the simulation output in Section 8.4. Finally, I summarize the findings.

8.1 Introduction

I evaluate the uncertainty before and after calibration to complete the full process of parameter handling. For this purpose, I propagate both the flat priors and the posterior distribution obtained from the calibration in Section 6.4.3.3 to evaluate the distribution of the quantity of interest: the flow through the bottleneck. Again, I focus on the bottleneck scenario described in Section 3.2.1. From the established methods for uncertainty analysis, I choose Monte Carlo sampling for this application since it is a flexible approach that can be used for all input parameter distributions. In addition, I demonstrate how propagation can be performed with a generalized polynomial chaos expansion which is of particular interest for computationally demanding scenarios. I compare the uncertainty in the response at three steps in the process to show how the calibration of the influential parameters reduces the uncertainty in the simulation output. First, I propagate the flat priors that I used for the sensitivity analysis. Second, I analyze the setting in which I fix all non-influential parameters. Finally, I propagate the joint posterior distribution of all influential parameters.

Research questions addressed in this chapter

Q3 How can we quantify the uncertainty in the prediction for the bottleneck scenario?

Q3.1 Which uncertainty analysis methods are suited for crowd simulations?

Q3.2 How large is the uncertainty in the prediction for the bottleneck scenario before and after calibrating influential parameters?

8.2 State of the art on uncertainty analysis in crowd simulation

In the crowd dynamics community, studies in which uncertain parameters are propagated are often referred to as "sensitivity analysis" even though no sensitivity metric is defined. In Table B.1, these studies are summarized. Most of them use a one-factor-at-a-time (OAT) approach where only one parameter is propagated at a time while the others are fixed, instead of propagating all parameter distributions. The reason is that they aim to perform a sensitivity analysis without employing specific methods for this task. They judge the sensitivity of the parameters based on the propagation of each factor. Different from my approach, their focus is not on the distribution of the quantity of interest. With the OAT approach, the total uncertainty in the output cannot be evaluated. Therefore I do not go into more detail here.

So far, there are only a few studies using systematic methods from uncertainty quantification for propagation. Several studies from the research group at the Munich University of Applied Sciences employ spectral methods based on the generalized polynomial chaos expansion. All of them rely on the chaospy package [Feinberg and Langtangen, 2015] for uncertainty analysis and Vadere for crowd simulation. Sivers et al. study the impact of three parameters on the number of agents still in a danger zone with the pseudo-spectral approach for polynomial chaos expansion. They simulate the helping behavior observed during the subway attacks in London in 2005 [von Sivers et al., 2016b]. The uncertain parameters are the number of people sharing a social identity, the percentage of injured pedestrians, and the speed of a helper with an injured person. All three parameters are part of the helping model implemented in the social identity model application which is an extension of Vadere. Dietrich et al. study uncertainties for de-boarding a train with the same approach [Dietrich et al., 2018]. They analyze the impact of the number of pedestrians on the platform and on the train, and the free-flow speed mean on the total number of pedestrians on the platform. Instead of generating an expansion for the microscopic model, the gradient navigation model [Dietrich and Köster, 2014], Dietrich et al. introduce a data-driven surrogate on closed observables as an intermediate step. The computational effort then lies in the model evaluations for the generation of the surrogate model instead of for the expansion. Both, Sivers et al. and Dietrich et al. , study time-dependent quantities of interest by generating one expansion for each time step in the simulation. In [Kurtc et al., 2021], a corridor scenario similar to the experiments of [Zhang et al., 2011] is analyzed. Kurtc et al. use point collocation to quantify the impact of the mean and standard deviation of the free-flow speed, and the number of pedestrians on the density in the corridor. They propose two ways to deal with the stochasticity in the simulator: averaging a fixed number of repetitions and building several expansions from individual repetitions. Finally, in [Rahn et al., 2021], we study the uncertainty in the length of a demonstration march when the number of participants and the standard deviation of the free-flow speed of the participants are uncertain. In a smaller version

of the scenario, both the collocation and the pseudo-spectral approach are compared to Monte Carlo sampling. The results of the pseudo-spectral approach are closer to the Monte Carlo simulations and therefore we select this method for the original scenario size where Monto Carlo simulations are not feasible. Similar to Dietrich et al. [2018] and von Sivers et al. [2016a], a time-dependent analysis is performed. The stochasticity in the simulator is reduced by averaging ten repetitions at each parameter set. The results are published in [Rahn et al., 2021] and also briefly described in Section 5.4.5 as far as they fit into the context of this thesis.

8.3 Methods for uncertainty analysis

There are several approaches to quantify the impact of uncertain parameters on the model response: direct evaluation, perturbation methods, sampling methods, and spectral methods. Figure 8.1 provides a brief overview.

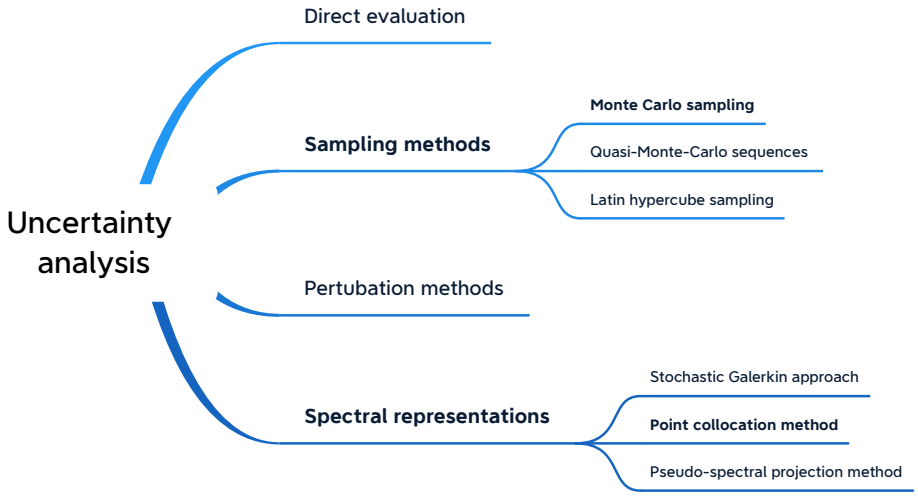


Figure 8.1: Overview of methods for uncertainty analysis. Methods that were selected for crowd simulations are highlighted.

Direct evaluation is suited only for linearly parameterized problems. The mean and covariance of the model response can be found by evaluating the mean and covariance of the parameter distributions of the uncertain parameters. Consequently, no model evaluations are necessary. However, most models are not linearly parameterized. Perturbation methods use Taylor expansions to represent the model response. They aim at nonlinearly parameterized problems. The accuracy depends on the order of the expansion. Direct evaluation provides an estimate for the mean and variance or covariance. However, the distribution of the quantity of interest is not calculated. Perturbation methods are also referred to as propagation of moments, second-moment approach, or propagation of errors since only moments of the distribution are determined. Another approach

for nonlinearly parameterized problems is sampling. It is suitable for all parameter distributions. That includes posterior distributions obtained from Bayesian techniques or experiments as discussed in Chapter 6. The methods are intuitive and easy to implement. Sampling methods obtain an ensemble of responses and can therefore evaluate the density of the response. They are the best choice for problems with correlated parameters or sufficiently large parameter dimensions [Smith, 2014]. The core of sampling methods is the choice of samples, that is the parameter values at which the model is evaluated. Two established approaches for sampling methods are Monte Carlo and Latin hypercube sampling. Monte Carlo sampling is simple and universally applicable. Yet it suffers from its slow convergence of $\mathcal{O}(\frac{1}{\sqrt{M}})$ with the number of samples M. However, the convergence is independent of the number of parameters. Latin hypercube sampling provides optimal coverage of the parameter space by separating the distribution into bins and therefore yields faster convergence than Monte Carlo sampling. In addition, there are multiple other sampling schemes as well as specific sampling sequences.

For computationally demanding models or scenarios, evaluating the model at several parameter values may be too time-consuming. In this case, surrogate models can help. Instead of the model, the surrogate is evaluated. Model evaluations are only necessary for the generation of the surrogate. Spectral methods try to construct a surrogate model that is designed to propagate input parameter distributions. The surrogate is a polynomial expansion. Evaluations of the original model are only needed for the construction of the surrogate. A well-known spectral method is polynomial chaos expansion. Here, "chaos" does not relate its use for dynamical systems but rather means a stochastic process [Sullivan, 2015]. While polynomial chaos implies Gaussian random variables, generalized polynomial chaos is used for all parameter distributions.

There are three common approaches for calculating the coefficients of generalized polynomial chaos expansions: stochastic Galerkin, point collocation, and discrete projection.

We distinguish between intrusive and non-intrusive methods: For intrusive methods, the computerized model, the code, has to be adapted which is costly, complex, and model-specific. Non-intrusive methods, on the other hand, do not require changes to the model and are therefore more generally applicable. They might, however, require a longer computation time.

Stochastic Galerkin is an intrusive approach based on projection. From the model, a Galerkin system is derived. The resulting equations are often coupled. That means the coefficients have to be calculated from a system of coupled ordinary differential equations. Stochastic Galerkin requires orthogonal polynomials which means that we are limited regarding the distribution of the input parameters. For the common distributions, Table 8.1 lists the respective family of polynomials. The orthogonal polynomials can be computed using the three-term recurrence. Stochastic Galerkin provides higher accuracy than point collocation and discrete projection because there is no interpolation and integration error. However, its complexity increases considerably for multiple uncertain parameters. This method is commonly used for deterministic partial differential equations [Sullivan, 2015].

Table 8.1: Orthogonal polynomials for common probability distributions for the uncertain input parameters.

Distribution	Family of polynomials
Gaussian	Hermite
Gamma	Laguerre
Beta	Jacobi
Uniform	Legendre

Point collocation, or stochastic collocation, is a non-intrusive interpolation or regression scheme. It can be used for general parameter distributions [Smith, 2014, p. 225]. The approach decouples stochastic and deterministic components of the model. Depending on the number of samples, it requires either an interpolation or regression scheme. If the number of nodes is equal to the degree of the highest polynomial and therefore to the number of coefficients, we need interpolation to determine the coefficients of the expansion. Interpolation is not very robust, especially for a multi-dimensional input parameter space. In addition, the accuracy in between the nodes is difficult to assess and control [Xiu, 2017]. Moreover, interpolation suffers from Runge's phenomenon if the sample points are chosen equidistant [Smith, 2014, p. 251]. As a workaround, Chebyshev polynomials can be utilized. Regression alleviates the problems regarding the robustness of the interpolation especially when the samples of the solution, the model response, include noise [Xiu, 2017]. The prerequisite for the regression scheme is that the number of samples is larger than the number of coefficients. In this case, the system is overdetermined. The coefficients can be calculated using least squares. Typically, a linear oversampling $M \approx \alpha N$ with $\alpha \in [1.5, 3]$ is used for a polynomial of degree N [Xiu, 2017]. The nodes can be chosen by any sampling method, for example, Monte Carlo or quasi-Monte Carlo sampling. For point collocation in higher dimensions, sparse grids can be employed [Xiu, 2007]. Finally, discrete projection, or the pseudo-spectral approach, is a non-intrusive integration scheme. Discrete projection shares attributes of both stochastic Galerkin and point collocation: As for the Galerkin ansatz, discrete projection requires orthogonal polynomials. Like point collocation, discrete projection decouples stochastic and deterministic parts of the model response. It can be understood as a discrete version of generalized polynomial chaos expansions [Xiu, 2007].

All three approaches need to approximate integrals. For large parameter dimensions, stochastic quadrature techniques are optimal for the approximation. For lower dimensions, deterministic quadrature approaches can be employed [Smith, 2014, p. 239]. Instead of using polynomials for the expansion, expansion can also be performed using splines or radial basis functions, wavelets, or even non-smooth basis functions [Sullivan, 2015].

From the presented methods, sampling is suitable for crowd simulations because it provides a distribution of the quantity of interest. In addition, it works with posteriors obtained with Bayesian inference methods. Monte Carlo is a common approach, its only downside is the slow convergence of the method. For computationally demanding models

or scenarios, a non-intrusive polynomial chaos expansion is preferable. That concerns, in particular, simulations that cover large areas or include many agents. Both point collocation and discrete projection are non-intrusive. Only point collocation is, however, applicable to general parameter distributions.

8.3.1 Monte Carlo

For propagation with Monte Carlo, I randomly sample from the parameter distributions and evaluate the model at each sample. For each realization, I collect the model response. Typically, the histogram of the responses is visualized to analyze the distribution of the quantity of interest. In addition, mean, variance or covariance, and higher-order moments can be empirically calculated.

8.3.2 Generalized polynomial chaos expansion with point collocation

When using polynomial chaos, we construct a surrogate model for the model response using a low-order expansion. An expansion in general is an approximation of a function by a sum of other functions that are common, well-studied and therefore easier to handle. Polynomial chaos expansions describe a class of expansions which utilize polynomials as basis functions. Since the term "chaos" can be misleading, they are often also called spectral expansions. The model response $f(x)$ is represented by an infinite expansion [Smith, 2014, p. 208]

$$f(x) = f_0\hat{P}_0 + \sum_{i_1=1}^{\infty} f_{i_1}\hat{P}_1(Q_{i_1}) + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{\infty} f_{i_1,i_2}\hat{P}_2(Q_{i_1}, Q_{i_2}) \tag{8.1}$$

$$+ \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{\infty}\sum_{i_3=1}^{\infty} f_{i_1,i_2,i_3}\hat{P}_3(Q_{i_1}, Q_{i_2}, Q_{i_3}) + \dots$$

$$= \sum_{k=0}^{\infty} f_k\Psi_k(Q_1, Q_2, \dots) \tag{8.2}$$

with deterministic coefficients $f_0, f_{i_1}, f_{i_2}, \dots$ for the basis functions $\Psi_k(Q)$ with increasing interactions terms. $\Psi_k(Q)$ are orthogonal polynomials. When using only K terms, the expansion approximates the model response:

$$f(x) \approx \sum_{k=0}^{K} f_k\psi_k(Q_1, Q_2, \dots) \tag{8.3}$$

This expansion separates the deterministic and stochastic parts into the coefficients and basis functions, respectively. The orthogonal polynomials are chosen based on the distribution of the uncertain input parameters, compare Table 8.1. Point collocation, as well as stochastic Galerkin and discrete projection, are approaches to calculate the coefficients for the expansion. Point collocation is a non-intrusive method and provides high flexibility due to the regression approach. We solve the regression problem with

least-squares using $M \geq K+1$ samples from the parameter space, that are the collocation points q_m,

$$\min_{f_k} \parallel f(q_m) - \sum_{k=0}^{K} f_k \psi_k(q_m) \parallel . \tag{8.4}$$

Based on the expansion, the statistical moments of the model response can be evaluated [Smith, 2014, p. 209]:

$$\mathrm{E}[f(x)] \approx \mathrm{E}[f^K] = f_0(t) \tag{8.5}$$

$$\mathrm{Var}[f(x)] \approx \mathrm{Var}[f^K] = \sum_{k=1}^{K} f_k^2(t)\gamma_k \tag{8.6}$$

where $\gamma_k = \mathrm{E}[\Psi_k^2(Q)]$ is a normalization factor.

Especially when propagating multiple parameters, the collocation nodes need to be chosen carefully. Regularly sampled points can produce spurious oscillations and lead to ill-conditioned collocation matrices [Smith, 2014, p. 217]. In addition, Lagrange polynomials can be used to avoid large collocation matrices due to a high number of necessary collocation points.

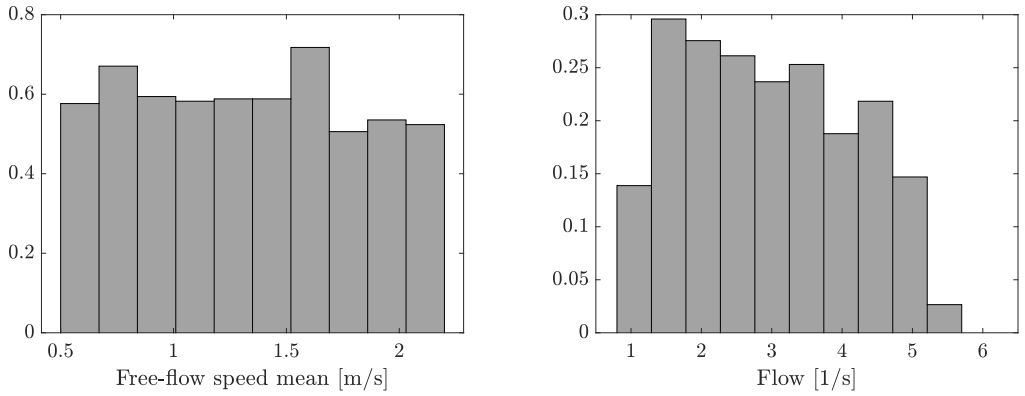
8.4 Studying impact of uncertain parameters on the prediction uncertainty

At first, I demonstrate Monte Carlo sampling for propagating one uncertain parameter in a single bottleneck scenario. In addition, I show how forward propagation can be performed with generalized polynomial chaos using the point collocation approach. After these rather didactic examples, I use forward propagation to quantify the uncertainty in the model response for the scenario with five bottlenecks. I do this at three stages of the process: (1) Before sensitivity analysis, using flat prior distributions for each parameter according to Section 3.2.1, (2) with factor fixing in which all non-influential parameters are fixed to arbitrary values within their prior ranges, and (3) using the joint posterior distribution obtained by calibrating the influential parameters.

8.4.1 Propagation with Monte Carlo sampling

First, we take a look at how propagation with Monte Carlo sampling works by evaluating the uncertainty in the model response for a single bottleneck. I choose the widest bottleneck (1.2 m) of the scenario with five bottlenecks. I propagate 1000 samples of the uniform distribution for the free-flow speed mean, $\mathcal{U}(0.5, 2.2)$, through the optimal steps model and evaluate the flow through the bottleneck. Figure 8.2 shows the histogram of the Monte Carlo samples at which the model was evaluated together with the histogram of the model responses. We observe that the flow values are not uniform but asymmetrically distributed. This is a consequence of a nonlinear relationship between free-flow speed mean and flow. The flow varies between $1\,\mathrm{s}^{-1}$ and $6\,\mathrm{s}^{-1}$. While this is

a large variation, it is not surprising because of the wide distribution of the free-flow speed mean.



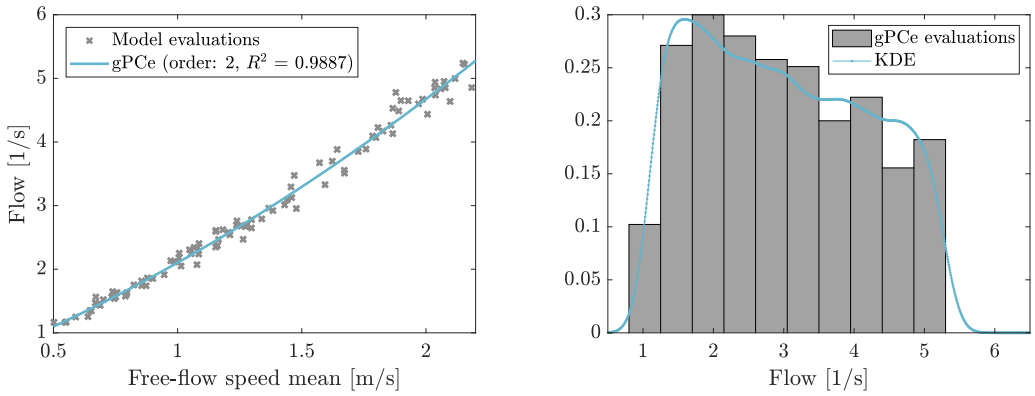
(a) Prior samples **(b)** Model response

Figure 8.2: Forward propagation of prior parameter interval for free-flow speed mean in the bottleneck scenario using Monte Carlo sampling.

8.4.2 Propagation with generalized polynomial chaos expansion

Second, we look into propagation using a generalized polynomial chaos expansion. I calculate an expansion of order 2 from 100 model evaluations without averaging repetitions. I determine the coefficients of the expansion using point collocation. That means I perform a regression over the flow values. The expansion nodes are generated by Monte Carlo sampling. The resulting expansion yields a coefficient of determination of $R^2 = 0.9887$. That means it fits the data well. Figure 8.3a shows the expansion for the flow, the quantity of interest.

Now, we can calculate statistical modes of the distribution of flow directly from the expansion. We are, however, also interested in the distribution of the flow values. In order to obtain the distribution of the flow values, I employ kernel density estimation (KDE). Typically, KDE is used to empirically estimate the probability density function for a random variable for a given set of realizations. We obtain a data set by evaluating the expansion at several points, here 10.000 points. This is cheap because it requires only evaluations of a polynomial. Then, I apply the Gaussian KDE from chaospy to estimate the probability density function of the quantity of interest, the flow. An important step in kernel density estimation is smoothing. An important parameter is the kernel bandwidth of the smoothing operator, often termed h or ϵ. In chaospy, the rules of thumb by Scott [Scott, 1992, p. 144] and Silverman [Silverman, 1986, p. 45ff] are used to choose the bandwidth. Figure 8.3b shows the probability density function of the flow obtained by the Gaussian KDE. The distribution of the flow values agrees with the results for Monte Carlo sampling implying that the expansion represents the model well.



(a) Generalized polynomial chaos expansion for the flow

(b) Kernel density estimation (KDE) for the probability density of the flow

Figure 8.3: Forward propagation of the free-flow speed mean in the bottleneck scenario to simulate the flow through the bottleneck using generalized polynomial chaos expansion (gPCe) by point collocation approach.

8.4.3 Measuring the reduction of uncertainty in the simulation output due to calibration

After the rather didactic examples of both Monte Carlo sampling and generalized polynomial chaos expansion for propagation, we look into the propagation for the scenario with five bottlenecks. We compare the flat priors with factor fixing and the joint posterior obtained from calibration. I choose Monte Carlo sampling for propagation because it allows us to utilize the posterior samples. Propagation using polynomial chaos expansion cannot make use of posterior samples.

8.4.3.1 Propagation of initial parameter intervals

At first, I take the flat prior used for the sensitivity analysis presented in Section 3.2.1 and propagate them through the model. Figure 8.4 illustrates the distribution of the flow for each bottleneck width. The obtained distribution is strongly asymmetric due to the nonlinear relationship between uncertain parameters and the quantity of interest. The probability mass lies mainly between the flow values of $0.5\,\mathrm{s}^{-1}$ and $2.5\,\mathrm{s}^{-1}$. As a measure for the variation in the posterior, we take a look at the standard deviation. In Table 8.2, we can see that the variation in the flow increases with increasing bottleneck width. The standard deviation is about one-third of the average flow for each width.

8.4.3.2 Propagation with factor fixing

In the next step, I employ factor fixing: I fix the parameters that were deemed non-influential in the sensitivity analysis, the number of agents, and the minimum step length, to arbitrary values within their ranges. Figure 8.5 displays the resulting distribution of

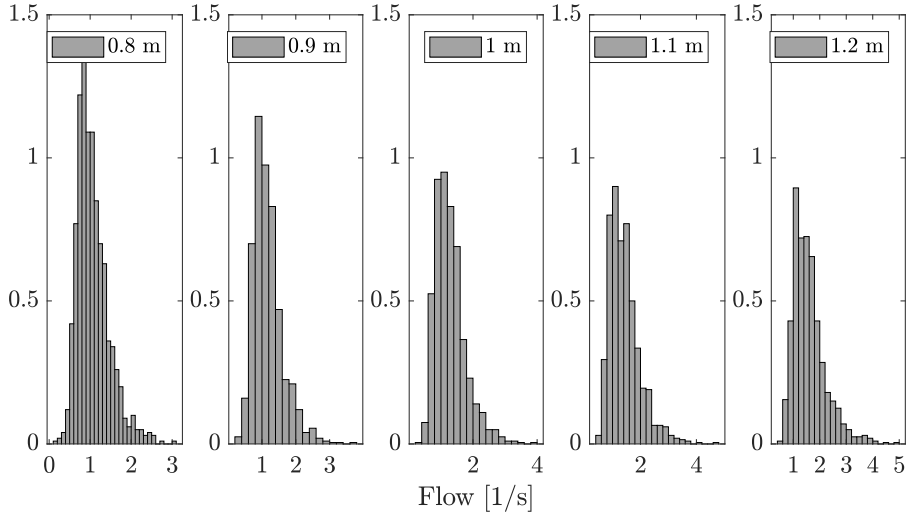


Figure 8.4: Histogram of flow values for each bottleneck width (from 0.8 m to 1.2 m) for propagating the uniform distributions for the uncertain parameters.

Table 8.2: Variation in flow after propagating initial flat parameter distributions. The coefficient of variation is the ratio of the standard deviation to the mean.

Bottleneck width [m]	0.8	0.9	1.0	1.1	1.2
Standard deviation of flow [1/s]	0.3885	0.4413	0.4819	0.5493	0.6146
Coefficient of variation	0.3649	0.3761	0.3745	0.3834	0.3853

the flow. From the standard deviation in Table 8.3, we can see that fixing the non-influential parameters has no impact on the propagation. This is what I expected. However, I vary fewer parameters with factor fixing and therefore need fewer evaluations.

Table 8.3: Variation in flow after propagating parameters not affected by factor fixing.

Bottleneck width [m]	0.8	0.9	1.0	1.1	1.2
Standard deviation of flow [1/s]	0.3641	0.4057	0.4569	0.5166	0.5715
Coefficient of variation	0.3454	0.3468	0.3562	0.3620	0.3590

8.4.3.3 Propagation of posterior distribution obtained with Bayesian inference

Even after calibration, there is still uncertainty in the output due to uncertainty in the data used for calibration, uncertainties in the estimation process, and stochasticity in the simulator when propagating samples. When we perform a simulation with a calibrated model, this uncertainty must be quantified. I analyze the specific model for the bottleneck scenario. Therefore, I keep non-influential parameters fixed. The influential parameters are drawn from the joint posterior distribution obtained in Section

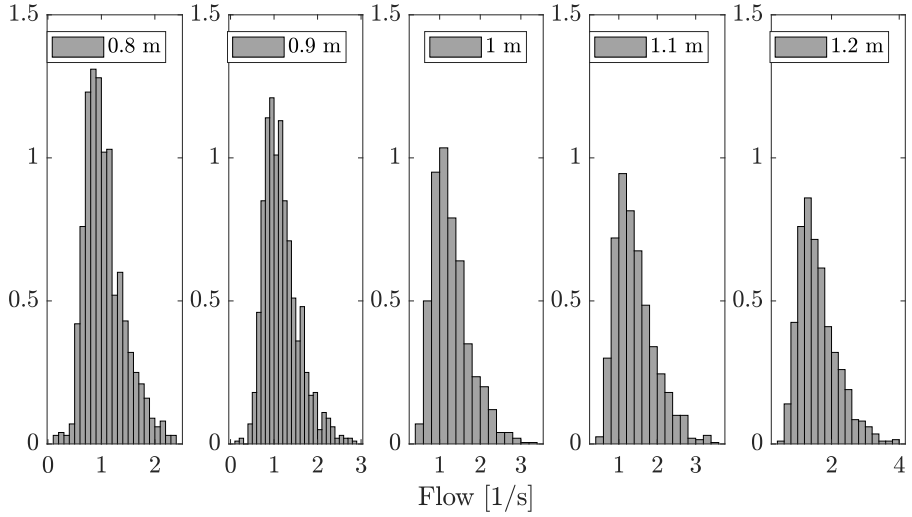


Figure 8.5: Histogram of flow values for each bottleneck width for propagating the flat priors on the influential parameters, non-influential parameters were fixed to defaults (factor fixing).

6.4.3.3. When propagating this model, we obtain the flow distribution illustrated in Figure 8.6. The width of the posterior is considerably reduced compared to the previous propagations. The probability mass lies between $1.25\,\mathrm{s}^{-1}$ and $2.5\,\mathrm{s}^{-1}$. When we take a look at the standard deviation, in Table 8.4, we observe a considerable reduction of the uncertainty in the propagation by a factor of about 3.3. In the application, this makes a major difference to, say, the designer of an evacuation route who has to judge the capacity of the doors in order to allow for a certain flow.

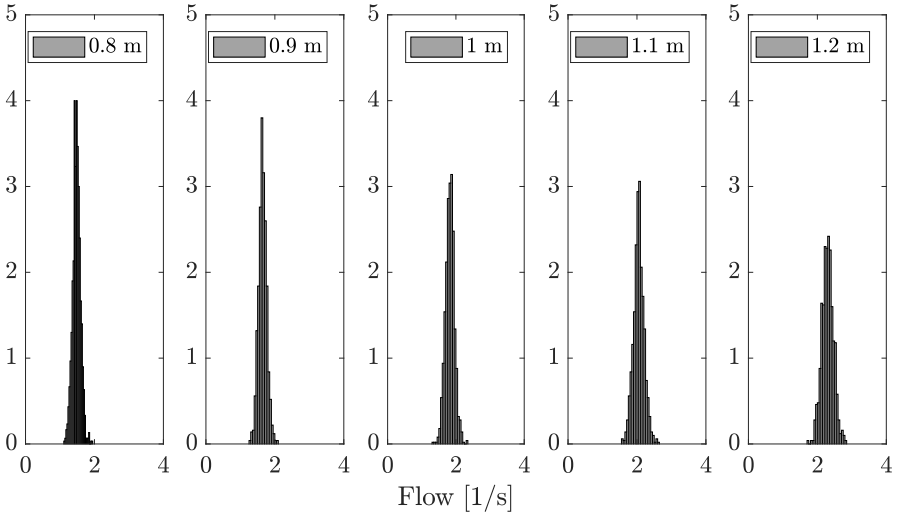


Figure 8.6: Histogram of flow values for each bottleneck width for propagating the joint posterior for free-flow speed mean, free-flow speed standard deviation, personal space strength, and obstacle repulsion obtained with Bayesian inference. Non-influential parameters are fixed.

Table 8.4: Variation in flow when propagating the joint posterior for free-flow speed mean, free-flow speed standard deviation, personal space strength, and obstacle repulsion.

Bottleneck width [m]	0.8	0.9	1.0	1.1	1.2
Standard deviation of flow [1/s]	0.1137	0.1192	0.1307	0.1558	0.1692
Coefficient of variation	0.0767	0.0725	0.0716	0.0758	0.0736

8.5 Summary

This chapter focused on answering the research question *Q3: "How can we quantify the uncertainty in the prediction for the bottleneck scenario?"*. I answered two sub-questions:

Q3.1: Which uncertainty analysis methods are suited for crowd simulations?
I presented two methods for uncertainty analysis that are suitable for crowd simulations: Monte Carlo sampling and a generalized polynomial chaos expansion based on point collocation. The former works with all parameter distributions, even with posterior distributions obtained with Bayesian inference. The latter performs well when simulating scenarios with high computational demand due to a large topography or many agents. I illustrated how the methods work on a didactic example. I propagate the free-flow speed according to the prior distribution from Section 3.2.1 and evaluated the quantity of interest, the flow.

Q3.2 How large is the uncertainty in the prediction for the bottleneck scenario before and after calibrating influential parameters?

I applied Monte Carlo propagation to study the uncertainty before and after calibration. First, I propagated the flat priors for all parameters and evaluated the uncertainty in the flow. Second, I performed factor fixing which means I fixed the non-influential parameters to arbitrary values within their ranges. As expected, this does not affect the output uncertainty. It does, however, reduce the input parameter space. Finally, I used the four-dimensional joint posterior distribution obtained from Bayesian inference in Section 6.4.3.3 for propagation. The results showed that the uncertainty in the output was reduced to less than a third of the original output uncertainty into a range that makes, e. g., the evaluation of escape routes much clearer. As a measure for the uncertainty in the response, I used the standard deviation of the distribution of the flow values. My results demonstrate how a careful identification and estimation of parameters can considerably reduce the uncertainty in the prediction. Even after calibration, residual uncertainty is present in the response. Quantification of uncertainty in the simulation output is therefore also necessary when predicting crowd behavior with calibrated models.

9 Summary, conclusions, and future directions

In this final chapter of my thesis, I briefly summarize the work presented in the previous chapters and review what has been accomplished (Section 9.1). Then, I draw conclusions (Section 9.2) and give an outlook on future work (Section 9.3).

9.1 Summary

The thesis started with a motivation for uncertainty analysis for crowd dynamics: For simulations to contribute to pedestrian safety, we must know how reliable they are. We need to identify and consider uncertainties in the modeling and simulation process to quantify and reduce the uncertainty in the prediction.

In Chapter 2, I discussed the modeling of crowd behavior based on the modeling cycle. Two essential steps are verification and validation. Uncertainty quantification can be understood as a part of the validation. It needs to be included in all steps of modeling and simulation to assess and improve the reliability of predictions. I described different types of uncertainty and explained how they are introduced in the modeling process. In this thesis, I focused on uncertainty in parameters due to lack of knowledge, which is an epistemic uncertainty.

In the next chapter, Chapter 3, I introduced the models and scenarios that I examine. Throughout the thesis, I analyzed a bottleneck scenario. Since constrictions can lead to high densities and delays in evacuations, studying bottlenecks scenarios is crucial for pedestrian safety. In addition, I investigated the multi-directional flow in a train station overpass. With real-world trajectory data provided by Swiss Federal Railways (SBB), I conducted a feasibility study on obtaining origin-destination matrices for dynamic initialization of the simulation. After introducing the scenarios, I described two locomotion models, the optimal steps model and an emulator for the social force model. I used both to simulate the flow through a bottleneck scenario. In addition, I briefly introduced the crowd simulation framework Vadere with which I performed the simulations. Finally, I described two effects observed in the simulation outcomes, stochasticity and noise. I explained why and where stochastic terms are introduced in the simulation and outlined how I handle them.

Chapter 4 was dedicated to finding suitable software to perform uncertainty quantification for crowd simulations. I derived the functional (FR) and non-functional requirements (NFR) on such software. Then, I presented state-of-the-art software frameworks and their shortcomings with respect to these requirements. While there are several

frameworks available, none of them fulfills all mandatory requirements. Therefore, I decided to design and implement my own framework. The architecture of the framework assures modularity (NRF1). More precisely, I employ the strategy pattern to separate algorithms from infrastructure so that the framework is easily expandable. For increased readability, the framework follows the Python coding style, PEP 8 (NFR2). The framework provides routines for parameter identification (FR3), parameter estimation (FR4), and uncertainty analysis (FR5). The methods are adapted to crowd simulations and can be applied to all models implemented in Vadere (FR2). They are carefully tested by unit tests which are carried out in a continuous-integration pipeline (NFR4). Existing interfaces such as the suq-controller to Vadere are supported (FR1). The results are reproducible (FR6) and can be stored to file (FR7). The resulting framework is provided open-source (NFR3), and I used it throughout the thesis.

In Chapter 5, I presented methods from uncertainty quantification to identify influential parameters and to rank them by their impact on the simulation outcome. I chose two methods that are particularly suitable for crowd simulations: Sobol' indices and activity scores. I calculated both measures for the flow through the bottleneck. As a byproduct of the active subspace method, I also obtained the first eigenvector metric and the derivative-based global sensitivity metrics. All metrics rank the parameters in the bottleneck scenario consistently: The most influential parameter is the free-flow speed mean, followed by the free-flow speed standard deviation, the obstacle repulsion, and the personal space strength. The individual free-flow speed of each agent is the speed which with it moves unhindered through a space. It is drawn from a truncated normal distribution. The parameters obstacle repulsion and personal space strength ensure that agents keep a natural distance to each other and to walls. The analysis also showed that interactions are only present between free-flow speed mean and standard deviation. Two parameters, the minimum step length, and the number of agents were deemed non-influential. For subsequent studies, these parameters can be fixed to an arbitrary value within their range. In addition to ranking the individual parameters, the active subspace method also found a one-dimensional subspace of the input parameter space from which a reduced-order model can be derived.

The next chapter, Chapter 6, focused on the calibration of the influential parameters in the bottleneck scenario. So far, calibration in crowd dynamics has mainly been performed through point estimates such as maximum likelihood estimates. In contrast, Bayesian inference provides full posterior distributions which consider residual uncertainty after calibration. I identified two approaches for Bayesian inference that are suitable for crowd simulation: the likelihood-based random walk Metropolis algorithm and the likelihood-free approximate Bayesian computation. I successfully performed a proof-of-concept with artificial data for both methods. Then, I compared approximate Bayesian computation as a representative for Bayesian inference to a point estimate, the posterior mean, in three case studies. The results highlighted that point estimates can be sufficient when the posterior distribution is symmetric and unimodal. However, when a non-symmetric or multi-modal posterior is present, the full posterior distribution needs to be considered for subsequent studies. It can only be provided by Bayesian inference. For multi-dimensional calibration, the joint posterior distribution contains additional

information about the parameters and the calibration process: For the calibration of the four influential parameters, the shape of the univariate posterior distribution showed that the free-flow speed mean, the obstacle repulsion, and the personal space strength were well informed by the dataset employed for calibration. The free-flow speed standard deviation, however, was not informed to the same degree. Additionally, the posterior revealed correlations between the free-flow speed mean and each other parameter concerning the experimental data.

In Chapter 7, I tackled the challenge of a dynamic initialization for real-time simulations of crowds during events. The origins and destinations of the agents are crucial for this. Origin-destination matrices describe the popularity of combinations of origins and destinations. I conducted a feasibility study on whether we can predict origin-destination matrices for the simulation from live sensor data in form of density heatmaps. These heatmaps can be reliably and quickly obtained from a variety of sensors. For this study, Swiss Federal Railways (SBB) provided real-world trajectory data from the train station overpass obtained from stereo sensors. I employed two statistical learning models for the estimation: a multivariate linear regression model and random forest. Both models are simple, robust, and explainable. The linear regression successfully predicts the OD matrices. To address the nonlinearities in the input-output relationship, I applied a random forest model which captured the shape of the output space better but overall performed slightly worse in terms of R^2 score. My results show that both learning models successfully predict origin-destination matrices from density heatmaps.

The last chapter, Chapter 8, was dedicated to quantifying the uncertainty in the prediction for the bottleneck scenario after calibration. I presented two methods for uncertainty analysis that are suitable for crowd simulations: Monte Carlo sampling and a generalized polynomial chaos expansion based on point collocation. The former works with all parameter distributions, even with posterior distributions obtained with Bayesian inference. The latter is in particular well suited for scenarios with high computational demand, e. g. when the topography is large or when there are many agents. I applied Monte Carlo propagation to compare the uncertainty before and after calibration. I propagated the prior parameter distributions and the four-dimensional joint posterior distribution obtained from Bayesian inference. The results showed that the uncertainty in the output was reduced to less than a third of the original output uncertainty.

9.2 Conclusions

In this thesis, I investigated the impact of parameter uncertainty on the simulation results and proposed a three-step approach to form a specific model with reduced uncertainty: (1) identification of influential and non-influential parameters, (2) calibration of influential parameters, (3) quantification of the uncertainty in the simulation output for the specific model. Throughout the thesis, I investigated an essential dynamic in all ingress and egress scenarios: the flow through a bottleneck. My results show that calibration of the influential parameter considerably reduced the uncertainty in the prediction. This is crucial for a reliable simulation.

In particular, I addressed one requirement and answered four research questions:

R1: Choose or design a software for uncertainty quantification for crowd simulations I defined the functional and non-functional requirements for an uncertainty quantification software for crowd simulations. Since none of the available frameworks fulfill all requirements, I designed and implemented my own framework. It provides implementations of methods for parameter identification, parameter estimation, and uncertainty analysis adapted to crowd simulations. I designed the framework in a modular way by employing the strategy pattern so that algorithms are separated from infrastructure. The framework is the basis for all studies that I performed throughout this dissertation. Moreover, it is available open-source for the scientific community to perform uncertainty quantification studies for crowd simulation. This closes a gap and I expect it to promote the use of uncertainty quantification methods in crowd dynamics.

Q1: How can we identify influential parameters in the optimal steps model for the bottleneck scenario? I identified Sobol' indices and activity scores as suitable metrics for parameter ranking in crowd simulations. For the bottleneck scenario, both metrics ranked the parameters consistently: The most influential parameter is the free-flow speed mean, followed by the free-flow speed standard deviation, the obstacle repulsion, and the personal space strength. We need to concentrate our efforts on these parameters (factor prioritization). The analysis revealed that interactions are only present between free-flow speed mean and standard deviation. Two parameters, the minimum step length, and the number of agents were deemed non-influential. These parameters are fixed to arbitrary values within their range for subsequent studies (factor fixing). This is helpful in practice because it reduces the number of model evaluations necessary for calibration and forward propagation. Moreover, the active subspace method which provides the activity scores revealed a one-dimensional active subspace in the input parameter space. That means the model response varies primarily along a linear combination of the input parameters. Especially for computationally expensive scenarios, a reduced-order model based on the active subspace allows otherwise infeasible parameter studies.

Q2: How can we calibrate the influential parameters in the bottleneck scenario? So far, calibration of crowd dynamics models is typically carried out with point estimates such as maximum likelihood estimates. I proposed Bayesian inference methods for calibration which provide a full posterior distribution of the parameters instead of a single value. The posterior reflects the remaining uncertainty after calibration due to measurement noise in the data and model error. This uncertainty has to be considered for subsequent studies. In three case studies, I compared a Bayesian inference method, approximate Bayesian computation, to point estimation. While the point estimate appeared sufficient when the posterior is unimodal and symmetric, it fails when the posterior is multimodal. In the case study, this lead to miscalculated egress times at a bottleneck. When one plans large events, this can endanger pedestrian safety. This example also exposed a need for caution when transferring calibrated parameters from one safety scenario to another. When multiple parameters are calibrated, the posterior distribution provides insight into parameter correlations with respect to the data and how well each parameter is informed by the data. High correlations among parameters highlight the need to infer all parameters simultaneously rather than calibrating them separately.

Q2: Can we predict origin-destination matrices for the initialization of origins and destinations in the simulation from live sensor data in form of density heatmaps?* One crucial step towards real-time simulations is the online initialization of the simulation. For this, initial and boundary conditions need to be continuously estimated. Origin-destination (OD) matrices contain all information regarding the agents' sources and destinations. These are mandatory for the simulation. I performed a case study in which I estimated dynamic OD matrices from density heatmaps. Density heatmaps can nowadays be automatically extracted from sensor data. For this study, Swiss Federal Railways (SBB) provided trajectory data obtained with stereo sensors in the overpass of the Basel train station. I proposed two statistical learning models, multivariate linear regression and random forest for the estimation because they are both easy to understand, robust, and explainable. Both models successfully predicted the OD matrices. However, there is still room for improvement in terms of accuracy. I consider this a promising direction for new research. Altogether, the case study showed how classical parameter estimation can be complemented with new methods when it comes to real-time simulations.

Q3: How can we quantify the uncertainty in the prediction for the bottleneck scenario? For the bottleneck scenario, I applied Monte Carlo propagation to measure the effect of the calibration of the influential parameters: First, I propagated the prior parameter distribution through the model. Then, I employed the specific model defined by the joint posterior distribution obtained from calibration and by fixing non-influential parameters and performed propagation. The uncertainty in the prediction of the specific model was reduced to less than a third compared to the original parameter intervals into a range that makes, e. g., the evaluation of escape routes much clearer. My results demonstrate how a careful identification and estimation of parameters can significantly reduce the uncertainty in the prediction. They also highlight the importance of uncertainty analysis because they showed that there is still uncertainty in the simulation output present even after careful calibration.

9.3 Future directions

Real-time predictions are an emerging challenge in crowd dynamics. They are an essential step towards decision support systems. The three-step approach of this thesis aims at beforehand studies. It is computationally expensive and therefore not necessarily suitable for real-time predictions. It helps however to focus on influential parameters when the quantities of interests are known beforehand. However, the estimation of parameters and initial and boundary conditions based on sensor data as well as propagation must be performed in real-time. Future work must therefore deal with the identification and adaption of efficient methods that ideally run continuously alongside the prediction throughout an event. Along these lines, Chapter 7 focused on the estimation of initial and boundary conditions for the dynamic initialization of a simulation. My results encourage spending more time on the identification of a suitable nonlinear statistical learning model that represents the data better.

Crowd simulations are being used for ever larger and computationally more demanding scenarios. To speed up their calibration, Bayesian inference can be carried out in an active subspace instead of the full input parameter space. This reduces the effective number of parameters to calibrate. I described the approach in Section 6.4.4. One central step is the construction of a so-called ridge approximation, a surrogate model based on the active subspace. In order to make this approach available to the scientific community, the construction of the surrogate model needs to be implemented and a proof-of-concept should be performed.

Dynamic surrogate models for crowd dynamics models could address two challenges: non-continuous parameter types in conventional models and the comparison of reality to simulation. In a surrogate model, all parameter types can be sampled continuously, independent of the parameter type in the original model. Surrogate models based on the Koopman operator rely on the eigenfunctions of the system that is to be emulated. These eigenfunctions can be used to compare simulations and real data. Such a comparison serves as a basis for parameter estimation, which requires that the model and the data represent the same system. In addition, surrogate models can be evaluated cheaply. This allows parameter studies for large scenarios with many agents. Hence, I consider surrogate models a promising approach to perform parameter studies for models with non-continuous parameter types to compare underlying dynamics between observations and simulations.

This thesis focused on the influence of parameter uncertainties on the prediction. However, beyond that, there are other uncertainties and errors in the process. One is model error, which is the discrepancy between model and reality. It is well known that a model is a simplified representation of reality and that this simplification introduces errors. However, the size and distribution of the errors are usually unknown. For a complete analysis of uncertainty, this uncertainty needs to be estimated. This certainly is a gigantic task. Especially since observation of crowds is not comparable to measuring a device. Nevertheless, modelers should always stay aware of it.

Parameter studies such as those presented in this thesis are limited by the nature and structure of the models. Ideally, the models have continuous input parameters and there is a continuous relationship between input variables and simulation results. When designing new models, parsimony and comprehensibility are often emphasized, and rightly so. However, one should also make sure that the models can be sufficiently analyzed to increase the reliability of the simulations and thus enhance the confidence in the simulations. In particular, heuristic models often violate the assumptions that are necessary for parameter studies due to numerous yes-no decisions and true-false parameters. As a consequence, approaches such as the active subspace method that relies on gradients, cannot be applied to these models. I consider the design of a new model for crowd dynamics that are easy to analyze as a challenging continuation of the ideas presented in this work.

Bibliography

Brian M. Adams, William J. Bohnhoff, Keith R. Dalbey, Mohamed S. Ebeida, John P. Eddy, Michael S. Eldred, Gianluca Geraci, Russell W. Hooper, Patricia D. Hough, Kenneth T. Hu, John D. Jakeman, Mohammad Khalil, Jason A. Maupin, Kathryn A. Monschke, Elliott M. Ridgway, Ahmad A. Rushdi, J.A. Stephens, Laura P. Swiler, Dena M. Vigil, Timothy M. Wildey, and Justin G. Winokur. Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.11 User's manual. Technical report, Sandia National Laboratories, 2014. URL `https://dakota.sandia.gov/sites/default/files/docs/6.11/Users-6.11.0.pdf`.

Juliane Adrian, Nikolai Bode, Martyn Amos, Mitra Baratchi, Mira Beermann, Maik Boltes, Alessandro Corbetta, Guillaume Dezecache, John Drury, Zhijian Fu, Roland Geraerts, Steve Gwynne, Gesine Hofinger, Aoife Hunt, Tinus Kanters, Angelika Kneidl, Krisztina Konya, Gerta Köster, Mira Küpper, Georgios Michalareas, Fergus Neville, Evangelos Ntontis, Stephen Reicher, Enrico Ronchi, Andreas Schadschneider, Armin Seyfried, Alastair Shipman, Anna Sieben, Michael Spearpoint, Gavin Brent Sullivan, Anne Templeton, Federico Toschi, Zeynep Yücel, Francesco Zanlungo, Iker Zuriguel, Natalie van der Wal, Frank van Schadewijk, Cornelia von Krüchten, and Nanda Wijermans. A glossary for research on human crowd dynamics. *Collective Dynamics*, 2019. doi:10.17815/CD.2019.19.

Yoongho Ahn, Tomoya Kowada, Hiroshi Tsukaguchi, and Upali Vandebona. Estimation of passenger flow for planning and management of railway stations. *Transportation Research Procedia*, 25:315–330, 2017. doi:10.1016/j.trpro.2017.05.408. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016.

Amani A. Alahmadi, Jennifer A. Flegg, Davis G. Cochrane, Christopher C. Drovandi, and Jonathan M. Keith. A comparison of approximate versus exact techniques for Bayesian parameter inference in nonlinear ordinary differential equation models. *Royal Society Open Science*, 7(3):191315, 2020. doi:10.1098/rsos.191315.

American Society of Mechanical Engineers. *Guide for Verification and Validation in Computational Solid Mechanics*. ASME V&V. American Society Of Mechanical Engineers (ASME), 2006. ISBN 9780791873168.

Gianluca Antonini, Michel Bierlaire, and Mats Weber. Discrete choice models of pedestrian walking behavior. *Transportation Research Part B: Methodological*, 40(8):667–687, 2006. doi:10.1016/j.trb.2005.09.006.

Jason D. Averill. Five grand challenges in pedestrian and evacuation dynamics. In Richard D. Peacock, Erica D. Kuligowski, and Jason D. Averill, editors, *Pedestrian and Evacuation Dynamics*, pages 1–11, Boston, MA, 2011. Springer US. doi:10.1007/978-1-4419-9725-8_1.

Gabriel Baglietto and Daniel R. Parisi. Continuous-space automaton model for pedestrian dynamics. *Physical Review E*, 83(5):056117, 2011. doi:10.1103/PhysRevE.83.056117.

Muhammad Baqui and Rainald Löhner. Towards real-time monitoring of the hajj. In *Proceedings from the 9th International Conference on Pedestrian and Evacuation Dynamics*, pages 394–402, 2020. doi:10.17815/CD.2020.75.

Chris P. Barnes, Sarah Filippi, Michael P. H. Stumpf, and Thomas Thorne. Considerate approaches to constructing summary statistics for abc model selection. *Statistics and Computing*, 22(6):1181–1197, 11 2012. doi:10.1007/s11222-012-9335-7.

Michaël Baudin, Anne Dutfoy, Bertrand Iooss, and Anne-Laure Popelin. *OpenTURNS: An Industrial Software for Uncertainty Quantification in Simulation*, pages 2001–2038. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-12385-1_64.

Dietmar Bauer. Estimating origin-destination-matrices depending on the time of the day from high frequent pedestrian entry and exit counts. *IET Intelligent Transport Systems*, 6(4):463–473, 2012. doi:10.1049/iet-its.2011.0156.

Mark A. Beaumont, Wenyang Zhang, and David J. Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002. ISSN 0016-6731. URL `https://www.genetics.org/content/162/4/2025`.

Mario Bebendorf. A Note on the Poincaré Inequality for Convex Domains. *Zeitschrift für Analysis und ihre Anwendungen*, 22(4):751–756, 2003. doi:10.4171/ZAA/1170.

Michael G.H. Bell. The estimation of origin-destination matrices by constrained generalised least squares. *Transportation Research Part B: Methodological*, 25(1):13–22, 1991. doi:10.1016/0191-2615(91)90010-G.

Nicola Bellomo and Abdelghani Bellouquid. On multiscale models of pedestrian crowds from mesoscopic to macroscopic. *Communications in Mathematical Sciences*, 13(7): 1649–1664, 2015. doi:10.4310/cms.2015.v13.n7.a1.

Jean L. Berrou, Jonathan Beecham, Philippe Quaglia, Marios A. Kagarlis, and Alex Gerodimos. Calibration and validation of the legion simulation model using empirical data. In Nathalie Waldau, Peter Gattermann, Hermann Knoflacher, and Michael Schreckenberg, editors, *Pedestrian and Evacuation Dynamics 2005*, pages 167–181, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. doi:10.1007/978-3-540-47064-9_15.

Bert Bettonvil and Jack P.C. Kleijnen. Searching for important factors in simulation models with many factors: Sequential bifurcation. *European Journal of Operational Research*, 96(1):180–194, 1997. doi:10.1016/S0377-2217(96)00156-7.

Carlo Bianca and Caterina Mogno. A thermostatted kinetic theory model for event-driven pedestrian dynamics. *The European Physical Journal Plus*, 133, 2018. doi:10.1140/epjp/i2018-12055-5.

Daniel H. Biedermann, Carolin Torchiani, Peter M. Kielar, David Willems, Oliver Handel, Stefan Ruzika, and André Borrmann. A hybrid and multiscale approach to model and simulate mobility in the context of public events. *Transportation Research Procedia*, 19:350–363, 2016. doi:10.1016/j.trpro.2016.12.094. Transforming Urban Mobility. mobil.TUM 2016. International Scientific Conference on Mobility and Transport.

Jean-Baptiste Blanchard, Guillaume Damblin, Jean-Marc Martinez, Gilles Arnaud, and Fabrice Gaudier. The Uranie platform: an open-source software for optimisation, meta-modelling and uncertainty analysis. *EPJ Nuclear Sci. Technol.*, 5(4):1–32, 2019. doi:10.1051/epjn/2018050.

Michael G.B. Blum. Choosing the summary statistics and the acceptance rate in approximate Bayesian computation. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 47–56, Heidelberg, 2010. Physica-Verlag HD. doi:10.1007/978-3-7908-2604-3_4.

Eric Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7280–7287, 2002. doi:10.1073/pnas.082080899.

Emanuele Borgonovo. A new uncertainty importance measure. *Reliability Engineering & System Safety*, 92(6):771–784, 2007. doi:10.1016/j.ress.2006.04.015.

Emanuele Borgonovo and George E. Apostolakis. A new importance measure for risk-informed decision making. *Reliability Engineering & System Safety*, 72(2):193–212, 2001. doi:10.1016/S0951-8320(00)00108-3.

Emanuele Borgonovo and Elmar Plischke. Sensitivity analysis: A review of recent advances. *European Journal of Operational Research*, 248(3):869–887, 2016. doi:10.1016/j.ejor.2015.06.032.

André Borrmann, Angelika Kneidl, Gerta Köster, Stefan Ruzika, and Markus Thiemann. Bidirectional coupling of macroscopic and microscopic pedestrian evacuation models. *Safety Science*, 50:1695–1703, 2012. doi:10.1016/j.ssci.2011.12.021.

Ernst Bosina. *A New Generic Approach to the Pedestrian Fundamental Diagram*. PhD thesis, Institut für Verkehrsplanung und Transportsysteme (IVT), ETH Zürich, 2018. URL `https://doi.org/10.3929/ethz-b-000296226`.

Ernst Bosina and Ulrich Weidmann. Estimating pedestrian speed using aggregated literature data. *Physica A: Statistical Mechanics and its Applications*, 468:1–29, 2017. doi:10.1016/j.physa.2016.09.044.

Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and Regression Trees*. Chapman & Hall, London, 1st edition, 1984. ISBN 9780412048418.

Hans-Joachim Bungartz, Stefan Zimmer, Martin Buchholz, and Dirk Pflüger. *Modeling and Simulation: An Application-Oriented Introduction*. Springer Undergraduate Texts in Mathematics and Technology. Springer, Berlin Heidelberg, 2014. doi:10.1007/978-3-642-39524-6.

Tom Burr and Alexei Skurikhin. Selecting summary statistics in approximate Bayesian computation for calibrating stochastic models. *BioMed Research International*, 2013: 210646, 2013. doi:10.1155/2013/210646.

Mario C. Campanella, Serge P. Hoogendoorn, and Winnie Daamen. A methodology to calibrate pedestrian walker models using multiple-objectives. In Richard D. Peacock, Erica D. Kuligowski, and Jason D. Averill, editors, *Pedestrian and Evacuation Dynamics*, pages 755–759, Boston, MA, 2011. Springer US. doi:10.1007/978-1-4419-9725-8_69.

Ennio Cascetta. Estimation of trip matrices from traffic counts and survey data: A generalized least squares estimator. *Transportation Research Part B: Methodological*, 18(4):289–299, 1984. doi:10.1016/0191-2615(84)90012-2.

K.S. Chan, William H.K. Lam, L.Q. Ouyang, and S.C. Wong. Simultaneous estimation of the pedestrian origin-destination matrix and parameter of the activity/destination choice model. *Journal of the Eastern Asia Society for Transportation Studies*, 6:1760–1773, 2007. doi:10.11175/easts.7.1760.

Jiayan Chen, Jia Yu, Jiahong Wen, Chuanrong Zhang, Zhan'e Yin, Jianping Wu, and Shenjun Yao. Pre-evacuation time estimation based emergency evacuation simulation in urban residential communities. *International Journal of Environmental Research and Public Health*, 16(23):4599, 11 2019. doi:10.3390/ijerph16234599.

Xu Chen, Martin Treiber, Venkatesan Kanagaraj, and Haiying Li. Social force models for pedestrian traffic – state of the art. *Transport Reviews*, 38(5):625–653, 2018. doi:10.1080/01441647.2017.1396265.

Mohcine Chraibi, Armin Seyfried, and Andreas Schadschneider. Generalized centrifugal-force model for pedestrian dynamics. *Physical Review E*, 82(4):046111, 2010. doi:10.1103/PhysRevE.82.046111.

Chih-Yuan Chu. A computer model for selecting facility evacuation design using cellular automata. *Computer-Aided Civil and Infrastructure Engineering*, 24(8):608–622, 2009. doi:10.1111/j.1467-8667.2009.00619.x.

Andrew Collins, Terra Elzie, Erika Frydenlund, and R. Michael Robinson. Do groups matter? An agent-based modeling approach to pedestrian egress. *Transportation Research Procedia*, 2:430–435, 2014. doi:10.1016/j.trpro.2014.09.051. The Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014).

Annachiara Colombi, Marco Scianna, and Alessandro Alaia. A discrete mathematical model for the dynamics of a crowd of gazing pedestrians with and without an evolving environmental awareness. *Computational and Applied Mathematics*, 36:1113–1141, 2017. doi:10.1007/s40314-016-0316-x.

Paul G. Constantine. *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies*. Society for Industrial and Applied Mathematics, 2015. doi:10.1137/1.9781611973860.

Paul G. Constantine and Paul Diaz. Global sensitivity metrics from active subspaces. *Reliability Engineering & System Safety*, 162:1–13, 2017. doi:10.1016/j.ress.2017.01.013.

Paul G. Constantine, Carson Kent, and Tan Bui-Thanh. Accelerating Markov Chain Monte Carlo with active subspaces. *SIAM Journal on Scientific Computing*, 38(5): A2779–A2805, 2016. doi:10.1137/15M1042127.

Winnie Daamen and Serge P. Hoogendoorn. Calibration of pedestrian simulation model for emergency doors by pedestrian type. *Transportation Research Record*, 2316(1): 69–75, 2012. doi:10.3141/2316-08.

Winnie Daamen, Mario Campanella, and Serge P. Hoogendoorn. Calibration of Nomad parameters using empirical data. In Valery V. Kozlov, Alexander P. Buslaev, Alexander S. Bugaev, Marina V. Yashina, Andreas Schadschneider, and Michael Schreckenberg, editors, *Traffic and Granular Flow '11*, pages 109–120, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-39669-4_11.

Maria Davidich and Gerta Köster. Towards automatic and robust adjustment of human behavioral parameters in a pedestrian stream model to measured data. *Safety Science*, 50(5):1253–1260, 2012. doi:10.1016/j.ssci.2011.12.024.

Maria Davidich and Gerta Köster. Predicting pedestrian flow: A methodology and a proof of concept based on real-life data. *PLoS ONE*, 8(12):1–11, 12 2013. doi:10.1371/journal.pone.0083355.

Etienne de Rocquigny, Nicolas Devictor, and Stefano Tarantola, editors. *Uncertainty in Industrial Practice*. John Wiley & Sons, Ltd, 2008. doi:10.1002/9780470770733.

Bert Debusschere, Khachik Sargsyan, Cosmin Safta, and Kenny Chowdhary. The uncertainty quantification toolkit (UQTk). In Roger Ghanem, David Higdon, and Houman Owhadi, editors, *Handbook of Uncertainty Quantification*, pages 1807–1827. Springer International Publishing, 2017. doi:10.1007/978-3-319-12385-1_56.

Charitha Dias and Ruggiero Lovreglio. Calibrating cellular automaton models for pedestrians walking through corners. *Physics Letters A*, 382(19):1255–1261, 2018. doi:10.1016/j.physleta.2018.03.022.

Charitha Dias, Miho Iryo-Asano, Hiroaki Nishiuchi, and Tomoyuki Todoroki. Calibrating a social force based model for simulating personal mobility vehicles and pedestrian mixed traffic. *Simulation Modelling Practice and Theory*, 87:395–411, 2018. doi:10.1016/j.simpat.2018.08.002.

Felix Dietrich and Gerta Köster. Gradient navigation model for pedestrian dynamics. *Physical Review E*, 89(6):062801, 2014. doi:10.1103/PhysRevE.89.062801.

Felix Dietrich, Florian Künzner, Tobias Neckel, Gerta Köster, and Hans-Joachim Bungartz. Fast and flexible uncertainty quantification through a data-driven surrogate model. *International Journal for Uncertainty Quantification*, 8:175–192, 2018. doi:10.1615/Int.J.UncertaintyQuantification.2018021975.

John Drury. Recent developments in the psychology of crowds and collective behaviour. *Current Opinion in Psychology*, 35:12–16, 2020. doi:10.1016/j.copsyc.2020.02.005.

Dorine C. Duives. *Analysis and Modelling of Pedestrian Movement Dynamics at Large-scale Events*. PhD thesis, Delft University of Technology, 2016. URL `https://doi.org/10.4233/uuid:08831f69-9b8e-44cf-8afe-f4a3e7bc9a9c`.

Dorine C. Duives, Winnie Daamen, and Serge P. Hoogendoorn. Continuum modelling of pedestrian flows - Part 2: Sensitivity analysis featuring crowd movement phenomena. *Physica A: Statistical Mechanics and its Applications*, 447:36–48, 2016. doi:10.1016/j.physa.2015.11.025.

Christian Eilhardt and Andreas Schadschneider. Stochastic headway dependent velocity model for 1d pedestrian dynamics at high densities. *Transportation Research Procedia*, 2:400–405, 2014. doi:10.1016/j.trpro.2014.09.043. The Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014).

Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):419–474, 2012. doi:10.1111/j.1467-9868.2011.01010.x.

Jonathan Feinberg and Hans P. Langtangen. Chaospy: An open source tool for designing methods of uncertainty quantification. *Journal of Computational Science*, 11:46–57, 2015. doi:10.1016/j.jocs.2015.08.008.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, Boston, MA, 1994. ISBN 0201633612.

Ziyou Gao, Yunchao Qu, Xingang Li, Jiancheng Long, and Hai-Jun Huang. Simulating the dynamic escape process in large public places. *Operations Research*, 62(6):1344–1357, 2014. doi:10.1287/opre.2014.1312.

Alan E. Gelfand. Gibbs sampling. *Journal of the American Statistical Association*, 95 (452):1300–1304, 2000. doi:10.1080/01621459.2000.10474335.

Andrew Gelman, Gareth O. Roberts, and Walter R. Gilks. Efficient Metropolis jumping rules. In *Bayesian Statistics*, volume 5, pages 599–607, 1996. ISBN 9780198523567.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, Nov 1984. doi:10.1109/TPAMI.1984.4767596.

Roger Ghanem, David Higdon, and Houman Owhadi, editors. *Handbook of uncertainty quantification*. Springer, Cham, 06 2017. doi:10.1007/978-3-319-12385-1.

Peter G. Gipps and Bertil S. Marksjö. A micro-simulation model for pedestrian flows. *Mathematics and Computers in Simulation*, 27(2–3):95–105, 1985. doi:10.1016/0378-4754(85)90027-8.

Marion Gödel, Rainer Fischer, and Gerta Köster. Applying Bayesian inversion with Markov Chain Monte Carlo to Pedestrian Dynamics. In *UNCECOMP 2019, 3rd ECCOMAS Thematic Conference on Uncertainty Quantification in Computational Sciences and Engineering*, 2019a. doi:10.7712/120219.6322.18561.

Marion Gödel, Rainer Fischer, and Gerta Köster. Towards inferring input parameters from measurements: Bayesian inversion for a bottleneck scenario. In *Traffic and Granular Flow '19*, 2019b. doi:10.1007/978-3-030-55973-1_12.

Marion Gödel, Rainer Fischer, and Gerta Köster. Sensitivity analysis for microscopic crowd simulation. *Algorithms*, 13, 2020a. doi:10.3390/a13070162.

Marion Gödel, Gerta Köster, Daniel Lehmberg, Manfred Gruber, Angelika Kneidl, and Florian Sesser. Can we learn where people go? *Collective Dynamics*, 2020b. doi:10.17815/CD.2020.43.

Marion Gödel, Luca Spataro, and Gerta Köster. Can we learn where people come from? Retracing of pedestrians origins in merging situations. Technical report, Munich University of Applied Sciences, 2020c. URL `https://doi.org/10.48550/arXiv.2012.11527`.

Marion Gödel, Nikolai Bode, Gerta Köster, and Hans-Joachim Bungartz. Bayesian inference methods to calibrate crowd dynamics models for safety applications. *Safety Science*, 147:105586, 2022. doi:10.1016/j.ssci.2021.105586.

Ren-Yong Guo, Hai-Jun Huang, and S.C. Wong. Route choice in pedestrian evacuation under conditions of good and zero visibility: Experimental and simulation results. *Transportation Research Part B: Methodological*, 46(6):669–686, 2012. doi:10.1016/j.trb.2012.01.002.

Stephen J. Guy, Jur van den Berg, Wenxi Liu, Rynson Lau, Ming C. Lin, and Dinesh Manocha. A statistical similarity measure for aggregate crowd dynamics. *ACM Trans. Graph.*, 31(6):190:1–190:11, 2012. doi:10.1145/2366145.2366209.

Heikki Haario, Eero Saksman, and Johanna Tamminen. Adaptive proposal distribution for random walk metropolis algorithm. *Computational Statistics*, 14:375–395, 1999. doi:10.1007/s001800050022.

Jacques Hadamard. Sur les problèmes aux dérivés partielles et leur signication physique. *Princeton University Bulletin*, 13:49–52, 1902.

Panagiotis E. Hadjidoukas, Panagiotis Angelikopoulos, Costas Papadimitriou, and Petros Koumoutsakos. Π4U: A high performance computing framework for Bayesian uncertainty quantification of complex models. *Journal of Computational Physics*, 284: 1–21, 2015. doi:10.1016/j.jcp.2014.12.006.

Milad Haghani, Majid Sarvi, and Abbas Rajabifard. Simulating indoor evacuation of pedestrians: The sensitivity of predictions to directional-choice calibration parameters. *Transportation Research Record*, 2018. doi:10.1177/0361198118796351.

Edward T. Hall. *The Hidden Dimension.* Doubleday, New York, 1966.

Flurin S. Hänseler, Nicholas A. Molyneaux, and Michel Bierlaire. Estimation of pedestrian origin-destination demand in train stations. *Transportation Science*, 51(3):981–997, 2017. doi:10.1287/trsc.2016.0723.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi:10.1038/s41586-020-2649-2.

Florian Hartig, Justin M. Calabrese, Björn Reineking, Thorsten Wiegand, and Andreas Huth. Statistical inference for stochastic simulation models - theory and application. *Ecology Letters*, 14(8):816–827, 2011. doi:10.1111/j.1461-0248.2011.01640.x.

Wilfried K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. doi:10.2307/2334940.

Dirk Helbing and Anders Johansson. *Pedestrian, Crowd and Evacuation Dynamics*, pages 697–716. Springer New York, New York, NY, 2011. doi:10.1007/978-1-4419-7695-6_37.

Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51(5):4282–4286, 1995. doi:10.1103/PhysRevE.51.4282.

Dirk Helbing, Illés Farkas, and Tamás Vicsek. Simulating dynamical features of escape panic. *Nature*, 407:487–490, 2000. doi:10.1038/35035023.

Jon Herman and Will Usher. SALib: An open-source Python library for sensitivity analysis. *The Journal of Open Source Software*, 2(9), 01 2017. doi:10.21105/joss.00097.

K. Hirai and K. Tarui. A simulation of the behavior of a crowd in panic. In *Proc. of the 1975 International Conference on Cybernetics and Society*, page 409, 1975.

Serge P. Hoogendoorn and Piet H. L. Bovy. Pedestrian route-choice and activity scheduling theory and models. *Transportation Research Part B: Methodological*, 38(2):169–190, 2004. doi:10.1016/S0191-2615(03)00007-9.

Serge P. Hoogendoorn and Winnie Daamen. Microscopic parameter identification of pedestrian models and implications for pedestrian flow modeling. *Transportation Research Record*, 1982(1):57–64, 2006. doi:10.1177/0361198106198200108.

Serge P. Hoogendoorn and Winnie Daamen. Microscopic calibration and validation of pedestrian models: Cross-comparison of models using experimental data. In Andreas Schadschneider, Thorsten Pöschel, Reinhart Kühne, Michael Schreckenberg, and Dietrich E. Wolf, editors, *Traffic and Granular Flow'05*, pages 329–340. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-47641-2_29.

Serge P. Hoogendoorn, Femke L.M. van Wageningen-Kessels, Winnie Daamen, and Dorine C. Duives. Continuum modelling of pedestrian flows: From microscopic principles to self-organised macroscopic phenomena. *Physica A: Statistical Mechanics and its Applications*, 416:684–694, 2014. doi:10.1016/j.physa.2014.07.050.

Roger L. Hughes. The flow of large crowds of pedestrians. *Mathematics and Computers in Simulation*, 53(4):367–370, 2000. doi:10.1016/S0378-4754(00)00228-7.

Mohamed Hussein and Tarek Sayed. A methodology for the microscopic calibration of agent-based pedestrian simulation models. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3773–3778, Nov 2018. doi:10.1109/ITSC.2018.8569395.

John Kells Ingram. *A history of political economy.* Cambridge University Press, 1888.

Bertrand Iooss and Paul Lemaître. *A review on global sensitivity analysis methods*, pages 101–122. Springer US, Boston, MA, 2015. doi:10.1007/978-1-4899-7547-8_5.

Tsutomu Ishigami and Toshimitsu Homma. An importance quantification technique in uncertainty analysis for computer models. In *[1990] Proceedings. First International Symposium on Uncertainty Modeling and Analysis*, pages 398–403, 1990. doi:10.1109/ISUMA.1990.151285.

Michiel J.W. Jansen. Analysis of variance designs for model output. *Computer Physics Communications*, 117(1):35–43, 1999. doi:10.1016/S0010-4655(98)00154-4.

Anders Johansson, Dirk Helbing, and Pradyumn Shukla. Specification of the social force pedestrian model by evolutionary adjustment to video tracking data. *Advances in Complex Systems*, 10:271–288, 2007. doi:10.1142/S0219525907001355.

Fredrik Johansson, Anders Peterson, and Andreas Tapani. Waiting pedestrians in the social force model. *Physica A: Statistical Mechanics and its Applications*, 419:95–107, 2015. doi:10.1016/j.physa.2014.10.003.

Hsuan Jung and Paul Marjoram. Choice of summary statistic weights in approximate Bayesian computation. *Statistical applications in genetics and molecular biology*, 10 (23089822):45, 09 2011. doi:10.2202/1544-6115.1586.

Armel U. Kemloh Wagoum, Antoine Tordeux, and Weichen Liao. Understanding human queuing behaviour at exits: an empirical study. *Royal Society Open Science*, 4(1): 160896, 2017. doi:10.1098/rsos.160896.

Armel Ulrich Kemloh Wagoum, Bernhard Steffen, Armin Seyfried, and Mohcine Chraibi. Parallel real time computation of large scale pedestrian evacuations. *Advances in Engineering Software*, 60-61:98–103, 2013. doi:10.1016/j.advengsoft.2012.10.001. CIVIL-COMP: Parallel, Distributed, Grid and Cloud Computing.

Sultan D. Khan, Stefania Bandini, Saleh Basalamah, and Giuseppe Vizzari. Analyzing crowd behavior in naturalistic conditions: Identifying sources and sinks and characterizing main flows. *Neurocomputing*, 177:543–563, 2016. doi:10.1016/j.neucom.2015.11.049.

Jack P.C. Kleijnen. Review of random and group-screening designs. *Communications in Statistics - Theory and Methods*, 16(10):2885–2900, 1987. doi:10.1080/03610928708829548.

Benedikt Kleinmeier and Gerta Köster. Experimental setups to observe evasion maneuvers in low and high densities. In Iker Zuriguel, Ángel Garcimartín, and Raúl Cruz, editors, *Traffic and Granular Flow 2019*, Springer Proceedings in Physics. Springer, 2020. doi:10.1007/978-3-030-55973-1_15.

Benedikt Kleinmeier, Benedikt Zönnchen, Marion Gödel, and Gerta Köster. Vadere: An open-source simulation framework to promote interdisciplinary understanding. *Collective Dynamics*, 4, 2019. doi:10.17815/CD.2019.21.

Benedikt Kleinmeier, Gerta Köster, and John Drury. Agent-based simulation of collective cooperation: From experiment to model. *Journal of the Royal Society Interface*, 17: 20200396, 2020. doi:10.1098/rsif.2020.0396.

Angelika Kneidl. S^2UCRE - Sicherheit in Städtischen Umgebungen: Crowd Monitoring, Prädiktion und Entscheidungsunterstützung; Teilvorhaben: Regelkreis zwischen

Realdaten und Simulation für realitätsnahe Prädiktion: Schlussbericht, 2021. URL `https://www.tib.eu/de/suchen/id/TIBKAT%3A1756553068`.

Moonsoo Ko, Taewan Kim, and Keemin Sohn. Calibrating a social-force-based pedestrian walking model based on maximum likelihood estimation. *Transportation*, 40(1): 91–107, 01 2013. doi:10.1007/s11116-012-9411-z.

Anna Kormanová. A review on macroscopic pedestrian flow modelling. *Acta Informatica Pragensia*, 2013(2):39–50, 2013. doi:10.18267/j.aip.22.

Gerta Köster and Benedikt Zönnchen. Queuing at bottlenecks using a dynamic floor field for navigation. *Transport Research Procedia*, pages 344–352, 2014. doi:10.1016/j.trpro.2014.09.029. The Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014).

Gerta Köster, Franz Treml, and Marion Gödel. Avoiding numerical pitfalls in social force models. *Physical Review E*, 87(6):063305, 2013. doi:10.1103/PhysRevE.87.063305.

Gerta Köster, Franz Treml, Michael J. Seitz, and Wolfram Klein. Validation of crowd models including social groups. In Ulrich Weidmann, Uwe Kirsch, and Michael Schreckenberg, editors, *Pedestrian and Evacuation Dynamics 2012*, pages 1051–1063. Springer International Publishing, 2014. doi:10.1007/978-3-319-02447-9_87.

Gerta Köster, Daniel Lehmberg, and Angelika Kneidl. Walking on stairs: Experiment and model. *Phys. Rev. E*, 100:022310, 08 2019. doi:10.1103/PhysRevE.100.022310.

George Kouskoulis, Ioanna Spyropoulou, and Constantinos Antoniou. Pedestrian simulation: Theoretical models vs. data driven techniques. *International Journal of Transportation Science and Technology*, 7(4):241–253, 2018. doi:10.1016/j.ijtst.2018.09.001. Special Issue on Advances in Transportation Modeling and Policy in the Modern Era.

Tobias Kretz, Anna Grünebohm, and Michael Schreckenberg. Experimental study of pedestrian flow through a bottleneck. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(10):P10014, 10 2006. doi:10.1088/1742-5468/2006/10/P10014.

John K. Kruschke. *Doing Bayesian Data Analysis: A Tutorial with R, JAGS and Stan.* Academic Press, Inc., 2nd edition, 2015. ISBN 9780124058880.

Sergei Kucherenko, Daniel Albrecht, and Andrea Saltelli. Exploring multi-dimensional spaces: a comparison of Latin hypercube and quasi Monte Carlo sampling techniques. *arXiv*, 2015. doi:10.48550/arXiv.1505.02350.

Florian Künzner. *Efficient non-intrusive uncertainty quantification for large-scale simulation scenarios.* Dissertation, Technical University of Munich, Munich, 2021. URL `http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20210118-1576066-1-1`.

Valentina Kurtc, Gerta Köster, and Rainer Fischer. Sensitivity analysis for resilient safety design: Application to a bottleneck scenario. In John Littlewood, Robert J. Howlett, and Lakhmi C. Jain, editors, *Sustainability in Energy and Buildings 2020*, volume 203, pages 255–264. Springer Science + Business Media, 2021. doi:10.1007/978-981-15-8783-2_21.

Gustave Le Bon. *La Psychologie des Foules*. Alcan, 1895.

Kurt Lewin. *Field theory in social science: Selected theoretical papers*. Harper, New York, 1951.

Yan Li, Majid Sarvi, and Kourosh Khoshelham. Pedestrian origin-destination estimation in emergency scenarios. In *2019 9th International Conference on Fire Science and Fire Protection Engineering (ICFSFPE)*, pages 1–5, 2019. doi:10.1109/ICFSFPE48751.2019.9055868.

Weichen Liao, Armin Seyfried, Jun Zhang, Maik Boltes, Xiaoping Zheng, and Ying Zhao. Experimental study on pedestrian flow through wide bottleneck. *Transportation Research Procedia*, 2:26–33, 2014. doi:10.1016/j.trpro.2014.09.005. The Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014).

Jack Liddle, Armin Seyfried, Wolfram Klingsch, Tobias Rupprecht, Andreas Schadschneider, and Andreas Winkens. An experimental study of pedestrian congestions: Influence of bottleneck width and length. *arXiv*, 0911.4350(v2), 2009. URL `https://arxiv.org/abs/0911.4350`.

Ruggiero Lovreglio, Enrico Ronchi, and Daniel Nilsson. Calibrating floor field cellular automaton models for pedestrian dynamics by using likelihood function optimization. *Physica A: Statistical Mechanics and its Applications*, 438:308–320, 2015. doi:10.1016/j.physa.2015.06.040.

Lili Lu, Ching-Yao Chan, Jian Wang, and Wei Wang. A study of pedestrian group behaviors in crowd evacuation based on an extended floor field cellular automaton model. *Transportation Research Part C: Emerging Technologies*, 81:317–329, 2017. doi:10.1016/j.trc.2016.08.018.

Stefano Marelli and Bruno Sudret. *UQLab: A Framework for Uncertainty Quantification in Matlab*, pages 2554–2563. American Society of Civil Engineers, 2014. doi:10.1061/9780784413609.257.

Paul Marjoram, John Molitor, Vincent Plagnol, and Simon Tavaré. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26): 15324–15328, 2003. doi:10.1073/pnas.0306899100.

Francisco Martinez-Gil, Migual Lozano, and Fernando Fernández. Strategies for simulating pedestrian navigation with multiple reinforcement learning agents. *Autonomous Agents and Multi-Agent Systems*, 2015. doi:10.1007/s10458-014-9252-6.

Ryan G. McClarren. *Introduction to Uncertainty Quantification and Predictive Science*, pages 3–17. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-99525-0_1.

Wes McKinney. Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010. doi:10.25080/Majora-92bf1922-00a.

Kerrie L. Mengersen, Pierre Pudlo, and Christian P. Robert. Bayesian computation via empirical likelihood. *Proceedings of the National Academy of Sciences*, 110(4): 1321–1326, 2013. doi:10.1073/pnas.1208827110.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21:1087–1092, 1953. doi:10.1063/1.1699114.

Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 8th edition, 2013. ISBN 9781118146927.

Max D. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2):161–174, 1991. doi:10.1080/00401706.1991.10484804.

Mehdi Moussaïd, Simon Garnier, Guy Theraulaz, and Dirk Helbing. Collective information processing and pattern formation in swarms, flocks, and crowds. *Topics in Cognitive Science*, 1(3):469–497, 2009. doi:10.1111/j.1756-8765.2009.01028.x.

Neelamkavil. *Computer Simulation and Modelling*. John Wiley & Sons, Inc., USA, 1987. ISBN 0471911305.

Katsuhiro Nishinari, Ansgar Kirchner, Alireza Namazi, and Andreas Schadschneider. Extended floor field CA model for evacuation dynamics. *IEICE TRANSACTIONS on Information and Systems*, E87-D(3):726–732, 03 2004. URL `https://arxiv.org/abs/cond-mat/0306262`.

William L. Oberkampf and Christopher J. Roy. *Verification and Validation in Scientific Computing*. Cambridge University Press, Cambridge, 2010. ISBN 9780521113601.

Tinsley Oden, Robert Moser, and Omar Ghattas. Computer predictions with quantified uncertainty, Part I, 2010. URL `https://archive.siam.org/pdf/news/1842.pdf`.

John O'Keefe and Lynn Nadel. *The Hippocampus as a Cognitive Map*. Oxford: Clarendon Press, 1978. ISBN 0198572069.

Edoardo Patelli. *COSSAN: A Multidisciplinary Software Suite for Uncertainty Quantification and Risk Management*, pages 1–69. Springer International Publishing, Cham, 2016. doi:10.1007/978-3-319-11259-6_59-1.

Karl Pearson and Francis Galton. VII. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352):240–242, 1895. doi:10.1098/rspl.1895.0041.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL `https://arxiv.org/abs/1201.0490`.

Anselmo Ramalho Pitombeira Neto, Francisco Moraes de Oliveira Neto, and Carlos Felipe Grangeiro Loureiro. Statistical models for the estimation of the origin-destination matrix from traffic counts. *Transportes*, 2017. doi:10.14295/transportes.v25i4.1344.

Matti Pouke, Jorge Goncalves, Denzil Ferreira, and Vassilis Kostakos. Practical simulation of virtual crowds using points of interest. *Computers, Environment and Urban Systems*, 57:118–129, 2016. doi:10.1016/j.compenvurbsys.2016.02.004.

Ernesto E Prudencio and Karl W Schulz. The parallel `C++` statistical library 'QUESO': Quantification of uncertainty for estimation, simulation and optimization. In *Euro-Par 2011: Parallel Processing Workshops*, pages 398–407. Springer, 2012. doi:10.1007/978-3-642-29737-3_44.

Vincenzo Punzo, Marcello Montanino, and Biagio Ciuffo. Do we really need to calibrate all the parameters? Variance-based sensitivity analysis to simplify microscopic traffic flow models. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):184–193, 02 2015. doi:10.1109/TITS.2014.2331453.

Simon Rahn, Marion Gödel, Rainer Fischer, and Gerta Köster. Dynamics of a simulated demonstration march: An efficient sensitivity analysis. *Sustainability*, 13(6):3455, 2021. doi:10.3390/su13063455.

Craig W. Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference*, pages 763–782, San Jose, CA, 1999. Miller Freeman Game Group, San Francisco, CA. URL `http://www.red3d.com/cwr/papers/1999/gdc99steer.html`.

RiMEA. *Guideline for Microscopic Evacuation Analysis*. RiMEA e.V., 3.0.0 edition, 2016. URL `http://www.rimea.de/`.

Thomas Robin, Gianluca Antonini, Michel Bierlaire, and Javier Cruz. Specification, estimation and validation of a pedestrian walking behavior model. *Transportation Research Part B: Methodological*, 43(1):36–56, 2009. doi:10.1016/j.trb.2008.06.010.

Enrico Ronchi, Erica D. Kuligowski, Paul A. Reneke, Richard D. Peacock, and Daniel Nilsson. The process of verification and validation of building fire evacuation models. Technical Note 1822, National Institute of Standards and Technology (NIST), U. S. Department of Commerce, 2013. URL `https://doi.org/10.6028/NIST.TN.1822`.

Enrico Ronchi, Paul A. Reneke, and Richard D. Peacock. A method for the analysis of behavioural uncertainty in evacuation modelling. *Fire Technology*, 50:1545–1571, 2014. doi:10.1007/s10694-013-0352-7.

David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1):125–141, 2008. doi:10.1007/s11263-007-0075-7.

Christian Rudloff, Thomas Matyus, and Stefan Seer. Comparison of different calibration techniques on simulated data. In Ulrich Weidmann, Uwe Kirsch, and Michael Schreckenberg, editors, *Pedestrian and Evacuation Dynamics 2012*, pages 657–672, Cham, 2014. Springer International Publishing. doi:10.1007/978-3-319-02447-9_55.

Tobias Rupprecht, Armin Seyfried, Wolfram Klingsch, and Maik Boltes. Bottleneck capacity estimation for pedestrian traffic. In *Proceedings of the Interflam 2007*, pages 1423–1430, 2007. URL `https://juser.fz-juelich.de/record/59573`.

Tobias Rupprecht, Wolfram Klingsch, and Armin Seyfried. Influence of geometry parameters on pedestrian flow through bottleneck. In Richard D. Peacock, Erica D. Kuligowski, and Jason D. Averill, editors, *Pedestrian and Evacuation Dynamics*, pages 71–80. Springer US, 2011. doi:/10.1007/978-1-4419-9725-8_7.

Andrea Saltelli and Ilya M. Sobol'. About the use of rank transformation in sensitivity analysis of model output. *Reliability Engineering & System Safety*, 50(3):225–239, 1995. doi:10.1016/0951-8320(95)00099-2.

Andrea Saltelli, Marco Ratto, Stefano Tarantola, and Francesca Campolongo. Sensitivity analysis for chemical models. *Chem. Rev.*, 105(7):2811–2828, July 2005. doi:10.1021/cr040659d.

Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global Sensitivity Analysis. The Primer*. John Wiley & Sons, Ltd., 2008. doi:10.1002/9780470725184.

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(2): 259–270, 2010. doi:10.1016/j.cpc.2009.09.018.

Andreas Schadschneider. Cellular automaton approach to pedestrian dynamics - theory. In Michael Schreckenberg and Som Deo Sharma, editors, *Pedestrian and Evacuation Dynamics*, pages 75–86. Springer, 2001. URL `https://arxiv.org/abs/cond-mat/0112117`.

Stewart Schlesinger. Terminology for model credibility. *Simulation*, 32(3):103–104, 1979. doi:10.1177/003754977903200304.

David W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, Inc., 1992. doi:10.1002/9780470316849.

Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with Python. In *9th Python in Science Conference*, 2010. doi:10.25080/MAJORA-92BF1922-011.

Stefan Seer, Norbert Brändle, and Carlo Ratti. Kinects and human kinetics: A new approach for studying pedestrian behavior. *Transportation Research Part C: Emerging Technologies*, 48(0):212–228, 2014a. doi:10.1016/j.trc.2014.08.012.

Stefan Seer, Christian Rudloff, Thomas Matyus, and Norbert Brändle. Validating social force based models with comprehensive real world motion data. *Transportation Research Procedia*, 2:724–732, 2014b. doi:10.1016/j.trpro.2014.09.080. The Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014).

Michael J. Seitz. *Simulating pedestrian dynamics: Towards natural locomotion and psychological decision making*. PhD thesis, Technical University of Munich, Munich, Germany, 2016. URL `http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20160623-1293050-1-6`.

Michael J. Seitz and Gerta Köster. Natural discretization of pedestrian movement in continuous space. *Physical Review E*, 86(4):046108, 2012. doi:10.1103/PhysRevE.86.046108.

Michael J. Seitz and Gerta Köster. How update schemes influence crowd simulations. *Journal of Statistical Mechanics: Theory and Experiment*, 2014(7):P07002, 2014. doi:10.1088/1742-5468/2014/07/P07002.

Michael J. Seitz, Gerta Köster, and Alexander Pfaffinger. Pedestrian group behavior in a cellular automaton. In Ulrich Weidmann, Uwe Kirsch, and Michael Schreckenberg, editors, *Pedestrian and Evacuation Dynamics 2012*, pages 807–814. Springer International Publishing, 2014. doi:10.1007/978-3-319-02447-9_67.

Michael J. Seitz, Nikolai W. F. Bode, and Gerta Köster. How cognitive heuristics can explain social interactions in spatial movement. *Journal of the Royal Society Interface*, 13(121):20160439, 2016. doi:10.1098/rsif.2016.0439.

James. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996. doi:10.1073/pnas.93.4.1591.

Armin Seyfried, Oliver Passon, Bernhard Steffen, Maik Boltes, Tobias Rupprecht, and Wolfram Klingsch. New insights into pedestrian flow through bottlenecks. *Transportation Science*, 43:395–406, 2009. doi:10.1287/trsc.1090.0263.

Georges Sfeir, Constantinos Antoniou, and Nivine Abbas. Simulation-based evacuation planning using state-of-the-art sensitivity analysis techniques. *Simulation Modelling Practice and Theory*, 89:160–174, 2018. doi:10.1016/j.simpat.2018.09.017.

Meng Shi, Eric Wai Ming Lee, and Yi Ma. A newly developed mesoscopic model on simulating pedestrian flow. *Procedia Engineering*, 211:614–620, 2018. doi:10.1016/j.proeng.2017.12.055. 2017 8th International Conference on Fire Science and Fire Protection Engineering (ICFSFPE 2017).

Meng Shi, Eric Wai Ming Lee, Yi Ma, Wei Xie, and Ruifeng Cao. The density-speed correlated mesoscopic model for the study of pedestrian flow. *Safety Science*, 133: 105019, 2021. doi:10.1016/j.ssci.2020.105019.

Anna Sieben, Jette Schumann, and Armin Seyfried. Collective phenomena in crowds — where pedestrian dynamics need social psychology. *PLOS ONE*, 12(6):1–19, 2017. doi:10.1371/journal.pone.0177328.

Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall/CRC, 1st edition, 1986. ISBN 9780412246203.

Ralph C. Smith. *Uncertainty Quantification: Theory, Implementation, and Applications*. Computational Science and Engineering. Society for Industrial and Applied Mathematics, 2014. ISBN 9781611973211.

Ilya M. Sobol'. Sensitivity estimates for nonlinear mathematical models. *Mathematical Modelling and Computational Experiment*, 1:407–414, 1993.

Ilya M. Sobol'. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, 55:271–280, 2001. doi:10.1016/S0378-4754(00)00270-6.

Ilya M. Sobol' and Sergei Kucherenko. Global sensitivity indices for nonlinear mathematical models. Review. *Wilmott magazine*, 2005. URL `http://www.broda.co.uk/gsa/sobol_global_sa.pdf`.

Ilya M. Sobol' and Sergei Kucherenko. Derivative based global sensitivity measures and their link with global sensitivity indices. *Mathematics and Computers in Simulation*, 79(10):3009–3017, 2009. doi:10.1016/j.matcom.2009.01.023.

Martijn Sparnaaij, Dorine C. Duives, Victor L. Knoop, and Serge P. Hoogendoorn. Multiobjective calibration framework for pedestrian simulation models: A study on the effect of movement base cases, metrics, and density levels. *Journal of Advanced Transportation*, 2019. doi:10.1155/2019/5874085.

Charles Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. URL `http://www.jstor.org/stable/1412159`.

Albert Steiner, Michel Philipp, and Alex Schmid. Parameter estimation for a pedestrian simulation model. In *Swiss Transport Research Conference*, 2007. URL `https://digitalcollection.zhaw.ch/handle/11475/14141`.

Miroslav Stoyanov, Damien Lebrun-Grandie, John Burkardt, Drayton Munster, and US-DOE. Tasmanian, 9 2013. URL `https://www.osti.gov//servlets/purl/1631296`.

Andrew M. Stuart. Inverse problems: A Bayesian perspective. *Acta Numerica*, 19: 451–559, 2010. doi:10.1017/s0962492910000061.

Bruno Sudret. Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering & System Safety*, 93(7):964–979, 2008. doi:10.1016/j.ress.2007.04.002. Bayesian Networks in Dependability.

Tim J. Sullivan. *Introduction to Uncertainty Quantification*. Springer International Publishing, 1st edition, 2015. doi:10.1007/978-3-319-23395-6.

Neda Taherifar, Homayoun Hamedmoghadam, Sushmitha Sree, and Meead Saberi. A macroscopic approach for calibration and validation of a modified social force model for bidirectional pedestrian streams. *Transportmetrica A: Transport Science*, 15(2): 1637–1661, 2019. doi:10.1080/23249935.2019.1636156.

Floris Takens. Detecting strange attractors in turbulence. *Lecture Notes in Mathematics*, pages 366–381, 1981. doi:10.1007/bfb0091924.

Ming Tang and Hongfei Jia. An approach for calibration and validation of the social force pedestrian model. In *Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*, pages 2026–2031, 2011. doi:10.1109/TMEE.2011.6199614.

Till Tantau. The tikz and pgf packages, 2021. URL `https://pgf-tikz.github.io/`.

Simon Tavaré, David J. Balding, R. C. Griffiths, and Peter Donnelly. Inferring coalescence times from dna sequence data. *Genetics*, 145(2):505–518, 1997. ISSN 0016-6731. URL `https://www.genetics.org/content/145/2/505`.

Mario Teixeira Parente, Daniel Bittner, Steven A. Mattis, Gabriele Chiogna, and Barbara Wohlmuth. Bayesian calibration and sensitivity analysis for a karst aquifer model using active subspaces. *Water Resources Research*, 55(8), 2019. doi:10.1029/2019WR024739.

Kardi Teknomo. *Microscopic Pedestrian Flow Characteristics: Development of an Image Processing Data Collection and Simulation Model*. PhD thesis, Tohoku University, Japan, 2002. URL `https://doi.org/10.48550/arXiv.1610.00029`.

Kardi Teknomo and Gloria P. Gerilla. Sensitivity analysis and validation of a multi-agents pedestrian model. *Journal of the Eastern Asia Society for Transportation Studies*, 6:198–213, 2005. doi:10.11175/easts.6.198.

Kardi Teknomo and Gloria P. Gerilla. *Mesoscopic Multi-Agent Pedestrian Simulation*, pages 323–336. Nova Science Publishers Inc, 2008. ISBN 9781604560312.

Charles Tong. *Problem Solving Environment for Uncertainty Analysis and Design Exploration*, pages 1695–1731. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-12385-1_53.

Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael P.H. Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of The Royal Society Interface*, 6(31):187–202, 2009. doi:10.1098/rsif.2008.0172.

Adrien Treuille, Seth Cooper, and Zoran Popović. Continuum crowds. *ACM Transactions on Graphics (SIGGRAPH 2006)*, 25(3):1160–1168, 2006. doi:10.1145/1141911.1142008.

Mark A. Tumeo. *The Meaning of Stochasticity, Randomness and Uncertainty in Environmental Modeling*, pages 33–38. Springer Netherlands, 1994. doi:10.1007/978-94-011-1072-3_3.

Jur van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation, 2008 (ICRA 2008)*, pages 1928–1935, 2008. doi:10.1109/ROBOT.2008.4543489.

Elske van der Vaart, Dennis Prangle, and Richard M. Sibly. Taking error into account when fitting models using approximate Bayesian computation. *Ecological Applications*, 28(2):267–274, 2018. doi:10.1002/eap.1656.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R.J. Nelson, Eric Jones, Robert Kern, Eric Larson, C.J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, 2020. doi:10.1038/s41592-019-0686-2.

Daniil Voloshin, Dmitriy Rybokonenko, and Vladislav Karbovskii. Optimization-based calibration for micro-level agent-based simulation of pedestrian behavior in public spaces. *Procedia Computer Science*, 66:372–381, 2015. doi:10.1016/j.procs.2015.11.043. 4th International Young Scientist Conference on Computational Science.

Cornelia von Krüchten and Andreas Schadschneider. A cognitive, decision-based model for pedestrian dynamics. In Iker Zuriguel, Angel Garcimartin, and Raul Cruz, editors, *Traffic and Granular Flow 2019*, pages 141–147, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-55973-1_18.

Isabella von Sivers and Gerta Köster. Dynamic stride length adaptation according to utility and personal space. *Transportation Research Part B: Methodological*, 74:104–117, 2015. doi:10.1016/j.trb.2015.01.009.

Isabella von Sivers, Florian Künzner, and Gerta Köster. Pedestrian evacuation simulation with separated families. In *Proceedings of the 8th International Conference on Pedestrian and Evacuation Dynamics (PED2016)*, Hefei, China, 10 2016a. doi:10.17815/CD.2016.11.

Isabella von Sivers, Anne Templeton, Florian Künzner, Gerta Köster, John Drury, Andrew Philippides, Tobias Neckel, and Hans-Joachim Bungartz. Modelling social identification and helping in evacuation simulation. *Safety Science*, 89:288–300, 2016b. doi:10.1016/j.ssci.2016.07.001.

Harvey M. Wagner. Global sensitivity analysis. *Operations Research*, 43(6):948–969, 1995. doi:10.1287/opre.43.6.948.

Chen Wang, Qingyun Duan, Charles H. Tong, Zhenhua Di, and Wei Gong. A GUI platform for uncertainty quantification of complex dynamical models. *Environmental Modelling & Software*, 76:1–12, 2016. doi:10.1016/j.envsoft.2015.11.004.

Ulrich Weidmann. *Transporttechnik der Fussgänger*, volume 90 of *Schriftenreihe des IVT*. Institut für Verkehrsplanung, Transporttechnik, Strassen- und Eisenbahnbau (IVT) ETH, Zürich, 2nd edition, 1993. doi:10.3929/ethz-b-000242008.

Richard David Wilkinson. Approximate Bayesian computation (abc) gives exact results under the assumption of model error. *Statistical Applications in Genetics and Molecular Biology*, 12(2):129–141, 2013. doi:10.1515/sagmb-2013-0010.

David Wolinski, Stefen Guy, Anne-Helene Olivier, Ming Lin, Dinesh Manocha, and Julien Pettré. Parameter estimation and comparative evaluation of crowd simulations. *Comput. Graph. Forum*, 33(2):303–312, 2014. doi:10.1111/cgf.12328.

Szechun C. Wong and Chungon O. Tong. Estimation of time-dependent origin-destination matrices for transit networks. *Transportation Research Part B: Methodological*, 32(1):35–48, 1998. doi:10.1016/S0191-2615(97)00011-8.

Simon N. Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466:1102—1104, 2010. doi:10.1038/nature09319.

Dongbin Xiu. Efficient collocational approach for parametric uncertainty analysis. *Communications in computational physics*, 2(2):293–309, 2007. ISSN 1991-7120.

Dongbin Xiu. Stochastic collocation methods: A survey. In R. Ghanem, H. Owhadi, and D. Higdon, editors, *Handbook of uncertainty quantification*, volume 45, pages 699–716. Springer International Publishing, 2017. doi:10.1007/978-3-319-12385-1_26.

Shanwen Yang, Tianrui Li, Xun Gong, Bo Peng, and Jie Hu. A review on crowd simulation and modeling. *Graphical Models*, 111:101081, 2020. doi:10.1016/j.gmod.2020.101081.

Weiliang Zeng, Hideki Nakamura, and Peng Chen. A modified social force model for pedestrian behavior simulation at signalized crosswalks. *Procedia - Social and Behavioral Sciences*, 138:521–530, 2014. doi:10.1016/j.sbspro.2014.07.233. The 9th International Conference on Traffic and Transportation Studies (ICTTS 2014).

Weiliang Zeng, Peng Chen, Guizhen Yu, and Yunpeng Wang. Specification and calibration of a microscopic model for pedestrian dynamic simulation at signalized intersections: A hybrid approach. *Transportation Research Part C: Emerging Technologies*, 80:37–70, 2017. doi:10.1016/j.trc.2017.04.009.

Jun Zhang, Wolfram Klingsch, Andreas Schadschneider, and Armin Seyfried. Transitions in pedestrian fundamental diagrams of straight corridors and T-junctions. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(06):P06004, 2011. doi:10.1088/1742-5468/2011/06/P06004.

Jinghui Zhong and Wentong Cai. Differential evolution with sensitivity analysis and the powell's method for crowd model calibration. *Journal of Computational Science*, 9: 26–32, 2015. doi:10.1016/j.jocs.2015.04.013. Computational Science at the Gates of Nature.

Benedikt Zönnchen. *Efficient parallel algorithms for large-scale pedestrian simulation.* Dissertation, Technical University of Munich, Munich, 2021. URL `http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20210521-1593965-1-9`.

Benedikt Zönnchen and Gerta Köster. A parallel generator for sparse unstructured meshes to solve the eikonal equation. *Journal of Computational Science*, 32:141–147, 2018. doi:10.1016/j.jocs.2018.09.009.

Benedikt Zönnchen and Gerta Köster. GPGPU computing for microscopic pedestrian simulation. In Ian Foster, Gerhard R. Joubert, Luděk Kučera, Wolfgang E. Nagel, and Frans Peters, editors, *Parallel Computing: Technology Trends*, volume 36, pages 93–104, 2020. doi:10.3233/APC200029.

Appendix

A Infrastructure

I would like to give a brief overview of the tools used for this thesis. Most of the tools are open-source or at least freeware. I am grateful to the developers for providing them.

The uncertainty quantification framework designed and implemented in this thesis is written in Python 3.7. I chose JetBrains' PyCharm[1] community edition 2020.2.3 as development environment. The framework depends on the following external software packages: suq-controller[2], SALib [Herman and Usher, 2017], chaospy [Feinberg and Langtangen, 2015]. I used established Python packages such as NumPy [Harris et al., 2020], SciPy [Virtanen et al., 2020], statsmodels [Seabold and Perktold, 2010], scikit-learn [Pedregosa et al., 2011], and Pandas [McKinney, 2010].

For tasks related to the open-source crowd simulation framework Vadere [Kleinmeier and Köster, 2020] such as programming new output processors, I relied on JetBrains IntelliJ IDEA[3] community edition 2020.2 and OpenJDK[4] 11.0.2. For version control and collaborative software development, Git[5] and Gitlab[6] were used.

The document was created with LaTeXusing TeXstudio[7] 3.1.1. I built on the LaTeX template provided by the Chair of Integrated Systems[8]. Citations were managed using JabRef[9] 5.1. I plotted the results of my studies with Matlab (R2020b) with colorbrewer schemes[10]. UML diagrams were created with draw.io[11] 15.5.8. Snapshots of Vadere simulations are supported by Vadere's Tikz [Tantau, 2021] generator. The mind maps that provide an overview of uncertainty quantification methods were created with XMind 2021.

[1] `https://www.jetbrains.com/pycharm/`
[2] `https://gitlab.lrz.de/vadere/suq-controller`
[3] `https://www.jetbrains.com/idea/`
[4] `https://openjdk.java.net/`
[5] `https://git-scm.com/`
[6] `https://gitlab.lrz.de/vadere`
[7] `https://texstudio.org/`
[8] `https://github.com/TUM-LIS/tum-dissertation-latex`
[9] `https://www.jabref.org/`
[10] Charles (2021). cbrewer : colorbrewer schemes for Matlab (`https://www.mathworks.com/matlabcentral/fileexchange/34087-cbrewer-colorbrewer-schemes-for-matlab`), MATLAB Central File Exchange.
[11] `https://www.draw.io`

B State of the art of parameter identification in crowd simulations

Table B.1: State-of-the-art forward propagations that analyze the variation in the output for different parameters for crowd dynamics models.

Reference	[Sparnaaij et al., 2019]
Model	Pedestrian dynamics by INCONTROL, locomotion by social forces model
Scenario	5 scenarios, bidirectional, corner (low and high density), t-junction, bottleneck
Method	OAT approach
Quantity of interest	Distribution of instantaneous speeds of all pedestrians
Parameter(s)	7 parameters, 2 from route following, 5 from collision avoidance, among them minimum desired speed and personal distance
Stochasticity	Number of necessary replications calculated using convergence method similar to [Ronchi et al., 2013], Result: $30-100$ replications depending on scenario
Note	Parameters are classified as "not sensitive to", "slightly sensitive", and "sensitive".
Reference	[Kouskoulis et al., 2018]
Model	Social force model
Scenario	Aboveground platform of Moschato metro station in Athens
Method	OAT approach
Quantity of interest	Root Mean Squared Percentage Error between simulated and observed trajectories
Parameter(s)	9 parameters, among them desired speed (mean and standard deviation), repulsive forces from obstacles and pedestrians
Stochasticity	No information on how stochasticity is handled, even though it is mentioned that the agents' speeds are drawn from a Gaussian distribution.
Reference	[Baglietto and Parisi, 2011]
Model	Contractile particle model (continuous-space automaton combined with social forces)
Scenario	Circular racetrack
Method	OAT
Quantity of interest	Velocity-density-diagram (fundamental diagram)
Parameter(s)	Minimum agent radius r_{min}, maximum agent radius r_{max}, exponent in relationship between radius and desired speed β
Stochasticity	Random initial position of particles, handling of stochasticity is not discussed.
Reference	[Colombi et al., 2017]
Model	Microscopic-discrete mathematical model based on the concept of walker behavioral strategy

Scenario	Bottleneck scenario in which 200 agents pass from one room through a constriction to an adjacent room to reach a door in the second room
Method	OAT approach, visual comparison of variation in the simulation outcome after propagation
Quantity of interest	Evacuation time (for all parameters except visual region), trajectories for visual region parameters
Parameter(s)	Eleven parameters from five groups: visual region, wall repulsion, interpersonal repulsion, interpersonal contract, and size of the crowd.
Stochasticity	Starting positions are random; they are fixed for the sensitivity analysis.

Reference	[Collins et al., 2014]
Model	Extended social force model with a group model
Scenario	Bottleneck, evacuation of 500 agents through a single exit
Method	No information provided. Judging from the resulting figure, both parameter were varied simultaneously.
Quantity of interest	Time to evacuate
Parameter(s)	Two parameters of the group extension, maximum group size and bias towards exit
Stochasticity	For the group extension, a stochastic selection between heading options is chosen. There is information provided on how the authors dealt with the stochasticity for the sensitivity analysis.

Reference	[Haghani et al., 2018]
Model	Combined microscopic model with social force model for locomotion and a new probabilistic discrete-choice route choice model for the tactical model.
Scenario	Controlled laboratory experiment of evacuation with a choice among four exits
Method	OAT approach
Quantity of interest	Average total evacuation time and average individual evacuation time
Parameter(s)	All parameters of the tactical model $\beta_1, \ldots, \beta_5$ (DIST, CONG, FLTOVIS, VIS, FLTOINVIS)
Stochasticity	Probabilistic route-choice model. For each parameter value, 50 repetitions were performed.

Reference	[Johansson et al., 2015]
Model	Social force model with extension for waiting behavior
Scenario	Straight corridor scenario with a group of waiting agents in the middle, a second group of agents passes by to reach their destination.
Method	No information is provided on the sampling approach.
Quantity of interest	Density along the corridor
Parameter(s)	Non-physical parameter M for the waiting extension, no clear behavioral or physical interpretation.
Stochasticity	No information provided.

Reference	[Davidich and Köster, 2013]
Model	Floor field cellular automaton model with a hexagonal grid
Scenario	Station hall in a German train station where agents can head to platforms and food stands
Method	OAT approach
Quantity of interest	Fitness of simulated density to observed density
Parameter(s)	Five parameters, mean and standard deviation of free-flow speed, density-flow relationship, pedestrian appearance schedule, and origin-destination distributions

Stochasticity	The CA model contains stochastic terms. No information provided on handling of the stochasticity.

Reference	[Shi et al., 2021]
Model	Grid-based mesoscopic model with an irregular grid using a static or dynamic floor field (CA)
Scenario	Bottleneck, agents evacuate from a room with a wide exit.
Method	Full factorial design
Quantity of interest	Average evacuation time
Parameter(s)	Time step δ, free moving velocity V, dynamic field (static vs. dynamic), pedestrian number N
Stochasticity	100 repetitions are performed for each configuration
Note	One parameter, the time step, is ranked as non-influential for the evacuation time.

Reference	[Teknomo and Gerilla, 2005]
Model	Force-based model with feedback loop inspired by social forces model
Scenario	Not clear, probably a pedestrian crossing in Sengai
Method	Full factorial analysis
Quantity of interest	1) Mean of average speed, 2) slope of the speed-density graph, 3) free-flow speed
Parameter(s)	Separate analyzes for two groups of parameters: 1) control variables of feedback loop, maximum speed and number of agents, 2) motion parameters, mass m, α, β, χ. α, β, and χ are non-physical parameters that serve to generalize the model.
Stochasticity	Averaging over 10 simulations (for motion parameters)
Note	For the motion parameters α, β, χ, m, only interactions show influence on the chosen quantities of interest.

Reference	[Lu et al., 2017]
Model	Extended Floor-field cellular automaton that considers group behavior
Scenario	Hall with two exits, resembling a university hall
Method	OAT approach
Quantity of interest	Evacuation time
Parameter(s)	Parameters of the group extension k'_S, k_d, and k_i: k_d is the strength of the attraction to the leader, k'_S is the strength of the static floor field when group members make decisions, k_i is the strength of adherence to the group leader
Stochasticity	Different sources of stochasticity in the model: 1) sequential update of agents according to a random number assigned to each agent, 2) random starting positions, 3) probabilistic transition rules. For model calibration, 30 simulations are averaged. However, for propagation, no information is provided.

Reference	[Bianca and Mogno, 2018]
Model	Event-driven microscopic thermostatted kinetic theory model
Scenario	Agents leaving a metro station with n gates
Method	Runge-Kutta formula, different parameter combinations are evaluated
Quantity of interest	Time evolution of the distribution function and the shape of the asymptotic state
Parameter(s)	1) Initial conditions (L, R, C, H), 2) magnitude F of the external force field, 3) number of gates n
Stochasticity	No information provided.

Reference	[Shi et al., 2018]
Model	Grid-based mesoscopic model using a static floor field (CA)
Scenario	Evacuation from a room with a single exit (bottleneck)
Method	Single parameter is varied, no details on the sampling method.
Quantity of interest	Evacuation time
Parameter(s)	Time step length δ
Stochasticity	Probabilistic transition rules for the cellular automaton, 1000 repetitions are performed.
Reference	[Duives et al., 2016]
Model	Macroscopic continuum model
Scenario	4 scenarios - uni-directional short bottleneck, uni-directional corner, bi-directional straight, intersecting flow scenario (two groups of agents)
Method	Equidistant sampling, full factorial setup
Quantity of interest	Spatial distribution of density and velocity as proxies for traffic state, for the intersecting flows, also the number of lanes are counted to analyze the traffic state.
Parameter(s)	For scenarios with one groups, one parameter, the local β_l route choice component is studied. The global route component is a dependent parameter $\beta_g = 10 - \beta_l$. When two groups of agents are present, also $\beta_{\delta=d}$ is studied, which is concerned with the avoidance of high density areas, are studied (dependent parameter $\beta_{\delta\neq d} = 10 - \beta_{\delta=d}$).

Table B.2: State-of-the-art sensitivity analysis for crowd dynamics models.

Reference	[Zhong and Cai, 2015]
Model	Social force model
Scenario	Two case studies, 1) evacuation from a room (bottleneck), 2) real-world cross-walk with observational data
Method	Probably an OAT approach with equidistant sampling (uniform sampling)
Metric	Entropy-based measure
Quantity of interest	For calibration, a distance measure for the local density is used; for sensitivity analysis, no information is provided.
Parameter(s)	Interaction strength A and range B for other agents and obstacles, k_1, k_2, free-flow speed v_0, maximum speed v_max, time step (discretization) τ
Stochasticity	Stochastic terms in the model, e. g. random initial positions. For calibration, 30 repetitions are averaged, for sensitivity analysis, no information is provided.
Reference	[Chen et al., 2019]
Model	Random forest models trained on questionnaire data, 1) with pre-evacuation time, 2) without pre-evacuation time
Scenario	Evacuation of a skyscraper in an urban residential area, Luoshanqicun Community in Shanghai
Method	OAT approach
Metric	Local sensitivity index from [Saltelli et al., 2005]
Quantity of interest	Total evacuation time

Parameter(s)	Walking speed, shoulder width, acceleration time, distance to obstacles, comfortable distance, Cd_Q from queue density, persist time, collision response time
Stochasticity	No information provided.
Reference	[Kurtc et al., 2021]
Model	Optimal steps model
Metric	Sobol' indices
Scenario	Corridor scenario
Method	1) Monte Carlo calculation of Sobol' indices and 2) generalized polynomial chaos expansion for model output, Sobol' indices are calculated from the expansion
Quantity of interest	Density in the corridor
Parameter(s)	Mean and standard deviation of free-flow speeds, number of agents
Stochasticity	Model contains stochastic term, handling by averaging 10 repetitions for the gPCe.

Table B.3: State-of-the-art sensitivity analysis for traffic models.

Reference	[Sfeir et al., 2018]
Model	Microscopic traffic simulator
Metric	Sobol' indices
Scenario	Evacuation of a parking garage in Athens, Greece. Four different configurations of the scenario are analyzed.
Method	Scatter plots for a pre-selection of influential parameters and then Sobol' indices (using Monte Carlo sampling and a low discrepancy sampling)
Quantity of interest	Evacuation time
Parameter(s)	Maximum acceleration, normal deceleration, maximum speed, reaction time (of the driver)
Stochasticity	Averaging of 10 simulations to "eliminate" the randomness of a single simulation.
Reference	[Punzo et al., 2015]
Model	microscopic traffic flow models, social force model for car-following
Metric	Sobol' total indices
Scenario	NGSIM I80-1 data set
Method	Variance-based global sensitivity analysis
Quantity of interest	Goodness-of-fit between observed and simulated trajectories
Parameter(s)	Six parameters: follower's desired speed V_f^{Max}, a_f^{Max} follower's maximum acceleration, comfort deceleration rate b_f, portion of the desired distance from the leader ΔS_1, minimum time headway between leader and follower T, constant α
Stochasticity	No information provided.
Note	Identifies non-influential parameters and applies factor fixing setting to obtain a reduced model version.

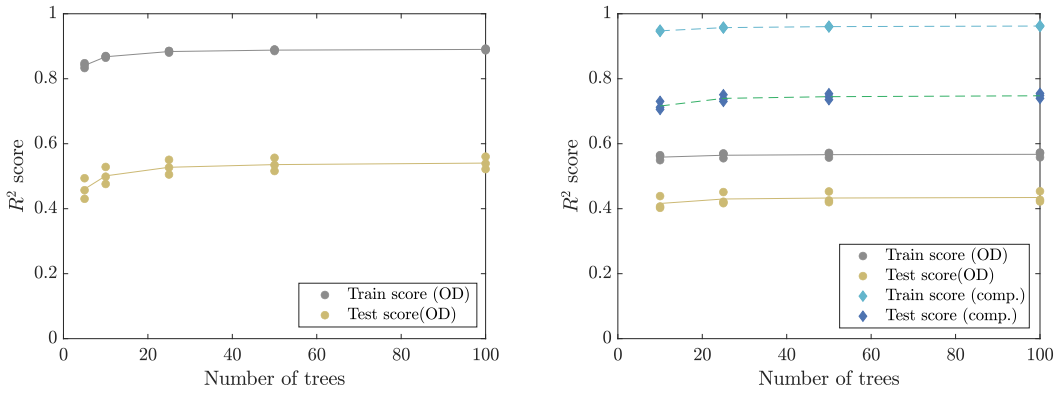
C Estimation of initial and boundary conditions

C.1 Random forest parameters

There are three main parameters to tune the performance of random forest: First, the number of estimators or trees that are trained to the data. Second, the maximum depth for each tree. Third, the number of features used in each split. We study the impact of each parameter separately to find a suitable configuration for our problem. For each parameter, I analyze the impact regarding two configurations: predicting the full OD matrix and predicting two components of the target PCA.

C.1.1 Number of trees

In Figure C.1 the performance of random forest for different numbers of trees is shown (number of features $0.75 \cdot 321$, maximum depth: 15). I choose 100 trees.



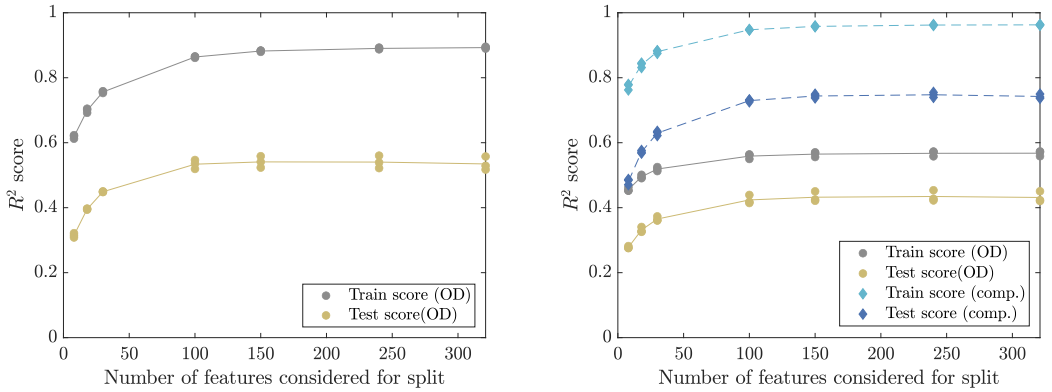
(a) Full origin-destination (OD) matrix

(b) First two components of principal component analysis of the OD matrix

Figure C.1: Performance of random forest against the number of trees.

C.1.2 Number of features for split

Figure C.2 displays the results for random forests with different numbers of features considered when looking for the best split (number of trees: 100, maximum depth: 15). I choose all features (321) to be considered for the split.



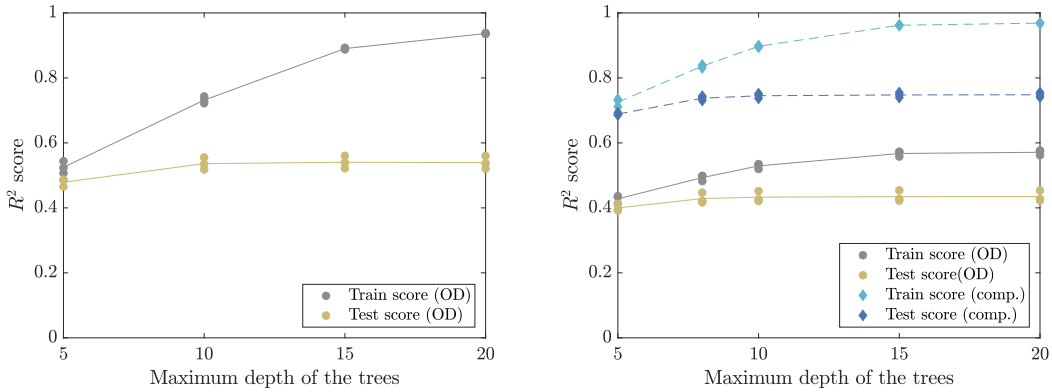
(a) Full origin-destination (OD) matrix

(b) First two components of principal component analysis of the OD matrix

Figure C.2: Performance of random forest against the number of features considered when looking for the best split.

C.1.3 Maximum depth of the trees

Figure C.3 depicts the scores for random forest with different values for the maximum depth of the trees (number of trees: 100, max features: $0.75 \cdot 321$). The maximum depth has the largest impact on the results. The best test scores with a minimum distance between train and test score are obtained with a maximum depth of 10.



(a) Full origin-destination (OD) matrix

(b) First two components of principal component analysis of the OD matrix

Figure C.3: Performance of random forest against the maximum depth.

C.2 Overlapping data set

In order to increase the sample size, I generate a data set with overlapping time intervals. Overlapping means that consecutive samples are no longer from disjoint time intervals, but the time intervals of ten seconds are now chosen only one second apart. This mimics Taken's time-delayed embedding [Takens, 1981], a technique used to reconstruct a dynamic system from a sequence of observations. The new dataset contains approximately 130000 samples.

Due to the large amount of input data, I use the incremental PCA approach by Ross et al. [Ross et al., 2008] implemented in [Pedregosa et al., 2011]. That means, the PCA is fitted iteratively with batches of $5 \cdot 20520$ samples.

The results using overlapping time intervals are shown in Figure C.4. The performance of random forest could not be improved using overlapping time intervals.

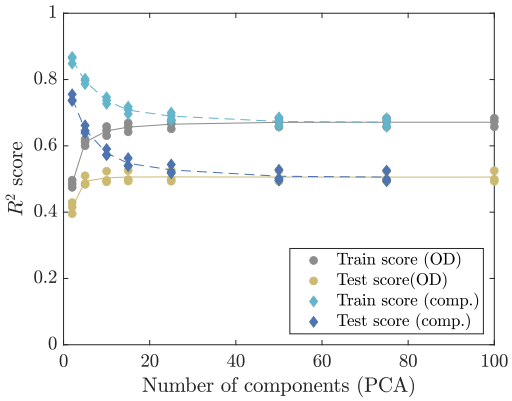


Figure C.4: Performance of random forest model using overlapping time intervals for the input samples. Instead of full origin-destination (OD) matrices, components of the principal component analysis (PCA) are predicted.