TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG

The University of Resources. Since 1765.

On Safe Usage of Shared Data in Safety-Critical Control Systems

By the Faculty of Mathematics and Informatics
of the Technische Universität Bergakademie Freiberg

approved

Thesis

to attain the academic degree of

Doktor-Ingenieur
(Dr.-Ing.)

submitted by **M. Sc. Georg Jäger**

born on the 28. September 1991 in Halberstadt

Assessor: Prof. Dr. Sebastian Zug
Prof. Dr. António Casimiro
Prof. Dr. habil. Rudolf Kruse

Date of the award: Freiberg, July 04th, 2022

Abstract

The introduction of paradigms such as *Internet of Things (IoT)* and *Cyber-Physical System (CPS)* form the basis of profound industrial transformations addressed by the term *Industry 4.0.* The already introduced flexibility and efficiency can be even increased when combined with concepts of automation and intelligent systems. It is envisioned that autonomous mobile systems dynamically share their environmental perceptions to cooperate and thereby generate emerging behaviors. Depending on the contextual situation evolving at their run-time, participating systems are considered to discover available sources of data and temporarily integrate them to increase their performance.

This next step requires switching from a static to a dynamic system composition. Instead of requiring knowledge about all components at design-time, systems with an open architecture featuring a dynamic integration of available resources at run-time are needed. Such dynamically composed systems, however, bring current processes for guaranteeing their safety into question. In the endeavor of identifying the arising challenges, this work starts by examining the functional safety standard IEC 61508. From the derived challenges, the need for a run-time safety assessment methodology is formulated. It is argued that only at run-time the system's composition and the failure characteristics of its components are available. Therefore, its safety can be assessed only at run-time as well.

Following this reasoning, two objectives are formulated. Firstly, a failure model capable of describing the failure characteristics of shared data is required. Secondly, a run-time safety assessment is required that enables analyzing such a failure model and verifying that the performance of implemented safety functions of a system is met.

For fulfilling both, state-of-the-art approaches are reviewed, assessed, and built upon. In that endeavor, a *Generic Failure Model (GFM)* for representing one- and multi-dimensional failure characteristics unambiguously is presented in conjunction with a processing chain enabling to generate such a model.

In the following step, a run-time safety assessment methodology to analyze such failure models of shared data is defined. For that, the theorem of *Region of Safety (RoS)* is defined. Once fulfilled by a given control policy, it provides a guarantee that the controlled system will not leave a region of safe states even when confronted with the specified failure characteristics and/or uncertainties.

To evaluate both concepts, the use case of collision avoidance for robots sharing their position data is considered. It is shown that not only a guarantee can be provided for the safety of the robots when faced with the considered failure characteristics affecting the shared data but also that their configuration can be safely adapted in accordance with changed failure characteristics.

The work is concluded by stating the limitations of the presented concepts and deriving directions for future work.

Acknowledgements

The journey to this work was full of learning, gaining insights, acquiring experiences, and ultimately solving challenges.

On this journey, the support I received from my supervisor, Prof. Sebastian Zug was invaluable. It was not only your patience in discussing my research in great detail that allowed me to grasp its very essence and formulate this thesis but also your infectious motivation that enabled me to grow from the challenge.

I would also like to express my gratitude to Prof. António Casimiro who supported me from the very beginning. Our discussions and your insightful feedback provided me with the means to bring me and my work to a higher level.

I'm very grateful to Madhura Thosar and Christian Müller with whom I argued and thought about approaching experiments and structuring research in length as well. In that line, I would also like to thank Martin Reinhardt, who was always open for discussions on implementation details and provided me with valuable insights about the process accompanying this dissertation.

After all, I would like to take the chance to thank my colleagues and friends at the Otto-von-Guericke Universität Magdeburg as well as at the TU Bergakademie Freiberg. Your kindness, patience, and happiness have encouraged me on my way and supported me whenever I encountered a trouble spot.

My deepest gratitude, however, goes to my friends and family, especially to my parents, brother, and grandparents, whose support I was always sure of. It is only their love, their unwavering support, and their undiminishing motivation that enabled me to pursue this dissertation from start to finish. I owe it all to you!

Contents

List of Acronyms

ACC Adaptive Cruise Control.

ADAS Advanced Driver Assistance System.

AI Artificial Intelligence.

ANN Artificial Neural Network.

ASIL Automotive Safety Integrity Level.

CACC Cooperative Adaptive Cruise Control.

CBF Control Barrier Function.

CDF Cumulative Distribution Function.

CLF Control Lyapunov Function.

ConSert Conditional Safety Certificate.

CPS Cyber-Physical System.

CWT Continuous Wavelet Transformation.

DDI Digital Dependability Identity.

DSC Dynamic Safety Contract.

EMD Earth Mover's Distance.

EUC Equipment Under Control.

EUCCS *Equipment Under Control (EUC)* Control System.

FMEA Failure Mode and Effect Analysis.

FSC Functional Safety Concept.

FTA Fault Tree Analysis.

GFM Generic Failure Model.

GNSS Global Navigation Satellite System.

GP Gaussian Process.

GUM Guide to the Expression of Uncertainty in Measurement.

HARA Hazard Analysis & Risk Assessment.

ICDF Inverse Cumulative Distribution Function.

IoT Internet of Things.

KARYON Kernel-based ARchitecture for safetY-critical cONtrol.

LiDAR Light Detection and Ranging.

LoS Level of Service.

LSTM Long-Short Term Memory.

MAE Mean Absolute Error.

MBDA Model-Based Dependability Analysis.

ML Machine Learning.

MLP Multi Layer Perceptron.

MSE Mean Squared Error.

ODD Operational Design Domain.

ODE Ordinary Differential Equation.

PDF Probability Density Function.

PFD Probability of Fails on Demand.

RBF Radial Basis Function.

RL Reinforcement Learning.

RoA Region of Attraction.

RoS Region of Safety.

Safe RL Safe Reinforcement Learning.

SDE Stochastic Differential Equation.

SGD Stochastic Gradient Descent.

SIL Safety Integrity Level.

SOTIF Safety of the Intended Functionality.

SSE Sum of Squared Errors.

TBF Time Between Failure.

TDNN Time-Delay Neural Network.

TSC Technical Safety Concept.

TtR Time to Repair.

List of Theorems

List of Definitions

List of Figures

List of Tables

1. Introduction – Safety in Future Smart Industries

Increased performance of wired and wireless communication, new approaches to process automation, and advances in intelligent algorithms form the basis of a profound industrial transformation termed *Industry 4.0*. The concept comprises paradigms, such as *Internet of Things (IoT)* [1] and *Cyber-Physical System (CPS)* [2], that promise efficient and sustainable industrial processes paired with increased flexibility to meet the demand for individualized products. A key-enabler to reach these goals are *Digital Twins* [3]. They are virtual replicates of physical processes which are updated regularly using sensor observations. As such, they closely follow the states of the physical process, which enables intelligent algorithms to optimize it, for instance, by predicting failures, scheduling maintenance, and reducing down-times. This introduction of intelligent algorithms to complex control systems motivated to add the attribute *Smart* to affected industries (e.g. *Smart Traffic* [4], *Smart Manufacturing* [2], *Smart Warehouses* [5], [6]).

For accelerating the *smartification* of these industries, the application of the paradigms will be expanded beyond separated industries. For instance, products of smart manufacturing processes can be delivered to smart warehouses by autonomous delivery systems. Once arrived at their destination, they may integrate with the infrastructure to share data and to navigate safely on site. As such shared data enhances the environmental perception of the autonomous delivery systems, it may even facilitate it to replenish the automated storage system directly.

To enable such a use case, a novel system architecture capable of integrating new sources of shared data dynamically at run-time is required. Simultaneously, such a system architecture has to support the well-known attribute of safety.

Commonly, systems are attributed with safety by considering all relevant factors at design-time and showing that they will behave safely at run-time. For that, applicable safety processes rely on complete knowledge about a system's components and their characteristics, cf. Fig. 1.3a. As sources of shared data are not available at design-time, their characteristics can not be taken into consideration, cf. Fig. 1.2b. Therefore, several questions arise.

Do existing approaches for guaranteeing safety cover systems that dynamically integrate sources of shared data? What challenges arise from the changed system architecture regarding safety? How can these challenges be addressed such that shared data can be used safely?

To better understand the consequences the changed system architecture poses on the process of guaranteeing safety for such systems, the next section firstly details the example of smart warehouses. By means of this example, the terms of *statically composed systems* and *dynamically composed systems* are introduced and defined to clarify the consequences of the changed system architecture. This clear distinction enables

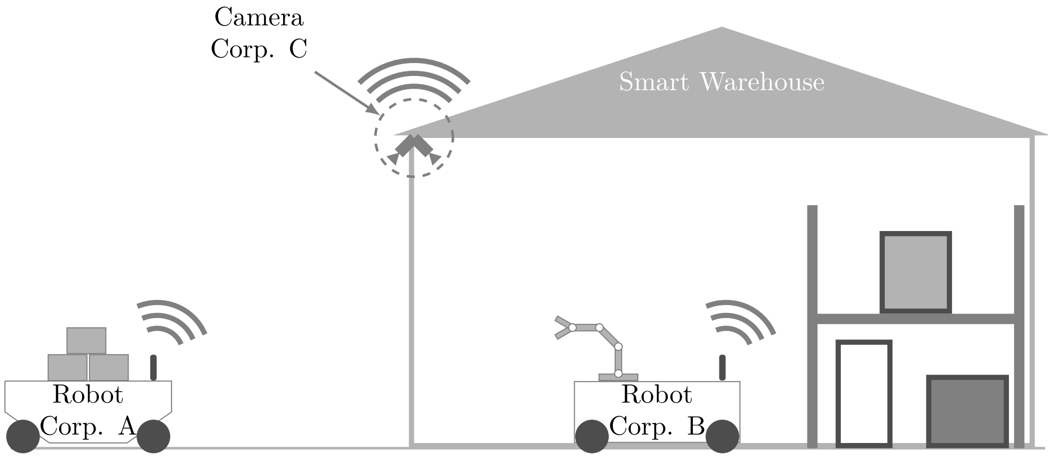


Fig. 1.1.: Schematic visualization of an automated delivery system interacting with a smart warehouse to safely transfer the transported goods.

identifying the arising challenges for guaranteeing safety in Section 1.2. These challenges form the basis from which the scope and objectives of this thesis are derived in Section 1.3.

1.1. The Example of Smart Warehouses

Smart warehouses [5], [6] are one example of the profound transformations enabled by the ideas of Industry 4.0. Their increased efficiency enables the rapid growth of e-commerce. A convincing feature is the so-called *Next-Day-Delivery* which promises customers that by finishing their order within a specified time, their bought products are delivered the next day.

Providing this service for a wide range of products is possible only through automation and information sharing as promised by Industry 4.0. For a smart warehouse, fulfilling the orders in minimal time is enabled by an automated storage system. Instead of having workers moving between racks to either restock them or pick items to fulfill orders, the automated storage system comprises a fleet of autonomous mobile robots moving the racks to designated restocking and picking stations. This eliminates the need for human workers to cover long distances between racks and enables them to focus on fulfilling the orders.

This changes the flow of commodities inside a smart warehouse. It now starts with the arrival of a delivery, commonly by a manually operated truck. Consisting of multiple products, the human workers identify the arrived items by scanning attached bar-codes. This very first step enables the smart warehouse and its automated storage system to become aware of the arrived items. They are brought to replenishment stations where human workers store the items in racks brought to them by the automated storage system. Once placed in a rack, which is observed by corresponding sensors, the warehouse's digital twin is updated. Thereby, not only the automated storage system is aware of the restock but also the online shop advertising the arrived products to its customers who can order these with only a minimal delay.

When an order is placed by a customer, the automated storage system is notified and instructs mobile robots to move the corresponding racks holding the items to the picking stations. Human workers are then fulfilling the customer's order. In a subsequent process, the selected items are packaged and sent to the customer.

Next to the employed technologies, a key-enabler for this efficient process is its static design. Although a multitude of sensors, versatile automation systems, and mobile robots are employed and combined to form a complex process, its components are determined at design-time. Moreover, human workers are employed at the intersection of technologies such that a static design is sufficient. For instance, the restocking and replenishment stations are designed in such a way that the workers have contact with exactly one rack brought to them by a mobile robot. The robots, on the other hand, are designed to operate in an enclosed space where no human workers[1] are allowed. Thus, the components and entities of the processes, as well as their interactions, are known completely at design time. Thereby, such a system fulfills the definition of a *statically composed system* [7].

Definition 1.1 (*Statically Composed System*)

> *A system whose components and their interactions are determined at design-time and remain static at run-time.*

As such, even highly automated smart warehouses scratch only at the surface of the opportunities provided by Industry 4.0. They predict flexible communication even between autonomous systems.

Combining, for instance, the idea of autonomous driving and smart logistics with the idea of a smart warehouse could provide an even higher degree of automation. Instead of having humans transport products to restock the warehouse, mobile robotic systems could provide automation of deliveries, cf. Fig. 1.1. A robotic system (Robot Corp. A in Fig. 1.1) could autonomously deliver goods to a smart warehouse. Once arrived at the destination, the warehouse could grant the robotic system access to deliver the goods to a specific location on site or even integrate with the automated storage system to directly replenish racks. Utilizing the available sensors (Camera Corp. C in Fig. 1.1) and sharing data with other mobile systems (Robot Corp. B in Fig. 1.1) operating the warehouse would further improve the process. For instance, sharing position data about static obstacles and other mobile systems can enable the arriving robot to avoid collisions and navigate safely.

However, it changes the nature of the system fundamentally. Instead of being a statically composed system, new sources of data, that is, shared data, are integrated at the system's run-time. The emerging system is therefore dynamically composed.

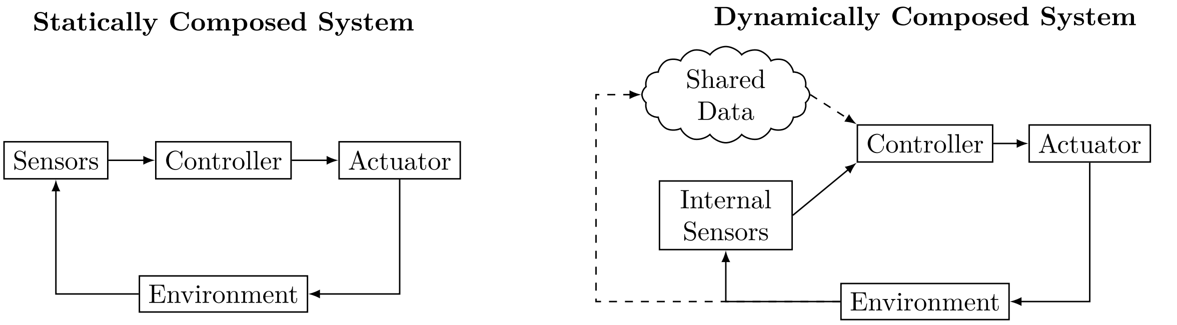
Definition 1.2 (*Dynamically Composed System*)

> *A system having an open architecture that enables the dynamic integration of multiple sources of data at run-time. These sources are considered unavailable and unknown at design-time.*

Considering the example visualized in Fig. 1.1, sources of shared position data, e.g. *Robot Corp. B*, is not known at design-time and become available only at run-time when the delivery robot arrives at a specific destination.

For the sake of clarity, both system types are abstracted and compared in Fig. 1.2. At a superficial level, a statically composed system observes its environment using local sensors, analyzes the provided data to determine appropriate control actions, and

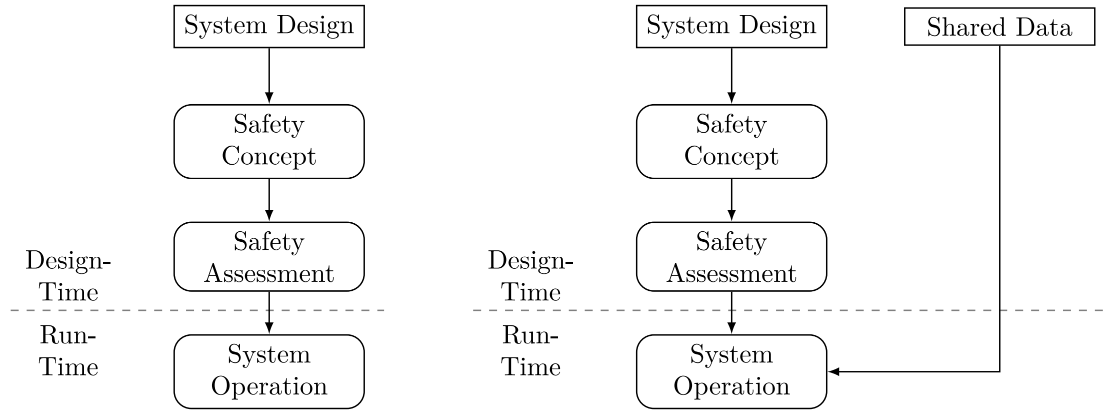
[1] With the exception of specialized maintenance personnel



(a) The simplified control loop of statically composed systems.

(b) The simplified control loop of dynamically composed systems.

Fig. 1.2.: Comparing statically and dynamically composed systems.



(a) Abstract safety process applicable to statically composed systems.

(b) Shared data are not considered during the safety process of statically composed systems.

Fig. 1.3.: Shared data in the safety process of statically composed systems.

performs these using actuators. The effect on the system's environment is observed through its sensors again, cf. Fig. 1.2a. The automated storage system, for instance, utilizes locally employed sensors to update the digital representation of the warehouse, its digital twin. Having knowledge about all available items enables it to instruct the mobile robots to move the correct racks to designated picking stations for fulfilling incoming orders. In turn, the mobile robots can be operated efficiently as all sources of data and their associated uncertainties are known at design-time. Minimal distances between moving robots can be maintained due to the design-time knowledge which thereby increases efficiency and safety.

In contrast, a delivery robot entering the smart warehouse and sharing its position data would require the smart warehouse and its internal mobile robots to temporarily incorporate the additional source of data. Represented by a cloud in Fig. 1.2b, the previously static control loop is temporarily extended by external sources of data. While this extension promises increased flexibility and efficiency as the environmental perception of all participating systems is increased, it requires a fundamental change in designing such systems. Statically composed systems are, by definition, complete at design-time. Therefore, knowledge about all components is available. In dynamically composed systems, components providing data, even safety-critical data such as the position of other robots, are available solely at run-time.

This has severe consequences for the safety process of such systems, cf. Fig. 1.3. The current safety process applicable to statically composed systems starts with an initial system design which is complemented by a safety concept. After implementing both, the system and measures ensuring its safety, the process of safety assessment provides evidence that the system can be operated safely. The system is only executed after the safety assessment is completed, Fig. 1.3a.

Contrarily, for dynamically composed systems, shared data is available only at its runtime, cf. Fig. 1.3b. As a consequence, it can not be taken into account during safety assessment. This means that it can not be shown whether or not the system will behave safely when using the shared data and being confronted with uncertainties that may be associated with it. Therefore, as a consequence of the current safety process, shared data can not be considered for safety-relevant functionality. However, when considering autonomous systems as envisioned for the delivery system integrating with the smart warehouse, shared data is safety-relevant as it directly affects the robot's decision-making. The question arises: **how can the safety of a dynamically composed system be guaranteed at design-time if critical information about its components is available only at run-time?**

1.2. Functional Safety Standards

In the endeavor of answering the question of safety in dynamically composed systems, it has to be refined and examined in relation to relevant safety processes. For that, safety is to be defined first to (a) differentiate between different perspectives and (b) identify functional safety as most relevant to this work. This discussion underlines the central role of functional safety standards and enables deriving the outline of this section.

In this work, safety is defined as follows [8].

Definition 1.3 (*Safety*)

> *The freedom from unacceptable risk of physical injury or of damage to the health of people, either directly, or indirectly as a result of damage to property or to the environment.*

In other words, a system is considered safe if the risk associated with its operation is acceptable [8], [9]. Conversely, risk is the central quantity for assessing safety and has to be reduced to achieve the same. Depending on the source of risk, different perspectives on safety can be distinguished [10].

Primary Safety Risk directly related to operating a system's hardware, e.g. electric shocks.

Indirect Safety Risk indirectly posed from incorrect system operation, e.g. racks are incorrectly replenished such that their center of mass is elevated, which results in an increased risk of rolling over.

Functional Safety Risk depending on the safe operation of the actual system in question.

In this work, the focus is on functional safety, which is defined as follows [9].

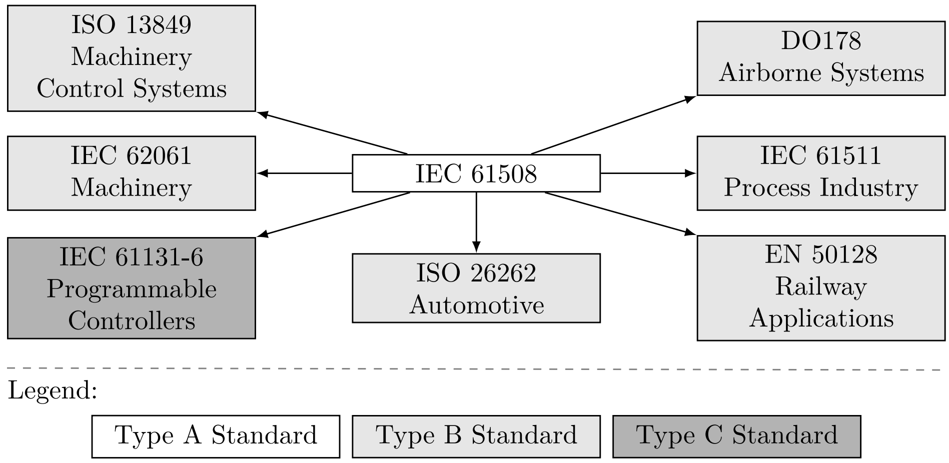


Fig. 1.4.: Non-exhaustive overview on industry-specific standards for functional safety. Figure inspired by [11]

Definition 1.4 (*Functional Safety*)

> *Functional safety is part of the overall safety that depends on a system or equipment operating correctly in response to its inputs.*

According to Definition 1.4, functional safety is an attribute that, once assigned, guarantees that a system performs operations to maintain a safe state when detecting potentially unsafe states.

Considering the mobile robots in a smart warehouse as described in Section 1.1, a potentially unsafe state occurs when their sensors detect an obstacle in their vicinity. The robot's collision avoidance then takes over to (a) re- or maintain a safe distance to the obstacle and (b) bypass the obstacle. To consider the robot safe, the correct functioning of the collision avoidance has to be proven at design-time. It is only then that the risk posed by collisions is shown to be sufficiently reduced such that the residual risk is tolerable.

For that, functional safety standards can be employed. Depending on the system under consideration and the targeted industry, different standards exist. In the endeavor of providing an overview, the next subsection discusses a non-exhaustive list of standards and identifies the IEC 61508 [8] as a central standard from which others are derived. Consequently, Section 1.2.2 reviews the standard in detail to determine challenges for guaranteeing the safety of dynamically composed systems.

1.2.1. Overview of Functional Safety Standards in Different Industries

To be accepted by society, physical systems are required to be functionally safe before brought into operation. While this means in general that the risk of operating these systems has to be at a tolerable level, legislative authorities are needed to define what is tolerable. In this endeavor, the targeted industry has to be taken into account. The benefits gained by operating systems have to be compared to the risk posed by them differently for different industries. From that consideration, different perspectives on tolerable risk as well as on how to achieve it developed over time and resulted in three different types of functional safety standards. Fig. 1.4 provides an exemplary overview of standards for some domains, which is discussed in this subsection.

Type C standards are most specific and target individual products. IEC 61131-6 [12], for instance, targets programmable controllers that may be used within safety-related systems. As such, not a specific safety function is addressed but rather controllers that can be used to implement such a safety function at a system level. Regarding the example of a smart warehouse, this standard can be applied to the embedded controllers implementing the navigation functionality of the mobile robots moving the racks.

Type B standards are more abstract and are domain- or system-specific. ISO 26262 [13] addresses functional safety of road vehicles while EN 50128 [14] and DO178 [15] are the corresponding versions addressing the railway or avionics domain. These types of standards can also address the manufacturing industry, e.g. IEC 61511 [16] which specifically targets process industries, or machinery within those industries, e.g. ISO 13849 [17] or IEC 62061 [18]. For instance, ISO 13849 [17] could be applied to the automated storage system of a smart warehouse and, individually, to its robots.

Lastly, type A standards are the most basic and provide general principles applicable across industries. IEC 61508 [8] is a prominent example that addresses "Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems". As such, the focus of this standard is on electronic systems in general. It, therefore, applies to embedded controllers used in the automotive industry, to control systems of airborne systems, and to the automated storage system of smart warehouses. Exactly this applicability across industries matches the transformative nature of *Industry 4.0* and its paradigms of *IoT* and *CPS* which result in dynamically composed systems. Therefore, the next subsection will discuss the approach to functional safety as prescribed by the IEC 61508 in detail and identify challenges for the usage of shared data in such systems.

1.2.2. IEC 61508

While the IEC 61508 was firstly introduced in 1998, the current revision dates to 2010. It is a normative standard that defines cross-industry processes and methods for designing safety-relevant programmable electronic systems. Following the definition of safety (cf. Definition 1.3), the standard takes a risk-based approach. It acknowledges the fact that risk can be reduced but not eliminated. In this endeavor, it proposes an overall safety life cycle comprising 16 phases (cf. Fig. 1.5) which guide engineers from designing the first concept of the envisioned system to generating an appropriate safety concept and to operating the system safely.

To analyze the safety life cycle in detail, the next paragraph introduces a selection of the terms used in IEC 61508 before the following paragraphs examine the individual phases with regard to the challenges arising for safety in dynamically composed systems. As some of the identified challenges are addressed by the standard ISO 21448 [19], its main characteristics and differences to IEC 61508 are briefly summarized in a concluding paragraph.

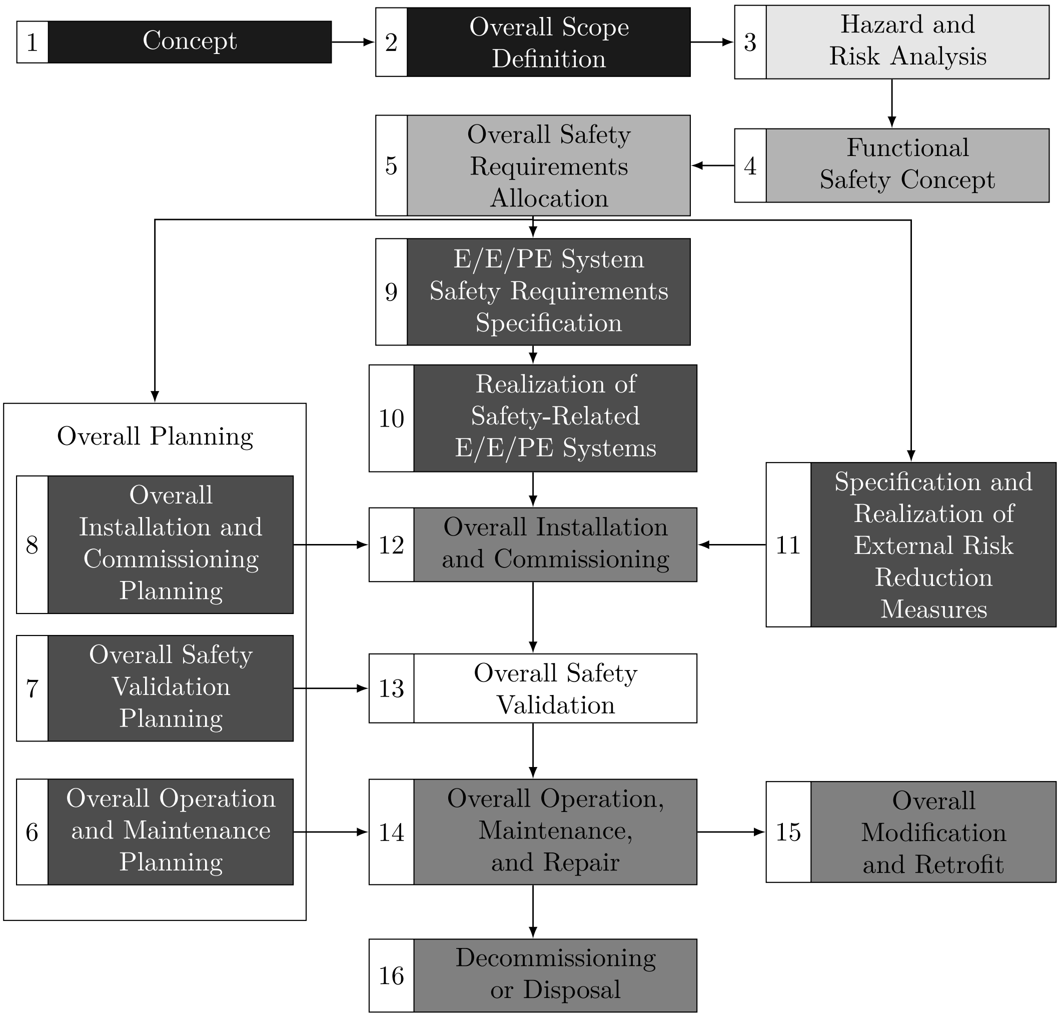

1 Concept
2 Overall Scope Definition
3 Hazard and Risk Analysis
4 Functional Safety Concept
5 Overall Safety Requirements Allocation
9 E/E/PE System Safety Requirements Specification
10 Realization of Safety-Related E/E/PE Systems
Overall Planning
8 Overall Installation and Commissioning Planning
7 Overall Safety Validation Planning
6 Overall Operation and Maintenance Planning
12 Overall Installation and Commissioning
11 Specification and Realization of External Risk Reduction Measures
13 Overall Safety Validation
14 Overall Operation, Maintenance, and Repair
15 Overall Modification and Retrofit
16 Decommissioning or Disposal

Fig. 1.5.: The figure lists the 16 phases of the safety life cycle of IEC 61508, which are arranged into 5 groups. The boxes are colored to highlight their affinity to these groups. Fig. 1.7 rearranges these groups and builds upon the coloring to clarify the associations. The figure is inspired by [11].

Terms of IEC 61508

Most central to the IEC 61508 is the definition of *Equipment Under Control (EUC)*, which enables differentiating between the initial system whose safe operation shall be achieved and other systems contributing to the actual safety. IEC 61508 defined *EUC* as follows [8].

Definition 1.5 (*Equipment Under Control*)

> *Equipment, machinery, apparatus or plant used for manufacturing, process, transportation, medical or other activities.*

The *EUC* is the hardware providing the function the overall system is designed for. For instance, the hardware platform of the smart warehouse's mobile robot would be a *EUC*. To fulfill its goal of moving from a starting position to an end position (e.g. to move one rack to a picking station) it requires a *EUC Control System (EUCCS)* which reacts to inputs and generates the required actions [8].

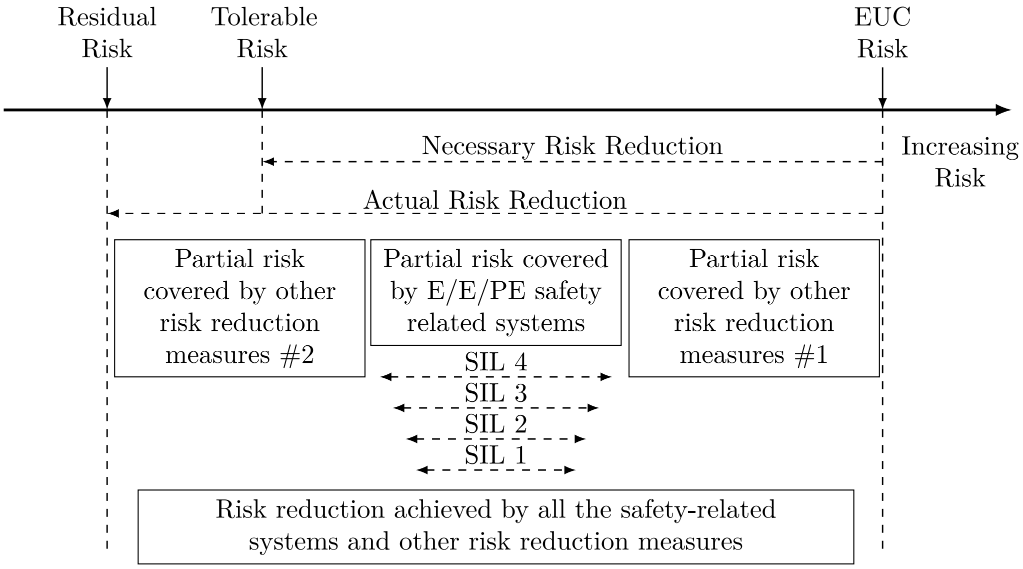

Residual Risk
Tolerable Risk
EUC Risk
Necessary Risk Reduction
Increasing Risk
Actual Risk Reduction
Partial risk covered by other risk reduction measures #2
Partial risk covered by E/E/PE safety related systems
Partial risk covered by other risk reduction measures #1
SIL 4
SIL 3
SIL 2
SIL 1
Risk reduction achieved by all the safety-related systems and other risk reduction measures

Fig. 1.6.: Risk reduction model of IEC 61508 showing the effect of different *Safety Integrity Level (SIL)* [8].

Definition 1.6 (*EUC Control System*)

> *System which responds to input signals from the process and/or from an operator and generates output signals causing the EUC to operate in the desired manner.*

The combination of a *EUC* and its control system provides the intended functionality of the system. For instance, the hardware of a mobile robot (*EUC*) and its navigation controller (*EUCCS*) together form a system. However, its operation may pose a risk, e.g. the risk of colliding with obstacles. In the endeavor of assessing the risk posed by a *EUC* and reducing it if necessary, the IEC 61508 assumes the Risk Reduction Model (cf. Fig. 1.6) and defines risk as follows [20].

Definition 1.7 (*Risk*)

> *Risk R is proportional to a measure for the probability P of an event (frequency, likelihood) and the consequences C of an event (impact, effect on objectives):*
>
> $$R = P \otimes C \tag{1.1}$$

In this definition, *proportionality* is abstracted to $\otimes$ and symbolizes that an application- and domain-specific derivation of risk from its parameters P and C is required. A multiplication, indicated by "$\cdot$" and used by other works, might not be sufficient.

Independently of the used symbol, risk depends on the likeliness of unwanted events to occur and their consequences. An unwanted event posing a potential safety-threat is called a *hazard*.

Definition 1.8 (*Hazard*)

> *A potential source of harm.*

with *harm* being defined as follows [8].

Definition 1.9 (*Harm*)

> *Damage to human health, to property, or to the environment.*

Therefore, the total risk posed by a *EUC* when not considering any safety-related measures is the sum of risks of all hazards.

For the smart warehouse and its automated storage system, this means that all hazards have to be considered. Colliding mobile robots moving the filled racks is only one hazard. Another hazard is the possibility of a rack falling over due to unevenly distributed load or a mobile robot not coming to a full stop due to overloading. The sum of the risks associated with these and other, not-listed hazards forms the total risk of the automated storage system.

To determine whether or not this is acceptable, the IEC 61508 defines tolerable risk [8].

Definition 1.10 (*Tolerable Risk*)

> *Risk which is accepted in a given context based on the current values of society.*

At this point, tolerable risk can not be defined in more detail as the standard does neither apply to a certain industry nor to a specific legislative authority. It is therefore an example of why type B and C standards are needed.

Despite its vague definition, the tolerable risk conceptually defines the maximal risk acceptable to operate a *EUC*. In other words, a system can be considered safe only, if the risk it poses is less or equal to what legislative authorities deem tolerable.

Conversely, if the initial risk posed by a *EUC* exceeds this threshold, risk reduction measures are required. More precisely, the difference between the initial risk of a *EUC* and the tolerable risk describes the required performance of risk reduction measures and thereby defines the required effort to provide safety for the system in question. This is described by the risk reduction model of IEC 61508 which is illustrated in Fig. 1.6 [8].

To derive appropriate risk reduction measures, safety requirements are formulated and associated with the identified hazards. Maintaining their fulfillment shall prevent the hazards from occurring once the system is in operation. Therefore, safety requirements are defined as follows.

Definition 1.11 (*Safety Requirement*)

> *Specification of operating condition(s) that have to be maintained by a safety function to guarantee the overall system's safety.*

According to this definition, a safety requirement states the condition(s) that have to be fulfilled for the overall system to be safe. Due to their association to specific hazards, each safety requirement enables a partial risk reduction, cf. Fig. 1.6. Depending on the overall required risk reduction, a set of safety requirements may be needed for the system to be safe.

For the automated storage system, for instance, each hazard has to be addressed by one or more safety requirements. One of which would address the necessity for avoiding collisions of employed robots.

However, safety requirements only enable risk reduction. For the risk to be indeed reduced, they have to be implemented such that the conditions they state are fulfilled. This is achieved by designated electric, electronic or programmable electronic safety-related systems. These safety-related systems implement a so-called *safety function*.

Tab. 1.1.: Available *Safety Integrity Level (SIL)* and their associated *Probability of Fails on Demand (PFD)* in IEC 61508 [8].

Safety Integrity Level	Low-demand mode of operation	High-demand mode of operation
4	$\geq 10^{-5}$ to $< 10^{-4}$	$\geq 10^{-9}$ to $< 10^{-8}$
3	$\geq 10^{-4}$ to $< 10^{-3}$	$\geq 10^{-8}$ to $< 10^{-7}$
2	$\geq 10^{-3}$ to $< 10^{-2}$	$\geq 10^{-7}$ to $< 10^{-6}$
1	$\geq 10^{-2}$ to $< 10^{-1}$	$\geq 10^{-6}$ to $< 10^{-5}$

Definition 1.12 (*Safety Function*)

> *Function to be implemented by an E/E/PE safety-related system that is intended to achieve or maintain a safe state for the EUC, in respect of a specific hazardous event.*

Thus, the safety function directly aims at preventing a specific hazard to occur, that is, preventing its consequences C by reducing its likeliness to occur (P). For that, the safety function itself has to be available, that is, be able to perform the required functionality, with high probability such that it does not fail when needed.

This is a direct consequence of the definition of risk, cf. Definition 1.7. If the safety function preventing a negative consequence is not sufficiently available, the hazard will occur with an intolerable probability nonetheless and the associated risk is not sufficiently reduced.

Therefore, the required probability for a safety function to be available is derived from the initial risk identified for a hazard. In the context of IEC 61508, it is termed as the *safety integrity* of the safety function. The standard defines four different levels to be assigned to a safety function.

Definition 1.13 (*Safety Integrity Level*)

> *Discrete level (one out of four), corresponding to a range of safety integrity values, where safety integrity level 4 is the highest and 1 is the lowest.*

Following the idea that a safety function reduces risk by limiting the likeliness of a hazard to occur, the *SILs* are mapped to a maximal *Probability of Fails on Demand (PFD)*, cf. Table 1.1. During the development of a safety function, it has to be shown that its *PFD* is less or equal to what is specified in the assigned *SIL*. In that regard, the *SIL* can be seen as the *safety performance* of a safety function.

The standard distinguishes between a low-demand and high-demand mode of operation of a safety function. A safety function assigned to a high-demand mode of operation generally has to have an increased availability, which in turn requires a reduced *PFD*.

For instance, the collision avoidance of a mobile robot in an automated storage system addresses the hazard of colliding robots and is, therefore, a safety function for which a *SIL* needs to be assigned. Moreover, as the robots operate in shared spaces, collisions might occur frequently (if not prevented) which puts the collision avoidance

functionality in a high-demand mode of operation.
While having a collision avoidance functionality in place is intuitive for a mobile robot, identifying necessary safety functions for a *EUC*, in general, does not have to be similarly straight forward. To provide a better understanding of the development of a hazard and consequently enable defining appropriate safety functions, the IEC 61508 adopts the notion of the *Fault-Error-Failure* chain [8], [10].

Definition 1.14 (*Fault*)

> *Abnormal condition that may reduce or stop the capability of a functional unit to perform a required function.*

As such a fault describes an internal state diverging from normal behavior. It affects a functional unit, which does not necessarily mean that it results in a hazard. However, it can be a source of a hazard. If not detected, a fault propagates through a system causing an *Error*.

Definition 1.15 (*Error*)

> *Discrepancy between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition.*

At this point, the error describes a state diverging from what is specified. Therefore, an error can be detected and, possibly, prevented from propagating. As such, detecting an error can form the basis of implementing a safety function.
Contrarily, if an error is allowed to propagate, it can cause a failure.

Definition 1.16 (*Failure*)

> *Termination of the ability of a functional unit to provide a required function.*

In other words, a failure of a functional unit occurs when it can not fulfill its specified function or service. It follows directly, that the behavior of a functional unit may vary from what is specified in versatile ways. These different aspects are described by its failure characteristics[2].

Definition 1.17 (*Failure Characteristics*)

> *The set of observable behaviors of a functional unit in case of a single or multiple errors or failures.*

This definition builds upon the concept of a functional unit as a component providing a specified service. Once it fails to provide this service, a failure occurs. As this is a consequence of an error, the set of errors also dictates the specific behavior of the functional unit in case of a failure. In other words, the failure characteristics of a functional unit describe its behavior in case of failure. Note that, within this work, a model or representation of failure characteristics is referred to as a failure model. This term is defined more clearly in Definition 2.2 in Chapter 2.
With respect to the automated storage system, one can consider a single distance sensor of a mobile robot. A fault of this sensor could be an incorrectly implemented rounding functionality. Once it is activated, it causes an internal error where the generated distance value differs from the true distance. If allowed to propagate, this incorrect value is provided to an application as a sensor observation. As the sensor does not

[2] The concept of failure characteristics is not part of IEC 61508 but is added here by the author for clarification.

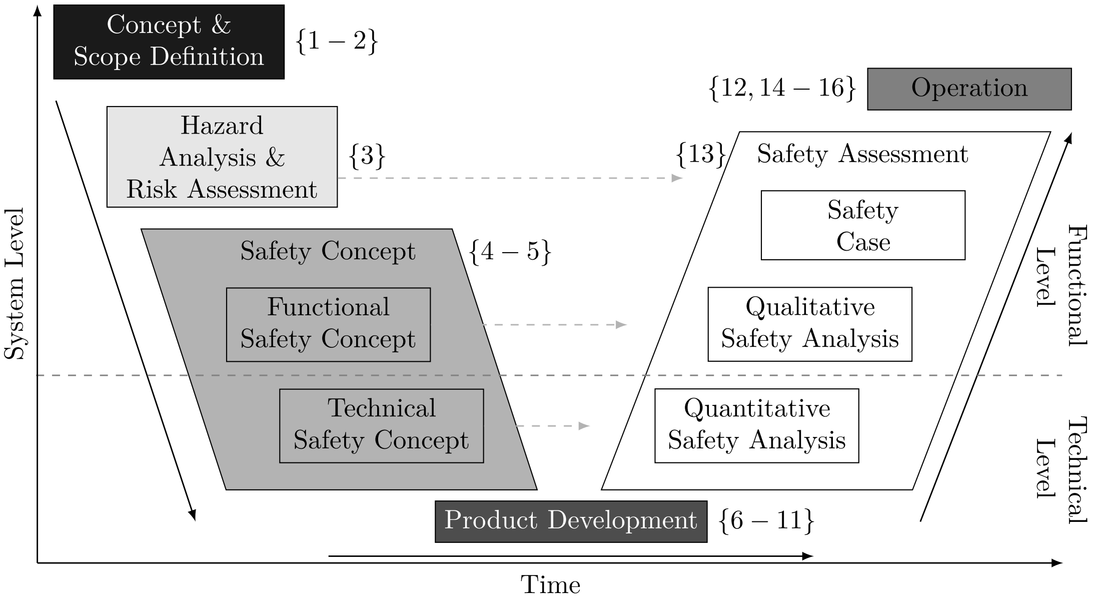


Fig. 1.7.: Simplified safety life cycle according to IEC 61508. The individual phases of Fig. 1.5 are referenced in braces.

adhere to its specified function, the sensor fails and a failure occurs. In this case, the difference between the specification and the provided observation determines its failure characteristics.

Within the IEC 61508, the failure characteristics of a functional unit are categorized into random failures or systematic failures. While random failures are commonly associated with wear and tear of hardware, systematic failures are caused by deficiencies of the system's design, for instance, an incorrectly implemented rounding functionality.

Both classes of failures are addressed independently by the standard. While the likeliness of random failures causing hazards is reduced by redundancy, systematic failures are addressed by following a prescribed development process which is part of the overall safety life cycle.

Safety Life Cycle of IEC 61508

The IEC 61508 is a normative standard. As such, it provides processes and methods to be applied during product development to design a safe system. These processes and methods are organized in a so-called *safety life cycle* based on the general life cycle of the system under consideration. To identify the challenges arising when such a system is a dynamically composed system, the IEC 61508 safety life cycle is discussed briefly.

For that, the 16 phases of the cycle defined by the standard (cf. Fig. 1.5) are arranged into 5 groups that form a V-Model[3], cf. Fig. 1.7. Each of these groups is discussed in detail in the following paragraphs.

[3] A development model that graphically represents the relation of each phase during software development with the corresponding testing phase [21]

Concept and Scope Definition Covering phases 1-2 of the IEC 61508 safety life cycle, the goal is to define the *Equipment Under Control (EUC)*, its control system (*EUCCS*), and its *Operational Design Domain (ODD)*. While the *EUC* and its control system describe the initial system and how it achieves the envisioned functionality, the *Operational Design Domain (ODD)* restricts its operational context. It defines under which environmental conditions the system can be operated and the assumptions made during operation respectively.

The first challenge for dynamically composed systems arises here. At design-time, neither the number of systems sharing their data nor the kind or quality of data being available at run-time can be foreseen. Moreover, the conditions under which the available systems may share their data are unclear.

Within the use case of a smart warehouse, situations may reach from a single delivery robot up to a great number of robots to support the scalability of the approach. Heterogeneity of the systems will even increase complexity and decrease the ability to specify an *ODD* in detail.

Challenge 1.1 (*ODD* Coverage) The increased complexity and limited knowledge about systems sharing their data decreases the ability to fully specify the *Operational Design Domain (ODD)* of a dynamically composed system at design-time.

It needs to be noted that Challenge 1.1 applies to complex systems operating on open environments in general and may not be limited to dynamically composed systems.

Hazard Analysis and Risk Assessment Given the system description and its operational context, the *Hazard Analysis & Risk Assessment (HARA)* has to be executed. Its goal is to identify all hazards of the *EUC* to determine its initial risk, specify functional descriptions of safety functions that shall mitigate the risk of each hazard, and assign corresponding *SIL* to them.

For the first step of identifying all hazards, resources such as historic data, field studies, or brainstorming sessions with involved engineers [10], are used. The process can be structured with methods such as *Fault Tree Analysis (FTA)* [22] and *Failure Mode and Effect Analysis (FMEA)* [23], which additionally enable analyzing their root causes and consequences. Both of which are central artifacts for the next step, the risk assessment.

Implied by the definition of risk (cf. Definition 1.7), the consequences of a hazard as well as its likeliness of occurrence have to be determined to assess the risk. Although not applicable for all *EUC*, the IEC 61508 proposes to split the likeliness of occurrence into three parameters, having in total four risk parameters (**C**, **F**, **P**, **W**). The parameters, their values, and descriptions are provided in Table A.1 in the appendix for reference.

The already mentioned parameter *Consequences* **C** captures the implications a hazard may have when occurred. Ranging from minor injuries ($\mathbf{C}_1$) to the death of many people ($\mathbf{C}_4$), the IEC 61508 focuses on harm to the health of people. The standard, however, notes that other classification schemes may be developed to take damage to property or monetary costs into account as well.

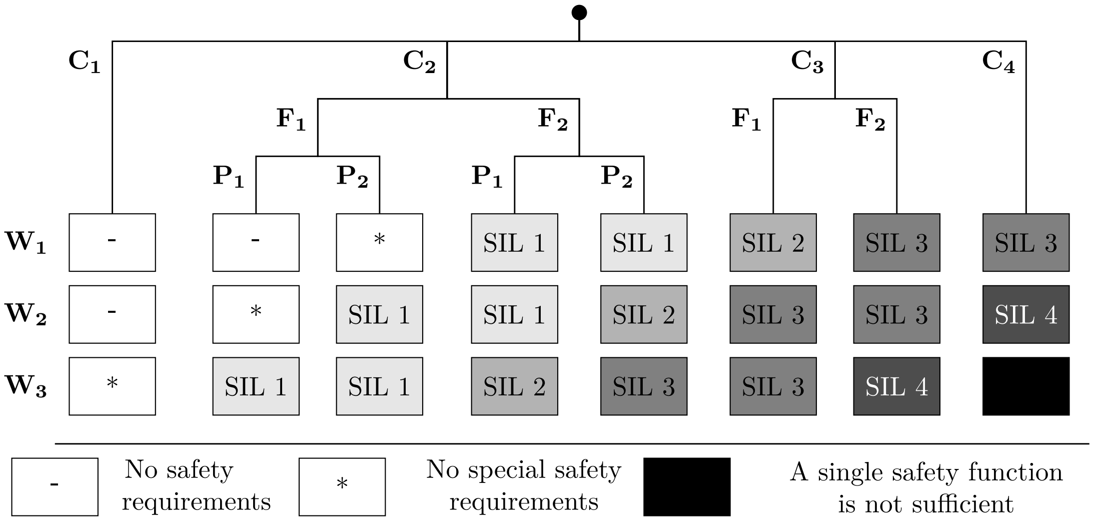

C_1
C_2
C_3
C_4
F_1
F_2
F_1
F_2
P_1
P_2
P_1
P_2
W_1 - - * SIL 1 SIL 1 SIL 2 SIL 3 SIL 3
W_2 - * SIL 1 SIL 1 SIL 2 SIL 3 SIL 3 SIL 4
W_3 * SIL 1 SIL 1 SIL 2 SIL 3 SIL 3 SIL 4
- No safety requirements
* No special safety requirements
A single safety function is not sufficient

Fig. 1.8.: Determining the required *SIL* of safety function depending on the risk of the associated hazard. Figure inspired by [24].

The remaining three parameters serve the purpose of classifying the likeliness of a hazard occurring. For that, the frequency **F** describes how often situations fostering the conditions for a hazard to occur while the probability **P** takes into account whether the hazardous event can be avoided by operating personal or external circumstances. Finally, the probability of the unwanted occurrence **W** describes how likely the occurrence of the hazard is without any implemented safety function.

Using these parameters, the risk of each hazard in question is assessed. From that, appropriate safety requirements, safety functions and their required safety performance (in terms of *SIL*) have to be determined. This can be achieved, for instance, using the risk graph of Fig. 1.8.

Hazards having only negligible consequences and are unlikely to occur do not entail any safety requirements while hazards with slightly increased risks may not need a *special* safety requirement. Any hazard deemed unacceptable, however, requires a safety requirement and a corresponding safety function of at least *SIL* 1. The level is increased up to the point where consequences and likeliness of occurrences require more than a single safety function.

It is only the set of these hazards and safety functions of at least *SIL* 1 that have to be accounted for during the following phases. The assumption is, that reducing the risk associated with them by implementing corresponding safety functions is sufficient to provide safety for the system. This, however, presumes that the set of identified hazards is complete and their risks are assessed correctly.

Contrarily, the concept of dynamically composed systems causes the lack of knowledge regarding its components during design-time, that is, when *Hazard Analysis & Risk Assessment (HARA)* is executed. As a result, relevant hazards might not be detected or their risk may be misjudged. Challenge 1.2 describes the former, while Challenge 1.3 addresses the latter.

Challenge 1.2 (Hazard Coverage) The inherent lack of knowledge about all components of a dynamically composed system at design-time poses a threat to the completeness of the hazard analysis. Relevant hazards might not be identified.

Challenge 1.3 (Risk Assessment) The failure characteristics of shared data can not be determined at a dynamically composed system's design-time. Thus, the risk of a hazard can not be fully assessed. This either results in overly restrictive safety functions or an underestimation of the posed risk.

Consider the example of a smart warehouse having a mobile robot with a robotic arm tasked to lift goods from a delivery robot, cf. Fig. 1.1. This, however, introduces the hazard of the robotic arm damaging the delivery robot or the transported goods. From the perspective of a manufacturer of a delivery robot, this hazard can be identified only if the option of a robotic arm lifting the goods is considered during *HARA*. Contrarily, when considering only the movement of the delivery robot and its goal to reach a target position, this hazard might not be identified.

Similarly, as described by Challenge 1.3, identified risks might not be assessed correctly due to missing information. Consider, for instance, the situation in which a delivery robot enters a warehouse. At this point, it might use the warehouse's internal localization system due to *Global Navigation Satellite System (GNSS)* being unavailable. Depending on the accuracy of provided position information, the robot might not be able to navigate indoors which might cause collisions. The risk of this hazard depends on the quality of the provided position information, which can be known only at run-time and not during *HARA* at design-time. As a consequence, the assigned *SIL* might not be appropriate.

Safety Concept Definition The *HARA* identified hazards and defined top-level safety requirements and safety functions with associated *SIL* for managing their associated risk. These, however, are defined in an implementation-independent manner and have to be refined to derive specifications for the actual product development. In this endeavor, the goal of the safety concept definition (comprising phases 4 and 5 of the overall safety life cycle) is to derive functional and technical safety requirements to guide the product development.

In the context of IEC 61508, no specific distinction between different types of safety requirements is provided. Therefore, the definitions of functional and technical safety requirements as proposed by the ISO 26262 [13] (a Type B functional safety standard derived from the IEC 61508) are used here.

The standard defines functional safety requirements as refinements of top-level safety goals which specify measures to reduce associated risk in an implementation-independent way. Thus, functional and logical entities of the preliminary system architecture are referenced without specifying their actual implementation. As a consequence, the first step is to refine the top-level requirements into functional safety requirements by following a top-down approach that takes the preliminary system architecture into account. The set of all functional safety requirements forms the *Functional Safety Concept (FSC)*.

The potential hazard of a mobile robot colliding with an obstacle, for instance, could be refined to the requirement that a minimum distance of $\hat{D}_{min}$ should be kept at all times.

The functional safety requirements are subsequently refined to technical safety requirements. They are implementation-dependent and support the proper realization of the intended safety functions. Therefore, they define the specific safety mechanisms to be implemented. Similar to the *FSC*, the set of all technical safety requirements form the *Technical Safety Concept (TSC).*

With respect to the collision avoidance of a mobile robot, the controller for maintaining a safe distance could be specified as a P-controller. This choice entails that target distance D_{min}, with which the P-controller might be configured, will not be met always, e.g. due to the well-known overshoot behavior [25]. A technical safety requirement therefore could be that $D_{min} > \hat{D}_{min}$ to ensure that the functional safety requirement is met nonetheless.

Product Development This group covers phases 6-11 of IEC 61508 in which the actual system is developed according to the previously generated specifications. Specific decisions are made on implementing functional and technical safety requirements, which are thereby refined to hardware and software requirements. Plans for maintenance, verification, and validation, as well as for installation of the system, are made.

Safety Assessment The safety assessment (phase 13) forms another critical activity in the IEC 61508 safety life cycle. Its objective is to reuse the artifacts of previous phases to generate the so-called *safety case* which documents that all safety requirements are met, all safety functions adhere to their assigned *SIL*, and ultimately the overall risk is reduced to an acceptable level. For that, not only the documentation of the overall development process is used but also quantitative and qualitative safety analyses are applied. Together, they provide evidence of safety on each abstraction level.

Starting with unit- and integration tests to show the fulfillment of hardware and software requirements, fault injection campaigns complementing simulations and field tests might be used to assess the fulfillment of technical safety requirements. Furthermore, these approaches can be used to generate reliability metrics, such as failure rates, which are inputs to quantitative safety analysis methods.

Approaches such as *FTA* [22] and *FMEA* [23] can be used in that regard again. In contrast to *HARA*, however, the focus is on the already implemented safety functions this time. They use a component-wise view of the system architecture and analyze it deductively (*FTA*) or inductively (*FMEA*). In the case of *FTA*, the interactions of the individual components are analyzed to track top-level failures to their possible causes. In *FMEA*, a qualitative safety analysis method, failures of individual components are considered and their effect on the overall system is analyzed. Causes, as well as effects, should now be covered by appropriate safety functions which thereby successfully mitigate the posed risks. On a functional level, field tests and system-level tests may provide empirical data to support the claims of fulfilled functional safety requirements and to show the integrity of implemented safety functions.

Such tests, however, require knowledge about failure characteristics to expect from the individual components of a system, which forms a challenge for dynamically composed systems.

Challenge 1.4 (Safety Assessment) Missing information about the failure characteristics of shared data does not allow assessing the effectiveness of implemented safety functions on mitigating hazards or their associated risk. Thus, their adherence to their assigned *SIL* can not be evaluated.

Especially the focus of quantitative safety analysis methods on reliability metrics is challenging when considering dynamically composed systems. Missing information about the uncertainty of shared data that is possibly used in safety functions prohibits assessing their effectiveness.
Considering the delivery robot arriving at a smart warehouse again where the performance of its collision avoidance now depends on the uncertainty of the position data shared by the warehouse or other mobile robots. Without knowledge about this uncertainty, it can not be assessed whether the implemented controller and its configuration of D_{min} can successfully prevent collisions.

Operation Finally, the safety assessment provided evidence for the safety case arguing the safety of the system by taking the implemented safety functions into account. This allows the system to be brought into operation, which covers phases 12 and 14-16 of the overall safety life cycle and concludes the same.

For dynamically composed systems, however, the safety life cycle can not be applied to guarantee safety without overcoming the identified challenges. These start as soon as defining the context and conditions for the operation of the envisioned system (cf. Challenge 1.1), span to the identification and assessment of hazards (cf. challenges 1.2 and 1.3) and finally address the question of how to evaluate whether safety has been achieved (cf. Challenge 1.4).

ISO 21448 – Safety of the Intended Functionality

Some of the challenges identified in the last subsection are not only valid for dynamically composed systems but became apparent also by the advent of technologies of *Machine Learning (ML)* and *Artificial Intelligence (AI)* [26]. Especially in the automotive sector, were *Advanced Driver Assistance System (ADAS)* and autonomous driving accelerate the development of complex functionalities, Challenge 1.1 and Challenge 1.2 where recognized. Thus, in 2019, the ISO 21448 - “Road vehicles - *Safety of the Intended Functionality (SOTIF)*” was introduced.

Definition 1.18 (*Safety of the Intended Functionality*)

> *The absence of unreasonable risk due to hazards resulting from functional insufficiencies of the intended functionality or by reasonably foreseeable misuse by persons is referred to as the Safety Of The Intended Functionality.*

With that definition, the standard does not focus on hazards caused by failure or malfunctioning of the system in question but by performance limitations of the intended function or unexpected operation conditions. In the endeavor of reducing the risk of such hazards, the normative standard emphasizes the idea of use cases and scenario-based development instead of the identification of requirements. It reflects the standard's assumption that neither the set of all hazards, nor all operating conditions can be foreseen at design-time. This is also acknowledged by the fact that the standard does not prescribe a methodology for classifying risk according to *SIL* or *Automotive Safety Integrity Level (ASIL)* (the automotive version of *SIL* according to ISO 262626 [13]). Instead, measures for assessing risk and whether or not it could be reduced is defined individually for each use case.

1.3. Scope of this Thesis

The challenges identified in the previous section clarify that a changed system design process is needed to guarantee safety in dynamically composed systems. Moreover, they enable deriving the scope of this thesis, which is the focus of this section.

In that endeavor, the previous section showed that challenges 1.1 and 1.2 are already addressed by the recently introduced standard ISO 21448 [19].

However, it does not propose a detailed process for risk assessment, but only vaguely defines that this depends on the scenarios and use cases defined for the system under consideration. This is in line with Challenge 1.3 which describes the problem from the perspective of dynamically composed systems. As knowledge about the failure characteristics of shared data is missing at design-time, the risk a hazard poses may be misjudged. The situation is aggravated by the fact that for determining the risk of a hazard, its contextual and environmental conditions have to be known. Although being overly restrictive, one solution assumed for this work is to presume the highest risk and therefore require the highest *SIL* (*SIL* 4) to be fulfilled by a safety function associated with the hazard.

Despite this restrictive assumption, safety can not be guaranteed because the missing knowledge remains the driver of Challenge 1.4. Assuming that a relevant hazard has been identified (e.g. collision of a delivery robot with other mobile robots in a smart warehouse environment) and a corresponding safety function has been implemented with *SIL* 4 (a collision avoidance controller), it still needs to be shown that the safety performance is actually met. In other words, the safety assessment of the implemented safety function has to prove that it successfully maintains a safe state. For that, the failure characteristics of shared data have to be taken into account, which is available only at run-time. Thus, the central question addressed in this thesis is:

How can a run-time safety assessment of a safety function guarantee the safety of a dynamically composed system?

In the endeavor of approaching this question, it is examined in the next subsection to derive the objectives of this thesis. Accordingly, Section 1.3.2 briefly discusses this work's contributions before an outline of the following chapters is given in Section 1.3.3.

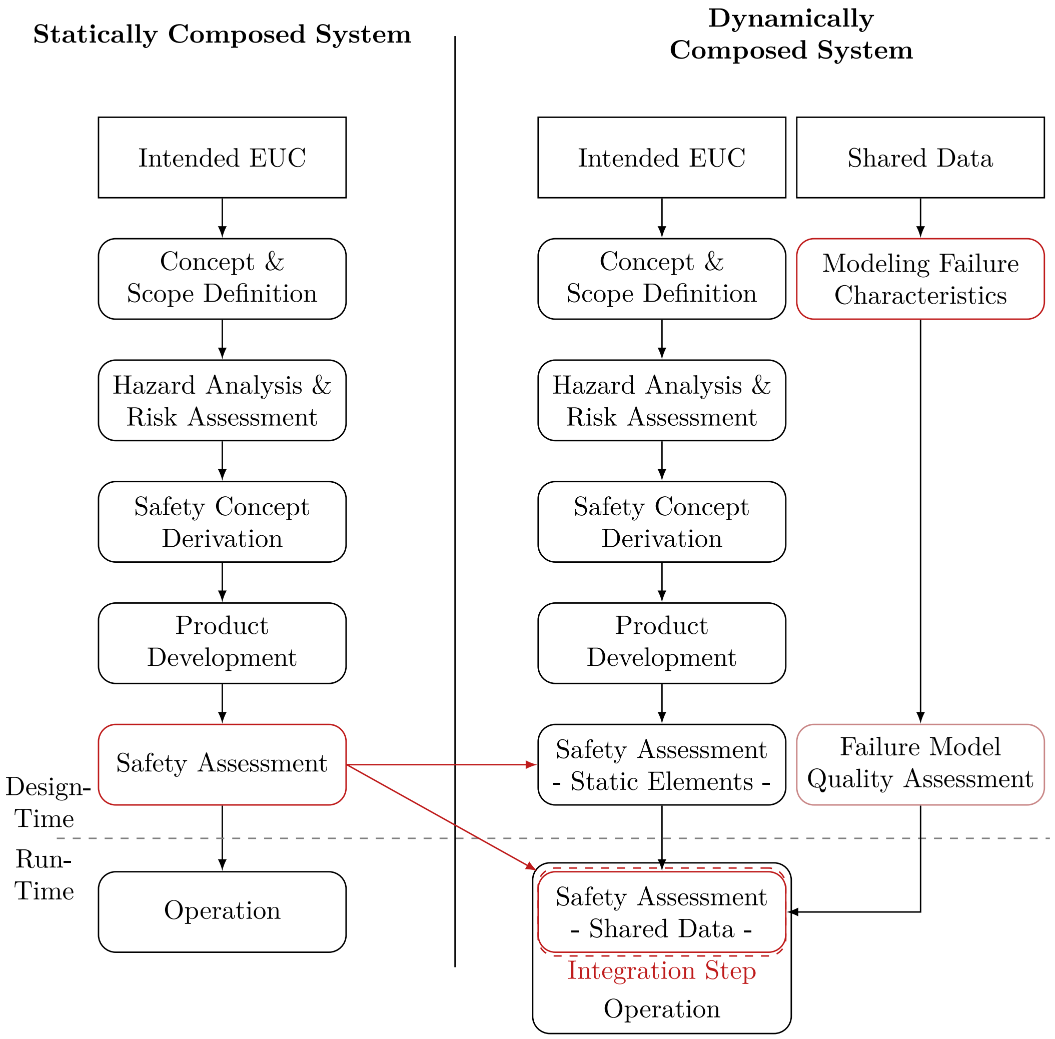

Statically Composed System
Dynamically Composed System
Intended EUC
Concept & Scope Definition
Hazard Analysis & Risk Assessment
Safety Concept Derivation
Product Development
Safety Assessment
Operation
Design-Time
Run-Time
Intended EUC
Shared Data
Concept & Scope Definition
Modeling Failure Characteristics
Hazard Analysis & Risk Assessment
Safety Concept Derivation
Product Development
Safety Assessment - Static Elements -
Failure Model Quality Assessment
Safety Assessment - Shared Data -
Integration Step
Operation

Fig. 1.9.: Comparing the abstract safety processes of statically and dynamically composed systems. The shift of parts of the safety assessment into the run-time is highlighted along with the activities addressed in this work.

1.3.1. Objectives

The discussion of the identified challenges clarified that for guaranteeing the safety of dynamically composed systems the activity of safety assessment has to be split and partially shifted into its run-time as it is only then that all required information is available. In this section, this shift is analyzed and the envisioned approach is discussed to derive objectives for this thesis.

In this endeavor, Fig. 1.9 visualizes the envisioned change in the safety process. While the left side shows the simplified safety process of IEC 61508 considering no shared data, the right-hand side schematically sketches required changes. The highlighted phases represent activities targeted by this work.

While the first phases involving the specification and development of the *EUC*, *EUCCS*, and its safety concept require to take sharing data with other systems into account, central changes have to be applied to the phase of safety assessment. It is proposed that the process of safety assessment is split into two parts. Static elements not using shared data can be assessed using traditional approaches following the process of IEC 61508.

Dynamic elements, that is, safety functions using shared data, have to be assessed at run-time. Such a run-time safety assessment forms the first objective.

Objective 1.1 (Run-Time Safety Assessment) The first objective of this thesis is to develop a run-time safety assessment method allowing to determine whether or not an envisioned safety performance is achieved by a given safety function that relies on shared data.

Such a method is a key-enabler to decide whether or not shared data can be used in a safety-critical functionality. This, however, requires that a description of the failure characteristics of shared data is available. From this, the second requirement follows.

Objective 1.2 (Generic Failure Model) The second objective of this thesis is to develop a generic failure model capable of representing failure characteristics of shared data in such a way that it can be used for run-time safety assessment methods.

For deriving what is necessary to fulfill both objectives, a set of criteria to assess the same is required. Correspondingly, the following paragraphs take the objectives into account and derive criteria for assessing their fulfillment in the following chapters.

Criteria for Run-Time Safety Assessment

The purpose of the run-time safety assessment is to guarantee that a safety function using shared data will meet its safety performance despite failures impairing the data. Consequently, the first criterion is that a suitable approach takes a shared failure model as an input to perform the assessment.

Shared Failure Model The run-time safety assessment method takes a failure model describing the failure characteristics of shared data as an input to derive whether or not a safety function can meet its required performance.

This criterion is central to ensure that the knowledge missing at design-time is incorporated into the decision process at run-time. This is required to guarantee at design-time that the method will maintain the safety of the system at run-time. To support providing this guarantee, the functional correctness of the approach has to be provable at design-time.

Functional Correctness The elements comprising the run-time safety assessment have to be proven functionally correct at design-time such that its result is valid at run-time.

In this sense, the approach provides an indirection for guaranteeing safety. Instead of showing that a safety function generally fulfills its envisioned safety performance, as it is done in IEC61508, it is ensured that the functionality assessing the safety performance of a safety function regarding the specific failure characteristics of shared data works correctly and comes to a valid conclusion. Fulfillment of this criterion enables to argue

the safety of the system by arguing the correctness of the implemented run-time safety assessment approach.
To supplement this criterion, it is required that a run-time safety assessment provides a *run-time certificate*.

Run-time Certification The central result to be generated by the run-time safety assessment is a binary decision on whether or not the safety function under consideration meets its safety performance when using a specific source of shared data. In other words, the run-time safety assessment certifies dynamically that a combination of a statically composed system with a source of shared data will either maintain its safety or not.

This criterion builds upon the central task of safety assessment, which is to check whether or not a safety function is meeting its required performance. Depending on the outcome of this check, the run-time safety assessment either certifies a dynamically composed system as safe or unsafe. With these two options available, the criterion implies that the possibility of not being able to use shared data due to safety concerns has to be accounted for at a system's design-time.
Finally, it has to be noted that the run-time safety assessment and these criteria rely on the fact that the shared failure model indeed represents the failure characteristics of shared data. That is, incorrect failure models shared with malicious intentions are not addressed in this work.

Criteria for Generic Failure Modeling

For the run-time safety assessment to decide whether or not a safety function meets its performance when using shared data, a model describing the failure characteristics that have to be expected is required. Described by Objective 1.2, a suitable failure model has to fulfill five requirements to meet the objective. As this question is partially addressed by Jäger, Zug, and Casimiro [7] already, the relevant criteria are repeated here.
First and foremost, the failure model needs to be interpretable by a run-time safety assessment method. As the system providing a failure model for shared data and the system interpreting the model to execute the assessment are possibly designed independently of each other, the interpretation of the failure model has to be clear. Therefore, the first criterion for a suitable failure model is *Clarity* [7].

Clarity The means used in a failure model to represent failure characteristics must be such that these characteristics will be interpreted unambiguously by an automated mechanism, e.g. a run-time safety assessment.

Fulfilling this requirement is mandatory to ensure a semantically correct safety assessment. Otherwise, a receiving system may misinterpret the failure model and potentially underestimate the threat the shared data poses to the system's safety.
Another aspect requiring an analytic interpretation is the requirement for supporting versatile methods of run-time safety assessment. Reaching from temporarily limited scenarios where only a superficial safety assessment can be executed up to long-lasting scenarios with sufficient computational resources enabling a fine-grained analysis of the represented failure characteristics. To cover this wide range, the failure model

has to be comparable to safety requirements under the conditions provided by an application [7].

Comparability For the flexible use of a failure model when comparing failure characteristics and application needs, the representation of failure characteristics must allow for interpretations with various levels of granularity.

Requiring comparability underlines the necessity to interpret a shared failure model within a run-time safety assessment. As already mentioned, these may be versatile and follow different approaches depending on the safety function whose performance shall be assessed. Therefore, comparability entails that the failure model is general in its definition to support a wide range of run-time safety assessment approaches [7].

Generality An appropriate failure model is required to have a generic approach to the representation of failure characteristics. This shall enable an application-independent description of failure characteristics that can be transformed into an application-specific representation when needed.

However, despite generality, a failure model remains a model which abstracts reality by omitting irrelevant information and focusing on application-specific information. In that sense transforming a model can not generate missing information. Therefore, a failure model for shared data should aim at providing a detailed description of the failure characteristics to enable receiving systems to generate their internal representations required for the implemented run-time safety assessment. Moreover, to enable versatile applications, a wide range of failure characteristics has to be representable [7].

Coverage An appropriate failure model must be capable of representing various failure characteristics in a versatile way.

Finally, based on the described failure characteristics, a run-time safety assessment is required to guarantee the safety of a system. However, this is possible only if one can rely on the correctness of the modeled failure characteristics. To express this, a confidence value should accompany a suitable failure model.

Confidence The confidence of the failure model combines the aspects of verification and validation to provide evidence that the modeled failure characteristics represent the actual characteristics sufficiently close.

The confidence value should be continuous to express belief in the correctness of what is modeled. It can be used by the receiving system to decide whether the modeled failure characteristics are accepted and further analyzed by a run-time safety assessment. In the latter, the safety function can not rely on the shared data, but the system's safety is not endangered by an invalid assessment result.

Fulfilling these requirements, a failure model is suitable to be shared from one system to another for a run-time safety assessment.

1.3.2. Contributions

The overarching goal of this thesis is to provide a run-time safety assessment that enables the safe use of shared data in a safety function. To that end, the two objectives identified in the previous section have to be fulfilled. In this thesis, they are addressed by the following contributions.

Generic Failure Model (GFM) The *GFM* is a model capable of representing versatile failure characteristics in an unambiguous way. Aimed at dynamically composed systems, it is mathematically defined to allow sharing it between systems. The receiving system can then interpret and analyze the failure model, e.g., to perform a run-time safety assessment.

In that endeavor, the model is defined as nothing but a set of so-called *failure types*[4] where each failure type represents a specific aspect of the overall failure characteristics. For that, a failure type comprises a deterministic failure pattern, describing its effect over time, as well as a stochastic scaling of the same. This representation is combined with distributions stating the conditions under which a failure type becomes active to acknowledge the random occurrence of failures. These elements of an individual failure type enable to represent systematic and random failures as described by the IEC 61508. Consequently, these aspects are available for the overall failure model.

Moreover, due to the fulfillment of the clarity criterion by the *GFM*, its parameters can be specified in an manual process. This allows reviewing the final parameters, modifying an existing failure model by hand, and designing failure characteristics. This transparency in the *GFMs* generation process supports its usage in both, safety assessments at design- and run-time. Finally, the definition of a confidence value enabling to assess whether or not failure characteristics are represented in their entirety complements this usage.

On the other hand, this flexibility in representing failure characteristics requires the model to have parameters encoding the same. For deriving these parameters from a given dataset and thereby simplifying the usage of *GFMs* a processing chain is proposed. It provides an automation for identifying and parameterizing failure types for generating failure models and ultimately enables determining its confidence values.

Region of Safety (RoS) With the *GFM* enabling to share a failure model along with the actual data in a dynamically composed system, the input for a run-time safety assessment is available. For addressing this task, the approach of *RoS* is proposed.

Building on the approach of *Region of Attraction (RoA)*, which provides guarantees for control systems on their asymptotic stability, the approach of *RoS* aims at guaranteeing that a control system does not leave a region of safe states. For that, a system model capable of integrating versatile sources of uncertainty and failure characteristics (one of which is shared data) is introduced in a first step. In a second step, the theorem of *RoS* enables estimating which regions of the state space are safe for a given control policy and with respect to the specified uncertainties.

Analyzing the produced *RoS*, therefore, informs about whether or not a system will be safe when using shared data. Moreover, due to its mathematical approach, the concept provides guarantees that the system will not leave the determined *RoS*. Therefore, when using shared data only if a valid *RoS* can be calculated at run-time provides the guarantee that the system will be safe already at design-time.

[4] A definition of a failure type is provided in Section 2.2.4

1.3.3. Outline

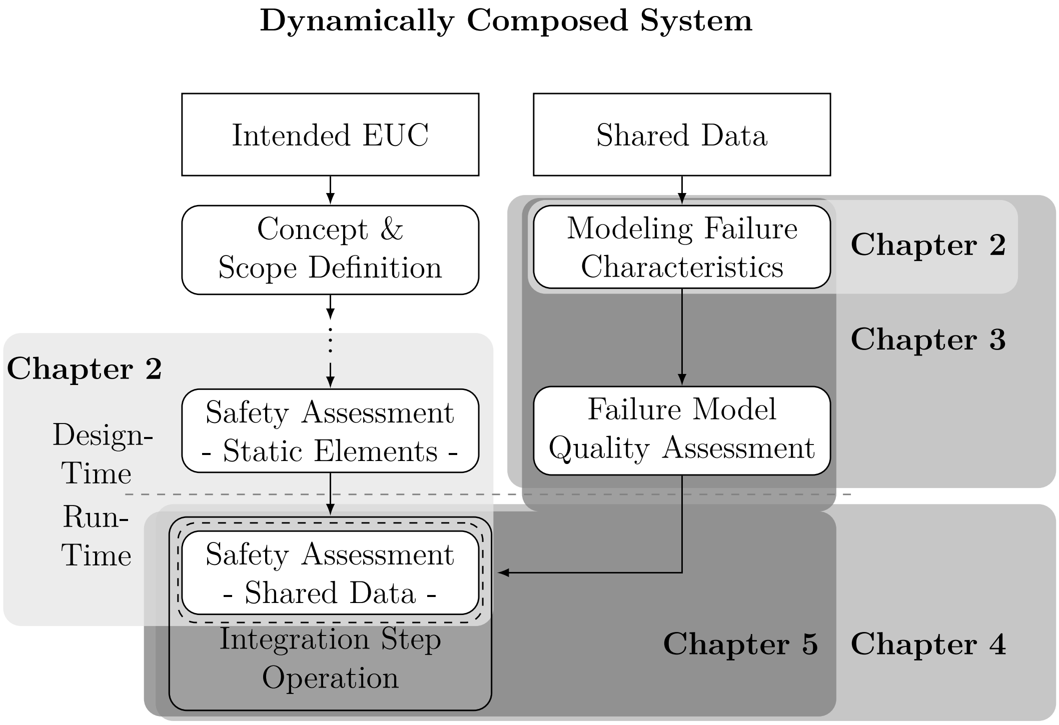

Dynamically Composed System
Intended EUC
Shared Data
Concept & Scope Definition
Modeling Failure Characteristics
Chapter 2
Chapter 3
Chapter 2
Design-Time
Run-Time
Safety Assessment - Static Elements -
Failure Model Quality Assessment
Safety Assessment - Shared Data -
Integration Step Operation
Chapter 5
Chapter 4

Fig. 1.10.: Simplified safety process from Fig. 1.9 showing the contents addressed by the individual chapters of this thesis.

The main contributions described in the previous section are derived, described, and evaluated in the following five chapters. The association of their contents with the abstract components of the introduced safety process is illustrated in Fig. 1.10.

Using the stated criteria to fulfill Objective 1.1 and Objective 1.2, the state of the art regarding both is reviewed in the following chapter.

For that, the first part focuses on works regarding safety assessment and ideas on how to execute the activity at a system's run-time. In the end, one concludes that approaches targeting the functional level are available but are missing for the technical level.

Similarly, approaches to modeling failures, especially sensor failures, are available but focus on statically composed systems without the need of communication descriptions of failures. Therefore, the clarity criterion, which is most central, is not fulfilled.

Having identified these gaps, Chapter 3 introduces the *GFM* and a corresponding processing chain for deriving a failure model from given time-series data. Using artificial data, the effectiveness of the processing chain and the fulfillment of the criteria are shown in a preliminary evaluation.

With a suitable approach to modeling failure characteristics at hand, Chapter 4 introduces the concept of *RoS* as an extension of the approach of *Region of Attraction (RoA)*. Motivated by the problem of the inverse pendulum, the concept of *RoS* is similarly evaluated preliminary as well.

However, as neither of the previous evaluations addressed the use case of a dynamically composed system, Chapter 5 takes upon the use case of the smart warehouse. Assuming a delivery robot operating next to other mobile robots, the safety function of collision avoidance is assessed using the concept of *RoS* and in regard with different failure models to show that not only different distances D_{min} are calculated in relation to different failure models but also the resulting parameterization of the safety function indeed maintains the safety of the overall system. With this evaluation in mind, the thesis is concluded in Chapter 6 and future work is described.

1.4. Related Publications by the Author

In accordance with the outline of this thesis, the author has published the following articles as the leading or co-author.
The following contributions were made to the state of the art and have lead to the central ideas presented in this work.

- G. Jäger, S. Zug, T. Brade, A. Dietrich, C. Steup, C. Moewes, and A. Cretu, "Assessing neural networks for sensor fault detection," in *2014 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, 2014, pp. 70–75. DOI: 10.1109/CIVEMSA.2014.6841441
- T. Brade, G. Jäger, S. Zug, and J. Kaiser, "Sensor- and environment dependent performance adaptation for maintaining safety requirements," in *Computer Safety, Reliability, and Security*, A. Bondavalli, A. Ceccarelli, and F. Ortmeier, Eds., Cham: Springer International Publishing, 2014, pp. 46–54. DOI: 10.1007/978-3-319-10557-4_7
- G. Jaeger, T. Brade, and S. Zug, "Using failure semantics to maintain safety for dynamic composed systems," in *ARCS 2016; 29th International Conference on Architecture of Computing Systems*, 2016, pp. 1–7
- J. Höbel, G. Jäger, S. Zug, and A. Wendemuth, "Towards a sensor failure-dependent performance adaptation using the validity concept," in *Computer Safety, Reliability, and Security*, S. Tonetta, E. Schoitsch, and F. Bitsch, Eds., Cham: Springer International Publishing, 2017, pp. 270–286

The initial ideas forming Chapter 3 were published in the following articles but are refined and corrected in this work.

- G. Jäger, S. Zug, and A. Casimiro, "Generic sensor failure modeling for cooperative systems," *Sensors*, vol. 18, no. 3, 2018. DOI: 10.3390/s18030925
- G. Jäger, K. Kirchheim, F. Schrödel, and S. Zug, "Multi-dimensional failure modeling for shared data in cooperative systems," *IFAC-PapersOnLine*, 2020, IFAC World Congress 2020

Similarly to the *GFM*, the initial ideas of Chapter 4, the concept of *RoS*, were published in the following article but are refined in this thesis.

- Analyzing Regions of Safety for Handling Shared Data in Cooperative Systems G. Jäger, J. Schleiss, S. Usanavasin, S. Stober, and S. Zug, "Analyzing regions of safety for handling shared data in cooperative systems," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 628–635. DOI: 10.1109/ETFA46521.2020.9211932

Moreover, the author was involved in application-specific publications focusing on generic, fault-tolerant robotic systems, which are listed below.

- G. Jäger, C. A. Mueller, M. Thosar, S. Zug, and A. Birk, “Towards robot-centric conceptual knowledge acquisition,” *Robots that Learn and Reason Workshop in IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018
- M. Thosar, C. A. Mueller, G. Jäger, J. Schleiss, N. Pulugu, R. Mallikarjun Chennaboina, S. V. Rao Jeevangekar, A. Birk, M. Pfingsthorn, and S. Zug, “From multi-modal property dataset to robot-centric conceptual knowledge about household objects,” *Frontiers in Robotics and AI*, vol. 8, p. 87, 2021. DOI: `10.3389/frobt.2021.476084`
- M. Thosar, C. A. Mueller, G. Jaeger, M. Pfingsthorn, M. Beetz, S. Zug, and T. Mossakowski, “Substitute selection for a missing tool using robot-centric conceptual knowledge of objects,” in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 972–979

1.5. Mathematical Notation

This thesis provides mathematical formulations complementing the linguistic description of the presented concepts. To support the reader, a brief description of the most central elements of the employed notation is provided hereafter.

Scalars and Vectors Variables are written in bold letters to denote vectors while not-bold letters indicate scalar values. For instance $x \in \mathbb{R}$ is a scalar, but $\mathbf{x} \in \mathbb{R}^n$ is a vector with $n \geq 2$.

Sets Sets are denoted using calligraphic letters. For instance, $\mathcal{X} = \{1, 2, \dots, N\}$ is a set of N integer numbers.

Domains Domains, for instance the set of real numbers, are written in double-struck letters. An example is the already used set of real numbers $\mathbb{R}$.

Euclidean Norm if not stated otherwise, the following notation is used to indicate the Euclidean norm: $\|\dots\|$.

2. State of the Art

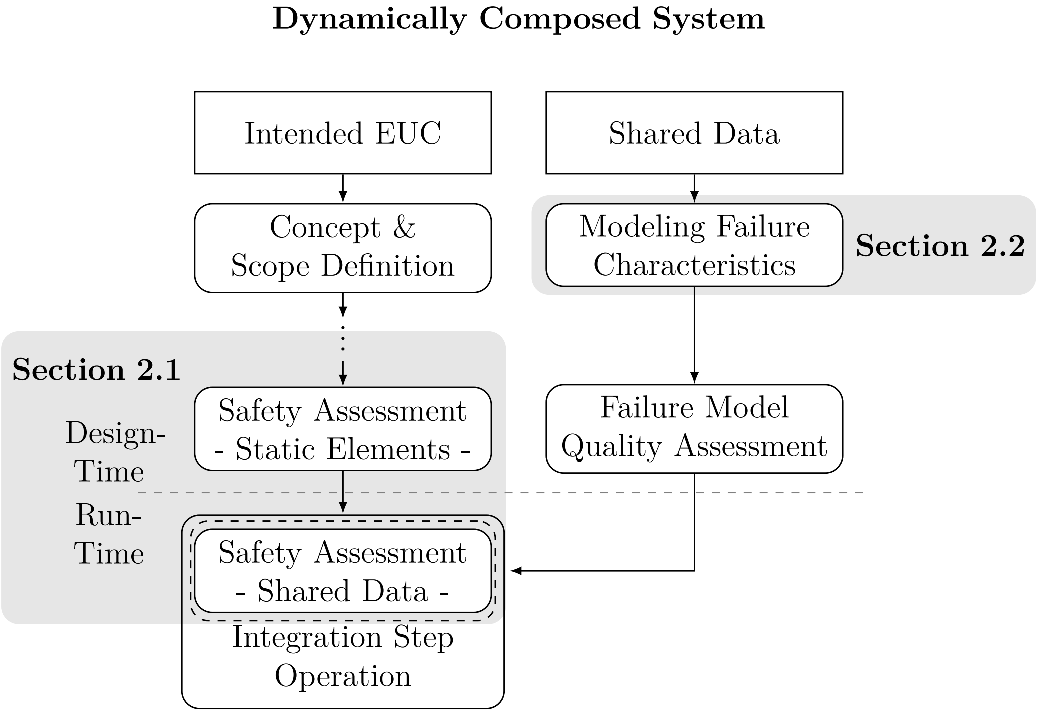

Dynamically Composed System
Intended EUC
Shared Data
Concept & Scope Definition
Modeling Failure Characteristics
Section 2.2
Section 2.1
Design-Time
Run-Time
Safety Assessment - Static Elements -
Failure Model Quality Assessment
Safety Assessment - Shared Data -
Integration Step Operation

Fig. 2.1.: Simplified safety process from Fig. 1.9 showing the contents addressed in this chapter.

Starting with the review of the IEC 61508 standard with its application to dynamically composed systems in mind, the last chapter motivated the need to shift parts of a system's safety assessment to its run-time to assess the performance a safety function can maintain when incorporating shared data. From that, two objectives of this thesis were derived. According to these, this chapter discusses state-of-the-art approaches.

An overview of the central sections is provided in Fig. 2.1. Section 2.1 targets Objective 1.1 and reviews work on run-time safety assessment. For structuring this discussion the presented approaches are categorized with respect to the level of system abstraction they are aiming at. In accordance with Section 1.2.2 and the phase of *Safety Concept Definition*, the functional (implementation-independent) and technical (implementation-dependent) level are distinguished[1].

At first, approaches targeting the functional level, that is, approaches considering implementation-independent safety assessment methods requiring only knowledge about the functional entities of a system are presented. Using the predefined criteria

[1] It needs to be noted, that approaches can not always be unambiguously associated with the functional or technical level, in which case the decision was made for the sake of structuring the argumentation.

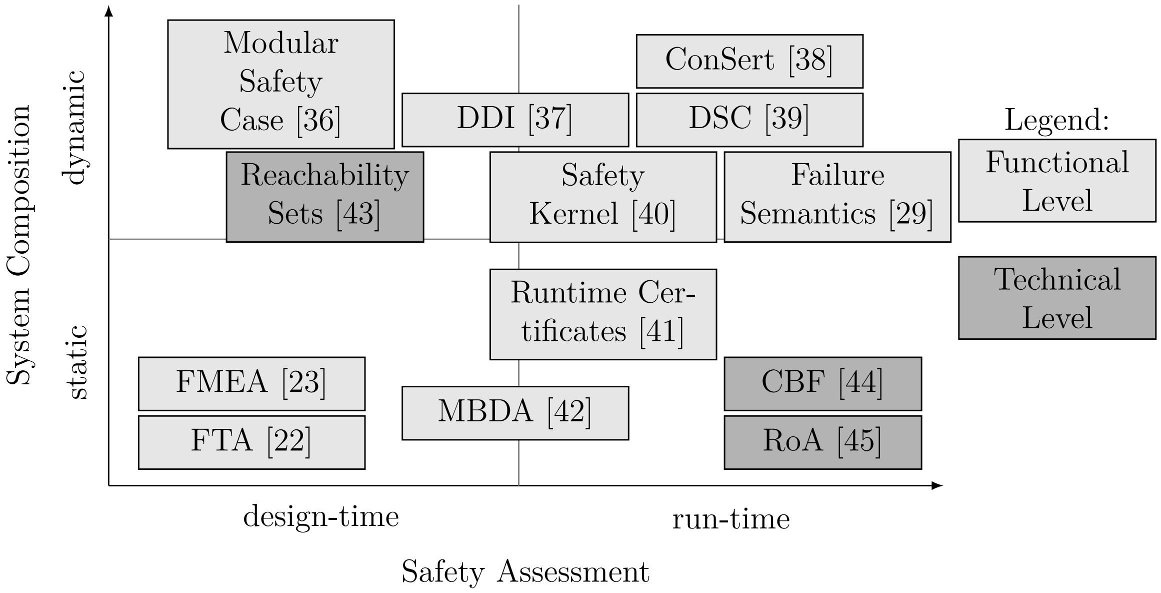


Fig. 2.2.: Overview and categorization of reviewed approaches to safety assessment.

presented in the previous chapter, it can be shown that different strategies towards run-time safety assessment exist.

In contrast, the same is missing for the technical level. At this level, approaches targeting specific safety mechanisms and implementation-dependent safety assessment strategies are in question.

However, the lack of those approaches motivates the development of a run-time safety assessment method focusing on the technical level. Following the predefined criteria, such a method shall facilitate analyzing the failure model of shared data for deciding on whether or not to use the same. Consequently, Section 2.2 targets Objective 1.2 and examines approaches to failure modeling. Due to the targeted level of control systems, sensor failure models are considered. They are assessed with respect to the criteria presented in Section 1.3.1. As none of the reviewed approaches fulfills these sufficiently, it is concluded that currently no failure model is suitable to be used in the envisioned use case of dynamically composed systems.

These findings are summarized and discussed in Section 2.3.

2.1. State of the Art in Run-Time Safety Assessment

The discussion on IEC 61508 in Section 1.2.2 brought not only challenges regarding dynamically composed systems to light but also clarified that safety has to be shown at all levels of abstraction. In that regard, the ISO 26262 [13] standard distinguishes between the functional (implementation-independent) and technical (implementation-dependent) level. Although not all approaches can be unambiguously categorized into one or the other level, the differentiation is used to structure the following subsections where approaches to run-time safety assessment are reviewed. An overview is given in Fig. 2.2. Next to the abstraction level, the system composition (static, dynamic) and execution time of the safety assessment (design-time, run-time) are studied. Using this classification the criteria from Section 1.3.1 are used to determine the applicability of the presented approaches.

For that, the next subsection reviews approaches targeting the functional level before approaches of the technical level are presented.

2.1.1. Approaches at the Functional Level

At a functional level of system abstraction, a wide range of approaches to safety assessment exist. Starting with traditional methods focusing on static system compositions (*FTA*, *FMEA*) at design-time, recent developments in *Model-Based Dependability Analysis (MBDA)* and efforts to shift safety assessment to a system's run-time are discussed. Finally, approaches directly targeting run-time safety assessment methods are presented.

FTA* and *FMEA As part of the IEC 61508 safety life cycle, safety assessment is traditionally applied to statically composed systems at design-time, cf. Section 1.2.2. Applicable already at *HARA* to identify relevant hazards and their associated risks, *Fault Tree Analysis (FTA)* [22] and *Failure Mode and Effect Analysis (FMEA)* [23] are common approaches for verifying that these risks have been successfully reduced by implemented safety functions during safety assessment as well. They consider the structure of the system in question to determine the origin and sequences of events causing failures or undesired system states.

For that, *FTA* starts with an undesired system state or failure and traces it back to its origin using Boolean logic. During this process, a tree structure of events is generated. The leaves of the tree state the basic events potentially leading to the considered failure. They can be associated with failure rates, which enables calculating the overall probability of the considered failure to occur. Having appropriate safety functions implemented, this overall failure rate associated with the occurrence of a hazard should be tolerable such that the system can be deemed safe.

For that, however, not only the components and failure rates of a system have to be available but also an analysis of events leading to a top-level failure or hazard. For dynamically composed systems, the calculation of an overall failure rate from a given *FTA* is possible at run-time in principle. However, deriving a fault tree as a result of a dynamic system composition at run-time is not reported yet.

It is precisely this analysis, which prevents using *FMEA* at run-time as well. In contrast to *FTA*, *FMEA* is a bottom-up, inductive approach. Using the architecture of the system in question, functional units, components, and subsystems are reviewed regarding their failure modes. For each failure mode, their causes and effects on the overall system are determined. Similarly to *FTA*, the approach can be combined with failure rates of components to derive a quantitative analysis.

Although this allows integrating failure models of components, the analysis of causes and effects is currently executed as brainstorming sessions of involved engineers. As such, the process is flexible but unstructured and can not be automated as part of a run-time safety assessment. Moreover, the process can not be evaluated to be functionally correct.

Model-Based Dependability Analysis (MBDA) On that account, the field of *Model-Based Dependability Analysis (MBDA)* aims at structuring not only the process of safety analysis but of dependability engineering in general. Assuming a system is designed with respect to models of its behavior, which are refined in an iterative process to implement the required functionality, the central idea is to reuse the generated models for dependability and safety analysis [42]. Thus, models of the system under consideration are analyzed using predefined algorithms to generate artifacts, e.g. fault trees, automatically. These are then used to calculate well-known failure rates and other dependability metrics.

This does not only reduces the manual effort required to generate a fault tree but does also increase the correctness of the approach as the generation process can be shown to be functionally correct. On the other hand, the lack of integrating failure models of shared data and descriptions of uncertainty hinders its application to dynamically composed systems along with the fact that only analysis of statically composed systems at design-time is targeted for now. However, the application of *MBDA* methods at run-time is proposed for future work [42].

Runtime Certification The idea of reusing artifacts from the design-time of a system at its run-time was formulated by Rushby [41] as well. They argue that the assurance (or safety) case of a system should not only be used as a safety guarantee at its design-time but provides opportunities for run-time analysis as well. In that endeavor, it is proposed to construct an assurance case with assumptions, goals, arguments, and elements of evidence to derive models that can be employed at run-time. They can be used not only for monitoring and failure detection but also to analyze occurred failures and synthesize corresponding recovery strategies. Through the explicit modeling of safety requirements, the adherence of generated recovery strategies to the same can be certified, which leads to the term "Runtime Certification" proposed by the authors. This approach additionally enables reacting to failures unknown at design-time, as these are analyzed at their occurrence at run-time.

As such, the approach does not only take a step towards run-time safety assessment but also fulfills the central requirement of being able to certify the result of the assessment. However, only models and information available already at design-time can be used, which requires a statically composed system. Consequently, no shared data or a failure model of its failure characteristics is supported.

Modular Safety Case An approach introducing the ability to dynamically integrate new components is presented by Jaradat *et al.* [36]. Motivated by the needs of Industry 4.0 and *Internet of Things (IoT)* where flexible factories of the future have to integrate new components of different suppliers frequently, the task of a system integrator to compile a complete safety case becomes costly and time-intensive. Therefore, the authors argue for a modular safety case allowing individual suppliers to contribute to the overall safety case. For that, a contract-based scheme is proposed which allows a supplier to explicitly state under which conditions or assumptions an element provides certain guarantees with the specified confidence. However, as a system integrator has to define the failure characteristics of the overall system in order to identify hazards and assess their risk (see *HARA* in Section 1.2.2) the elements additionally have to provide failure models. Identifying the challenge of completeness (cf. Challenge 1.2) of specifying the overall failure behavior, the approach targets dynamically composed systems but is limited to design-time.

Compared to the previous approaches, however, this work explicitly states the need for integrating failure models of not only shared data but of combined components in general. On the other hand, no process for assessing the safety of the resulting system is proposed such that the criterion of functional correctness can not be fulfilled.

Digital Dependability Identity (DDI) The challenge of sharing general dependability information for providing modularity in safety has been addressed by Schneider *et al.* [46] as well. They aim at formalizing dependability information to enable sharing the same and propose *Digital Dependability Identities (DDIs)* in that regard.

A *DDI* encompasses all information that uniquely describes the dependability characteristics, e.g. a failure model, of a component. For that, it is based on an open meta-model that enables different stakeholders to follow their own development processes but to derive *DDIs* that are exchangeable. The idea is that a component is developed at design-time where models generating dependability-related information are available. These are represented by a *DDI* which is certified and maintained during the component's lifetime. To bridge the gap to run-time dependability analysis, *DDIs* contain machine-readable information and models that may be used during run-time safety analysis. Therefore, *DDIs* are envisioned to support both, modular safety cases of systems integrated at design-time and run-time safety assessment of dynamically composed systems. However, no specific approach to the latter is described.

Conditional Safety Certificate (ConSert) In contrast, *ConSerts*, proposed by Schneider and Trapp [38], are considered as an implementation for using *DDIs* to certify the safety of a dynamically composed system at run-time [37]. *ConSerts* are based on the idea of services and contracts, which can be dynamically composed to generate top-level services. Similar to what is proposed in [36], *ConSerts* explicitly state for each component the assumptions under which it was developed and the safety demands that need to be fulfilled by the integrating environment to provide the stated safety guarantees. At run-time, these assumptions, demands, and guarantees are exchanged and need to be verified before integrating components. This requires two concepts.

Firstly, depending on the service that needs to be composed, additional *Runtime Evidence* needs to be acquired. The authors mention the independence of services, which can be evaluated only at run-time, as an example. However, one can think about analyzing a shared failure model to generate such run-time evidence as well. Secondly, functions that map a configuration of fulfilled/unfulfilled safety demands to a set of safety guarantees that can be given. Although any function is possible, it is proposed to use Boolean functions. With these functions, the safety of a specific configuration of a dynamically composed system can be assessed at run-time. Moreover, the rigorous checking of safety requirements enables to specify an algorithm whose functional correctness can be shown at design-time and which certifies the validity of a specific system combination at run-time.

Dynamic Safety Contract (DSC) Similar to *ConSert*, Müller and Liggesmeyer [39] build upon the idea of contracts. They acknowledge that safety-related data needs to be exchanged between systems to perform a run-time safety assessment of the dynamically composed system. To exchange the relevant information, they propose qualitative and quantitative safety contract modules called *DSC*. These state not only the safety demands that a component has and guarantees that it provides but also variants of these contracts. Comprising of inputs and output ports, each safety contract module explicitly states its requirements. The inputs may be run-time knowledge from other safety contract modules, which provides modularity, but also design-time knowledge, such as safety certifications. The central idea is, that safety contract modules can be combined to form required system services, such as *Cooperative Adaptive Cruise Control (CACC)*, the example given by the authors. For that, qualitative safety contracts allow binary checking whether or not a functionality can be guaranteed, while quantitative safety contracts facilitate gradual degradation through variants of the contract, for instance, to increase the safe distance to a leading vehicle in case of low quality (shared) data. Therefore, the approach does not only enable certifying a dynamic composition of safety contracts at run-time but also explicitly allows predefined levels of performance degradation. Depending on the quality of available data and modules, the level is adjusted to maintain safety.

Safety Kernel and *Level of Service (LoS)* While the work of Müller and Liggesmeyer [39] remains on a conceptual level, independent of specific implementations, the KARYON project [40], [47] proposed a system architecture for safe cooperative functions based on a safety kernel. Its task is to monitor the quality of data shared for cooperative functionalities in order to adapt the system's *Level of Service (LoS)* accordingly. Thus, instead of disengaging from cooperative functionalities when encountering reduced quality of sensory data, the performance of the functionality is reduced and safety is thereby maintained.

For that, the approach builds on the validity concept [48] which abstracts a sensor failure model to a scalar value informing an application about the confidence it can have in an observation. At design-time, the concept allows assessing the quality of data to be expected from a component while it enables using failure detectors and filters to update this representation at run-time. Consequently, at run-time, each component produces an output signal that is attributed with a validity value. This run-time representation is evaluated by the safety kernel, whose functional correctness is proven at design-time already. The kernel uses a rule-based system to adjust the *LoS* of the system, which eventually reduces performance but maintains safety. Depending on the chosen *LoS*, components providing cooperative functionality may be available or deemed unsafe. Assuming that cooperative functionality is possible only in higher *LoS*, the lower (and lowest) levels are restricted to non-cooperative functionalities that require inputs/outputs ensuring safety based on local components only.

Opposed to other approaches of the functional level, the concept of *LoS* does not directly provide complete modularity. Instead, it is assumed that the described system architecture is implemented in all systems participating in a cooperative maneuver. This, however, enables proving the safety kernel's functional correctness already at design-time, which ultimately provides confidence that shared data is used only when it is safe. In contrast, for the approach to work shared data has to be attributed with validity statements. As these are based on sensor failure models, their interpretation may be ambiguous, which endangers the validity of the result of the run-time safety assessment. Nevertheless, a prototypical implementation of the safety kernel in real hardware could be presented [40], which supports the conceptual idea.

Failure Semantics In its core, the safety kernel aims at comparing the quality of shared data with what is tolerated by the application under consideration. A similar idea is presented by Jaeger, Brade, and Zug [29]. The approach defines failure semantics from two perspectives: sensor-specific and application-specific. From a sensor's perspective, failure semantics describe the failure characteristics that are to be anticipated. From an application's perspective, a failure semantics states the tolerable failure characteristics. Based on a ranking, a matching of failure semantics is provided to dynamically decide on whether an application can tolerate the failure characteristics of a sensor or not. Although this takes upon the idea of comparing what is provided with what is required, a proof-of-concept evaluation is missing.

In summary, independently of the specific approach (failure semantics, *DDI*, *ConSert*, modular safety case or *LoS*), the presented concepts build upon the same ideas. Explicitly and unambiguously modeled information about the quality of data has to be shared between systems to enable cooperative functionalities and enable run-time safety assessment. Regarding the latter, two main approaches are identified. On the one hand, contract-based approaches are presented that promise truly open systems but require extensive efforts in modeling all required dependability information. On the other hand, the KARYON project proposed a system architecture based on the validity concept and a safety kernel that enables safe performance degradation in case of reduced sensor data quality at the expense of flexibility.
Although a prototypical implementation is presented in the latter case, both approaches remain on a functional level and refer to the fact that run-time evidence is required to finalize the run-time safety assessment. Accordingly, in the next section approaches aiming at providing safety assessment functionality at a technical level are reviewed.

2.1.2. Approaches at the Technical Level

In accordance with IEC 61508 and other safety standards, safety has to be shown at all system abstraction levels. Moreover, approaches to run-time safety assessment targeting the functional level were shown to rely on evidence generated by the technical level.
At this level, safety is addressed with respect to the state space of a control system under consideration. From that, two nonexclusive perspectives arise. On the one hand, safety can be provided in terms of stability of the control system as it shows that its behavior is predictable. On the other hand, safety can be shown by matching safety requirements to the states of the control system and showing that only states fulfilling the requirements are reachable. As reachability is closely related to stability, both aspects are addressed with regard to stability analysis of control systems. Therefore, in this subsection, approaches from this field are reviewed regarding their potential contribution to a run-time safety assessment method.

Reachability Analysis One of such stability analysis methods is *Reachability Analysis*. It is used by Kianfar, Falcone, and Fredriksson [43], for instance, who consider the exemplary scenarios of *Adaptive Cruise Control (ACC)* and *CACC*. During the *CACC* scenario, the vehicles are assumed to share their acceleration information. Within both scenarios, the safety of participating vehicles is maintained as long as their distance is greater than zero, that is, no collision occurs. In the endeavor of guaranteeing this, Kianfar, Falcone, and Fredriksson [43] use reachability analysis and invariant set theory.

Firstly, the authors assume a set of initial states. These define the distances the vehicles may have at the beginning of the driving scenarios. By simulating the vehicles using their kinematic model the set of all states the system may evolve to over time is determined. This forward reachability analysis allows differentiating between states resulting in collisions and collision-free states. As a consequence, the minimal safe distance to keep between both vehicles for maintaining their safety is determined.

Using backward reachability, where one asks what initial states a system might have been in to arrive at a given set of states, a *maximal asymptotic safe set* is calculated under the consideration of the previously defined minimum safe distance. This set comprises only states for which the system is guaranteed to stay within the set for all times. Given that this set does contain only states fulfilling the safety requirement, the safety of the system is guaranteed.

With this approach, the authors show that the safety of a system using shared data can be guaranteed. However, failures and uncertainties impairing the quality of the data are not considered. Furthermore, the analysis is performed only at design-time.

Control Barrier Function (CBF) In contrast, Cheng *et al.* [49] apply *CBFs* to guarantee the safety of a system during *Reinforcement Learning (RL)*, that is, at the system's run-time.

RL generally aims to solve the optimal control problem, originated in classical control theory, through machine learning and direct interaction with the system in question. The key idea is that taking an action $\mathbf{u}$ at a system state $\mathbf{x}$ produces a *reward* signal from the controlled system that informs about the appropriateness of the chosen action. Leveraging this signal as a feedback mechanism, *RL* aims at optimizing a control policy $\mathbf{u} = \pi(\mathbf{x})$ to maximize the cumulative reward over time. For that, phases of exploration (learning about the controlled system through random actions) and exploitation (applying currently optimal policy to maximize reward) alternate [50].

When directly executed on a physical system, for instance on a mobile robot, the exploration phase poses a safety threat to the system and its surroundings due to the random actions taken. Therefore, Cheng *et al.* [49] propose to use *CBFs* to provide a safety layer. A *CBF* can be considered as a cost function that drastically increases for unsafe states. It thereby encodes safety requirements.

Depending on the structure of the specified *CBF*, a controller adhering to the encoded safety requirements can be synthesized by solving the corresponding minimization problem. Cheng *et al.* [49] refer to these as *CBF* controllers. They are used to shield unsafe control actions during exploration and leveraged to guide the learning process during *RL*.

With this approach, the authors show that safety can be assessed at run-time using appropriate cost functions (*CBF* in this case). Furthermore, the process can be shown to be functionally correct, that is, it can be shown that the controller synthesized by minimizing the *CBF* maintains the safety of the system. However, neither a run-time certification is provided nor the integration of shared data or an analysis of its failure model.

Region of Attraction (RoA) **Estimation Using** ***Control Lyapunov Function (CLF)***

Similar to the usage of *CBF* as a cost function to assess safety, the stability of a given controller can be analyzed using a *Control Lyapunov Function (CLF)*. With such a function, an *RoA* for the controller under consideration can be estimated. The *RoA* is a set of states for which the controller is guaranteed to provide stabilizing control actions. For that, it is shown that at each state in the *RoA*, the *CLF* is minimized by the chosen control action [45]. The idea is visualized schematically in Fig. 2.3.

As a prerequisite to calculating a *RoA*, a continuous-time system model given as a system of *Ordinary Differential Equations (ODEs)* and a controller π of the control system in question are required, cf. Eqs. (2.1) and (2.2).

$$\dot{\mathbf{x}}(t) = \frac{\partial \mathbf{x}}{\partial t} = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{2.1}$$

$$\mathbf{u}(t) = \pi(\mathbf{x}(t)) \tag{2.2}$$

For the sake of simplicity, the time index t will be omitted in further equations. Additionally, a *CLF* encoding the distance to the control goal is required. It is a monotonically increasing, continuously differentiable function $V(\mathbf{x}) : \mathcal{X} \rightarrow \mathbb{R}_{\geq 0}$ with a global minimum of $V(\mathbf{x} = \mathbf{0}) = 0$. Similar to a *CBF*, it can be interpreted as a cost function that the controller aims to minimize over time. Intuitively, the set of states surrounding the global minimum for which the controller succeeds in this task is the *RoA*. Lemma 2.1 formalizes this idea [32, Lemma 1].

Lemma 2.1. *The origin of the dynamics in Eq.* (2.1) *is asymptotically stable within a level set,* $\mathcal{V}(c) = \{\mathbf{x} \in \mathcal{X} | V(\mathbf{x}) \leq c\}$ *with* $c \in \mathbb{R}_{>0}$ *and if* $\forall \mathbf{x} \in \mathcal{V}(c)$*:*

$$\dot{V}(\mathbf{x}) = \frac{\partial V(\mathbf{x})}{\partial t} = \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}}{\partial t} = \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \cdot f(\mathbf{x}, \mathbf{u}) < 0 \tag{2.3}$$

According to this lemma, a non-empty set $\mathcal{V}(c)$ is provided if the given controller π is able to asymptotically stabilize the system. Essentially, it means that at every state $\mathbf{x} \in \mathcal{V}(c)$ it can be shown that the system is driven closer to the stability point and therefore the gradient over time is negative, $\dot{V}(x) < 0$.

As this statement is directly derived from the gradient of the chosen *CLF*, its correctness is influenced by the function's appropriateness to reflect stability. Constructing a suitable *CLF*, however, is a challenging task. As no one-fits-it-all approach could be defined yet, it is a field of active research [44] and requires expert knowledge.

While Lemma 2.1 is defined on a continuous state space, checking the gradients of all states for determining the *RoA* is intractable. To solve this issue, Berkenkamp *et al.* [45] propose using Lipschitz continuity [32, Definition 1].

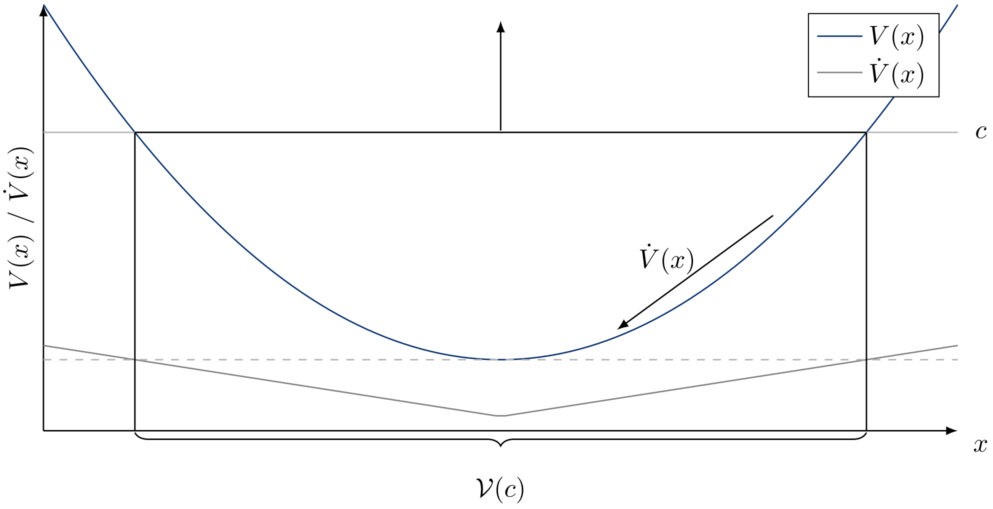


Fig. 2.3.: Estimating *Region of Attraction (RoA)* by maximizing c of Lemma 2.1.

Definition 2.1 (*Lipschitz Continuity*)

> *A function $\alpha(\mathbf{x}) : \mathcal{A} \to \mathcal{B}$ is called Lipschitz continuous if there exists a constant $L \in \mathbb{R}_{\geq 0}$ for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{A}$ such that:*
>
> $$\parallel \alpha(\mathbf{x}_1) - \alpha(\mathbf{x}_2) \parallel \leq L \cdot \parallel \mathbf{x}_1 - \mathbf{x}_2 \parallel \tag{2.4}$$

If L exists, it is referred to as a *Lipschitz constant* of the function $\alpha(\mathbf{x})$. Intuitively, it can be considered as an upper limit for the gradient of the function. Building upon that, a function is called *locally Lipschitz continuous* if one can find Lipschitz constants that are valid only for subsets of its domain.

Assuming that the gradient function $\dot{V}(\mathbf{x})$ is (locally) Lipschitz continuous with a Lipschitz constant $L_{\dot{V}}$, Eq. (2.3) can be refined for a discrete state space $\mathcal{X}_\tau$, cf. Eq. (2.5).

$$\dot{V}(\mathbf{x}) < -L_{\dot{V}} \cdot \tau \tag{2.5}$$

Fulfilling this condition implies that the gradient of the *CLF* will evolve over time τ at most to a value $\dot{V}(\mathbf{x}) < 0$. As this still matches the original requirement in Lemma 2.1, the stability of the system under consideration is guaranteed despite discretizing the state space. With this refinement, the *Region of Attraction (RoA)* for a system Eq. (2.1) controlled according to the policy π is estimated by maximizing c of the level set $\mathcal{V}(c)$ with respect to Lemma 2.1 and Eq. (2.5). This process is visualized in Fig. 2.3 where a control policy is assumed that successfully generates negative gradients over time for the chosen *CLF*, that is, where $\dot{V}(x) < 0$ for each x.

Region of Attraction (RoA)* in *Safe Reinforcement Learning (Safe RL) The guarantee provided by the concept of *RoA*, that is, asymptotic stability of the examined controller, combined with its applicability to general control systems enable Berkenkamp *et al.* [45] to use the same in *Safe Reinforcement Learning (Safe RL)*. Although it is commonly applied at a controller's design-time, the authors show that an application at run-time is possible as well.

In this case, however, the uncertainties affecting a system have to be considered as well. Therefore, Berkenkamp *et al.* [45] and Berkenkamp *et al.* [51] consider $U_{model}(\mathbf{x}, \mathbf{u})$ as the distribution of uncertainties and apply the concept to the changed system model presented in Eq. (2.6).

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) + U_{model}(\mathbf{x}, \mathbf{u}) \tag{2.6}$$

It describes not only the state change $\dot{\mathbf{x}}$ depending on the chosen control actions but also a model uncertainty $U_{model}(\mathbf{x}, \mathbf{u})$.

As U_{model} is a random distribution, however, the resulting gradient $\dot{\mathbf{x}}$ is a random distribution as well. To apply Eq. (2.5) for calculating an *RoA* nonetheless, the authors convert the distribution to an interval and consider only the maximal gradient. Thereby, the worst-case model uncertainty is assumed. If an *RoA* can be estimated nonetheless, it is shown that the controller stabilizes the system for states inside the resulting *RoA*.

Although this approach does not directly support analyzing a failure model of shared data to perform a run-time safety assessment, it provides guarantees about the stability of the system under consideration at run-time and allows specifying uncertainties.

However, these uncertainties are specified as a single distribution U_{model}. In contrast, versatile internal and external factors may contribute to the uncertainties affecting the state change $\dot{\mathbf{x}}$ of a system. They range from external disturbances, such as weather conditions, to impairments affecting sensors observing the system's environment, to internal uncertainties of model assumptions and parameters. Consequently, modeling uncertainties by a single distribution contradicts the idea of flexibility and dynamic integration as promised by dynamically composed systems.

In summary, approaches at the technical level address safety in two ways. Firstly, stability is considered as one aspect of safety as it enables limiting the control system's trajectory. This is combined with the second aspect of safety, which is to map safety requirements to the state space of a system in question. Then, by guaranteeing that only states deemed safe are visited, the overall safety is guaranteed. Moreover, as these guarantees are mathematically founded, functionally correct implementations for usage during a run-time safety assessment are possible.

However, these approaches are commonly applied at design-time. Only limited approaches from the field of *Safe RL* aim at providing guarantees at run-time, for instance using the idea of *RoA*. None of these approaches target dynamically composed systems or allow analyzing a failure model of shared data.

2.1.3. Conclusions

The previous subsection presented approaches to run-time safety assessment targeting the functional and technical level. An overview is given in Fig. 2.2.

At the functional level, the approaches of *FTA* and *FMEA* form the basis of the state of the art. These approaches can only be applied to static system compositions at design-time. In light of dynamically composed systems, the disadvantages and shortcomings of these approaches were already addressed. The required shift of safety assessment to run-time was addressed by Rushby [41] who proposed reusing system models generated at design-time to enable run-time certification. However, a dynamic system composition is missing.

Contrarily, Jaradat *et al.* [36] focused on the modularity of safety cases which enables the dynamic integration of new components and thereby forms a basis for dynamically composed systems. Both ideas, a run-time safety assessment and a dynamic integration of new system components, are required for dynamically composed systems. Therefore, approaches such as *ConSerts* and *DDI* but also the concept of *LoS* are proposed. With these approaches, run-time safety assessment is addressed at the functional level.

One requirement of these approaches is to interface with the technical level to provide run-time evidence about the safety of a dynamically composed system. Consequently, works targeting this level were reviewed as well. However, these were found to be focusing on statically composed systems.

Most promising are approaches from the field of *Safe RL*, where, for instance, the approach of *RoA* is shown to provide guarantees about the stability of a system at run-time. It thereby satisfies the criterion of run-time certification and can be shown at design-time to be functionally correct, cf. Section 1.3.1. However, the central criterion of being applicable to dynamically composed systems, that is, being able of analyzing failure models of shared data, is not fulfilled.

Therefore, while run-time safety assessment approaches targeting the functional level can be found, approaches providing the run-time evidence required by these are missing for the technical level.

2.2. State of the Art in Failure Modeling

In the endeavor of realizing a run-time safety assessment for dynamically composed systems, the need for a failure model describing the failure characteristics of shared data was identified in Section 1.3. Thus, in this section approaches to failure modeling are reviewed. To clarify what a failure model is, however, the next subsection starts with formally introducing its definition. Building upon that, Section 2.2.2, Section 2.2.3, and Section 2.2.4 discuss approaches based on intervals, distributions, and failure types respectively. Despite these differences, however, the approaches aim at failures of sensors and sensory data. These are focused as sensors form the basis for the environmental perception of autonomous systems. As such they are central sources of uncertainty that either directly or indirectly influence failure characteristics of shared data as well. The suitability of the presented models is discussed with respect to the predefined criteria of Section 1.3.1.

2.2.1. The Definition of (Sensor) Failure Model

Before approaches to failure modeling can be discussed, it has to be defined what a failure model is first. As specifically sensory data and its failure characteristics are targeted in this work, the definition of a sensor and its failures is provided first.

A sensor is a device observing a continuous phenomenon $e(t) \in \mathbb{R}^m$ in the real-world and converting it to a measurable, electrical signal. By generating a digital representation of this signal with a sampling period of T_s, a sensor produces a discrete time series of observations $\mathbf{o}_k = e(k \cdot T_s)$ with $\mathbf{o}_k \in \mathbb{O} \subseteq \mathbb{R}^m$, where $k \in \mathbb{N}_0$ is the discrete time index and $\mathbb{O}$ is the set of possible observations provided by a sensor [52], [53].

However, the conversion process is typically disturbed by internal (e.g. power supply when provided through battery) or external (e.g. weather conditions) factors [48].

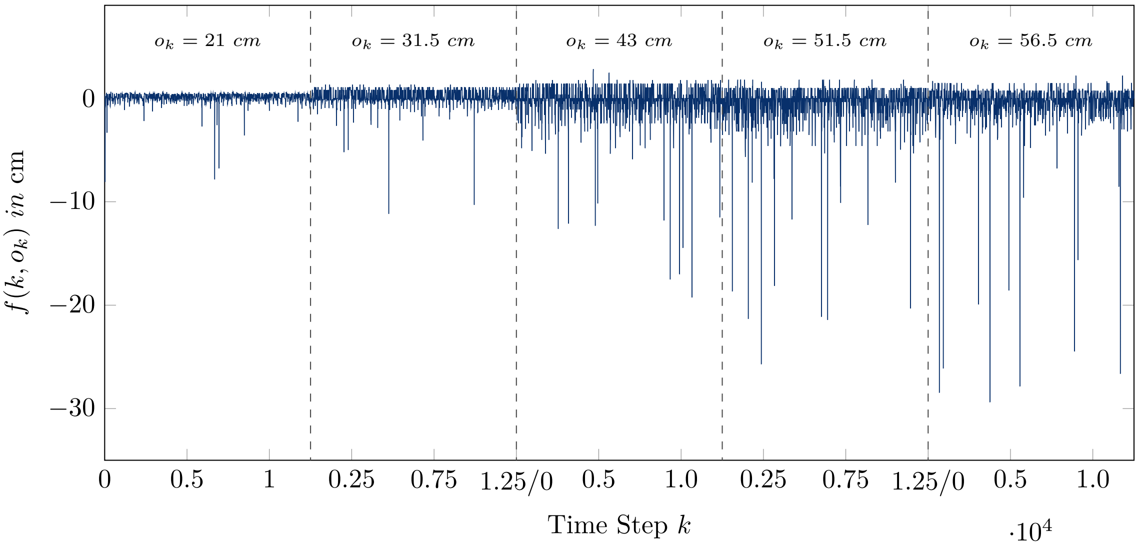


Fig. 2.4.: Failure amplitudes according to Eq. (2.7) of a Sharp GP2D12 distance sensor for different ground truth distances $o_k \in \{21\text{cm}, 31.5\text{cm}, 43\text{cm}, 51.5\text{cm}, 56.5\text{cm}\}$. The figures is inspired by [7].

These result in random or systematic failures impairing the observation produced by a sensor. Although systematic failures can be mitigated by properly calibrating a sensor [54], random failures are challenging to predict. Therefore, instead of providing the theoretically correct observation $\mathbf{o}_k$, the impaired observation $\hat{\mathbf{o}}_k$ is produced. The difference between both is the failure amplitude $f(k, \mathbf{o}_k)$ which is a discrete time series as well.

$$f(k, \mathbf{o}_k) = \hat{\mathbf{o}}_k - \mathbf{o}_k \tag{2.7}$$

Fig. 2.4 shows exemplary time series of failure amplitudes of a Sharp GP2D12 infrared distance sensor. Note that the failure amplitudes are shown for different ground truth distances $o_k \in \{21\text{cm}, 31.5\text{cm}, 43\text{cm}, 51.5\text{cm}, 56.5\text{cm}\}$ and shown in one plot for simplicity.

As one can see, the magnitude of $f(k, o_k)$ varies with the ground truth distance o_k. Similar to such value-correlations, failure amplitudes are reported to be possibly time-correlated as well [55], [56]. This is observed for distance information provided by 3D depth cameras [56], for instance. Due to these correlations, failure amplitudes $f(k, \mathbf{o}_k)$ are considered a function of time k and the ground truth observation $\mathbf{o}_k$.

Moreover, the failure amplitudes and their correlations resemble the internal and external disturbances influencing the conversion process of the sensor. Therefore, they resemble the sensor's failure characteristics. The task of a failure model is to describe exactly these. It is therefore defined as follows.

Definition 2.2 (*(Sensor) Failure Model*)

> *A failure model is a representation of a sensor's failure characteristics describing the inaccuracy and imprecision to be expected from its observations.*

Definition 2.2 provides a data-driven definition of a sensor failure model focusing on the observable consequences of a sensor failing to provide a sufficient adequate observation of a phenomenon.

It has to be noted that despite Definition 2.2 focusing on sensors, corresponding models might be applicable to shared data as well. On the one hand, shared data can be sensory

Tab. 2.1.: Qualitative assessment of the fulfillment of the criteria discussed in Section 1.3.1 by the interval-based failure modeling approaches.

Approach	Clarity	Comparability	Generality	Coverage	Confidence
Static Intervals [58], [59]	●	○	◐	○	○
Value-Correlated Intervals [59]	●	○	◐	○	○
Uncertainty Intervals [60]	●	○	◐	○	◐
GUM [61]	●	○	◐	○	◐

Legend: ○ – not fulfilled, ◐ – partially fulfilled, ● – fulfilled

data. On the other hand, shared data may stem from processing sensory data. In this case, the concept of *virtual sensors* [57] applies and enables considering the data as sensory data as well. Therefore, not only low-level data such as distance measurements are addressed but also processed and abstracted data.

Having a definition of a failure model in place, different types of models are reviewed next.

2.2.2. Interval-Based Failure Modeling

Due to the random nature of sensor failures, modeling them is a challenging task. The most simplistic approach is to state a lower and upper bound of $f(k, \mathbf{o}_k)$. Approaches using such interval-based failure models are reviewed in this subsection. An overview, as well as the fulfillment of the predefined criteria by the considered approaches, is given in Table 2.1.

A common document for the specification of interval-based failure models are sensor datasheets. For a long-range radar sensor, for instance, the accuracy of its distance and velocity observations is given by $\pm 0.1\,\mathrm{m}$ and $\pm 0.1\,\mathrm{km\,s^{-1}}$ respectively [58]. These are an example of static intervals, where the range of $f(k, o_k)$ does not change. Its interpretation is that the true value o_k is with the interval $[\hat{o}_k - 0.1\,\mathrm{m}, \hat{o}_k + 0.1\,\mathrm{m}]$. While this supports the clarity criterion defined in Section 1.3.1, it limits the fulfillment of the coverage criterion. The bounds of the specified interval have to be chosen according to the worst-case failure amplitudes. Therefore, a fine-grained representation of failure characteristics or a detailed analysis of the same is not possible. This renders the comparability criterion to be unfulfilled, as such a failure model does not provide sufficient information to be compared with the application needs. Consequently, the generality criterion is only partially fulfilled as, on the one hand, the representation is application-independent, but, on the other hand, transformation to an application-specific representation is limited.

Opposed to such a static interval, the failures of a laser range finder are modeled as a mixture of a static interval ($\pm 0.03\,\text{m}$ for observations up to 1 m) and a dynamic interval ($\pm 3\%$ of observation for ranges between $[1\,\text{m}, 4.095\,\text{m}]$) [59]. For the dynamic interval, the range in which the true value o_k is to be expected increases proportional to the observation: $o_k \in [0.97 \cdot \hat{o}_k, 1.03 \cdot \hat{o}_k]$. Therefore, the interval is value-correlated. This provides an equally clear interpretation but suffers from the same limitations causing the coverage, generality, and comparability criteria to be not or only partially fulfilled. Moreover, as the representation for both variants boils down to lower and upper bounds of $f(k, o_k)$, the confidence criterion is not fulfilled.

This is different for uncertainty intervals that are proposed by Moffat [60]. They define the true value o_k to lie in an interval $o_k \in [\hat{o}_k - \alpha, \hat{o}_k + \alpha]$ as well. However, α provides the significance, that is, the probability of o_k lying within the interval. For that, α can be set to 2σ, where σ is the standard deviation expected in the observations $\hat{o}_k$.

The considerations of Moffat [60] mainly influenced the "Guide to the Expression of Uncertainty in Measurement" [61]. It distinguishes random (Type A) and systematic (Type B) effects that introduce uncertainties into observations. For quantifying the uncertainties, multiple, independent test series are conducted. For each series, the mean value is calculated, resulting in one mean value per independent test. From that, the standard error is defined as the standard deviation of these mean values. As it follows a Gaussian distribution according to the Central Limit Theorem [62], the standard error can be used to derive a corresponding confidence interval as well, which enables giving confidence in the described failure model.

In summary, interval-based failure models focus on simplifying the representation of failure characteristics by providing only its bounds. Thereby, the coverage and comparability criteria can not be fulfilled, but the interpretation of the model is clear. Generality can be only partially fulfilled as an interval representation is general, but provides not sufficient information to transform it into an application-specific representation. Regarding the confidence criterion, only uncertainty intervals and the intervals based on the standard error of observations enable its partial fulfillment. In parts, both approaches build upon the distribution with which the failure amplitudes are distributed in the provided intervals.

Central to the restrictions of the interval-based failure models, however, is the limited information provided. Considering, for instance, the failure amplitudes shown in Fig. 2.4 it becomes clear that considering only the minimal and maximal value of $f(k, o_k)$ is not sufficient to comprehend the overall characteristics. Therefore, distribution-based approaches are reviewed next.

2.2.3. Distribution-Based Failure Modeling

To overcome the disadvantage of limited information when using interval-based failure models, distribution-based models can be used. Table 2.2 lists the approaches exemplary reviewed in this section. It needs to be noted that this is a non-exhaustive list with many more examples described in literature.

Within the body of research, approaches assuming zero-mean Gaussian distributions for representing sensor failure characteristics are common. Elnahrawy and Nath [63], for instance, build upon that assumption and propose a filter based on the Bayes' theory to clean noisy sensor observations of wireless sensor networks. Assuming in-

Tab. 2.2.: Qualitative assessment of the fulfillment of the criteria discussed in Section 1.3.1 by the distribution-based failure modeling approaches.

Approach	Clarity	Comparability	Generality	Coverage	Confidence
Gaussian Distribution [63]–[65]	●	◐	◐	○	○
Histogram [66], [67]	●	◐	◐	◐	●
Descriptive Variables [68]	●	○	◐	○	○

Legend: ○ – not fulfilled, ◐ – partially fulfilled, ● – fulfilled

dependence between consecutive sensor observations enables them to consider only a single observation at each time step k. The corrected sensor observation is provided by applying Eq. (2.8).

$$P(o_k|\hat{o}_k) = \frac{P(\hat{o}_k|o_k) \cdot P(o_k)}{P(\hat{o}_k)} \tag{2.8}$$

Assuming $\hat{o}_k$ to be the noisy sensor observation, the authors require three distributions. Firstly, $P(o_k)$ represents the probability of the true value being o_k. This is captured in a process model. Secondly, $P(\hat{o}_k)$ represents the distribution of noise observations. This is given by the failure model of the sensor. Thirdly, $P(o_k|\hat{o}_k)$ represents the probability of o_k being the true value given the current observation. The approach is evaluated on data provided by temperature sensors.

The assumption that noisy observations are Gaussian distributed clarifies the interpretation of the failure model and increases its comparability to be partially fulfilled as it now supports fine-grained analysis. This additionally enables transforming the distribution in application-specific representations, e.g. by reducing it to an interval. However, the assumption of sensor failures being Gaussian distributed limits the failure characteristics representable by this approach, which renders the coverage criterion to be unfulfilled. Similarly, no further measures of confidence that the Gaussian distribution matches the true failure characteristics are provided, which causes the associated criterion to be unfulfilled as well.

Nevertheless, for statically composed systems, the mathematical characteristics of Gaussian distributions often compensate for their lack of modeling true failure characteristics. Therefore, they are used, e.g. for Kalman filters [64], [65] to integrate and filter observations in an uncertainty-aware manner.

Contrarily, Cooper, Raquet, and Patton [66] use histograms to describe the failure characteristics of a *Light Detection and Ranging (LiDAR)*. They aim at quantifying the observation failures of a Hokuyo UST-20LX in different contextual situations. For that, the sensor is brought into varying distances to an observation target. For these distances, the surface material, the target's color, and the angular orientation of the

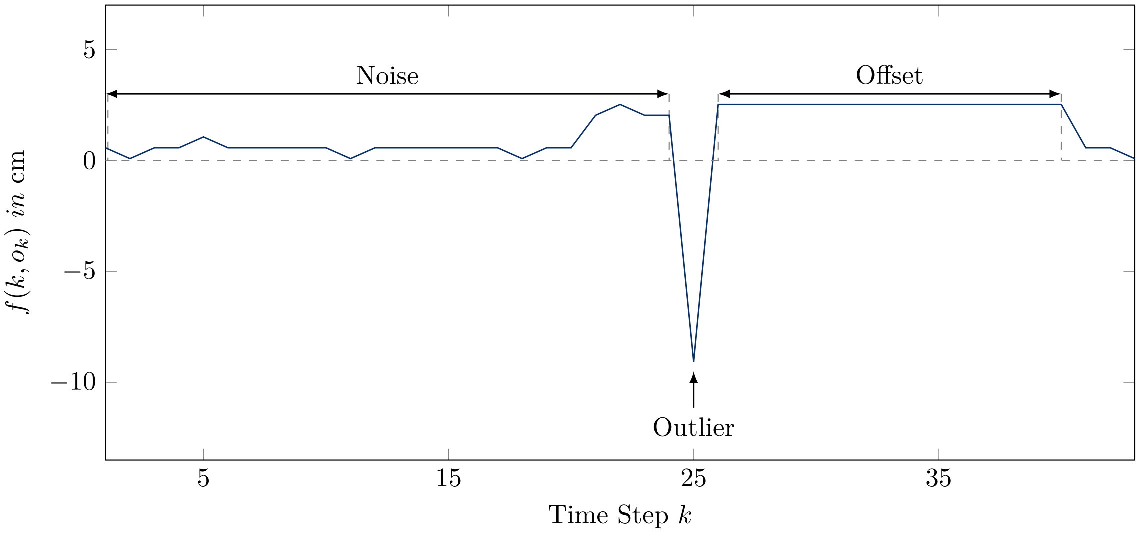


Fig. 2.5.: Temporal patterns within series of failure amplitudes $f(k, o_k)$ produced by a Sharp GP2D12 infrared distance sensor.

sensor towards the target were varied. By capturing the true distance using a motion capturing system with a high resolution, ground truth data ($\mathbf{o}_k$) were available so that failure amplitudes ($f(k, \mathbf{o}_k)$) could be calculated, cf. Eq. (2.7). Despite fitting Gaussian distributions, the authors additionally provide histograms to represent the failure characteristics. The interpretation of such a histogram is clear as it provides the failure characteristics in a non-parametric way. This supports detailed analysis that can provide application-specific representations. Moreover, the data reflects the true failure characteristics by directly providing samples, which enables assessing the confidence an application can have in the model.

In contrast, for dynamically composed systems where a failure model has to be communicated, a more compressed representation is favorable. One option is to calculate descriptive variables from a histogram first and share them instead. Dasika *et al.* [68], for instance, use box and whisker plots to describe the failure characteristics of a *LiDAR* sensor. Similar to the work presented before, they aim at analyzing the failure characteristics of the sensor with respect to different contextual situations. For that, the authors mount a *LiDAR* sensor on a mechanical testbed that moves it with varying velocities horizontally. Arranging the surface observed by the sensor to yield different but predefined heights, the sensor's ability to provide correct observations at different velocities is analyzed. For that, the distribution of observation failures is described by their mean, 0.25/0.5/0.75 quantiles, minimal, and maximal failure amplitude $f(k, \mathbf{o}_k)$. The interpretation of these variables is clear, however, the representation of failure characteristics and consequently the level of detail of analysis of the same is limited (e.g. considering multi-modal or multi-variate distributions). Similarly to interval-based approaches, no additional confidence in the correctness of the represented characteristics can be given.

The increased information provided when using distributions as failure models supports the fulfillment of the comparability criterion when compared to interval-based approaches. Thus, the coverage criterion is supported as well. Nevertheless, using distributions may not be sufficient to describe the characteristics of a sensor.

Firstly, when considering the example of the Sharp sensor once again, assuming a Gaus-

Tab. 2.3.: Qualitative assessment of the fulfillment of the criteria discussed in Section 1.3.1 by the failure-type-based failure modeling approaches.

Approach	Clarity	Comparability	Generality	Coverage	Confidence
Ni *et al.* [69]	○	◐	◐	●	○
Zug, Dietrich, and Kaiser [55]	○	◐	●	●	○
Jäger *et al.* [27]	◐	◐	◐	◐	○
Muhammed and Shaikh [70]	●	●	●	◐	○
Fagbemi, Perhinschi, and Al-Sinbol [71]	●	●	◐	◐	○

Legend: ○ – not fulfilled, ◐ – partially fulfilled, ● – fulfilled

sian distribution does not provide an appropriate match of the failure characteristics, cf. Fig. 2.4 [7]. On the one hand, the distribution of failure amplitudes does not follow a Gaussian distribution. On the other hand, the magnitude of failure amplitudes is value-correlated, which is not represented by such a model.

Secondly, distribution-bases failure models do not enable representing temporal patterns. The need for representing them is visualized in Fig. 2.5 where Offset patterns (a constant plateau), Noise (subsequent minor failure amplitudes of varying magnitude), and Outlier (a separated failure amplitude of increased magnitude) can be distinguished. Failure models expressing failure characteristics in terms of these patterns are failure-type-based failure models and are reviewed in the next subsection.

2.2.4. Failure-Type-Based Failure Modeling

Interval-based or distribution-based failure models are limited regarding the failure characteristics they can represent. In parts, this is because individual aspects of a failure characteristics are generalized. For instance, temporal patterns evolving systematically in failure amplitudes can not be represented. To overcome this restriction and describe failure characteristics more thoroughly, failure-type-based failure models are described in literature. Table 2.3 lists exemplary approaches.

To clarify this modeling approach, a failure type is defined as follows.

Definition 2.3 (*Failure Type*)

> *A failure type is a description or representation of a distinct property of a component's failure characteristics.*

In other words, the idea is to represent a failure model in terms of a collection of failure types, each describing a distinctive aspect of the overall failure characteristics. In the end, a failure model is nothing but a set of failure types.

Ni *et al.* [69], for instance, define eight failure types common to sensor networks. The failure types range from *Spike* and *Outlier* to *Noise*, which is defined as "random unwanted variation in data". After defining all failure types linguistically, they relate them to possible, physical causes such as battery depletion or sensor age. Afterward, they propose analytic, signal, and stochastic metrics based on which detection and filtering algorithms may be implemented. They exemplify their propositions using data sets from sensors observing chemicals, humidity, temperature, and light intensity.

Due to their linguistic definition, the failure types and the resulting failure model can not be interpreted unambiguously, which renders the clarity criterion unfulfilled. This also affects the fulfillment of the comparability and generality criteria. While the approach is general and allows to derive application-specific representations, the insufficient clarity imposes the challenge of retaining the correct information in the process and may result in misinterpretation. Moreover, no confidence in the correctness of the failure model is provided. In contrast, the approach analyzes the causes of sensor failures in detail, which means that the coverage of failure characteristics is high.

Although defined linguistically as well, Zug, Dietrich, and Kaiser [55] present a failure model comprising 14 failure types for which a qualitative visual representation is provided. The described failure types range from data failures, such as *Noise*, *Outliers*, and *Stuck-ats* to fixed and variable *Delays*. While data failures affect only the value of a sensor observation, delays affect their timely delivery by the sensor. The modeling of these failures increases the generality and renders the criterion to be fulfilled.

Using the introduced failure model, Jäger *et al.* [27] discuss different features from signal processing (e.g. *Signal-to-Noise Ratio* or Variance) that can be used to train a *Time-Delay Neural Network (TDNN)* for detecting the failure types in sensor data. To train the *Artificial Neural Network (ANN)*, four out of 14 failure types were selected to generate training data through simulation. Afterward, failure injection and simulation are used to assess the detection performance. The overarching goal is to show that by using learning-based methods, the process for tuning sensor failure detection methods can be automated partially.

On the one hand, the approach increases the fulfillment of the clarity criterion as mathematical descriptions of the selected failure types are provided. These are required for simulating the failure types. On the other hand, only four failure types are taken into account, which limits the coverage and thereby affects the generality and comparability.

With the goal of reviewing concepts for fault detection and isolation in wireless sensor networks, Muhammed and Shaikh [70] present a failure model comprising six failure types. They consider *Offsets*, *Gain* failures, *Stuck-at*, *Out-of-Bounds*, *Spike*, and *data loss* and provide mathematical descriptions. *Offset*, *Gain*, and *Stuck-at* failures, for instance, are represented by appropriate parameterization of Eq. (2.9).

$$\hat{o}_k = \alpha + \beta \cdot o_k + \eta \tag{2.9}$$

Here, α is an additive offset while β allows modeling gain failures. η represents the noise affecting an observations. Similar modeling approaches are provided for the remaining failure types.

Such a mathematical definition of failure types is the key to clear failure models. In contrast to linguistic definitions, they can be interpreted unambiguously. Moreover, the failure types are represented application-independent but can be transformed, e.g. through sampling, to application-dependent representations. This flexibility also allows for fine-grained analysis of a failure model during safety assessment. However, the set of failure types is restricted, which affects the fulfillment of the coverage criterion. Furthermore, no confidence in the correctness of the represented failure characteristics is provided.

Another field of research where mathematical definitions of failure types are mandatory is fault injection. Fagbemi, Perhinschi, and Al-Sinbol [71], for instance, leverage the idea in their fault injection and simulation tool targeting unmanned air systems. They propose a failure model comprising 8 failure types. These range from *Noise* failures, represented as a Gaussian distribution, to *Dropout* and *Drift* failures. For injecting the defined failure types a predefined hierarchy is given. This means that the effect of, for instance, a *Bias*, which introduces a constant offset, can be imposed by a *Saturation* failure, but not vice versa. On the one hand, this further clarifies the interpretation of the failure model, but, on the other hand, restricts the failure characteristics that it can represent. Similarly, with its goal of targeting unmanned air systems, the failure model is application-dependent which renders the generality requirement to be partially fulfilled.

2.2.5. Conclusions

In the endeavor of providing a run-time safety assessment method, the need for a suitable failure model describing the failure characteristics of shared data motivated to review existing approaches. For that, failure models targeting sensory data were focused. These, despite considering three categories of failure modeling approaches, only partially fulfill the predefined criteria, cf. Section 1.3.1. While interval and distribution-based failure models lack coverage and therefore do not fulfill the comparability and generality criteria, failure-type-based failure models lack clarity. The majority ([27], [55], [69]) of these approaches define failure types only linguistically. This renders the failure models unsuited to be shared in dynamically composed systems as their interpretation is ambiguous. Thus, the result of a run-time safety assessment could be incorrect and ultimately endanger the overall system's safety.

Approaches defining failure types mathematically ([70], [71]) overcome this challenge but are often defined with respect to a certain application, e.g. fault injection and simulation of unmanned air systems [71]. This limits their ability to represent failure characteristics and thereby renders the coverage criterion to be only partially fulfilled. Finally, non of the approaches provide a measure of confidence, that is, a value or meta-information providing confidence that the modeled failure characteristics match what is to be expected in reality. While significance levels of interval-based failure models can be interpreted as confidence values [60], [61], they are not explicitly focused on and therefore fulfill the criterion only partially.

2.3. Conclusions from the State of the Art

According to Objective 1.1 and Objective 1.2, approaches from corresponding fields have been reviewed. Section 2.1 focused on run-time safety assessment and showed that approaches such as *Conditional Safety Certificate (ConSert)* and *Level of Service (LoS)* fulfill the criteria defined in Section 1.3.1. However, they target only the functional abstraction level and assume that evidence supporting the claim for safety is generated by the technical level as well. As can be seen in Fig. 2.2, a suitable approach to provide such evidence for dynamically composed systems is missing. Works providing safety guarantees at the technical level either focus on design-time analysis or require a static system composition. The approach of using *RoA* to guarantee safety during *RL*, for instance, shows that safety assessment at run-time is possible, but lacks the ability of specifying failure characteristics of shared data. Thus, approaches specifically targeting dynamically composed systems are missing.

Similarly, the need for a suitable failure model to describe failure characteristics of shared data became clear when reviewing corresponding approaches in Section 2.2. The criterion of clarity, that is, the ability to unambiguously interpret a failure model when shared in dynamically composed systems, was fulfilled only by approaches that lack coverage and generality.

Consequently, a failure model focusing on clarity but fulfilling all predefined criteria is required. Based on such a generic failure model, a run-time safety assessment that enables analyzing the quality of shared data to provide safety guarantees has to be found. Both of these objectives are therefore addressed in the following chapters respectively.

3. Generic Failure Model

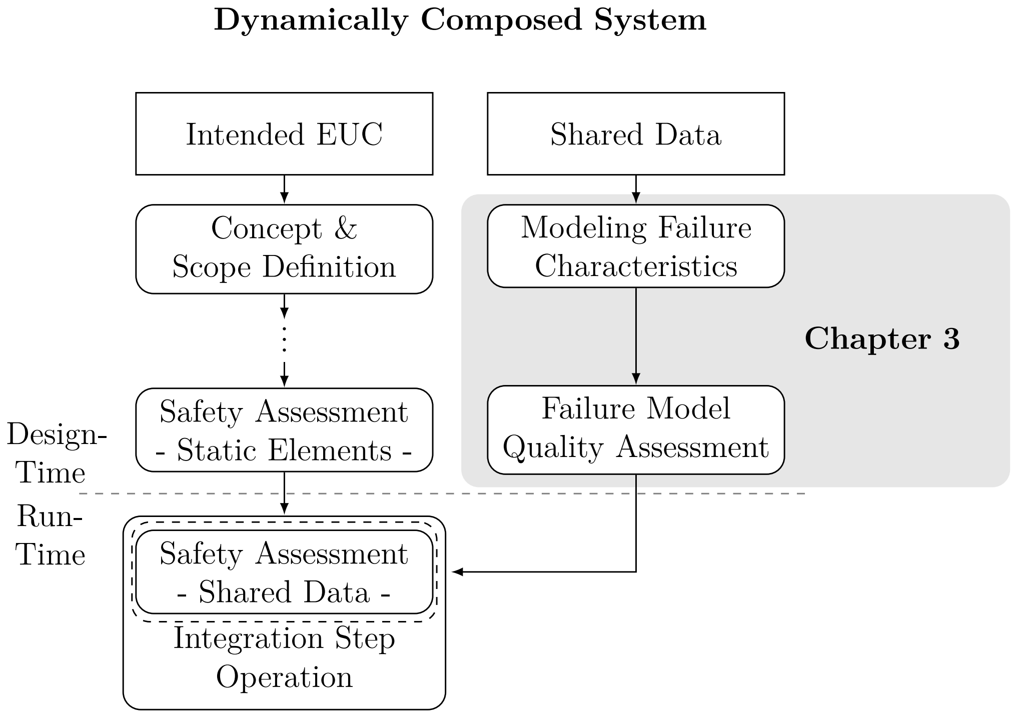

Dynamically Composed System
Intended EUC
Shared Data
Concept & Scope Definition
Modeling Failure Characteristics
Chapter 3
Safety Assessment - Static Elements -
Failure Model Quality Assessment
Design-Time
Run-Time
Safety Assessment - Shared Data -
Integration Step Operation

Fig. 3.1.: Simplified safety process from Fig. 1.9 showing the components addressed in this chapter.

In search of approaches fulfilling Objective 1.2, the last chapter reviewed state-of-the-art failure modeling approaches. While partial fulfillment of the required criteria could be shown for some approaches, none of the presented failure models are suitable for dynamically composed systems. Therefore, in this chapter, the *Generic Failure Model (GFM)*, initially proposed by Jäger, Zug, and Casimiro [7], is introduced, cf. Fig. 3.1.

The failure model focuses on clarity to prevent ambiguous interpretation but reuses concepts from the state of the art to fulfill the remaining criteria. Its definition, which will be discussed in Section 3.1, provides generality and coverage by building upon the idea of failure types while mathematical definitions of these guarantee clarity. By presenting an approach to transform a *GFM* to an interval-based representation in Section 3.2, the fulfillment of the comparability criterion is shown.

In [7], the authors additionally propose a processing chain for extracting a *GFM* from a given series of failure amplitudes, which underlines the applicability of the failure model. In this pursuit, the authors adopted evolutionary algorithms to enable flexibility. The entailed disadvantage of having to split the identification process of failure types into two stages is overcome in this work by presenting an alternative processing chain in Section 3.3. A single stage based on the *Continuous Wavelet Transformation*

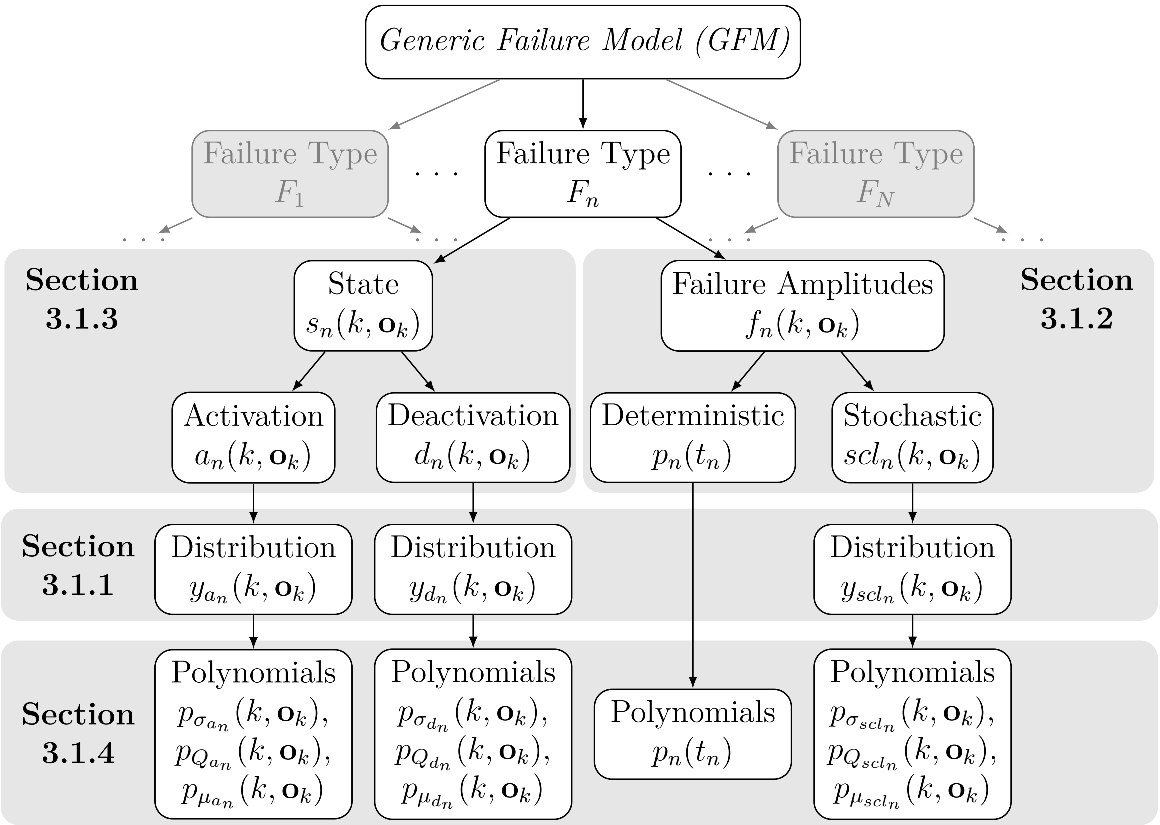

Generic Failure Model (GFM)
Failure Type F_1
. . .
Failure Type F_n
. . .
Failure Type F_N
Section 3.1.3
State $s_n(k, \mathbf{o}_k)$
Failure Amplitudes $f_n(k, \mathbf{o}_k)$
Section 3.1.2
Activation $a_n(k, \mathbf{o}_k)$
Deactivation $d_n(k, \mathbf{o}_k)$
Deterministic $p_n(t_n)$
Stochastic $scl_n(k, \mathbf{o}_k)$
Section 3.1.1
Distribution $y_{a_n}(k, \mathbf{o}_k)$
Distribution $y_{d_n}(k, \mathbf{o}_k)$
Distribution $y_{scl_n}(k, \mathbf{o}_k)$
Section 3.1.4
Polynomials $p_{\sigma_{a_n}}(k, \mathbf{o}_k)$, $p_{Q_{a_n}}(k, \mathbf{o}_k)$, $p_{\mu_{a_n}}(k, \mathbf{o}_k)$
Polynomials $p_{\sigma_{d_n}}(k, \mathbf{o}_k)$, $p_{Q_{d_n}}(k, \mathbf{o}_k)$, $p_{\mu_{d_n}}(k, \mathbf{o}_k)$
Polynomials $p_n(t_n)$
Polynomials $p_{\sigma_{scl_n}}(k, \mathbf{o}_k)$, $p_{Q_{scl_n}}(k, \mathbf{o}_k)$, $p_{\mu_{scl_n}}(k, \mathbf{o}_k)$

Fig. 3.2.: Hierarchical overview on the *Generic Failure Model (GFM)* motivated by Jäger, Zug, and Casimiro [7].

(CWT) and gradient descent is proposed. Moreover, an additional stage leveraging the transformation of a *GFM* to its interval-based representation to calculate confidence values is described. These values provide means to assess the correctness of the generated failure model with respect to the failure amplitudes specified as inputs to the processing chain.

Using both, the *GFM* and the presented processing chain, Section 3.4 provides a preliminary evaluation by means of artificial data. It will not only be shown that the *GFM* is capable of representing versatile failure characteristics but also the processing chain's ability to extract the same from given time series data.

Finally, Section 3.5 concludes this chapter by summarizing its contents and discussing the central findings of the evaluation.

3.1. Defining the Generic Failure Model

The central goal of this section is to define the *GFM* which shall fulfill Objective 1.2. For that, the failure amplitudes $f(k, \mathbf{o}_k)$ (cf. Eq. (2.7)) form the starting point. It is presumed that these resemble the failure characteristics of shared data that need to be modeled. They are assumed to be a function of time k and value $\mathbf{o}_k$ as they are generally considered to be time- and/or value-correlated [53], [55], [68].

The envisioned failure model capable of representing these correlations is schematically depicted in Fig. 3.2. Additionally, the sections in which the corresponding components of the failure model are explained in detail are referenced.

Building on the concept of failure types, a *GFM* is nothing but a set of failure types $\mathcal{FM} = \{F_1, \ldots, F_N\}$. It represents a failure characteristics as a composition of the effect $f_{F_n}(k, \mathbf{o}_k)$ of these failure types, cf. Eq. (3.1) [7].

$$f(k, \mathbf{o}_k) = \sum_{n=1}^{N} f_{F_n}(k, \mathbf{o}_k) = \sum_{n=1}^{N} \underbrace{s_n(k, \mathbf{o}_k) \cdot f_n(k, \mathbf{o}_k)}_{n\text{-th Failure Type } (F_n)} \tag{3.1}$$

For that, each failure type consists of a state function $s_n(k, \mathbf{o}_k) : \mathbb{R} \times \mathbb{R}^m \to \{0, 1\}$ and a failure amplitudes function $f_n(k, \mathbf{o}_k) : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$. While the state function merely describes at which point in time a failure type is active, a failure type's failure amplitudes describe the actual effect it has on the overall failure amplitudes. For that, a failure type encompasses a failure pattern $p_n(t_n)$ describing the shape of failure amplitudes it introduces over time as well as a scaling $scl_n(k, \mathbf{o}_k)$.

In contrast to the failure pattern, which is deterministic, the scaling $scl_n(k, \mathbf{o}_k)$ is a distribution that is time- and value correlated. Similarly to the failure amplitudes of a failure type, its state function $s_n(k, \mathbf{o}_k)$ consists of an activation distribution $a_n(k, \mathbf{o}_k)$ and a deactivation distribution $d_n(k, \mathbf{o}_k)$, both of which are time- and value-correlated as well. These correlations are represented separately by corresponding scaling (σ) and shifting (μ) function such that the actual distribution (Q) is assumed to be static. The resulting functions are represented by polynomials.

Thus, in the endeavor of defining the generic failure model, the concept of a time- and value-correlated random distribution is introduced first in the next subsection. This provides the basis on which the representation of a failure type's failure amplitudes $f_n(k, \mathbf{o}_k)$ is defined in Section 3.1.2. Similarly, the concept of a time- and value-correlated random distribution is reused to model a failure type's activation and deactivation for representing its state function in Section 3.1.3. While this complements the definition of a single failure type and thereby the definition of the *GFM*, an approach for representing the individual functions in terms of mathematical expressions is required. In that endeavor, Section 3.1.4 discusses the usage of polynomials to support the modeling of a wide range of failure characteristics for fulfilling the coverage criterion and to maintain clarity of the approach. Whether or not this is achieved is discussed with respect to the predefined criteria in a finalizing subsection.

3.1.1. Time- and Value-Correlated Random Distribution

In literature, two aspects of failure amplitudes are repeatedly reported. On the one hand, they occur with a stochastic nature. On the other hand, they are correlated with time k and value $\mathbf{o}_k$. The purpose of the time- and value-correlated random distribution, which shall be introduced in this subsection, is to represent both.

In that endeavor, the Z-score normalization [72] is utilized cf. Eq. (3.2).

$$Y' = \frac{Y - \mu_Y}{\sigma_Y} \tag{3.2}$$

The Z-score normalization assumes a random variable Y with mean μ_Y and standard deviation σ_Y. By subtracting the mean and dividing by σ_Y, the random variable is normalized to have zero mean and a standard deviation of one.

In this work, the mean and standard deviation are assumed to be functions of time k and value o_k. By inverting Eq. (3.2) and rewriting it using the assumptions of μ_Y and σ_Y being functions, the random variable Y can be represented as follows.

$$Y = \sigma_Y(k,\, \mathbf{o}_k) \cdot Y' + \mu_Y(k, \mathbf{o}_k) \tag{3.3}$$

This representation enables distinguishing between deterministic correlations due to time k and value $\mathbf{o}_k$ and stochastic magnitudes captured by the uncorrelated distribution Y'. To support representing arbitrary distributions and thereby to support the fulfillment of the coverage criterion, $Y' = Q_Y(z)$ is represented as an *Inverse Cumulative Distribution Function (ICDF)*, which is also called a quantile function [73]. In simulations, this representation can be used to generate random values of an arbitrary distribution from a uniformly distributed random variable $z \in \mathbb{U}(0, 1)$. To reflect this way of representing Y', Eq. (3.3) can be rewritten as shown in Eq. (3.4).

$$y(k,\, \mathbf{o_k}) = \sigma_Y(k,\, \mathbf{o_k}) \cdot Q_Y(z) + \mu_Y(k,\, \mathbf{o_k}) \tag{3.4}$$

Using a quantile function, Eq. (3.4) enables sampling the time- and value-correlated random distribution[1]. Firstly, a random value distributed according to Y' is generated by sampling $z \in \mathbb{U}(0, 1)$ and evaluating the quantile function $Q_Y(z)$. Secondly, the value is scaled and shifted by $\sigma_Y(k, \mathbf{o}_k)$ and $\mu_Y(k, \mathbf{o}_k)$ to introduce time- and value-correlations.

On the one hand, the use of a quantile function facilitates representing arbitrary distributions, which supports fulfilling the coverage criterion. On the other hand, it causes a challenge for representing multi-dimensional distributions.

A quantile function requires the cumulative distribution function of a random variable to be bijective, that is, invertible. This is the case for one-dimensional distributions as the probability $P(Y \leq y)$ can be unambiguously mapped to a single value of y. Therefore, $Q_Y(z) = Q_Y^{-1}(y)$ and $Q_Y(y) = P(Y \leq y)$. In contrast, if $\mathbf{y} \in \mathbb{R}^{m>1}$ multiple vectors can be mapped to the same probability, which means that $Q_{\mathbf{Y}}(\mathbf{y})$ can not be inverted. This is caused by the fact that no natural ordering for elements of multi-dimensional spaces ($m > 1$) exist [75], [76].

One approach to address this problem is proposed by Liu, Parelius, and Singh [77]. They use statistical depth functions (e.g. the Mahalanobis depth) and define quantiles as their superlevel sets. The mean of the represented distribution obtains the highest function value while values of reduced likeliness are having lower function values. Therefore, the function value can be interpreted similarly as the probability of an element with respect to the modeled distribution. Nevertheless, statistical depth-functions can not be inverted, which prohibits generating values from the underlying distribution and therefore renders the approach unsuitable in this context.

In contrast, [78] consider vectors $\mathbf{z} \in [0, 1]^m$ to resemble the probability of a multi-dimensional random value. Based on norm minimization, they realize a unique mapping $[0, 1]^m \Rightarrow \mathbb{R}^m$. However, the interpretation of $\|\mathbf{z}\|$ as a probability does not hold for $m > 1$ [75].

Another approach to multi-dimensional quantile functions stems from the field of simulation. The approach of *standard construction* [79], [80] utilizes the indices of the

[1] The use of a quantile function also turns the time- and value-correlated random distribution into a *generative model* [74]

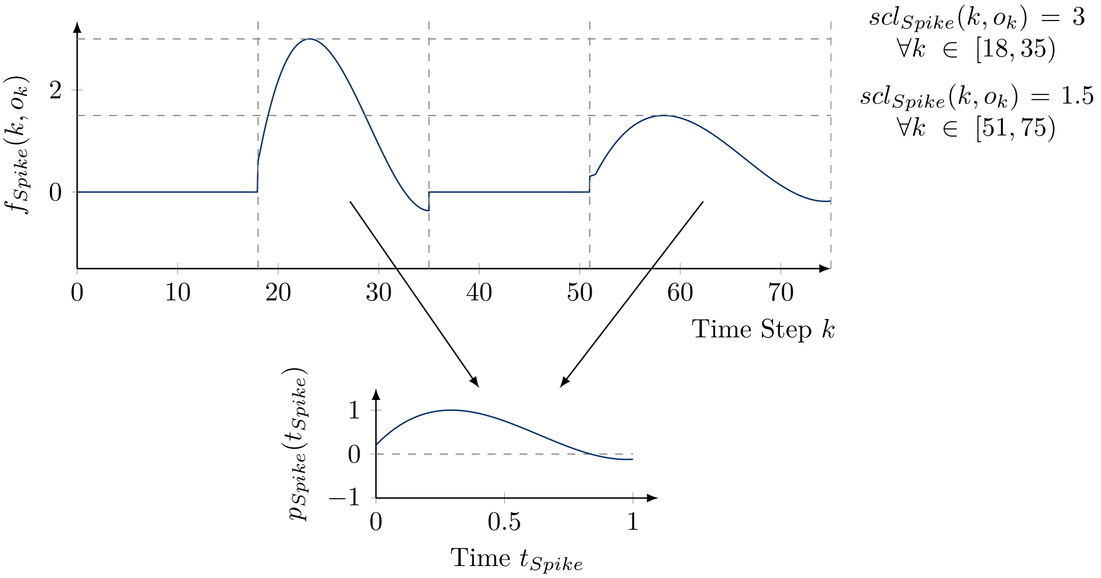


Fig. 3.3.: Example of a Spike failure type with two occurrences. Despite the differences in its instantiations, the failure pattern of the failure type is static, as indicated by the lower diagram. The figure is motivated by Jäger, Zug, and Casimiro [7].

dimensions as a natural ordering and was used to represent multi-dimensional quantile functions in Jäger *et al.* [31]. Assuming $\mathbf{Y} = [Y_1 \ \dots \ Y_m]^T$ to be a random vector in $\mathbb{R}^m$ and $\mathbf{z} = [z_1 \ \dots \ z_m]^T \in [0, 1]^m$, an arbitrary distribution of $Q_{\mathbf{Y}}(\mathbf{z})$ can be sampled as follows [79]:

$$y_1(z_1) = Q_{(Y_1)}(z_1) \tag{3.5}$$

$$y_2(z_1, z_2) = Q_{(Y_2|Y_1=y_1(z_1))}(z_2) \tag{3.6}$$

$$\dots \tag{3.7}$$

$$y_m(z_1, \dots, z_m) = Q_{(Y_{m-1}|\cap_{j=1}^{d-1} Y_j = y_j(z_1,\dots,z_j))}(z_b) \tag{3.8}$$

Here, $Q_{(Y_i|\dots)}(z)$ denotes the uni-variate quantile function of the marginal distribution of the i-th component given that all components $j < i$ have values $y_1, \cdots, y_j$. Thus, a dependency structure based on the chosen ordering of dimensions is imposed on the actual distribution of the random vector. Note that, opposed to the previous approach, the values $z_1, \dots, z_m$ retain their interpretation as probabilities. This underlines the fulfillment of the clarity criterion while employing the standard construction enables the representation of multi-dimensional distributions and therefore addresses the coverage criterion.

3.1.2. A Failure Type's Failure Amplitudes

The defined time- and value-correlated random distribution enables the representation of stochastic parts of failure types. These are present in its state function as well as its failure amplitudes. In this subsection, the latter is addressed. Note that the final representation of the individual functions of each time- and value-correlated random distribution is discussed in Section 3.1.4.

The failure amplitudes $f_n(k, \mathbf{o}_k)$ of a failure type describe a distinct property of the overall failure characteristics of a sensor or shared data. In literature, it is described that this property may be systematic or random. Moreover, as reported by Zug, Dietrich, and Kaiser [55], for instance, temporal patterns with random scaling can be observed as well. The representation of a failure type's failure amplitudes takes up on that observation and comprises two properties: its deterministic failure pattern $p_n(t_n)$ and a stochastic scaling $scl_n(k, \mathbf{o}_k)$, cf. Eq. (3.9).

$$f_n(k,\, \mathbf{o}_k) = \underbrace{scl_n(k, \mathbf{o}_k)}_{\text{stochastic}} \cdot \underbrace{p_n(t_n)}_{\text{deterministic}} \tag{3.9}$$

While the failure pattern models the temporal behavior introduced by the failure type, the stochastic scaling represents the randomness in magnitude with which the pattern may occur.

An exemplary illustration of how both parts of a failure type may manifest in the failure amplitudes $f_n(k, \mathbf{o}_k)$ is shown in Fig. 3.3. In this example, the failure type is named F_{Spike} for the sake of argumentation. Its pattern $p_{Spike}(t_{Spike}) \in [-1, 1]$ with $t_{Spike} \in [0, 1]$ shown in the lower diagram of the figure models the shape of the failure amplitudes.

As indicated in Fig. 3.3, a failure type's failure pattern is normalized in two ways. Firstly, the values are normalized over time ($t_n \in [0, 1]$). This allows to dilate and compress the pattern by sampling $p_n(t_n)$ accordingly. The specific length of a failure type's occurrence is modeled by its state-function $s_n(k, \mathbf{o}_k)$, which will be discussed in the next subsection.

Secondly, the pattern is normalized in its value domain ($p_n(t_n) \in [-1, 1]$) as it represents only the shape of the introduced failure amplitudes but does not represent a specific instance.

For that, the failure pattern is scaled according to $scl_n(k, \mathbf{o}_k)$, which is random in its nature. Therefore, $scl_n(k, \mathbf{o}_k)$ is represented by a time- and value-correlated random distribution as introduced in the last paragraph. Using Eq. (3.4) to replace $scl_n(k, \mathbf{o}_k)$ in Eq. (3.9) yields Eq. (3.10).

$$f_n(k, \mathbf{o}_k) = \underbrace{[\sigma_{f_n}(k, \mathbf{o}_k) \cdot Q_{f_n}(\mathbf{z}) + \mu_{f_n}(k, \mathbf{o}_k)]}_{\text{stochastic}} \cdot \underbrace{p_n(t_n)}_{\text{deterministic}} \tag{3.10}$$

This model for representing the failure amplitudes of a failure type enables flexibility and thereby supports fulfilling the coverage criterion. The scaling $scl_{spike}(k, o_k)$ of the example shown in Fig. 3.3, for instance, can be modeled in two ways. Firstly, the scaling of both occurrences may be purely random, which can be represented by a suitable quantile function $Q_{f_{spike}}(z)$ and constant values for $\sigma_{f_{spike}}(k, o_k) = 1$ and $\mu_{f_{spike}}(k, o_k) = 0$. Secondly, the scaling may be predefined by time- and/or value-correlations. In that case, the quantile function and standard deviation could be constant (e.g. $Q_{f_{spike}}(z) = 1$, $\sigma_{f_{spike}}(k, o_k) = 1$) and the mean would represent the scaling.

Note that these are only two ways of modeling the scaling of the failure type in this example. Other approaches, as well as mixtures of these, are possible.

3.1.3. A Failure Type's State Function

A failure type's failure amplitudes describe how a failure type contributes to the overall failure characteristics when it is active. Whether or not a failure type is active, on the

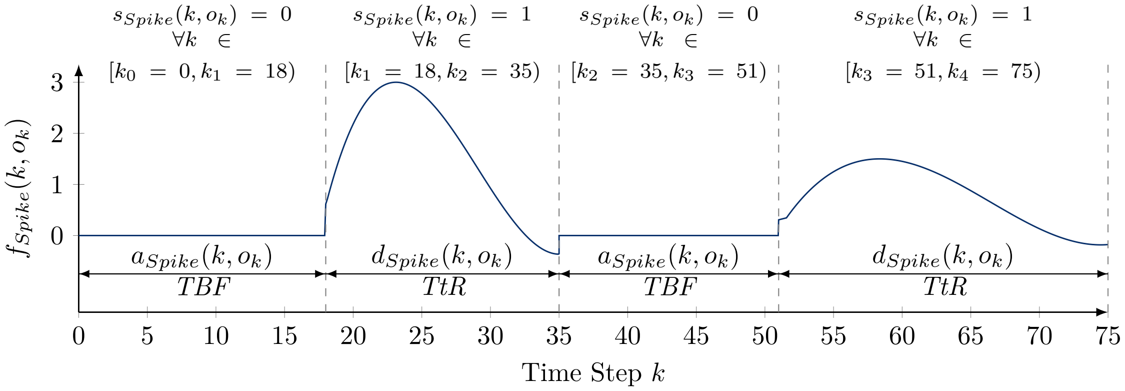


Fig. 3.4.: Schematic illustration of the components of a failure type's state function by means of an exemplary Spike failure type with two occurrences. The figure is motivated by Jäger, Zug, and Casimiro [7].

other hand, is modeled by the failure types state function $s_n(k, \mathbf{o_k})$, which will be discussed in this subsection.

For that, the example of the *Spike* failure type of the previous subsection is considered once again in Fig. 3.4. In the adapted figure it can be seen that the failure type is active ($s_{Spike}(k, o_k) = 1$) for $f_{Spike}(k, o_k) \neq 0$ and inactive ($s_{Spike}(k, o_k) = 0$) for $f_{Spike}(k, o_k) = 0$. Moreover, the first occurrence of the failure type lasts for 17 time units, corresponding to $s_{Spike}(k, o_k) = 1, \forall k \in [18, 35)$, while the second occurrence lasts for 24 time units ($k \in [51, 75)$). This illustrates the model's assumption that a failure type may occur with varying durations. Similarly, the time intervals during which the failure type is not active, in this case with lengths of 18 time units ($k \in [0, 18)$) and 16 time units ($k \in [35, 51)$), vary as well.

To facilitate representing these variations, the activation and deactivation of a failure type are modeled independently. For the sake of argumentation, one can use an analogy from the field of reliability engineering to explain the activation and deactivation distribution [81]. As the activation of a failure type denotes the time between two consecutive occurrences, that is, between two consecutive failure instances, it can be thought of as the distribution of *Time Between Failure (TBF)*. On the other hand, the deactivation of a failure type denotes the time for which a failure instance is active, that is, the time until the failure becomes inactive. Therefore, it defines the *Time to Repair (TtR)*. Both concepts are mapped to the corresponding intervals in Fig. 3.4.

From the distribution of *TBF* and *TtR* in this figure, one can see not only that they need to be modeled independently but also that they may vary stochastically. Therefore, the approach of time- and value-correlated random distributions is used once again. Here, $a_n(k, \mathbf{o}_k)$ is defined to represent the *TBF*, while $d_n(k, \mathbf{o}_k)$ represents the *TtR*.

$$a_n(k, \mathbf{o}_k) = \sigma_a(k, \mathbf{o}_k) \cdot Q_a(z) + \mu_a(k, \mathbf{o}_k) \tag{3.11}$$

$$d_n(k, \mathbf{o}_k) = \sigma_d(k, \mathbf{o}_k) \cdot Q_d(z) + \mu_d(k, \mathbf{o}_k) \tag{3.12}$$

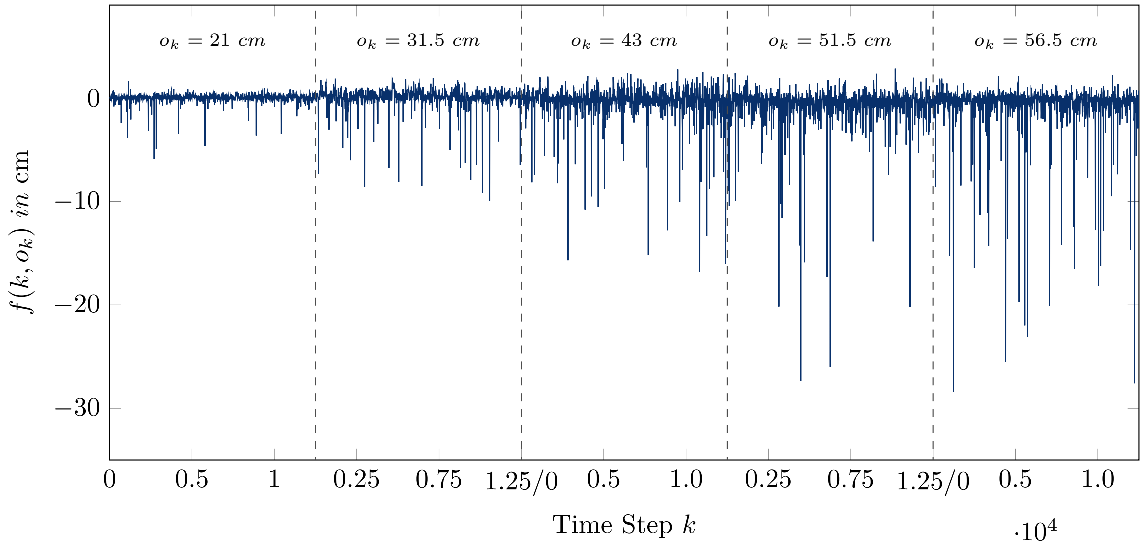


Fig. 3.5.: Failure amplitudes simulated by the *GFM* of the Sharp GP2D12 as presented in [7].

The state-function $s_n(k, \mathbf{o}_k)$ is defined by combining Eq. (3.11) and Eq. (3.12). For that, the fact that both functions define disjoint intervals is leveraged:

$$s_n(k, o_k) = \begin{cases} 0, & k \in [k_{2i}, k_{2i+1}),\ \forall i \in \mathbb{N}_0, \\ 1, & k \in [k_{2i+1}, k_{2i+2}),\ \forall i \in \mathbb{N}_0, \end{cases} \tag{3.13}$$

with the starting condition of $k_0 = 0$ and

$$k_j = \begin{cases} k_{j-1} + a_n(k_{j-1}, \mathbf{o}_{k_{j-1}}),\ j = 1, 3, 5, ..., \\ k_{j-1} + d_n(k_{j-1}, \mathbf{o}_{k_{j-1}}),\ j = 2, 4, 6, ... \end{cases} \tag{3.14}$$

Here, the sub-scripted time steps k_j define the border of the intervals of Eq. (3.13). The output of the state-function $s_{Spike}(k, o_k)$ of the exemplary failure type F_{Spike} is stated in Fig. 3.4 in accordance with the activation function $a_{Spike}(k, o_k)$ and deactivation function $d_{Spike}(k, o_k)$.

3.1.4. Polynomial Representation of a Failure Type

The core components of each failure type are its failure pattern and three time- and value-correlated random distributions representing its *TBF*, *TtR*, and random scaling. Each of these components is defined in terms of functions that need to be represented. To fulfill the clarity criterion, systems sharing failure models must agree on a common function approximation scheme that is mathematically defined. However, for choosing an appropriate approach, the coverage criterion has to be taken into account as well. It dictates employing a function approximation scheme that facilitates the representation of a wide range of failure characteristics.

From that perspective, polynomials are well suited as the Stone–Weierstrass theorem states that any continuous function defined on a closed interval can be approximated arbitrarily close by polynomials of the form of Eq. (3.15)[82]. It needs to be noted that here, p denotes a general polynomial and is not to be confused with a failure type's failure pattern p_n.

$$p(\mathbf{x}) = \sum_{j=0}^{D} \omega_j \cdot x^j \tag{3.15}$$

Depending on the values of the parameters ω_j and the polynomial degree D, the represented function changes. In essence, the Stone–Weierstrass theorem states that by increasing the polynomial degree D, the difference between the function represented by a polynomial and the function that is to be represented can be reduced to arbitrarily small values. Therefore, polynomials can be considered as universal function approximation schemes.

Neural networks inherit this property from polynomials [83]. Similar to polynomials, neural networks are defined mathematical functions whose structure is inspired by the inner workings of brains in humans and animals. So-called *neurons* use the weighted sum of their inputs to determine their activation using a non-linear activation function. The calculated output is passed to successive neurons or considered the output of the network. As such, the number of neurons and their arrangement in layers determine the expressiveness of the network. Similar to polynomials, the function represented by the neural network is determined by the parameters, that is, the weights of each neuron.

Although applicable to *Multi Layer Perceptron (MLP)* in general, a special form suitable for function approximation are *Radial Basis Function (RBF)* networks [84]. They were used in Jäger, Zug, and Casimiro [7] to represent the failure type functions and shown to be effective for modeling failures of an infrared distance sensor, cf. Fig. 3.5. However, two disadvantages come with neural networks. Firstly, determining its parameters is challenging. The procedure, often referred to as *training*, is an iterative approach. Given an appropriate loss function, e.g. the *Sum of Squared Errors (SSE)*, gradient descent is used to minimize this function and thereby find appropriate parameter values [85]. However, as this optimization problem is non-convex, finding a global minimum is NP-hard, which is why state-of-the-art algorithms find only a local minimum instead [86]. Secondly, the parameters of the neural network can not be interpreted or adapted easily. This contradicts the fulfillment of the clarity criterion.

In contrast, interpreting the parameters of a polynomial is not as challenging as interpreting the parameters of a neural network. As this not only enables experts to manually adjust polynomials but also allows them to assess the suitability of the fitted function, they are used to represent the functions of a *GFM* in this work.

In this endeavor, multi-variate and multi-response polynomial regression is employed where a polynomial is a function $G : \mathbb{R}^n \to \mathbb{R}^m$ where n denotes the number of independent variables and m denotes the number of response variables. Given the order of the polynomial as $D \in \mathbb{N}_{\geq 1}$ again, each response variable p_r is represented as [31]:

$$p_r(\mathbf{x}) = \omega_0 + \sum_{i=1}^{n} \sum_{d=1}^{D-(i-1)} \sum_{c \in C_i} \omega_l \cdot \prod_{j \in c} x_j^d \tag{3.16}$$

with $\hat{n} = \{i | i \in \mathbb{N}_{\geq 1} \wedge i \leq n\}$ being the set of indices of the independent variables and $C_i = \binom{\hat{n}}{i}$ denoting the set of their i-combinations. l is merely an index to distinguish the scalar parameters $\omega_l \in \Omega$ of the polynomial.

3.1.5. Discussion on the Fulfillment of the Predefined Criteria

The definition of the function approximation scheme to use for representing the individual functions concludes the definition of the *GFM*. As such, at a qualitative level, the suitability of the failure model for usage in a dynamically composed system with regard to the predefined criteria shall be discussed in this subsection.

Clarity The clarity criterion asks for an unambiguous interpretation of the failure model, which is provided for different reasons. Firstly, all functions are represented by mathematical equations enabling an automated interpretation. Secondly, each function has a specific purpose and meaning. The failure pattern describes the temporal behavior introduced by a failure type. Its scaling distribution models the magnitude of a single instance. The activation represents the time between two consecutive occurrences while the deactivation states the duration for which a failure type is active. Moreover, the time- and value-correlated random distribution forming the basis for these distributions enables representing the normalized distribution as a quantile function while using functions to represent time- and value-correlated mean and standard deviations. Thirdly, each function of a single failure type is represented as a polynomial. This enables engineers to interpret the weights and manually design failure types. Due to these reasons, clarity is supported by the definition of a failure type and thereby by the *GFM* as well.

Coverage The coverage criterion asks for the ability to represent versatile failure characteristics. For its fulfillment, the failure model takes upon the approach of failure types. Using these, complex failure characteristics can be decomposed into separate failure types, each representing a distinct part of the overall characteristics. Additionally, each failure type can represent time- and value-correlations by means of the introduced time- and value-correlated random distribution. This in itself provides different approaches to modeling correlations. Finally, the usage of polynomials enables to model arbitrarily complex functions (in theory) due to the Stone–Weierstrass theorem [82].

However, it needs to be noted that polynomials are in practice prone to oscillations when used for extrapolation. As a result, fitting, for instance, the quantile function for multi-dimensional data can be a challenge. Furthermore, the Stone-Weierstrass theorem requires a closed interval, which is not given for functions defined on time k.

Comparability To fulfill this criterion, the failure model has to support various levels of granularity. For that, the failure model can be evaluated in different ways. Firstly, as the failure model is generative, it can be directly used for simulation purposes. Using Monte-Carlo simulations, for instance, the number of simulations may be used to define different levels of granularity. Moreover, the severity of injected failures can be controlled by using not uniform values $\mathbf{z} \in \mathbb{U}^m(0, 1)$ to evaluate quantile functions but by increasing the likeliness to sample values close to 0 or 1. In this way, events occurring only rarely otherwise can be favored. Secondly, through the use of quantile functions, the minimal and maximal failure amplitudes introduced by each failure type can be determined directly. This supports calculating intervals restricting the range of possible failure amplitudes of a failure type.

Generality Generality asks for a representation that is application-independent but can be transformed into an application-dependent representation. The *GFM* in general is designed to be application-independent. Furthermore, due to the different levels of granularity at which the model can be analyzed, it can be transformed into application-specific representations as well. Therefore, this criterion is fulfilled as well.

Confidence Opposed to the previously discussed criteria, the confidence criterion is not addressed directly by the *GFM*. On the contrary, the use of polynomials enables representing a wide range of functions and correlations but does not provide an inherent mechanism to assess the goodness of fit. Therefore, this criterion is not fulfilled for now.

According to this discussion, the fulfillment of most criteria could be shown. However, on the one hand, the claim of providing different levels of granularity at which a *GFM* can be analyzed and its ability to be transformed into an application-specific representation has yet to be shown. On the other hand, the confidence criterion could not be shown to be fulfilled yet. In prerequisite to supporting the claim of granularity of analysis, transformability, and for defining a confidence measure, the next section discusses how a *GFM* can be transformed to an interval-based representation stating only the minimal and maximal failure amplitudes to expect.

3.2. Converting a Generic Failure Model to an Interval

The *GFM* defined in the last section was shown to fulfill most of the criteria defined in Section 1.3.1, two of which being comparability and generality. These ask for the ability to analyze a failure model at different levels of granularity and the ability to convert it into a (possibly) application-specific representation. In the endeavor of underlining that these criteria are met, one way of converting a *GFM* into an interval is presented in this section. On the one hand, intervals are shown in Section 2.2.2 to be a widely used approach to representing failure characteristics as well. On the other hand, the conversion of a *GFM* to an interval representation forms the basis for defining a confidence value in Section 3.3.3.

To convert a *GFM* into an interval, it is leveraged that it is nothing but a set of failure types. Therefore, the overall interval $\mathcal{I}_{\mathcal{FM}}$ is calculated according to Eq. (3.17).

$$\mathcal{I}_{\mathcal{FM}}(\mathbb{K}, \mathbb{O}) = \sum_{n=1}^{N} \mathcal{I}_{F_n}(\mathbb{K}, \mathbb{O}) \tag{3.17}$$

As the failure characteristics represented by the failure model $\mathcal{FM}$ comprise time- and value-correlations, the generated interval $\mathcal{I}_{\mathcal{FM}}$ does to. However, here $\mathbb{K}$ is the set or range of time steps and $\mathbb{O}$ is the set or range of observations for which the interval is valid, that is, the range of correlations covered by the interval.

The overall interval $\mathcal{I}_{\mathcal{FM}}(\mathbb{K}, \mathbb{O})$ is constructed by summing over the intervals $\mathcal{I}_{F_n}(\mathbb{K}, \mathbb{O})$ of the individual failure types, which are given by Eq. (3.18)

$$\mathcal{I}_{F_n}(\mathbb{K}, \mathbb{O}) = \left[\min_{k \in \mathbb{K}, \mathbf{o}_k \in \mathbb{O}} (f_n(k, \mathbf{o}_k)), \max_{k \in \mathbb{K}, \mathbf{o}_k \in \mathbb{O}} (f_n(k, \mathbf{o}_k))\right] \tag{3.18}$$

Thus, the minimal and maximal failure amplitudes possible for a given time horizon $\mathbb{K}$ and range of observations $\mathbb{O}$ have to be determined for each failure type. For that, two aspects of the failure type have to be determined. Firstly, the minimal and maximal failure amplitude it may generate with respect to the consider time- and value domain. Secondly, whether or not the failure type will be active within the same.

According to Section 3.1, both depend on the values of the respective time- and value-correlated random distributions. Therefore, the next subsection generally introduces the approach to calculating the interval of possible values of such a random distribution before Section 3.2.2 finalizes the calculation of a failure model's interval of failure amplitudes by defining the calculation of a failure type's interval $\mathcal{I}_{F_n}$.

3.2.1. Converting a Time- and Value-Correlated Random Distribution

A mandatory prerequisite for converting a failure type to an interval representation is the transformation of a time- and value-correlated random distribution (cf. Section 3.1.1) into the same. In accordance with its definition in Section 3.1.1, this means that its individual functions have to be transformed into intervals first before these can be combined to obtain the distribution's overall interval.

In this endeavor, the mean and standard deviation functions have to be evaluated with respect to the given time horizon $\mathbb{K}$ and range of observations $\mathbb{O}$ to obtain their respective minimal and maximal function values forming the required intervals. Similarly, the distribution's quantile function has to be evaluated to obtain the range of possible distribution values.

As all of these functions are represented by polynomials, a general strategy to find their minimal and maximal function values depending on a (sub-)set of their inputs has to be defined first. This strategy can then be applied to the quantile function, the mean function, and the standard deviation function.

Accordingly, the next paragraph discusses a sampling-based approach to extracting the interval of possible values for Lipschitz continuous functions (see Definition 2.1 of Lipschitz continuity). Here a sampling-based strategy is used to limit the requirements on the functions of the *GFM*. Having the strategy in place, calculating the interval of values from the distribution's quantile function is presented before the evaluation of the mean and standard deviation functions is considered.

Sampling-based Interval Reduction Using Lipschitz Continuity

Central to converting a *GFM* to an interval is the estimation of minimal and maximal function values depending on a set of valid inputs. More specifically, as the *GFM* is to be used in a run-time safety assessment, it is necessary to guarantee that the determined interval either matches the exact values or overestimates the range. Having the criterion of comparability and generality in mind, a sampling-based strategy is chosen here. It

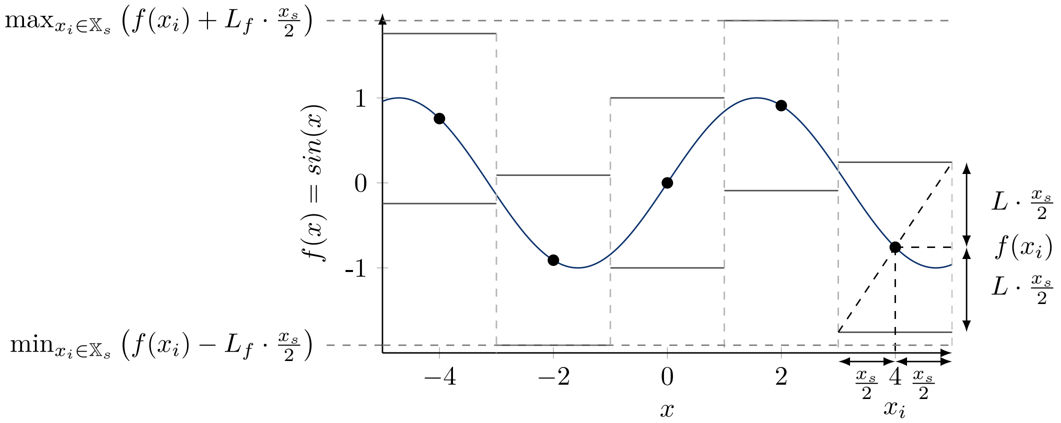


Fig. 3.6.: Sampling-based interval calculation of Lipschitz continuous function.

allows analyzing the *GFM* at various levels of granularity by adjusting the step size x_s with which a function is sampled. Moreover, it is application-independent.

The sampling-based approach is visualized in Fig. 3.6 and described by Eq. (3.19). For the sake of simplicity, only the one-dimensional case is discussed here. However, the concept can be directly extended to the multi-dimensional case.

$$\mathcal{I}_f(\mathbb{X}) = \left[\min_{x_i \in \mathbb{X}_s} \left(f(x_i) - L_f \cdot \frac{x_s}{2}\right), \max_{x_i \in \mathbb{X}_s} \left(f(x_i) + L_f \cdot \frac{x_s}{2}\right)\right] \tag{3.19}$$

Assuming a Lipschitz continuous function $f(x) : \mathbb{X} \rightarrow \mathbb{Y}$ (see Definition 2.1 of Lipschitz continuity) and its Lipschitz constant L_f, the interval bounding the minimal and maximal values of $f(x)$ can be calculated by sampling the function discretely at $x_i \in \mathbb{X}_s \subseteq \mathbb{X}$, where it is assumed that the steps x_i are distributed equidistantly according to the step size x_s. At each point x_i, the function value $f(x_i)$ is calculated. Using the Lipschitz constant L_f, which can be thought of as the maximal gradient of the function, the range of possible function values for the neighborhood of x_s is determined by adding and subtracting the value of $L_f \cdot 0.5 \cdot x_s$. As L_f is the upper limit of the gradient, multiplying it with half of the step size means that the gradient is "followed" to the left and right to calculate the minimal and maximal values. As the true gradient of $f(x)$ can be only equal to or smaller than L_f, the real function values have to be contained in the resulting interval. Therefore, by considering the Lipschitz constant, it is guaranteed that the value $f(x_i) \pm L_f \cdot 0.5 \cdot x_s$ is the smallest or greatest function value for $x \in [x_i - x_s \cdot 0.5, x_i + x_s \cdot 0.5]$. By searching for the minimal and maximal value over each $x_i \in \mathbb{X}_s$, the overall minimal and maximal function values of $f(x)$ are found or at least guaranteed to be overestimated.

Note that the Lipschitz constant L_f can be either global or local. In the latter, the Lipschitz constant can only be used for input values for which it is valid.

A drawback of this approach is the estimation of a function's Lipschitz constant. In literature, one distinguishes *white-box* approaches leveraging knowledge about the specific function and *black-box* approaches assuming no knowledge about the same [87]. In this work, the local Lipschitz constant of a function is calculated by sampling the function $f(x)$ at $x_i \in \mathbb{X}_s$ and determining the gradient using the central difference method Eq. (3.20).

$$L_f = \max_{x_i \in \mathbb{X}_s} \left\| \frac{f(x_i + \epsilon) - f(x_i - \epsilon)}{2\epsilon} \right\| \tag{3.20}$$

Although this provides a close estimate, it can not be guaranteed that L_f is indeed a valid Lipschitz constant of the considered function f. Therefore, alternative approaches (preferably providing guarantees) should be investigated in future work.

Interval Calculation for Quantile Function

Having a general strategy for determining the interval of a function's values given a range of input values in place enables realizing the calculation of an overall interval for a time- and value-correlated random distribution. For that, the first step of extracting an interval of values from its quantile function is addressed in this paragraph.

Recalling that the quantile function is the inverse of a cumulative distribution function it becomes apparent that a confidence interval can be calculated directly for the one-dimensional case, cf. Eq. (3.21).

$$\mathcal{I}_{Q_Y}(\alpha) = \left[Q_Y(\frac{\alpha}{2}), Q_Y(1-\frac{\alpha}{2})\right] \tag{3.21}$$

The confidence interval $\mathcal{I}_{Q_Y}(\alpha)$ defines the minimal and maximal value an associated random variable Y may take with the given significance α.

$$P\left(Q_Y(0.5\alpha) \leq Y \leq Q_Y(1-0.5\alpha)\right) = 1-\alpha \tag{3.22}$$

The probability as given in Eq. (3.22) is also called the *confidence level* of the confidence interval.

In contrast to the one-dimensional case, the quantile functions used in the *GFM* are multi-dimensional and use the approach of standard construction, cf. Eq. (3.8). While the same principle can be used for the first dimension, analytically calculating the interval for the second dimension results in Eqs. (3.23) to (3.26).

$$F_{(Y_1)}(y_1) = Q^{-1}_{(Y_1)}(z_1) \tag{3.23}$$

$$F_{(Y_2|Y_1=y_1)}(y_2) = Q^{-1}_{(Y_2|Y_1=y_1)}(z_1) \tag{3.24}$$

$$F_{Y_2}(y_2) = \int_{-\infty}^{\infty} F_{Y_1}(y_1) F_{(Y_2|Y_1=y_1)}(y_2)dy_1 \tag{3.25}$$

$$\mathcal{I}_{Q_{Y_2}}(\alpha) = \left[F^{-1}_{Y_2}(\alpha/2), F^{-1}_{Y_2}(1-\alpha/2)\right] \tag{3.26}$$

As one can see, the individual quantile functions of the first and second dimensions need to be inverted, combined, integrated and the resulting integral has to be inverted again. Thus, applying the analytic solution is not possible as polynomials used to represent the quantile function are not invertible in general. Instead, the sampling-based extraction of an interval of the quantile function as described in the previous paragraph is used. For that, the multi-dimensional quantile function $Q_{\mathbf{Y}}(\mathbf{z})$ is sampled on $\mathbf{z} \in [0.5\cdot\beta, 1-0.5\cdot\beta]^m$ with a step size z_s equal in each dimension. The significance β is calculated from α. The calculation presumes independence between the m dimensions as dependencies are already respected in the definition of the multi-dimensional quantile function. $Q_{(Y_2|Y_1=y_1)}(z_1)$, for instance, represents the quantile function of Y_2 for the case of $Y_1 = y_1$, which already encodes dependency.

The calculation is shown exemplarily for the two-dimensional case in Eqs. (3.27) to (3.33).

$$P\left(Y_{L_1} \leq Y_1 \leq Y_{U_1} \wedge Y_{L_2} \leq Y_2 \leq Y_{U_2}\right) = 1 - \alpha \tag{3.27}$$

$$P\left(Y_{L_1} \leq Y_1 \leq Y_{U_1}\right) \cdot P\left(Y_{L_2} \leq Y_2 \leq Y_{U_2}\right) = 1 - \alpha \tag{3.28}$$

$$P\left(Y_{L_1} \leq Y_1 \leq Y_{U_1}\right) = P_{Y_1} \tag{3.29}$$

$$P\left(Y_{L_2} \leq Y_2 \leq Y_{U_2}\right) = P_{Y_2} \tag{3.30}$$

$$P_{Y_1} \cdot P_{Y_2} = 1 - \alpha \tag{3.31}$$

$$P_{Y_1} \cdot P_{Y_2} = (1 - \beta)^2 = 1 - \alpha \tag{3.32}$$

$$\beta = 1 - \sqrt[m=2]{(1 - \alpha)} \tag{3.33}$$

Here, Y_{L_1} and Y_{L_2} denote the lower bounds of the interval $\mathcal{I}_{Q_{Y_1}}(\alpha)$ and $\mathcal{I}_{Q_{Y_2}}(\alpha)$ respectively. Similarly, Y_{U_1} and Y_{U_2} denote the upper bounds of the respective dimensions.

Note that due to the use of Lipschitz constants entailed by employing the sampling-based strategy to extract the interval from the quantile function, the equality in Eq. (3.27) does not hold. Thus, despite being based on the idea of confidence intervals, the interval extracted from the quantile function is **not** a confidence interval. However, it is guaranteed that Eq. (3.34) holds.

$$P\left(\bigwedge_{i=1}^{m} Y_i \in \mathcal{I}_{Q_{Y_i}}(\alpha)\right) \geq 1 - \alpha \tag{3.34}$$

Interval Calculation Considering Time- and Value-Correlations

Using the approach of the previous paragraph facilitates extracting an interval from a multi-dimensional quantile function. The time- and value-correlated random distribution defined in Section 3.1.1, however, additionally defines mean and standard deviation functions to model correlations. Thus, to determine the interval for the correlated distribution, these have to be taken into account as well.

In this endeavor, the sampling-based strategy presented in Section 3.2.1 is reused to determine the minimal and maximal scaling and shift values, cf. Eqs. (3.35) and (3.36)

$$\mathcal{I}_{\mu}(\mathbb{K}, \mathbb{O}) = \left[\min_{k \in \mathbb{K}, \mathbf{o}_k \in \mathbb{O}} \left(\mu_Y(k, \mathbf{o}_k)\right), \max_{k \in \mathbb{K}, \mathbf{o}_k \in \mathbb{O}} \left(\mu_Y(k, \mathbf{o}_k)\right)\right] \tag{3.35}$$

$$\mathcal{I}_{\sigma}(\mathbb{K}, \mathbb{O}) = \left[\min_{k \in \mathbb{K}, \mathbf{o}_k \in \mathbb{O}} \left(\sigma_Y(k, \mathbf{o}_k)\right), \max_{k \in \mathbb{K}, \mathbf{o}_k \in \mathbb{O}} \left(\sigma_Y(k, \mathbf{o}_k)\right)\right] \tag{3.36}$$

With these, interval arithmetic [88] can be used to calculate the final result, cf. Eq. (3.37).

$$\mathcal{I}_Y(\mathbb{K}, \mathbb{O}, \alpha) = \mathcal{I}_{\sigma_Y}(\mathbb{K}, \mathbb{O}) \cdot \mathcal{I}_{Q_Y}(\alpha) + \mathcal{I}_{\mu_Y}(\mathbb{K}, \mathbb{O}) \tag{3.37}$$

In accordance with Eq. (3.4), the interval extracted from the distribution's quantile function is scaled by the interval derived from the standard deviation function $\sigma_Y(k, \mathbf{o}_k)$ and finally shifted according to the interval of mean values extracted from $\mu_Y(k, \mathbf{o}_k)$.

Note that by specifying $\mathbb{K}$ and $\mathbb{O}$, the interval calculated from a time- and value-correlated random distribution is either valid for a time horizon and range of observations or, in case both sets contain a single value, for a specific point in time k and a single observation $\mathbf{o}_k$. Therefore, the approach supports analyzing the failure model at various levels of granularity and thereby underlines the fulfillment of the comparability criterion by the *GFM*.

3.2.2. A Failure Type's Interval

As the concept of a time- and value-correlated random distribution is used repeatedly to construct a failure type, it is central to extracting its interval of possible failure amplitudes as well. Thus, using the approach presented in the previous section the calculation of a failure type's interval can be defined.

Corresponding to Eq. (3.1), intervals of the state function $s_n(k, \mathbf{o}_k)$ and the failure amplitudes $f_n(k, \mathbf{o}_k)$ have to be extracted such that Eq. (3.38) can be applied.

$$\mathcal{I}_{F_n}(\mathbb{K}, \mathbb{O}) = \mathcal{I}_{s_n}(\mathbb{K}, \mathbb{O}) \cdot \mathcal{I}_{f_n}(\mathbb{K}, \mathbb{O}) \tag{3.38}$$

Both aspects are described in the following paragraphs. While the first addresses the extraction of an interval from a failure type's failure amplitudes function, the second addresses the determination of an interval from a failure type's state function.

Extracting an Interval of a Failure Type's Failure Amplitudes

Two components have to be considered when determining the interval of possible failure amplitudes for a failure type, its failure pattern $p_n(t_n)$ and the scaling thereof $scl_n(k, \mathbf{o}_k)$, cf. Eq. (3.9).

By using the sampling-based strategy to evaluate the failure type's pattern $p_n(t_n)$ on the time interval $t_n \in [0, 1]$ the range of normalized failure amplitudes $\mathcal{I}_{p_n}$ can be calculated. Similarly, the approach for obtaining an interval from a time- and value-correlated random distribution as discussed in Section 3.2.1 facilitates extracting $\mathcal{I}_{scl_n}(\mathbb{K}, \mathbb{O})$ from the scaling distribution. In accordance with Eq. (3.9), both intervals are multiplied to obtain the interval of possible failure amplitudes introduced by the n-th failure type, cf. Eq. (3.39).

$$\mathcal{I}_{f_n}(\mathbb{K}, \mathbb{O}) = \mathcal{I}_{p_n}([0, 1]) \cdot \mathcal{I}_{scl_n}(\mathbb{K}, \mathbb{O}) \tag{3.39}$$

Extracting an Interval of a Failure type's State Function

Whether or not the failure amplitudes stated by $\mathcal{I}_{f_n}(\mathbb{K}, \mathbb{O})$ are actually contributing to the overall failure characteristics depends on whether or not the failure type will be active. For that, the failure type's state function $s_n(k, \mathbf{o}_k)$ has to be evaluated. This is done in two steps. Firstly, intervals for the activation function $a_n(k, \mathbf{o}_k)$ and deactivation function $d_n(k, \mathbf{o}_k)$ are determined and combined to generate the likeliness of a failure type being active. Secondly, the likeliness is mapped to an interval representing the activation state of the considered failure type over $\mathbb{K}$ and $\mathbb{O}$.

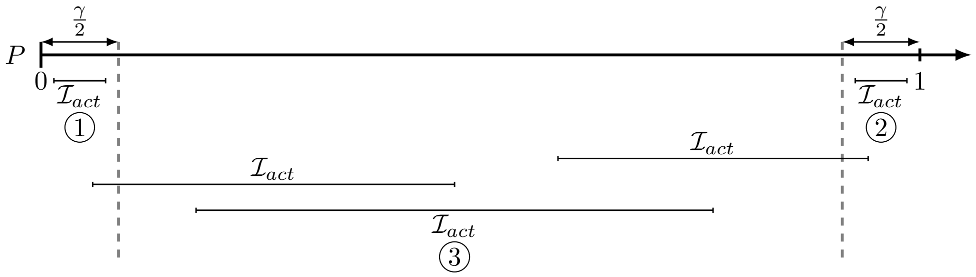


Fig. 3.7.: Using γ to map the activation interval $\mathcal{I}_{act}$ to the resulting state function interval $\mathcal{I}_{s_n}$.

Determining the Likeliness of Activation To obtain the likeliness of a failure type being active, the functions $a_n(k, \mathbf{o}_k)$ and $d_n(k, \mathbf{o}_k)$ are evaluated for generating the respective intervals $\mathcal{I}_{a_n}(\mathbb{K}, \mathbb{O})$ and $\mathcal{I}_{d_n}(\mathbb{K}, \mathbb{O})$ in a first step, cf. Eqs. (3.40) and (3.41).

$$\mathcal{I}_{a_n}(\mathbb{K}, \mathbb{O}) = \left[a_{min} = \min_{k \in \mathbb{K}, \mathbf{o}_k \in \mathbb{O}} (a_n(k, \mathbf{o}_k)), a_{max} = \max_{k \in \mathbb{K}, \mathbf{o}_k \in \mathbb{O}} (a_n(k, \mathbf{o}_k))\right] \tag{3.40}$$

$$\mathcal{I}_{d_n}(\mathbb{K}, \mathbb{O}) = \left[d_{min} = \min_{k \in \mathbb{K}, \mathbf{o}_k \in \mathbb{O}} (d_n(k, \mathbf{o}_k)), d_{max} = \max_{k \in \mathbb{K}, \mathbf{o}_k \in \mathbb{O}} (d_n(k, \mathbf{o}_k))\right] \tag{3.41}$$

In a second step, the duration of activations ($\mathcal{I}_{d_n}$), as well as the time between two activations ($\mathcal{I}_{a_n}$), need to be combined, cf. Eq. (3.42).

$$\mathcal{I}_{act} = \left[P_{min} = \frac{d_{min}}{d_{min} + a_{max}}, P_{max} = \frac{d_{max}}{d_{max} + a_{min}}\right] \tag{3.42}$$

The idea is to calculate the proportion of the time a failure type is active to the overall length of a period, that is, the time the failure type is active and inactive. Correspondingly, for the minimal likeliness, the shortest possible duration d_{min} is considered along with the maximal time between two occurrences a_{max}. For the maximal likeliness, on the other hand, the longest duration d_{max} is combined with the minimal time between two occurrences a_{min}. Finally, $\mathcal{I}_{act}$ states the likeliness of the failure being active.

Determining the Activation State of a Failure Type The interval $\mathcal{I}_{act}$ needs to be mapped to an interval representing the activation state of the failure type to generate $\mathcal{I}_{s_n}(\mathbb{K}, \mathbb{O})$. For that, three cases need to be distinguished. These are visualized in Fig. 3.7.

① **The failure type is always inactive** In this case, the failure type does not introduce any failure amplitudes. Therefore, the resulting interval has to be empty, that is, $\mathcal{I}_{s_n}(\mathbb{K}, \mathbb{O}) = [0, 0]$.

② **The failure type is always active** In this case the interval $\mathcal{I}_{f_n}(\mathbb{K}, \mathbb{O})$ already states the range of possible failure amplitudes. The interval resulting from the state function is $\mathcal{I}_{s_n}(\mathbb{K}, \mathbb{O}) = [1, 1]$.

③ **The failure type may be active or inactive** In this case, the failure type may produce failure amplitudes as described by the interval $\mathcal{I}_{f_n}(\mathbb{K}, \mathbb{O})$ or does not introduce failure amplitudes. To reflect this, the interval resulting from the state function is $\mathcal{I}_{s_n}(\mathbb{K}, \mathbb{O}) = [0, 1]$.

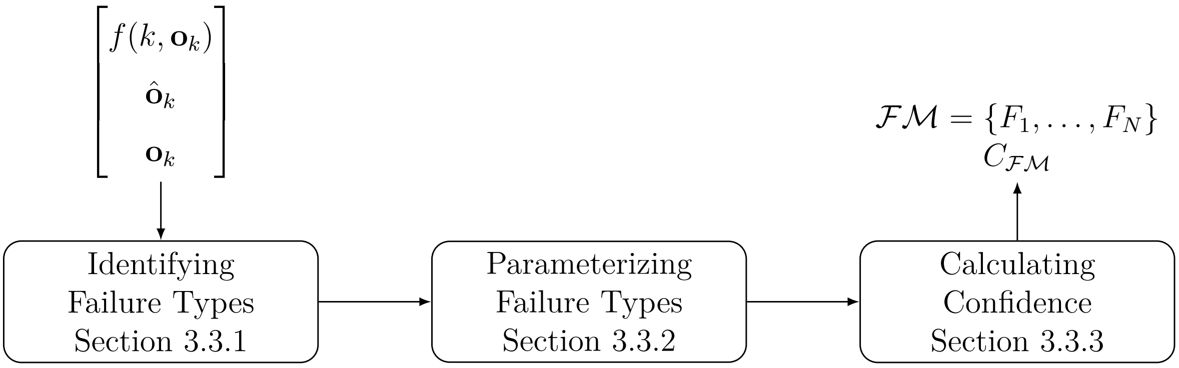

$\begin{bmatrix} f(k,\mathbf{o}_k) \\ \hat{\mathbf{o}}_k \\ \mathbf{o}_k \end{bmatrix}$
$\mathcal{FM} = \{F_1, \ldots, F_N\}$
$C_{\mathcal{FM}}$
Identifying Failure Types Section 3.3.1
Parameterizing Failure Types Section 3.3.2
Calculating Confidence Section 3.3.3

Fig. 3.8.: Overview of the proposed processing chain for generating a *GFM* from time series of failure amplitudes.

In the endeavor of generating the corresponding interval $\mathcal{I}_{s_n}(\mathbb{K},\mathbb{O})$, the interval stating the likeliness of the failure type being active is analyzed according to Eqs. (3.43) to (3.45).

$$s_{n_{min}} = \begin{cases} 1 & P_{min} \geq 1.0 - \frac{\gamma}{2.0} \\ 0 & otherwise \end{cases} \tag{3.43}$$

$$s_{n_{max}} = \begin{cases} 0 & P_{max} \leq \frac{\gamma}{2.0} \\ 1 & otherwise \end{cases} \tag{3.44}$$

$$\mathcal{I}_{s_n}(\mathbb{K},\mathbb{O}) = [s_{n_{min}}, s_{n_{max}}] \tag{3.45}$$

Here γ is a confidence value defining the threshold that has to be exceeded or undercut to consider a failure type being active or inactive respectively. The idea of this mapping is visualized in Fig. 3.7. The figure displays three positions at which the borders of $\mathcal{I}_{act}$ can be with regard to γ. In ①, P_{min} and P_{max} are smaller than $0.5 \cdot \gamma$. Therefore, the first case applies, the failure type is considered to be always inactive and $\mathcal{I}_{s_n}(\mathbb{K},\mathbb{O}) = [0,0]$ is assumed. Contrarily, in ② P_{min} and P_{max} are greater than $1 - 0.5 \cdot \gamma$. Consequently, it is assumed that the failure type is always active and $\mathcal{I}_{s_n}(\mathbb{K},\mathbb{O}) = [1,1]$ results. For any combination in between these two cases, as exemplary visualized in ③, the failure type has to be assumed to be active and inactive for the given ranges of $\mathbb{K}$ and $\mathbb{O}$ and therefore $\mathcal{I}_{s_n}(\mathbb{K},\mathbb{O}) = [0,1]$ is assumed.
Using the interval extracted from a failure type's state function, its overall interval is defined according to Eq. (3.38). Finally, the overall failure amplitudes of the *GFM* is nothing but the sum over the intervals defined by each failure type, cf. Eq. (3.17).

3.3. Processing Chain for Generating Generic Failure Models

The *Generic Failure Model (GFM)* introduced in Section 3.1 has yet to be shown applicable to modeling failure characteristics. Moreover, as required by the confidence criterion, a measure facilitating to assess the confidence an application can have in the correctness of the failure characteristics represented by a *GFM* has yet to be defined.

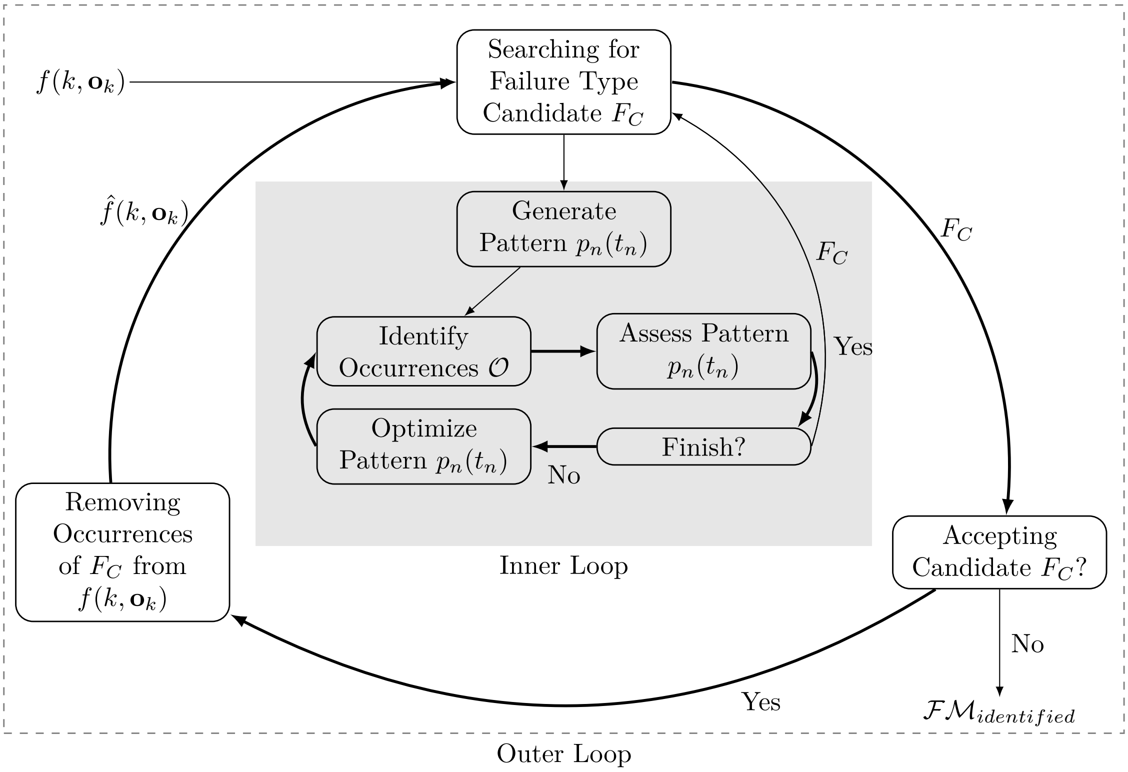

$f(k, \mathbf{o}_k)$
Searching for Failure Type Candidate F_C
$\hat{f}(k, \mathbf{o}_k)$
Generate Pattern $p_n(t_n)$
F_C
F_C
Identify Occurrences $\mathcal{O}$
Assess Pattern $p_n(t_n)$
Yes
Optimize Pattern $p_n(t_n)$
Finish?
No
Removing Occurrences of F_C from $f(k, \mathbf{o}_k)$
Inner Loop
Accepting Candidate F_C?
No
Yes
$\mathcal{FM}_{identified}$
Outer Loop

Fig. 3.9.: Iterative process for identifying failure types.

In this endeavor, this section proposes a processing chain that (i) identifies failure types in a given series of failure amplitudes $f(k, o_k)$, (ii) generates a *GFM* to represent the failure characteristics, and (iii) calculates a confidence value with respect to the initial failure amplitudes. As such, the stages described in this work substantially correct and extend the processing chain introduced in [7]. The individual stages are visualized in Fig. 3.8 and the associated sections are referenced.

Correspondingly, the next subsection describes an approach for identifying failure types in a given time series of failure amplitudes. The prototypical failure types are parameterized in the second stage, during which the polynomials modeling a failure type's individual functions are fitted. In the last stage, the generated failure model $\mathcal{FM}$ is converted to an interval-based representation as described in Section 3.2 to calculate a final confidence value expressing its quality.

3.3.1. Identifying Failure Types

Given a time series of failure amplitudes $f(k, \mathbf{o}_k)$ [2], the first stage of the processing chain aims at identifying candidates of failure types. For that, an iterative process encompassing an inner and outer loop is proposed, cf. Fig. 3.9.

The inner loop realizes a search for a failure type candidate F_C on the given failure amplitudes $f(k, \mathbf{o}_k)$. To that end, it starts with generating a random failure pattern $p_n(t_n)$ by randomizing the weights Ω of a polynomial. Identifying its occurrences

[2] The processing chain is capable of handling sets of time series as well. For the sake of simplicity, the text refers to only a single time series.

in $f(k, \mathbf{o}_k)$, assessing the pattern $p_n(t_n)$, and optimizing it accordingly results in an iterative process at whose end a candidate F_C is produced.
This candidate is examined by the outer loop to either accept or reject it. In the former case, the initial failure amplitudes $f(k, \mathbf{o}_k)$ are updated to prepare for the next iteration while the process is stopped in the latter case.
Both, the inner and outer loops, are discussed in detail in the next subsections.

Generating Failure Type Candidates - Inner Loop

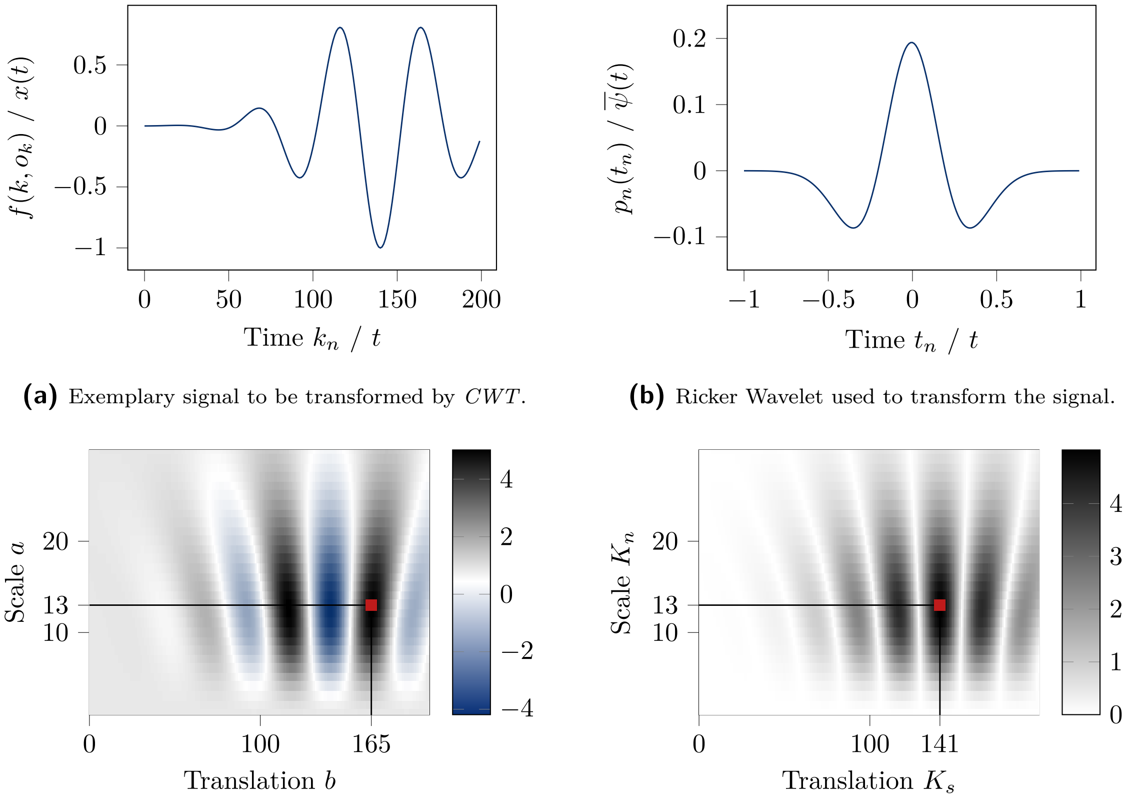
The goal of the inner loop is to generate a failure type candidate F_C. For that, a failure pattern $p_n(t_n)$ has to be generated, optimized, and its occurrences within the given failure amplitudes $f(k, \mathbf{o}_k)$ have to be found. The corresponding phases of the inner loop are described below, cf. Fig. 3.9.

Generate Pattern $p_n(t_n)$ The first step of the inner loop prepares its iterations. In accordance with the inner loop's goal of identifying a failure type's failure pattern, this step starts with generating a polynomial and randomizing its set of weights Ω to values $\omega \in [-1, 1]$. The polynomial is then considered to represent the failure pattern $p_n(t_n)$(cf. Eq. (3.16)) is to be optimized regarding its occurrences in $f(k, \mathbf{o}_k)$.

Identifying Occurrences $\mathcal{O}$ The randomly initialized failure pattern represents the central component of the failure type candidate to be produced. In the endeavor of assessing and optimizing it, its occurrences within the failure amplitudes have to be identified first. More specifically, for the failure pattern $p_n(t_n)$ it needs to be known at which time K_s an occurrence starts, for which duration K_n the failure type is active, and with which scaling $\mathbf{scl}$ the pattern occurs. This shall result in a set of occurrences $\mathcal{O} = \{O_1 = (K_s, K_n, \mathbf{scl}), O_2, \ldots\}$ of the considered failure pattern $p_n(t_n)$.
In the endeavor of identifying the occurrences in $f(k, \mathbf{o}_k)$, that is, identifying the parameters K_s and K_n for each occurrence, the *Continuous Wavelet Transformation (CWT)* is used [89].
In general, the *CWT* calculates the correlation of a signal $x(t)$ with a shorter signal, called wavelet, $\psi(t)$ [89]. It is defined as follows.

$$X(a,b) = \frac{1}{|a|^{1/2}} \int_{-\infty}^{\infty} x(t)\overline{\psi}\left(\frac{t-b}{a}\right) dt \tag{3.46}$$

Here $\overline{\psi}(\frac{t-b}{a})$ is the complex conjugate of the *mother wavelet.* It is the base function which is scaled, by parameter a, and translated, by parameter b, to generate daughter wavelets. These are convoluted with the signal $x(t)$ to obtain the transformed $X(a, b)$.
Fig. 3.10 shows an exemplary signal (Fig. 3.10a) which is transformed using the Ricker Wavelet (Fig. 3.10b). The result of the transformation is a matrix, which can be displayed as a heat map and is called a *scalogram*, cf. Fig. 3.10c. Its highest (and lowest) values indicate the time step (b) and duration (a) with which the Ricker Wavelet matches the signal $x(t)$ best.
Assuming $f(k, \mathbf{o}_k)$ to be the signal $x(t)$ and $p_n(t_n)$ the *wavelet* whose occurrences one is interested in, the parameter b would state the start of an identified occurrence K_s and



(a) Exemplary signal to be transformed by *CWT*.

(b) Ricker Wavelet used to transform the signal.

(c) *CWT* of signal according to Eq. (3.46). The scale a and translation b achieving the maximal correlation value is highlighted at $(b = 165, a = 13)$.

(d) Scalogram according to Eq. (3.47) used to identify occurrences of failure pattern. The scale K_n and translation K_s achieving the maximal correlation value is highlighted at $(b = 141, a = 13)$.

Fig. 3.10.: Scalogram according to Eq. (3.47) used to identify occurrences of failure pattern.

the parameter a the duration K_n of the same. With this in mind, Eq. (3.46) is adapted to match the task of identifying occurrences of a failure pattern, cf. Eq. (3.47).

$$X(K_n, K_s) = \left| \frac{1}{|K_n|^{1/2}} \sum_{k=-\infty}^{\infty} f(k, o_k) p_n \left(\frac{k - K_s}{K_n} \right) \right| \tag{3.47}$$

Here, the signal of which the scalogram is to be calculated are the failure amplitudes $f(k, o_k)$ while the mother wavelet is represented by the failure pattern $p_n(t_n)$. In contrast to the *CWT* as defined in Eq. (3.46), however, the absolute value of the discretized correlation is considered. The reason is that the original *CWT* indicates negative and positive correlations of the wavelet with the signal by correspondingly signed values of $X(a, b)$. These stem from the sign of the signal and wavelet functions. In case of the failure pattern $p_n(t_n)$ being the wavelet, the sign will be accounted for later by its scaling **scl**. Therefore, as one is interested in finding the maximal correlation independently of the signs, the absolute value is considered. The difference between both when searching for the maximal correlation value is clarified by comparing Fig. 3.10c and Fig. 3.10d. While the maximal correlation value of the signed *CWT* can be found at $b = 165$ with a value of 4.19, the maximal absolute correlation is found at $K_s = 141$ with a value of 5.02.

The produced scalogram is then featuring maximal values at scales K_n and translations K_s where the failure pattern $p_n(t_n)$ matches the failure amplitudes $f(k, \mathbf{o}_k)$ best. To identify the individual occurrences, these maximal correlations are searched for and K_n and K_s are extracted in such a way that the resulting occurrences $O_n \in \mathcal{O}$ do not overlap.

Note that the description provided here considers the one-dimensional case. For applying the same procedure to multi-dimensional data, the scalogram is calculated and summed for each dimension before occurrences are identified.

After identifying the temporal parameter for the occurrences of a failure pattern $p_n(t_n)$, its scaling value $\mathbf{scl}$ (cf. Eq. (3.9)) at each occurrence O_n has to be determined as well. For that, for each time step K_s of occurrence O_n where the pattern occurs with duration K_n a scaling value $\mathbf{scl}$ that maximizes the overlapping area of the failure pattern with the failure amplitudes $f(k, \mathbf{o}_k), k \in [K_s, K_s + K_n]$ has to be found. This is achieved by minimizing Eq. (3.48).

$$r(K_s, K_n) = \frac{\sum_{k=K_s}^{K_s+K_n} |f(k, \mathbf{o}_k) - p(\frac{k-K_s}{K_n}) \cdot scl|}{\sum_{k=K_s}^{K_s+K_n} |f(k, \mathbf{o}_k)|} \tag{3.48}$$

Here the absolute difference between the failure amplitudes and the scaled failure pattern is summed to obtain the area of failure amplitudes remaining after removing what is modeled by the pattern. Dividing it by the initial area of failure amplitudes provides a measure of reduction accomplished by the failure pattern at the occurrence when using a scaling value $\mathbf{scl}$. By iteratively choosing different $\mathbf{scl}$ values a sequential search is realized with which the optimal scaling is found. The procedure is repeated for all time steps K_s with duration K_n identified by the CWT approach before. As a result, a failure type candidate F_C now comprises a failure pattern $p_n(t_n)$ and a set of occurrences $\mathcal{O} = \{O_1 = (K_s, K_n, \mathbf{scl}), O_2, \ldots\}$

Assessing a Failure Pattern $p_n(t_n)$ The set of occurrences identified for a failure pattern by the previous phase is used to assess the same regarding its suitability to form a failure type candidate F_C. For that, a cost function is defined, cf. Eq. (3.49).

$$c(\mathcal{O}, p_n(t_n)) = \sum_{O_n \in \mathcal{O}} \sum_{k=K_s(O_n)}^{K_n(O_n)} \left[f(k, \mathbf{o}_k) - p_n\left(\frac{k - K_s}{K_n(O_n)}\right) \cdot \mathbf{scl}(O_n) \right]^2 \tag{3.49}$$

Note that $K_s(O_n), K_n(O_n), \mathbf{scl}(O_n)$ denote the starting time step K_s, duration K_n, and scaling value $\mathbf{scl}$ of the n-th occurrence O_n. Similar as before, the cost function assesses the overall reduction that is achieved by a failure type candidate by subtracting the scaled occurrences of the failure pattern from the given failure amplitudes. Considering all iterations of the inner loop, the failure type candidate producing the minimum cost $c(\mathcal{O}, p_n(t_n))$ is saved and returned when the process is finished.

Finishing the Process As depicted in Fig. 3.9, the assessment of a failure pattern according to its occurrences is used to decide on whether or not to continue the inner loop. For that, two conditions are checked.

Firstly, the loop is terminated if the predefined number of iterations $I_{Identification}$ is exceeded. At that point, it is assumed that no significant improvements are made.

Secondly, the loop is terminated if the iteration at which the best candidate was observed is I_{best} iterations ago. This similarly ensures that the loop is terminated if no significant improvements are made.

Optimizing the Failure Pattern $p_n(t_n)$ If the process is not finished, the failure pattern is optimized to better fit the failure amplitudes at its identified occurrences. This means that the weights Ω of the polynomial representing the failure pattern have to be adjusted such that Eq. (3.49) is to be minimized. In this endeavor, *Stochastic Gradient Descent (SGD)* is used in this work.
SGD is a technique for optimizing non-convex objective functions. It is commonly used, for instance, in training neural networks but is not limited to this type of functions [90]. The central idea is to repeatedly evaluate a cost function for calculating its gradients regarding the parameters of the used model and update the model parameter according to the gradients. With this iterative approach, the model parameters are successively optimized.
Here, the failure pattern $p_n(t_n)$ with its parameters Ω is the model which is to be optimized according to the cost function Eq. (3.49). Therefore, in this step, the gradients of the cost function are calculated and used for updating the failure pattern's weights. More specifically, automatic differentiation [91] is employed to obtain the gradients which are used to update the weights according to the AMSGrad update rule [92]. This rule was empirically shown to improve the performance of the *SGD* algorithm in general, other update rules could be used as well.

The result of the inner loop is the failure type candidate F_C encompassing a failure pattern $p_n(t_n)$ as well as its identified occurrences $\mathcal{O}$. Whether or not the resulting failure type is accepted to the identified failure model $\mathcal{FM}_{identified}$, however, is determined in the second step of the outer loop.

Decomposing Failure Amplitudes - Outer Loop

The inner loop for identifying failure types produced a failure type candidate F_C, which simultaneously addresses the first step of the outer loop. This loop, however, has to to decide on whether or not the candidate is acceptable and, if so, prepare the next iteration by removing the occurrences of the accepted failure type from the failure amplitudes to produce $\hat{f}(k, \mathbf{o}_k)$.

Accepting a Failure Type Candidate While the decision to accept or reject a candidate can be based on various aspects, two are focused on in this work. Firstly, the number of occurrences has to suffice, cf. Eq. (3.50).

$$|\mathcal{O}| \geq N_{\mathcal{O}} \tag{3.50}$$

By requiring a number of at least $N_{\mathcal{O}}$ occurrences, the next stage of the processing chain is prepared. In contrast to the failure pattern $p_n(t_n)$, the remaining functions of the failure type are not represented as polynomials yet. To determine their parameters in the next stage of the processing chain, sufficient samples, e.g. for the time between two consecutive occurrences, have to be available. This is ensured by Eq. (3.50).

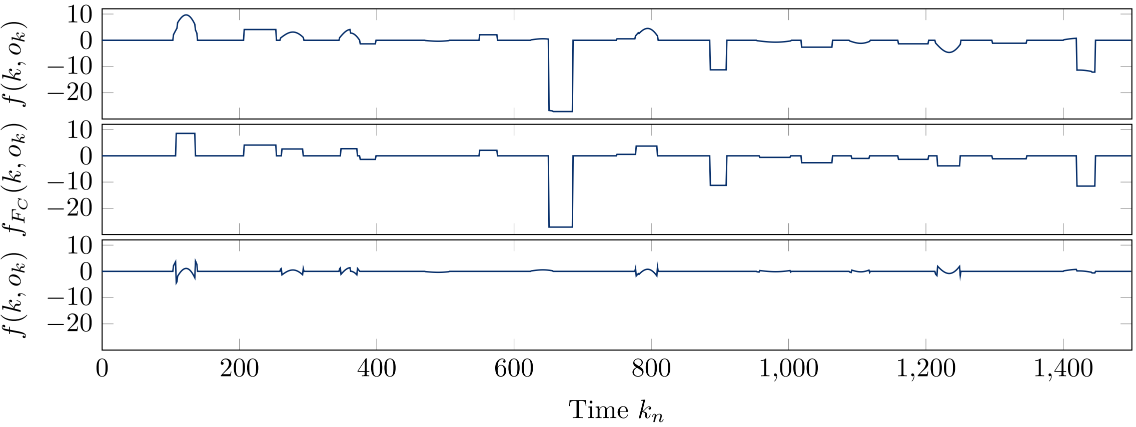


Fig. 3.11.: Exemplary visualization of removing occurrences of a failure type candidate F_C to generate updated failure amplitudes $\hat{f}(k, o_k)$, which become $f(k, o_k)$ in the iterative process of identifying failure types. For the sake of visualization, one-dimensional data is considered here.

The second condition ensures that the candidate matches the given failure amplitudes sufficiently. For that, the failure amplitudes of a failure type candidate are reconstructed as described by Eq. (3.51).

$$f_{F_C}(k, \mathbf{o}_k) = \\ \begin{cases} 0, & \nexists O_n \in \mathcal{O} \ s.t. \ k \in [K_s(O_n), K_s(O_n) + K_n(O_n)] \\ \mathbf{scl}(O_n) \cdot p_n\left(\frac{k - K_s(O_n)}{K_n(O_n)}\right), & O_n \in \mathcal{O} \ s.t. \ k \in [K_s(O_n), K_s(O_n) + K_n(O_n)] \end{cases} \tag{3.51}$$

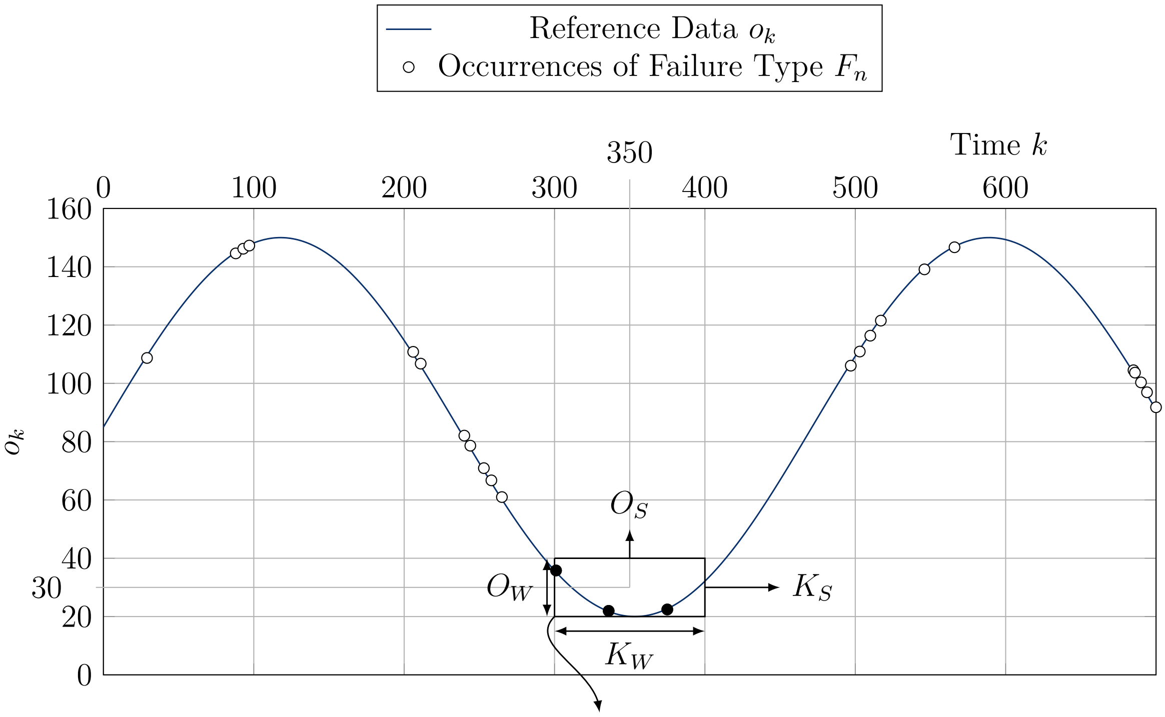
With these failure amplitudes in place, the second condition is given by Eq. (3.52).

$$\sum_{k}^{K} |f(k, \mathbf{o}_k) - f_{F_C}(k, \mathbf{o}_k)| \leq \sum_{k}^{K} |f(k, \mathbf{o}_k)| \tag{3.52}$$

Here K denotes the number of time steps in $f(k, \mathbf{o}_k)$. This condition requires a failure type candidate to minimize the overall area of failure amplitudes. It ensures that each failure type in the final failure model describes a sufficiently relevant property of the overall failure characteristics.

Removing Occurrences If the failure type candidate F_C generated by the inner loop fulfills the conditions listed in the previous paragraph, it is added to the preliminary failure model. To enable searching for another candidate focusing on currently undescribed failure amplitudes, however, the occurrences of the accepted failure type have to be removed.

The idea is visualized in Fig. 3.11. The upper diagram shows the initial failure amplitudes $f(k, o_k)$ passed to the first stage of the processing chain. From these, the failure type candidate comprising a failure pattern $p_n(t_n)$ and its occurrences were identified. By applying Eq. (3.51) these are used to construct $f_{F_C}(k, o_k)$, which is shown in the second diagram. Subtracting these reconstructed failure amplitudes that are described by the accepted failure type from the initial failure amplitudes $f(k, o_k)$ generates the updated failure amplitudes $\hat{f}(k, o_k)$ as shown in the lower diagram. These contain



$$\hat{\mathcal{O}}(k, o_k) = \left\{ q \in \mathcal{O} \middle| K_s(q) \in \left[k - \frac{K_W}{2}, k + \frac{K_W}{2}\right] \wedge o_k(q) \in \left[o_k - \frac{O_W}{2}, o_k + \frac{O_W}{2}\right] \right\} \tag{3.53}$$

Fig. 3.12.: Sliding window approach for identifying time- and value-correlations in the occurrences $O_n \in \mathcal{O}$ of a single failure type $F_C \in \mathcal{FM}_I$.

only failure amplitudes not considered by any of the previously accepted failure types. Therefore, the next iteration of the identification process is executed using these updated failure amplitudes.

Once the remaining failure amplitudes are sufficiently low, that is $\sum |f(k, \mathbf{o}_k)| \leq A_{f_{min}}$ or the failure type candidate produced by the inner loop is not accepted, the identification process is stopped and the identified failure model $\mathcal{FM}_{identified}$ is passed to the second stage of the processing chain.

3.3.2. Parameterizing Failure Types

The failure model $\mathcal{FM}_{identified}$ produced by the last stage consists of failure type candidates, each represented by a failure pattern $p_n(t_n)$ and a set of occurrences $\mathcal{O}$. These are used in this stage to parameterize each failure type. This means that the polynomials used to represent the time- and value-correlated random distributions $a_n(k, \mathbf{o}_k)$ (TBF), $d_n(k, \mathbf{o}_k)$ (TtR), and $scl_n(k, \mathbf{o}_k)$ (the scaling distribution) have to be fitted. As each of these is represented by a time- and value-correlated random distribution, which in turn requires three functions to be represented, a total of nine polynomials need to be fitted for each identified failure type.

In this endeavor, the first step is to generate training data representing the function to be modeled by a polynomial. The second step employs polynomial regression to calculate the optimal set of parameters Ω from the training data such that the resulting polynomial represents the required function.

To obtain the training data comprising the input values $(k, \mathbf{o}_k)$ and the output values of the function to be represented, a sliding window approach is used. The approach is visualized in Fig. 3.12 using artificial data generated for the sake of argumentation. For the failure type candidate F_C under consideration, the occurrences $O_n \in \mathcal{O}$ are associated with the reference signal $\mathbf{o}_k$. Then a window covering K_w time steps and spanning $\mathbf{O}_W$ in the value domain is defined. This window is shifted along the time axis with a step size of K_S and along the value axis with step size $\mathbf{O}_S$.

In Fig. 3.12 a single position of the window centered around $k = 350$ and $o_k = 30$ is exemplarily highlighted. Indicated by the different colors, only the occurrences that are within the current window (circles filled black) are considered to form $\hat{\mathcal{O}}$, cf. Eq. (3.53). The remaining occurrences outside the current window (circles filled white) are not considered.

$\hat{\mathcal{O}}$ is the basis on which the training data is calculated. To calculate the *Time to Repair (TtR)*, for instance, the mean, standard deviation, and normalized values of $K_n(O_n)$ are calculated, cf. Eqs. (3.54) to (3.56)

$$\hat{\mu}_d(k, \mathbf{o}_k) = \frac{1}{|\hat{\mathcal{O}}|} \sum_{O_n \in \hat{\mathcal{O}}} K_n(O_n) \tag{3.54}$$

$$\hat{\sigma}_d(k, \mathbf{o}_k) = \sqrt{\frac{1}{|\hat{\mathcal{O}}|} \sum_{O_n \in \hat{\mathcal{O}}} (K_n(O_n) - \hat{\mu}_d(k, \mathbf{o}_k))^2} \tag{3.55}$$

$$\hat{\mathcal{O}}'_d(k, \mathbf{o}_k) = \left\{ \frac{K_n(O_n) - \hat{\mu}_d(k, \mathbf{o}_k)}{\hat{\sigma}_d(k, \mathbf{o}_k)} \middle| \forall O_n \in \hat{\mathcal{O}} \right\} \tag{3.56}$$

While Eq. (3.54) and Eq. (3.55) consider the duration for which an occurrence is active to calculate the corresponding mean and standard deviation, Eq. (3.56) uses these values to normalize the durations $K_n(O_n)$. The intention is the following.

In contrast to the mean and standard deviation, which directly represent a single data point within the training data for the corresponding polynomials, the quantile function representing the normalized distribution is assumed to be constant for all occurrences. Therefore, while the mean and standard deviation are time- and value-correlated, meaning that their values depend on the occurrences within the current window, the normalized distribution should remain the same across all windows of the sliding window approach. Correspondingly, to obtain the training data for the normalized distribution, the normalized duration values are collected for all windows.

Similar to the deactivation, the training data for the activation distribution, that is, for the *Time Between Failure (TBF)* is calculated according to Eqs. (3.57) to (3.59).

$$\hat{\mu}_a(k, \mathbf{o}_k) = \frac{1}{|\hat{\mathcal{O}}|} \sum_{O_n \in \hat{\mathcal{O}}} (K_s(O_{n+1}) - K_s(O_n) - K_n(O_n)) \tag{3.57}$$

$$\hat{\sigma}_a(k, \mathbf{o}_k) = \sqrt{\frac{1}{|\hat{\mathcal{O}}|} \sum_{O_n \in \hat{\mathcal{O}}} ((K_s(O_{n+1}) - K_s(O_n) - K_n(O_n)) - \hat{\mu}_a(k, \mathbf{o}_k))^2} \tag{3.58}$$

$$\hat{\mathcal{O}}'_a(k, \mathbf{o}_k) = \left\{ \frac{(K_s(O_{n+1}) - K_s(O_n) - K_n(O_n)) - \hat{\mu}_a(k, \mathbf{o}_k)}{\hat{\sigma}_a(k, \mathbf{o}_k)} \middle| \forall O_n \in \hat{\mathcal{O}} \right\} \tag{3.59}$$

Opposed to the deactivation, here the starting time steps K_s of two consecutive occurrences O_n are considered together. Assuming that O_{n+1} is the direct successor of

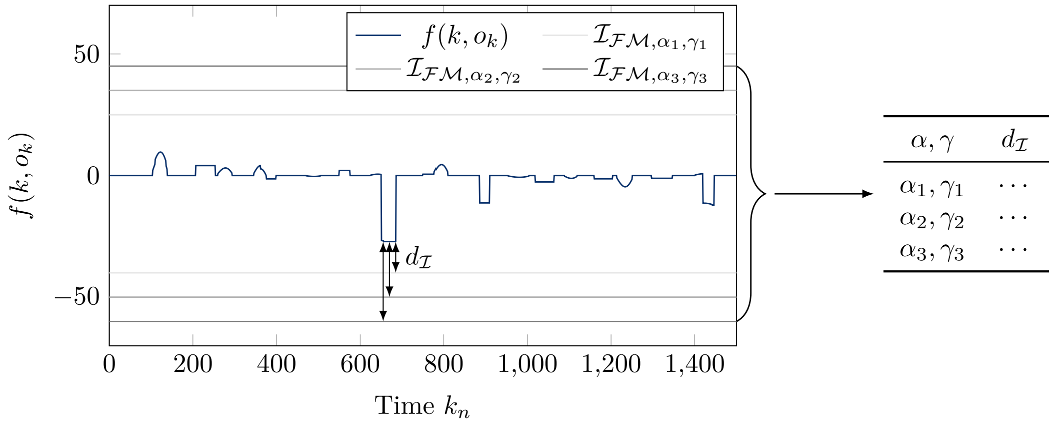


Fig. 3.13.: Calculating confidence values for *GFM* by extracting an interval and comparing its borders to the initial failure amplitudes.

O_n with respect to time, subtracting the starting time step $K_s(O_n)$ and the duration $K_n(O_n)$ results in the time between those consecutive occurrences. Again, the mean and standard deviation is calculated before the individual values are normalized.

The scaling values of the failure types occurrences are processed similar to the duration values, cf. Eqs. (3.60) to (3.62).

$$\hat{\mu}_{scl}(k, \mathbf{o}_k) = \frac{1}{|\hat{\mathcal{O}}|} \sum_{O_n \in \hat{\mathcal{O}}} \mathbf{scl}(O_n) \tag{3.60}$$

$$\hat{\sigma}_{scl}(k, \mathbf{o}_k) = \sqrt{\frac{1}{|\hat{\mathcal{O}}|} \sum_{O_n \in \hat{\mathcal{O}}} (\mathbf{scl}(O_n) - \hat{\mu}_{scl}(k, \mathbf{o}_k))^2} \tag{3.61}$$

$$\hat{\mathcal{O}}'_{scl}(k, \mathbf{o}_k) = \left\{ \frac{\mathbf{scl}(O_n) - \hat{\mu}_{scl}(k, \mathbf{o}_k)}{\hat{\sigma}_{scl}(k, \mathbf{o}_k)} \middle| \forall O_n \in \hat{\mathcal{O}} \right\} \tag{3.62}$$

Finally, generating the mean and standard deviation values for all windows of the sliding window and considering $(k, \mathbf{o}_k)$ at the center of the windows as the function's inputs, the training data for each polynomial is generated. Using the *Sum of Squared Errors (SSE)* as a cost function, finding the optimal weights of the polynomials is achieved by solving a linear equation. Here the Housholder QR decomposition is used.

Applying the procedure for each failure type candidate in $\mathcal{FM}_{identified}$ produces parameterized failure types F_n, which form the final failure model $\mathcal{FM} = \{F_1, \ldots, F_N\}$

3.3.3. Confidence Calculation

The parameterized failure model $\mathcal{FM}$ produced by the last stage resembles the final failure model already. However, until now, the quality of the generated model is not assessed. Moreover, as a confidence measure for the failure model has yet to be defined to fulfill the corresponding criterion, this section introduces the same.

The central goal of the envisioned confidence measure is to provide information to assess whether or not a failure model covers all relevant aspects of a represented failure characteristics. In that way, an application or system can reason about the suitability of a shared failure model with respect to its own safety. Especially with respect to sensory data, but also with respect to continuous data in general, this means that the maximal and minimal failure amplitudes have to be considered. As these can be extracted from a *GFM* using the approach presented in Section 3.2, the goal is to build upon this interval representation.

The conceptual idea is depicted in Fig. 3.13. Given the failure model $\mathcal{FM}$ and the initial failure amplitudes $f(k, \mathbf{o}_k)$, intervals $\mathcal{I}_{\mathcal{FM},\alpha,\gamma}$ are extracted for different values of α and γ, as well as the sets $\mathbb{K}$ and $\mathbb{O}$ which specify the time and value domain the intervals are valid for. While versatile combinations for $\mathbb{K}$ and $\mathbb{O}$ are possible, this work focuses on sets covering all initial failure amplitudes for the sake of simplicity. Where needed, these are specified with more detail. Using Eqs. (3.63) to (3.66), the failure amplitudes used to generate the failure model are compared to the bounds of intervals and the minimal distance is determined. For comparison reasons, this distance is normalized by the range of the interval under consideration. This enables taking the minimum value of all dimensions in case of multi-dimensional failure amplitudes.

$$\mathcal{I}_L = min(\mathcal{I}_{\mathcal{FM},\alpha,\gamma}) \tag{3.63}$$

$$\mathcal{I}_U = max(\mathcal{I}_{\mathcal{FM},\alpha,\gamma}) \tag{3.64}$$

$$range(\mathcal{I}_{\mathcal{FM},\alpha,\gamma}) = \mathcal{I}_U - \mathcal{I}_L \tag{3.65}$$

$$d_{\mathcal{I}}(f(k, \mathbf{o}_k), \alpha, \gamma) = min_{k \in \mathbb{K}, \mathbf{o}_k \in \mathbb{O}} \left(\frac{min(\mathcal{I}_U - f(k, \mathbf{o}_k), f(k, \mathbf{o}_k) - \mathcal{I}_L)}{range(\mathcal{I}_{\mathcal{FM},\alpha,\gamma})} \right) \tag{3.66}$$

While Eqs. (3.63) and (3.64) merely extract the lower, respective upper bound of the interval, Eq. (3.65) uses these to calculate the range covered by the interval. Eq. (3.66) defines the final confidence value depending on the interval under consideration as well as the failure amplitudes the interval is valid for. Specifying the confidence value for different α and γ values as indicated by Fig. 3.13 and the table therein, enables an application to assess the impact of changing these values has on the extracted interval. This enables applications to adjust these values with regard to their required safety performance level.

3.4. Exemplary Application to Artificial Failure Characteristics

The previous sections introduced the concept of a *Generic Failure Model (GFM)* and a processing chain for extracting the same from a given set of failure amplitudes. In this section, these concepts are investigated by an exemplary application to artificial, one-dimensional failure characteristics. This serves four goals.

1. By manually designing a *GFM* to represent the artificial failure characteristics, the meaning of individual functions of a failure type is clarified and the fulfillment of the clarity criterion is underlined (Section 3.4.1).
2. Showing that failure characteristics reported in the literature can be represented underlines furthermore the fulfillment of the coverage criterion.
3. Using the exact knowledge of the manually designed failure model, failure amplitudes are simulated and the ability of the proposed processing chain to generate a failure model is shown (Sections 3.4.2 to 3.4.4). The extracted failure model can further be compared with the original failure model to assess the performance of the individual steps of the processing chain.
4. The modeling performance of the *GFM* is compared to state-of-the-art approaches (Section 3.4.5).

Following these goals, the next subsection introduces the manually designed failure models, before the following subsections discuss the individual stages of the processing chain and its ability to retain the original failure models from simulated failure amplitudes. Finally, the extracted failure models are compared to state-of-the-art approaches for time series and failure modeling in Section 3.4.5.

3.4.1. Generating the Artificial Data Set – Manually Designing *GFMs*

The clarity criterion on *GFM* stems from the need of sharing a failure model between dynamically composed systems where it is of paramount importance that its interpretation is clear to ensure safety. In pursuit of this criterion, the *GFM* is designed using mathematical functions and polynomials. This results in an additional advantage for experts and engineers required to specify a failure model. It can not only be designed using the proposed processing chain but also by manually specifying the parameters of a *GFM*.

This is leveraged in this subsection where failure types, repeatedly reported in the literature but commonly defined only linguistically, are modeled manually. An overview of the considered failure types and their definitions is given in Table 3.1. For the Noise, Outlier, and Spike failure type, the definitions provided by Ni *et al.* [69] are considered. The *Offset* failure type considered here is known by varying names in the literature. It is defined as a *Calibration* failure in [69] or as *Intermittent Offset* or simply *Offset* in [93]. Due to this inconsistency, the latter name is used here.

Additionally to the aforementioned types, the Artificial failure type is listed. This type can be thought of as a special variant of the Spike failure type. It is used only in

Tab. 3.1.: Linguistic definitions of the considered failure types.

Failure Type	Definition
Noise [69]	Sensor values experience unexpectedly high variation or *noise*.
Outlier [69]	Isolated data point or sensor unexpectedly distant from model.
Offset [69], [93]	Sensor reports values that are offset from the ground truth.
Spike [69]	Multiple data points with a much greater than expected rate of change.
Artificial	A variant of the spike failure type considered only in this work to support the evaluation.

this evaluation to further analyze the capabilities and restrictions of (i) the *GFM* and (ii) the processing chain. To focus on this evaluation's goals, only one-dimensional data is considered here despite the *GFM*s ability to model multi-dimensional data, which was shown already in [31] and will be further discussed in Chapter 5.

In the endeavor of designing the failure types, the next subsection focuses on their failure patterns while the following subsection discusses their respective time- and value-correlated random distributions representing their scaling, activation, and deactivation distributions. Finally, the last subsection discusses the simulation of the failure models generated from the individual failure types which provides the artificial failure amplitudes. To these, the processing chain will be applied in the following sections.

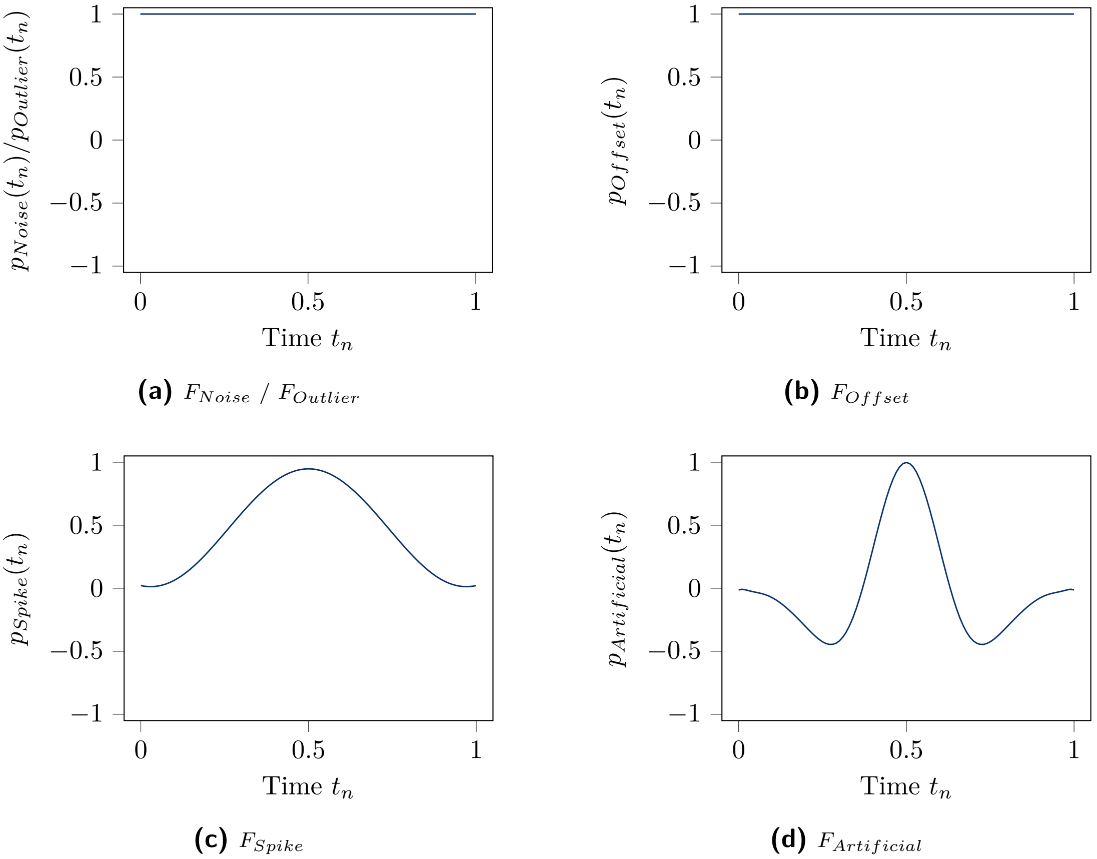
Failure Patterns

Central to the definition of the individual failure types are their failure patterns $p_n(t_n)$. In accordance with the definitions in Table 3.1, the patterns are represented by polynomials, which are sampled over $t_n \in [0, 1]$ and visualized in Fig. 3.14.

The definitions of Noise and Outlier state that, in the former case, sensor values experience high variations or, in the latter case, are unexpectedly distant from model. In both cases, however, sensor values are affected independently, meaning that no temporal pattern can be observed. Correspondingly, a constant "1" is considered here and represented by a polynomial of degree 0.

The failure pattern of the Offset failure type is modeled in the same way. In contrast to Noise and Outlier, however, it is expected that this failure pattern indeed affects multiple consecutive sensor values but with the same failure amplitude. Therefore, a constant is used here as well.

The Spike failure type, is defined with respect to its rate of change. Similar to the Offset failure type, multiple sensor values are affected, but the magnitude is changing. Therefore, the pattern is represented by a polynomial with degree 7 and was fitted to match the *Probability Density Function (PDF)* of a Gaussian distribution to resemble a bell-like shape.



(a) F_{Noise} / $F_{Outlier}$

(b) F_{Offset}

(c) F_{Spike}

(d) $F_{Artificial}$

Fig. 3.14.: Failure patterns of the failure types considered in Table 3.1.

The failure patterns are, until now, restricted to the value domain of $p_n \in [0, 1]$. Contrarily, a failure pattern is allowed to have values between -1 and 1, cf. Section 3.1.2. Therefore, the additional failure type called Artificial is added, cf. Fig. 3.14d. Its pattern resembles the Ricker Wavelet and is modeled by a polynomial of degree 15.

Time- and Value-Correlated Random Distributions

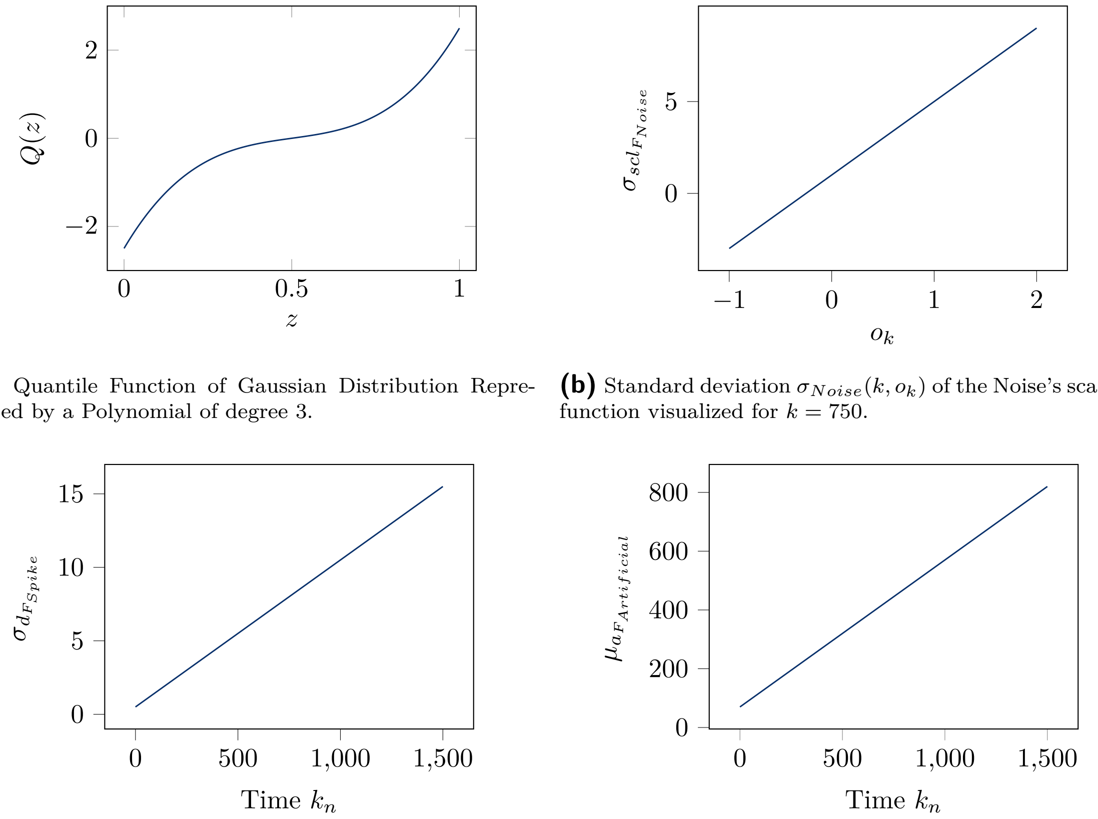
While the failure pattern of each failure type defines its deterministic part, the scaling of the pattern, as well as the failure type's activation and deactivation, are represented by time- and value correlated random distributions, cf. Section 3.1.1. The parameterization of these distributions is discussed in this subsection. An overview of the distribution serving as templates for this paramterization is provided in Table 3.2.

As stated by the table, the underlying distributions are either a Gaussian distribution $\mathbb{N}(0, 1)$ or a uniform distribution $\mathbb{U}(0, 1)$. While the former is represented by a polynomial of degree 3 (cf. Fig. 3.15a), the latter requires only a polynomial of degree 1 to represent the quantile function $Q_{uniform}(z) = 0 + 1 \cdot z$. As described in Section 3.1.1, these distributions are scaled and shifted according to μ and σ functions which potentially introduce time- and value-correlations.

For representing the scaling distribution of the Offset failure type, for instance, the Gaussian distribution is scaled by a standard deviation increasing in relation to o_k. While this means that $\mu_{scl_{Offset}}(k, o_k) = 0$ is represented by a polynomial of degree 0, the standard deviation $\sigma_{scl_{Offset}} = 10 + 1 \cdot o_k$, is a polynomial of degree 1. In contrast, the failure type's activation and deactivation distributions are not time- or

Tab. 3.2.: Template distributions used to generate the time- and value-correlated distributions of the modeled failure types

Failure Type	**Scaling**	**Activation**	**Deactivation**
Offset	$\mathbb{N}(0, 10 + o_k)$	$\mathbb{N}(100, 25)$	$\mathbb{U}(15, 50)$
Spike	$\mathbb{N}(0, 5)$	$\mathbb{N}(100, 25)$	$\mathbb{N}(35, 0.5 + 0.01 \cdot k)$
Artificial	$\mathbb{U}(7.5, 15)$	$\mathbb{N}(70 + 0.5 \cdot k, 10)$	$\mathbb{N}(35, 5)$
Noise	Q_{Noise} $(\mu = 0, \sigma = 1 + 4 \cdot o_k)$	0	1



(a) Quantile Function of Gaussian Distribution Represented by a Polynomial of degree 3.

(b) Standard deviation $\sigma_{Noise}(k, o_k)$ of the Noise's scaling function visualized for $k = 750$.

(c) Standard deviation $\sigma_{Spike}(k, o_k)$ of deactivation function of spike failure type visualized for $o_k = 0.5$.

(d) Mean $\mu_{Artificial}(k, o_k)$ of activation function of artificial failure type visualized for $o_k = 0.5$.

Fig. 3.15.: Sampled polynomials representing time- and value-correlations in the modeled failure types.

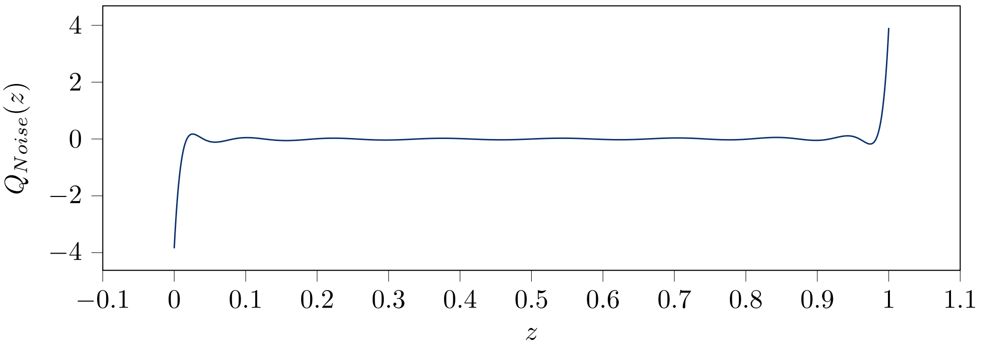


Fig. 3.16.: Quantile function of Noise failure type incorporating Outliers.

value-correlated. Therefore, the respective μ and σ functions are constants representing the mean and standard deviations stated in Table 3.2.

The Spike failure type, however, is modeled to have occurrences with increasing length over time. For that, the deactivation distribution representing its *TtR* is Gaussian with an increasing standard deviation. The sampled polynomial is visualized in Fig. 3.15c.

Regarding the Artificial failure type, only its activation distribution is time-correlated. This time, the mean value of its time between failure increases over time which means that its number of occurrences decreases. Again a linear correlation is assumed, as can be seen in Fig. 3.15d.

In contrast to the remaining failure types, Noise is defined to be always active but affect only single time steps. For that, the quantile function of the activation and deactivation distributions are set to zero ($Q_{a/d}(z) = 0$). The mean functions are used to model the constants, that is, the duration of a single occurrence is set to one time step ($d_{Noise}(k, o_k) = 1$) while the time between two occurrences is set to zero ($a_{Noise}(k, o_k) = 0$).

Moreover, the Noise failure type is additionally used to represent the Outlier failure type as well. To that end, the ability of a quantile function to represent arbitrary distributions is leveraged. It enables to impose a Gaussian distribution (for representing the Noise scaling) with a custom distribution adding high values that are only rarely occurring. The resulting quantile function represented by a polynomial of degree 18 is visualized in Fig. 3.16. By mapping approximately all values for $0.05 \leq z \leq 0.95$ to a value close to zero, small Noise amplitudes are represented with a high probability. Contrarily, the remaining values of z are mapped to high amplitudes that are occurring with a low probability. These encode the Outlier amplitudes. Simulating failure amplitudes from the failure type shows that it simultaneously represents Noise and Outlier, cf. Fig. 3.17a.

Generating the Artificial Data Set

In the endeavor of evaluating the proposed processing chain, time series of failure amplitudes, on which it can operate, are required. For that, this subsection starts by defining distinct failure models as sets of the manually defined failure types. After defining a reference signal required to evaluate modeled value-correlations, the simulation procedure is described. Examples of the generated failure amplitudes are discussed to conclude this subsection.

Tab. 3.3.: Mapping the sets of considered failure types to labels of failure models.

Failure Model	Failure Types	Failure Model	Failure Types
$\mathcal{FM}_{\{N\}}$	$\{F_{Noise}\}$	$\mathcal{FM}_{\{O,A\}}$	$\{F_{Offset}, F_{Artificial}\}$
$\mathcal{FM}_{\{O\}}$	$\{F_{Offset}\}$	$\mathcal{FM}_{\{S,A\}}$	$\{F_{Spike}, F_{Artificial}\}$
$\mathcal{FM}_{\{S\}}$	$\{F_{Spike}\}$	$\mathcal{FM}_{\{N,S,O\}}$	$\{F_{Noise}, F_{Spike}, F_{Offset}\}$
$\mathcal{FM}_{\{A\}}$	$\{F_{Artificial}\}$	$\mathcal{FM}_{\{N,O,A\}}$	$\{F_{Noise}, F_{Offset}, F_{Artificial}\}$
$\mathcal{FM}_{\{N,O\}}$	$\{F_{Noise}, F_{Offset}\}$	$\mathcal{FM}_{\{N,S,A\}}$	$\{F_{Noise}, F_{Spike}, F_{Artificial}\}$
$\mathcal{FM}_{\{N,S\}}$	$\{F_{Noise}, F_{Spike}\}$	$\mathcal{FM}_{\{S,O,A\}}$	$\{F_{Spike}, F_{Offset}, F_{Artificial}\}$
$\mathcal{FM}_{\{N,A\}}$	$\{F_{Noise}, F_{Artificial}\}$	$\mathcal{FM}_{\{N,S,O,A\}}$	$\{F_{Noise},F_{Spike},$ $F_{Offset},F_{Artificial}\}$
$\mathcal{FM}_{\{S,O\}}$	$\{F_{Spike}, F_{Offset}\}$		

The previously defined failure types described distinct failure amplitudes. By combining them, varying failure characteristics can be represented. This supports the analysis of the processing chain, for which failure amplitudes are to be generated. Therefore, all combinations of the presented failure types are considered. Starting from failure models having only one failure type, all failure models having two, three, and four failure types are considered. Consequently, the evaluation is based on 15 different failure models which are listed in Table 3.3.

To simulate the failure models, a reference signal o_k is needed to evaluate the defined value-correlations. For the sake of simplicity, a sinus curve is used here cf. Eq. (3.67).

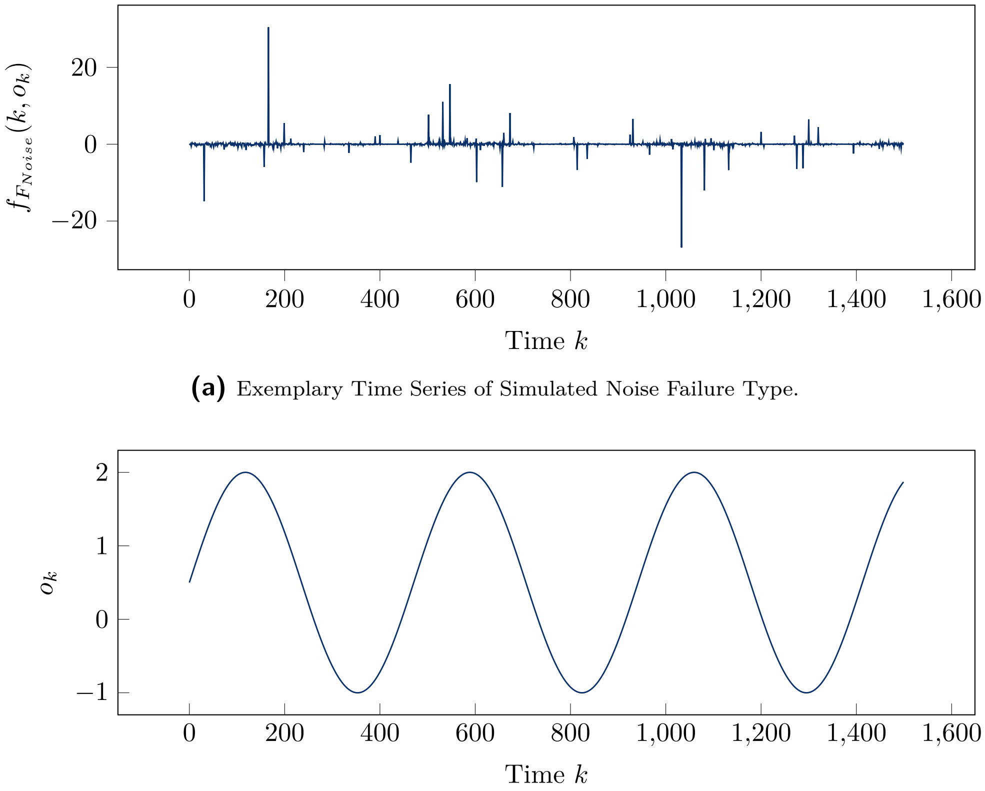
$$o_k = 3 \cdot \left(\frac{sin(\frac{k}{75})}{2} + 0.5 \right) - 1 \tag{3.67}$$

The sinus is sampled with a frequency of 0.013Hz and is scaled to be in the interval of $o_k \in [-2, 1]$. This enables evaluating positive and negative value-correlations.

The simulation process uses both, time k and the defined reference signal o_k, to sample the failure model and generate failure amplitudes $f(k, o_k)$. More specifically, the compositional characteristic of the *GFM* is leveraged (cf. Eq. (3.1)) which allows sampling each failure type independently to generate $f_{F_n}(k, o_k)$ and superimpose theses to obtain the overall failure amplitudes $f(k, o_k)$.

Therefore, the simulation of each failure type starts with evaluating its activation distribution $a_n(k, o_k)$ using k, o_k, and a random value of $z \in [0, 1]$. The result states the time between failure, that is, the time step $K_s = k + a_n(k, o_k)$ at which the failure type becomes active. When the failure type becomes active, the scaling distribution $scl_n(K_s, o_k)$ is sampled equally to generate the scaling with which the failure pattern occurs. Additionally, the deactivation distribution is sampled to determine the duration $K_n = d_n(K_s, o_k)$ for which the failure will be active. Once the failure type reaches the time step $k = K_s + K_n$ at which it becomes inactive, the procedure is repeated.

In essence, this cycle produces occurrences $\mathcal{O}_n$ of the failure type in question. From



(a) Exemplary Time Series of Simulated Noise Failure Type.

(b) Reference of Exemplary Time Series.

Fig. 3.17.: Failure amplitudes and reference signal for Noise failure type.

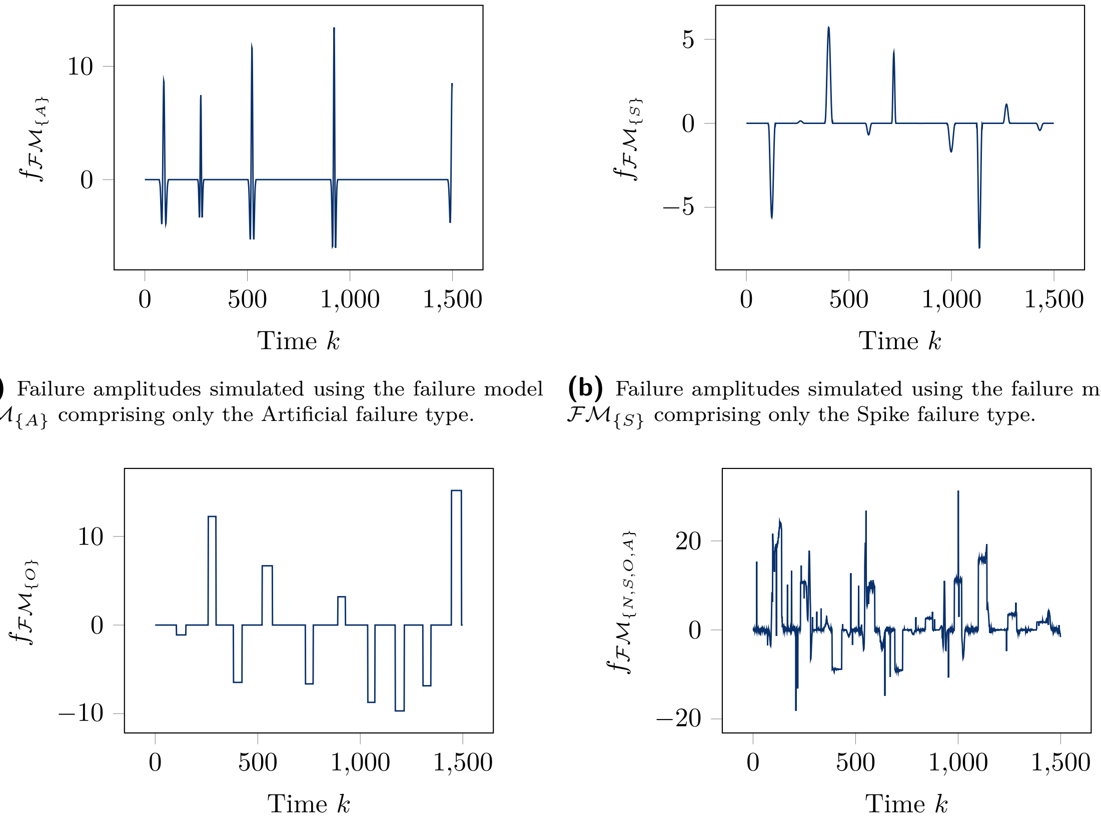
that set, Eq. (3.51) is used to construct the simulated failure amplitudes $f_{F_n}(k, o_k)$, from which the overall failure amplitudes $f(k, o_k)$ are generated by summing over all failure types, cf. Eq. (3.1).

For each of the failure models listed in Table 3.3, this simulation is repeated 6 times, each comprising 1500 time steps. In that way, time- and value-correlated effects are sampled multiple times which shall support the identification and parameterization process executed by the processing chain.

Fig. 3.18 shows exemplary time series of the simulated failure amplitudes. Starting with $\mathcal{FM}_{\{A\}}$ comprising only the Artificial failure type (cf. Fig. 3.18a), the reader can see that not only the complex shape is injected and scaled as modeled but also that the *TBF*, according to the modeled time-correlation, increases. While the time between two consecutive occurrences of the failure time is close to 100 in the beginning, it is increasing to approximately 700 in the end of the shown time series.

Similarly, the value-correlated scaling of Noise failures becomes apparent by comparing the reference signal in Fig. 3.17b with the corresponding failure amplitudes simulated using $\mathcal{FM}_{\{N\}}$ shown in Fig. 3.17a. The amplitudes increase with the minimal and maximal values of the sinus curve.

In contrast to these examples, the correlations modeled by the Spike and Offset failure are not as obvious as they affect the standard deviation of the respective distributions. In Fig. 3.18b a series of failure amplitudes of $\mathcal{FM}_{\{S\}}$ is shown. While the temporal pattern of the Spike failure is clearly visible, the increase in the standard deviation of the failure type's activation is subtle. Similarly, the effect of the increased standard



(a) Failure amplitudes simulated using the failure model $\mathcal{FM}_{\{A\}}$ comprising only the Artificial failure type.

(b) Failure amplitudes simulated using the failure model $\mathcal{FM}_{\{S\}}$ comprising only the Spike failure type.

(c) Failure amplitudes simulated using the failure model $\mathcal{FM}_{\{O\}}$ comprising only the Offset failure type.

(d) Failure amplitudes simulated using the failure model $\mathcal{FM}_{\{N,S,O,A\}}$.

Fig. 3.18.: Exemplary time series of failure amplitudes of selected failure models.

deviation in the scaling distribution of the Offset failure type is minor but observable, cf. Fig. 3.18c. Finally, Fig. 3.18d shows the failure amplitudes as simulated by the failure model $\mathcal{FM}_{\{N,S,O,A\}}$, which encompasses all failure types.

These figures underline that the *GFM* can be parameterized manually to represent failure characteristics reported in the literature. Furthermore, this shows that an intuitive interpretation of the individual components is possible and supports the fulfillment of the clarity criterion.

3.4.2. Identifying Failure Types

The failure amplitudes generated by simulating the manually designed failure models serve, together with the reference signal, as the input to the processing chain. As depicted in Fig. 3.8, its first stage aims at identifying failure type candidates comprising a failure pattern p_n and its occurrences $\hat{\mathcal{O}}_n$. In the endeavor of evaluating this process, the next subsection discusses the parameterization of this stage before the following subsection examines its output.

For that three aspects are discussed. Firstly, the optimization process for identifying and adjusting the failure patterns $p_n(t_n)$ is analyzed using the example of the Spike failure type. Secondly, the failure patterns generated by the processing chain are compared to the original patterns. Thirdly, the injected occurrences $\mathcal{O}_n$ are compared to the occurrences $\hat{\mathcal{O}}_n$ generated during the simulation of the failure model.

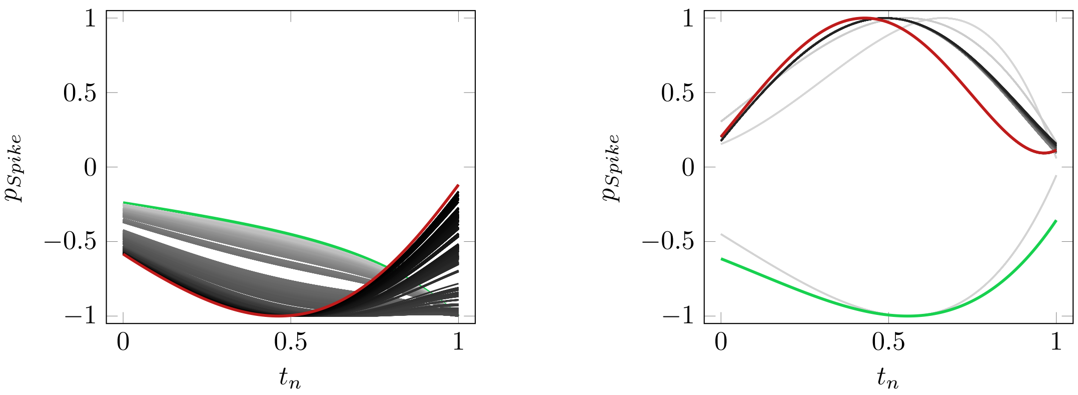
Tab. 3.4.: Parameterization of the identification stage of the proposed processing chain.

Parameter	Value	Description
$N_{\mathcal{O}}$	1	Number of occurrences a failure type must have at least to be accepted.
K_n	$[14, 56]$	Durations, that is, scale values of CWT (cf. Eq. (3.47)) with which the occurrences of a failure pattern are searched for.
D	5	Degree of the polynomial representing the failure pattern of an identified failure type.
$I_{Identification}$	10^6	Maximal number of iterations for optimizing a failure pattern.
η	-5^{-5}	Learning rate used by AMSGrad during gradient descent for optimizing a failure pattern.
N_{p_n}	32	Number of failure patterns optimized in parallel.

Note that, during this evaluation, the set $\hat{\mathcal{O}}_n$ denotes the occurrences of the n-the failure type identified by the processing chain while the set $\mathcal{O}_n$ denotes the occurrences of the same failure type as produced during simulation, that is, from which the original sets of failure amplitudes were generated. Similarly, $\hat{\mathcal{FM}}$ will denote failure models generated by the processing chain while $\mathcal{FM}$ will indicate the original, manually modeled failure models.

Parameterization

To underline the general applicability of the processing chain for extracting failure models, the parameters chosen for the identification step of failure types are the same for all failure models. An overview of the most central parameters is given in Table 3.4
The first stage identifies failure types by searching for occurrences of length $K_n \in [14, 56]$ of failure patterns represented by polynomials of degree 5. For the sake of simplicity, these parameters are set in accordance with the manually designed failure types. The range of length values is chosen with respect to the deactivation distributions of the modeled failure types. Similarly, the polynomial degree for representing failure patterns is chosen in accordance with the Spike failure type ($D = 5$). While the polynomial is, therefore, able to represent Noise, Offset, and Spike failures, the pattern of the Artificial failure type has to be approximated. Therefore, the degree is a trade-off between the complexity of the anticipated failure patterns.
For a failure type to be accepted, its pattern has to occur at least once. To that end, the pattern is optimized within up to $I_{Identification} = 10^6$ iterations, where a learning rate of -5^{-5} is applied. As the approach of SGD is known to potentially result in local optima [90], a total of $N_{p_n} = 32$ failure patterns are optimized in parallel while only the best according to Eq. (3.49) is considered as a candidate F_C.



(a) Optimizing $p_{Spike}(t_n)$ using a single polynomial.

(b) Optimizing $p_{Spike}(t_n)$ using multiple polynomials in parallel.

Fig. 3.19.: Visualizing the optimization process of single and parallel polynomials. The green curve shows the randomly initialized polynomial while the red curve depicts the optimized pattern. Polynomials obtained in between are colored gray where darker gray refers to polynomials later in the process.

Results

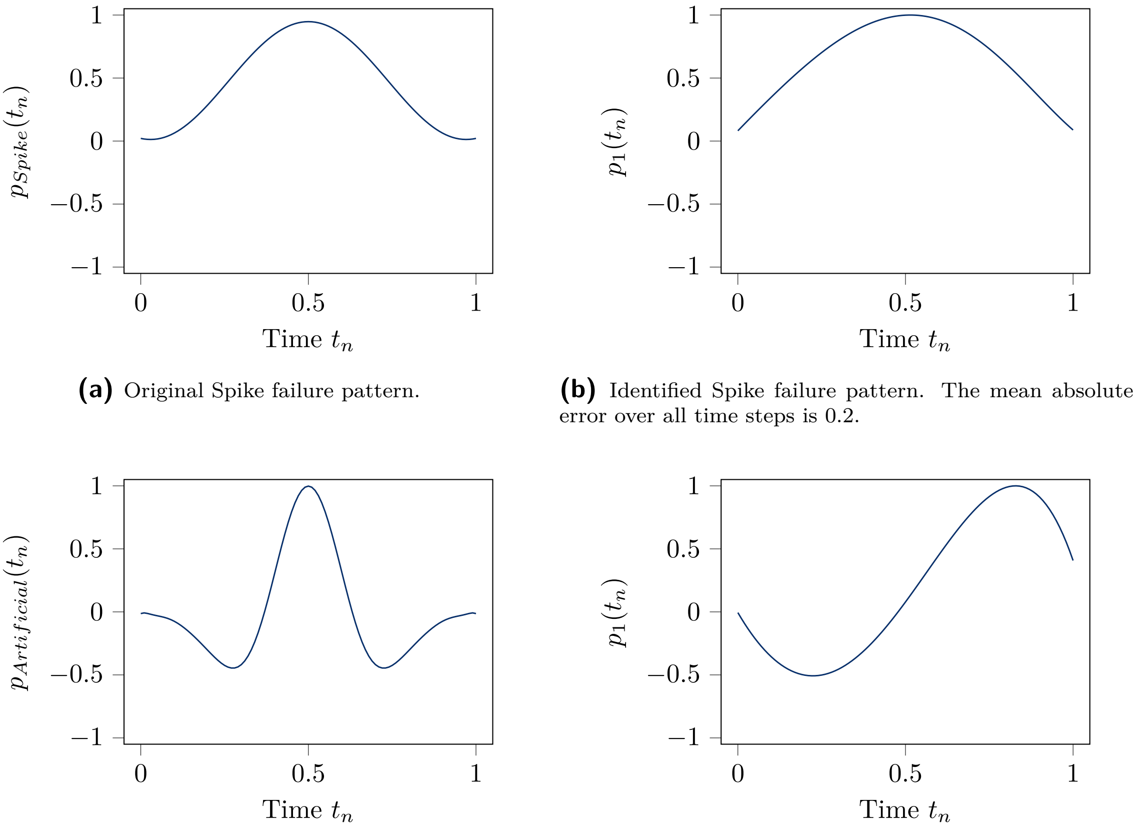
Using the parameters of Table 3.4, the first stage of the processing chain employs two nested loops to find failure patterns and identify a failure type candidate, cf. Section 3.3.1.

The inner loop employs *SGD* to optimize randomly initialized failure patterns. The process is visualized exemplarily in Fig. 3.19. It compares the optimization process of a single polynomial (Fig. 3.19a) with the same approach considering multiple polynomials in parallel (Fig. 3.19b). In both figures, the initial polynomial is depicted by a green curve while the best polynomial (according to Eq. (3.49)) obtained during optimization is colored red. The gray-colored lines in between correspond to new, best polynomials generated during the process. Light-gray-colored curves refer to polynomials early in the process while dark-gray-colored curves show polynomials closer to the end.

Considering Fig. 3.19a first, one can see that the initial polynomial is nothing but a decreasing linear that does not resemble a Spike pattern. During gradient descent, the parameters of the polynomials are adjusted such that the end of the curve is bent upwards. Thereby, the adjustment towards an inverted bell-like shape is observable. Note that the sign does not affect how well the pattern matches the failure amplitudes as it is compensated for by the scaling value *scl*. Despite the continuous optimization, only a local optimum is reached.

This becomes clear when comparing Fig. 3.19a with Fig. 3.19b. For the latter, multiple polynomials are optimized in parallel while only the best polynomial regarding Eq. (3.49) of each iteration is visualized. Moreover, in each iteration half of the polynomials achieving the worst scores are reinitialized with random parameters, which additionally introduces randomness and thereby counteracts the problem of local optima.

Although the best initial polynomial again resembles almost a linear curve (green), fewer iterations are needed to obtain better failure patterns. During these, the curve switches between positive and negative values. Moreover, clear, distinct curves are



(a) Original Spike failure pattern.

(b) Identified Spike failure pattern. The mean absolute error over all time steps is 0.2.

(c) Original Artificial Failure Pattern

(d) Identified Artificial failure pattern. The mean absolute error over all time steps is 0.6.

Fig. 3.20.: Comparing exemplary manually designed and automatically identified failure patterns.

observable which indicates that not a single polynomial was found to be the best during optimization, but multiple polynomials replaced each other as the best.

The parallel version requires only 436 candidates to achieve a Spike pattern compared to 4430 polynomials produced by the none parallel procedure, which further underlines that the parallel version is more efficient in finding suitable failure patterns.

As one can see in Fig. 3.20a and Fig. 3.20b, the procedure generates a close match between the original and the identified spike failure pattern for failure amplitudes simulated using $\mathcal{FM}_{\{S\}}$. The *Mean Absolute Error (MAE)* is 0.2, cf. Eq. (3.68).

$$MAE(p_{original}, p_{identified}) = \frac{1}{K_n} \sum_{k_n=0}^{K_n} \left| p_{original}(\frac{k_n}{K_n}) - p_{identified}(\frac{k_n}{K_n}) \right| \tag{3.68}$$

To calculate the value, both patterns are sampled for $K_n = 100$ time steps.

In contrast to the Spike failure pattern, the Artificial pattern can not be identified with such accuracy, cf. Figs. 3.20c and 3.20d. Using the failure amplitudes generated from $\mathcal{FM}_{\{A\}}$, a failure pattern with an *MAE* of 0.6 is generated. The reason is twofold. Firstly, the chosen degree of 5 is not sufficient to represent the complex failure pattern, which is originally represented by a polynomial of degree 15. Secondly, the increased complexity causes the optimization process to be more susceptible to terminating in a local optimum.

As one can see in Fig. 3.20d, approximately the first half of the original failure pattern is identified. From that point on, adjusting the pattern to represent the Artificial failure pattern in its entirety would require decreasing the overall score firstly.

In contrast to the Spike and Artificial failure pattern, the patterns of Noise and Offset could be identified with errors of 0 and 0.01 respectively. Due to these low values and as the patterns are constant zeros, no figure is shown here.

Similar results are observable when comparing the failure patterns identified for $\mathcal{FM}_{\{N,S,O,A\}}$. The failure amplitudes of this failure model comprise all failure types. Due to the imposition of multiple failure types, the individual patterns are occluded, which hinders their identification. Nevertheless, close approximations can be found, as shown in Fig. 3.21.

To compare the pattern, the original and identified patterns were matched based on their MAE values, cf. Eq. (3.68). Using these, the Offset pattern was matched with the first failure type F_1, the Spike failure type was matched with the second failure type F_2, the Artificial failure type was matched with F_3 , and Noise was matched with F_4. Note that the order $F_1, \ldots, F_4$ is the order in which the inner loop of the identification stage proposed the failure type candidates.

Compared to the previously identified patterns in Fig. 3.20d, the pattern associated with the Artificial failure type resembles the original more closely. This is due to the random initialization and indicates that more iterations or an increased number of parallel polynomials might result in better approximations.

Next to the actual failure pattern, the first stage of the processing chain additionally identifies its set of occurrences $\hat{\mathcal{O}}_n$. Similar to the failure patterns, the occurrences can be compared to the occurrences $\mathcal{O}_n$ generated by the initial simulation as well. For that, three metrics are applied.

Firstly, the ratio of identified occurrences to original occurrences is calculated for obtaining a general overview, cf. Eq. (3.69).

$$r_{\mathcal{O}} = \frac{\left|\hat{\mathcal{O}}_n\right|}{\left|\mathcal{O}_n\right|} \tag{3.69}$$

For the n-th failure type, the number of occurrences identified by the processing chain is divided by the number of occurrences used during simulation to generate the failure amplitudes.

Secondly, the temporal aspect of a failure type's occurrence is assessed by calculating the mean, temporal overlap of identified and original occurrences, cf. Eqs. (3.70) to (3.73).

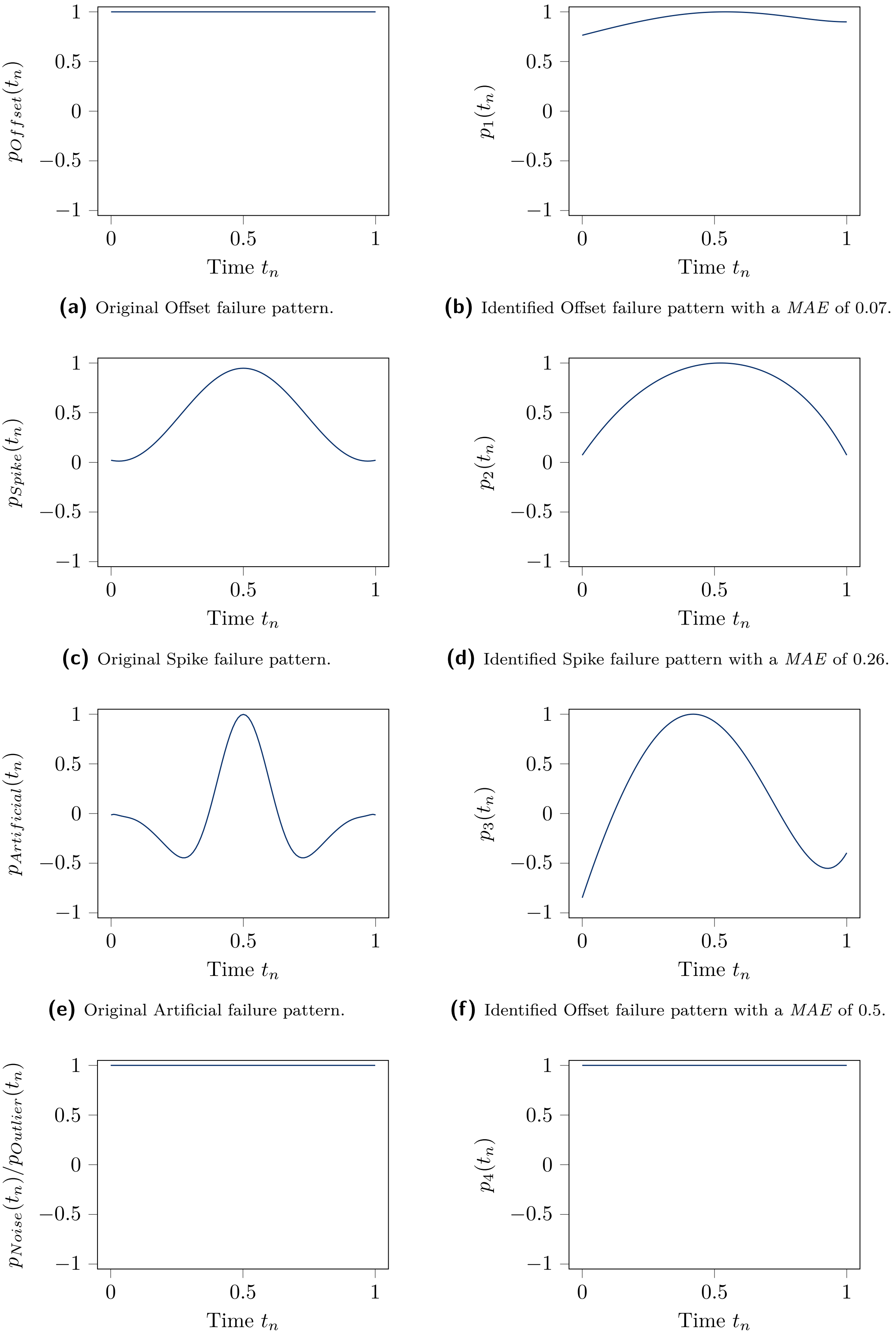
$$\mathcal{I}_{\cap}(\hat{O}, O) = \left[K_s(\hat{O}), K_s(\hat{O}) + K_n(\hat{O})\right] \cap \left[K_s(O), K_s(O) + K_n(O)\right] \tag{3.70}$$

$$d_{\cap}(\hat{O}, O) = \frac{\max(\mathcal{I}_{\cap}(\hat{O}, O)) - \min(\mathcal{I}_{\cap}(\hat{O}, O))}{K_n(O)} \tag{3.71}$$

$$\hat{\mathcal{O}} \cup \mathcal{O} = \left\{(\hat{O}, O) \,\middle|\, d_{\cap}(\hat{O}, O) > 0, \forall \hat{O} \in \hat{\mathcal{O}}, \forall O \in \mathcal{O}\right\} \tag{3.72}$$

$$\mu_{Overlap} = \frac{1}{\left|\hat{\mathcal{O}} \cup \mathcal{O}\right|} \sum_{(\hat{O},O) \in \hat{\mathcal{O}} \cup \mathcal{O}} d_{\cup}(\hat{O}, O) \tag{3.73}$$

For this, Eq. (3.70) is the interval obtained when intersecting the start and end time steps of an original occurrence O with an identified occurrence $\hat{O}$. Dividing the range



(a) Original Offset failure pattern.

(b) Identified Offset failure pattern with a *MAE* of 0.07.

(c) Original Spike failure pattern.

(d) Identified Spike failure pattern with a *MAE* of 0.26.

(e) Original Artificial failure pattern.

(f) Identified Offset failure pattern with a *MAE* of 0.5.

(g) Original Noise failure pattern.

(h) Identified Noise failure pattern with a *MAE* of 0.0.

Fig. 3.21.: Comparison of original and identified failure patterns for failure model $\mathcal{FM}_{\{N,S,O,A\}}$.

of this interval by the duration of the original occurrence (cf. Eq. (3.71)) determines the ratio to which both overlap. Eq. (3.72) uses this to determine the combinations of original and identified occurrences that actually overlap. Finally, $\mu_{Overlap}$ is calculated as the mean of the overlap values of this set and thereby states the average overlap achieved by the identified occurrences with the original occurrences. For the distance $d_{\cap}(\hat{O}, O)$, as well as for the mean $\mu_{Overlap}$, values close to one indicate that the original and identified occurrences overlap closely while values close to zero indicate the opposite.

Thirdly, the identified scaling value scl of an occurrence is assessed.

$$\mu_{scl_e} = \frac{1}{\left|\hat{\mathcal{O}} \cup \mathcal{O}\right|} \sum_{(\hat{O},O) \in \hat{\mathcal{O}} \cup \mathcal{O}} \frac{\left|scl(\hat{O}) - scl(O)\right|}{|scl(O)|} \tag{3.74}$$

Eq. (3.74) uses the set of combined occurrences according to Eq. (3.72). It firstly calculates the normalized scaling error for each occurrence and secondly determines the mean over all scaling errors. As such, it states the relative difference between the correct and identified scaling value[3].

In summary, while the $r_{\mathcal{O}}$ allows to generally assess whether or not the occurrences of the individual failure types are identified, the mean of overlap values $\mu_{Overlap}$ focuses on whether or not the start and end point (in time) are identified correctly. Lastly, the μ_{scl_e} enables to discuss the identification of correct scaling values. The results are shown in Table 3.5[4].

Starting with the failure models $\mathcal{FM}_{\{N\}}$, $\mathcal{FM}_{\{S\}}$, $\mathcal{FM}_{\{O\}}$, and $\mathcal{FM}_{\{A\}}$, one can see that almost all occurrences can be identified. The scaling error is low across all four models, but the mean of overlap values decreases. While the length of Noise and Offset failures can be identified, the value decreases for Spike and Artificial failures featuring more complex shaped failure patterns.

The consideration of an additional failure type, e.g. Noise, aggravates the identification process, as can be seen by $\mathcal{FM}_{\{N,S\}}$. The value of $r_{\mathcal{O}}$ drops drastically for the Spike failure type while the overlap value decreases as well. Furthermore, the scaling error increases for the Noise failure type. This is caused by the updating process of the failure amplitudes $f(k, o_k)$ after the acceptance of a failure type, cf. Fig. 3.11. If the length of a failure type's occurrence is not identified correctly or the shape does not sufficiently match the failure amplitudes, artifacts are introduced when removing the identified occurrences. As Noise is generally identified last, all artifacts introduced by previously identified failure types are captured by its occurrences. As a consequence, the scaling values of these occurrences do not match the original scaling values.

Therefore, the same increase in the scaling error is observable for $\mathcal{FM}_{\{N,A\}}$. Apart from that, the identification of the Artificial failure type is only disturbed by the Noise failure regarding the length of the occurrences, not regarding the number of occurrences. This implies that the identification of the Spike failure type is disturbed in general, as is underlined by $\mathcal{FM}_{\{S,O\}}$. The reason for that is the generally reduced magnitude with which a Spike failure occurs in the considered failure amplitudes, cf. Fig. 3.18b.

[3] The sign of the scaling value is considered in correspondence to the failure pattern. That means that if the failure pattern is inverted compared to its original pattern, the scaling value is inverted as well and taken into account during the calculation of Eq. (3.74)

[4] The results of all failure models are given in Table B.1 in the appendix.

Tab. 3.5.: Comparing the occurrences of identified failure types. The original failure model is stated in the most left column while the values of the applied metrics are summarizing the results obtained for the failure type identified by the processing chain. For failure models comprising multiple failure types, the individual evaluation values are stated for each failure type individually.

$\mathcal{FM}$	$r_{\mathcal{O}}$	$\mu_{Overlap}$	μ_{scl_e}
$\mathcal{FM}_{\{N\}}$	[1.00]	[1.00]	[0.00]
$\mathcal{FM}_{\{O\}}$	[0.98]	[0.99]	[0.01]
$\mathcal{FM}_{\{S\}}$	[0.97]	[0.80]	[0.08]
$\mathcal{FM}_{\{A\}}$	[1.00]	[0.49]	[0.04]
$\mathcal{FM}_{\{N,S\}}$	[1.00, 0.47]	[1.00, 0.71]	[8.40, 0.12]
$\mathcal{FM}_{\{N,A\}}$	[1.00, 1.00]	[1.00, 0.48]	[47.44, 0.02]
$\mathcal{FM}_{\{S,O\}}$	[0.63, 0.96]	[0.50, 0.95]	[2.88, 0.14]
$\mathcal{FM}_{\{S,O,A\}}$	[0.93, 0.86, 1.00]	[0.72, 0.35, 0.41]	[1.15, 1.05, 1.18]
$\mathcal{FM}_{\{N,S,O,A\}}$	[1.00, 0.15, 0.84, 0.76]	[1.00, 0.48, 0.85, 0.46]	[38.08, 0.50, 1.98, 0.39]

It is only for $\mathcal{FM}_{\{S,O,A\}}$ that the number of occurrences identified for the Spike failure type is at a higher level, while almost none of its occurrences are identified when considering $\mathcal{FM}_{\{N,S,O,A\}}$. Nevertheless, $\mathcal{FM}_{\{S,O,A\}}$ shows that even when imposed by each other, it is possible to identify the individual failure types to some degree. $\mathcal{FM}_{\{N,S,O,A\}}$ on the other hand underlines that Noise is challenging the identification process.

The evaluation of the first stage of the processing chain shows that by using the CWT-based occurrence identification and the gradient descent based learning of failure patterns, appropriate failure types can be identified.

3.4.3. Parameterizing Failure Types

The failure types identified in the first stage of the processing chain have to be represented in terms of parameters of corresponding polynomial functions. This parameterization is calculated in the second stage of the processing chain, which is the subject of this subsection. In the same manner as before, the parameterization of this stage is introduced before the results are discussed. For that, the time- and value-correlations manually designed for the original failure types are compared to what is represented by the automatically generated failure model.

Parameterization

Next to the failure pattern, each failure type identified in the processing chain's first step is accompanied by a set of occurrences $\hat{\mathcal{O}}$. Using the reference signal o_k, a sliding window approach is applied to these occurrences to extract the mean, standard devia-

Tab. 3.6.: Configuration for the parameterization stage of the proposed processing chain. Note that the parameters for generating the Noise failure type's differ to the parameters for generating the remaining failure types.

Parameter	Value	Description
$K_W/K_{W_{Noise}}$	750/50	Temporal width of the sliding window approach.
$O_W/O_{W_{Noise}}$	1.5/0.0125	Width of the sliding window approach in the value domain.
$K_S/K_{S_{Noise}}$	300/50	Temporal step size of the sliding window approach.
$O_S/O_{S_{Noise}}$	0.3/0.0125	Step size of the sliding window approach in the value domain.
$D_{Q_{scl}}$	3	Degree of polynomial used to represent the quantile function of a failure type's scaling distribution.
D	1	Degree of polynomial to represent other functions of a failure types scaling, activation, and deactivation distribution.

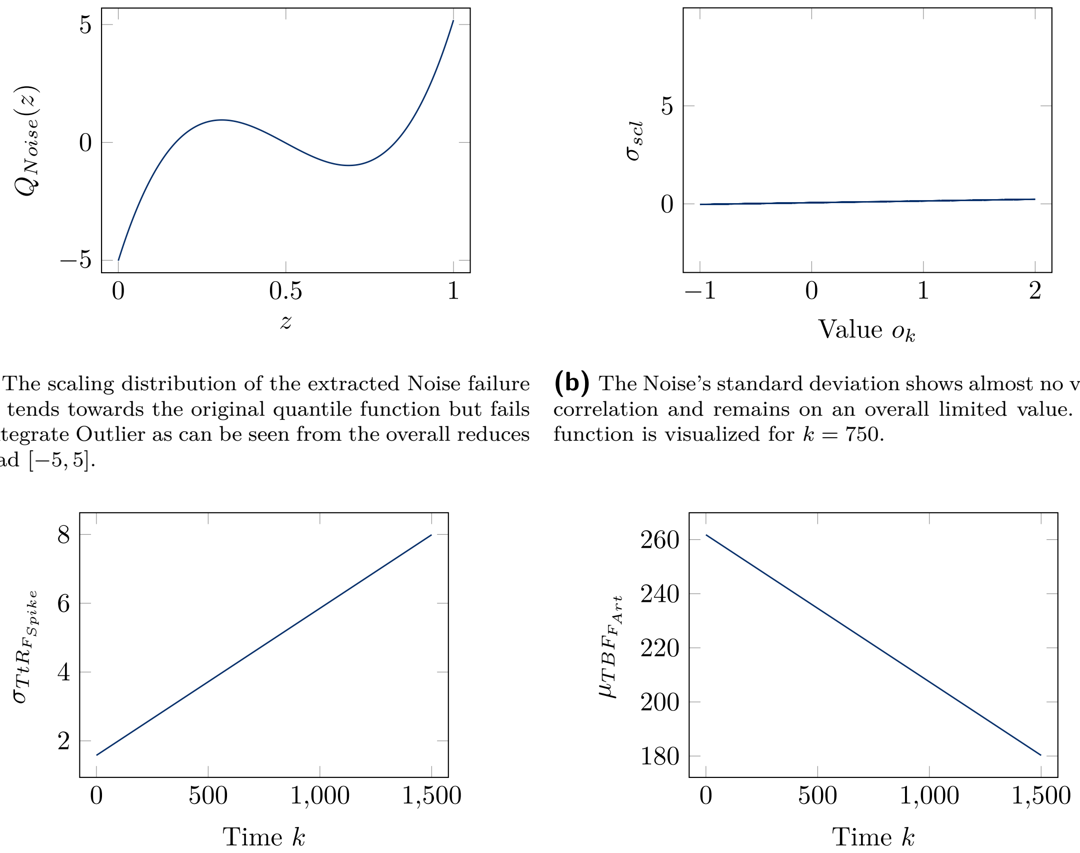
tion, and quantile function of the remaining time- and value-correlated random distributions for the failure type in question. Considering these as training data, polynomials are fitted to represent the corresponding functions. Table 3.6 lists the parameters used in this example.

The K_W and K_S parameters state the temporal width of the window and the step size used to *slide* the window over time. Correspondingly, O_W and O_S define the same attributes for the window's value domain.

One notices the difference between the Noise failure type and the remaining failure types. This is due to the difference in the number of occurrences. The Noise failure type is nothing but the last failure type identified during the first stage of the processing chain. It assumes each non-zero failure amplitude in $\hat{f}(k, o_k)$ as a single occurrence, which causes, in this specific example, $|\hat{\mathcal{O}}_{Noise}|= 9000$ occurrences. In contrast, failure type's identified through gradient descent reach a maximum of $|\hat{\mathcal{O}}|= 218$. Therefore, while small windows suffice to generate significant mean and standard deviation values for the Noise failure type, the window size must be increased for other failure types.

Similar to the window size, the stepping parameters K_S and O_S are adjusted. For the Noise failure type, the values are set to match the width of the window such that no overlapping windows are used during the process. Due to the increased number of occurrences, this ensures that each occurrence is used only once. Contrarily, the parameterization for other failure types is such that overlaps are explicitly generated. This time, the overlap ensures that the overall number of windows is increased, which means that the number of data points, to which the polynomials are fitted, is increased as well. This shall support the training process.

Finally, polynomials of degree $D = 1$ are fitted to represent the functions as it is known from the manually designed failure models that only linear correlations are to



(a) The scaling distribution of the extracted Noise failure type tends towards the original quantile function but fails to integrate Outlier as can be seen from the overall reduces spread $[-5, 5]$.

(b) The Noise's standard deviation shows almost no value-correlation and remains on an overall limited value. The function is visualized for $k = 750$.

(c) The extracted standard deviation of the Spike's deactivation distribution shows a minor time-correlation. The function is visualized for $o_k = 0.5$.

(d) The mean of the Artificial's *TBF* distribution shows a minor decrease over time. The function is visualized for $o_k = 0.5$.

Fig. 3.22.: Mean and standard deviation function's of the separately extracted failure types.

be expected. The quantile function of the scaling distributions, however, are fitted by polynomials of degree $D_{Q_{scl}} = 3$ as the Offset and Spike failure type's use a Gaussian distribution and the Noise failure type even uses a custom quantile function.

With these parameters, the occurrences of the identified failure types are analyzed using the sliding window approach and parameterized by fitting corresponding polynomials. Thus, the failure models' final representations are generated.

Results

For the sake of brevity and argumentation, only the failure models $\hat{\mathcal{FM}}_{\{N\}}$, $\hat{\mathcal{FM}}_{\{S\}}$, $\hat{\mathcal{FM}}_{\{O\}}$, and $\hat{\mathcal{FM}}_{\{A\}}$ are considered here as they feature a single failure type for which the identified and originally modeled failure pattern can be compared. In this endeavor, Fig. 3.22 shows the correlations as generated by the processing chain. Comparing these with the original correlations shown in Fig. 3.15 clarifies that this is a shortcoming of the processing chain. Correlations are identified in general but do not match the original correlations. While originally the mean of the activation $\mu_a(k, o_k)$ of the Artificial failure type is increasing over time (cf. Fig. 3.15d), the processing chain extracts a decreasing trend (cf. Fig. 3.22d). Similarly, the scaling $\sigma_{scl}(k, o_k)$ of the original Noise failure type is increasing proportionally to the value o_k while standard deviation as extracted by the processing chain remains at a constant level, cf. Fig. 3.22b. The trend provided for the standard deviation $\sigma_d(k, o_k)$ of the Spike's deactivation distribution (cf. Fig. 3.15c) is extracted correctly (cf. Fig. 3.22c). However, the function value is increased in general and exhibits a steeper gradient. Finally, the quantile function $Q_{scl}(z)$ of the scaling distribution of the Noise failure type is shown in Fig. 3.22a. It indicates a distribution with rarely occurring extreme values. Nevertheless, compared to the original quantile function (cf. Fig. 3.16) it fails to integrate Outliers as clearly.

On the one hand, these examples show that the processing chain is capable of identifying time- and value correlations of the initial failure amplitudes $f(k, o_k)$. On the other hand, these are not following the original correlations.

This, however, may not entail an incorrect failure model. In contrast, the *GFM* and more specifically the employed concept of a time- and value-correlated random distribution enables representing general trends in various ways. This entails that there is not a unique representation for a single failure type, which renders comparing the correlations of an identified failure type with its original version a challenging task. Thus, to assess the degree to which the failure model represents the initial failure characteristics of $f(k, o_k)$ the third stage is used.

3.4.4. Confidence Calculation

The failure models generated by the parameterization stage of the processing chain aim at representing the initial failure characteristics. However, the degree to which they succeed in this task is not stated yet. In this endeavor, Section 3.3.3 proposed a confidence calculation based on the extraction of intervals. As this requires to show the correct working of the interval calculation first, the next subsection uses a new set of manually modeled failure types to examine the aspects of the interval calculation before the following subsection applies the confidence calculation proposed in Section 3.3.3 to the failure models obtained by the processing chain.

Verifying Calculation of Intervals

As the confidence value calculated in the last stage of the processing chain is based on the extraction of intervals from a *GFM*, the approach has to be evaluated first. In this endeavor, this section firstly defines a new set of failure types specifically designed to highlight aspects of the interval extraction process. Secondly, Monte-Carlo simulation is used to verify that the determined intervals are indeed met by failure amplitudes of the corresponding failure types depending on the chosen α and γ values.

Failure Types and Failure Models The extraction of an interval $\mathcal{I}_{\mathcal{FM}}$ from a *GFM* requires considering varying aspects. To facilitate examining these individually, as well as in combination, a specific set of failure types is manually designed. Table 3.7 provides an overview of these and their definitions.

Firstly, Gaussian Noise is considered. Its scaling distribution is an uncorrelated Gaussian distribution. This failure type thereby enables analyzing the gradually increasing and decreasing range of extracted intervals depending solely on the value of α. Thus, it facilitates examining the effect of this parameter on the transformation of a time- and value-correlated random distribution to an interval.

Similarly, the γ parameter is used to differentiate the activation of a failure type as either being always active, always inactive, or maybe active. To show the effect of

Tab. 3.7.: Template distributions used to generate the time- and value-correlated distributions of the modeled failure types.

Failure Type	Scaling	Activation	Deactivation
Gaussian Noise	$\mathbb{N}(0, 0.5)$	0	1
Outlier	$Q_{Outlier}$ $(\mu = 0, \sigma = 15 + 10 \cdot o_k)$	$\mathbb{N}(750, 50)$	1
Constant Positive Offset	30	0	∞
Constant Negative Offset	-30	0	∞
Positive Spike	30	$\mathbb{N}(100, 25)$	$\mathbb{N}(35, 0.5 + 0.01 \cdot k)$
Negative Spike	-30	$\mathbb{N}(100, 25)$	$\mathbb{N}(35, 0.5 + 0.01 \cdot k)$

varying this parameter, the Outlier failure type is added. Equal to the Gaussian Noise, its deactivation is one. However, its activation distribution follows a Gaussian distribution with a mean of 750 time steps and a standard deviation of 50. Therefore, the failure type becomes active only rarely which allows examining the effect of considering a failure type to be always inactive.

Moreover, the failure type features a custom quantile function and a linear correlated standard deviation of $\sigma_{scl}(k, o_k) = 15 + 10 \cdot o_k$. While the correlated scaling facilitates showing that by limiting the time- and value domain for which an interval is valid, correlation information can be partially maintained, the custom quantile function shown in Fig. 3.23 underlines the ability to represent arbitrary distributions. However, due to its approximation by a polynomial of degree 5, the actual quantile function does not follow the targeted distribution.

Finally, the positive and negative Spike and Offset failures are considered to examine the feature of *cancellation*. As the interval of a failure model is nothing but the sum of the individual failure type's intervals, adverse intervals may cancel out each other, e.g. for constant positive and negative Offsets. However, for that, their occurrences have to be aligned timely, which is not the case for positive and negative Spike failures, as will be shown.

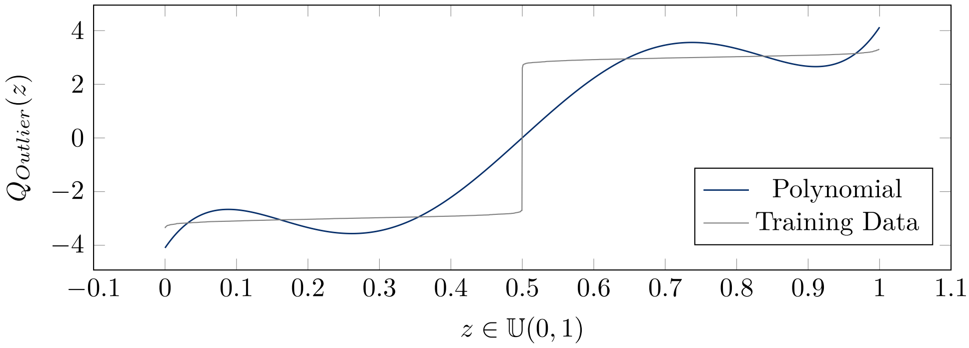


Fig. 3.23.: Comparing targeted and represented quantile function for scaling distribution of Outlier failure type.

Tab. 3.8.: Mapping the sets of considered failure types to labels of the associated failure models for evaluating the calculation of confidence values.

Failure Model	Failure Types	Failure Model	Failure Types
$\mathcal{FM}_{I_{\{O\}}}$	$\{F_{Outlier}\}$	$\mathcal{FM}_{I_{\{PO\}}}$	$\{F_{Constant\ Positive\ Offset}\}$
$\mathcal{FM}_{I_{\{N\}}}$	$\{F_{Gaussian\ Noise}\}$	$\mathcal{FM}_{I_{\{NS,O\}}}$	$\{F_{Negative\ Spike}, F_{Outlier}\}$
$\mathcal{FM}_{I_{\{NS\}}}$	$\{F_{Negative\ Spike}\}$	$\mathcal{FM}_{I_{\{PS,O\}}}$	$\{F_{Positive\ Spike}, F_{Outlier}\}$
$\mathcal{FM}_{I_{\{PS\}}}$	$\{F_{Positive\ Spike}\}$	$\mathcal{FM}_{I_{\{PS,NS\}}}$	$\{F_{Positive\ Spike}, F_{Negative\ Spike}\}$
$\mathcal{FM}_{I_{\{NO\}}}$	$\{F_{Constant\ Negative\ Offset}\}$	$\mathcal{FM}_{I_{\{NO,PO\}}}$	$\{F_{Constant\ Negative\ Offset},$ $F_{Constant\ Positive\ Offset}\}$

To evaluate these impositions of individual failure types, they are combined to form the failure models considered in this evaluation. They are listed in Table 3.8.

Examining and Evaluating Calculated Intervals For the defined failure models, the described approach for extracting intervals can be applied. In that endeavor, the time and value domain at which these shall be valid have to be defined. Here, the sinusoidal reference signal described in Eq. (3.67) is used for a time domain of $k \in [0, 1500]$ again.

As the defined failure models are not time-correlated, the intervals are extracted for $\mathbb{K} = [0, 1500]$, that is, the entire time domain. Contrarily, the value domain of the reference signal is $o_k \in [-1, 2]$. As a consequence of the discrete stepping, Eq. (3.67) produces a non-continuous reference signal with a discretization of $O_s = 1.7^{-6}$. Here, this discretization is used for calculating the intervals, which means that a separate interval is calculated for each time step ($\mathbb{O}(o_k) = [o_k - 0.85^{-6}, o_k + 0.85^{-6}]$). This allows following value-correlations closely. Lastly, the parameters α and γ have to be defined. For the sake of argumentation, two values are considered. Firstly, intervals for $\alpha = \gamma = 1.0$ are determined. These are expected to cover the modeled failure characteristics, that is, no failure amplitude outside this interval should occur. Secondly, intervals for $\alpha = \gamma = 0.75$ are calculated where failure amplitudes outside these are to be expected.

To evaluate these expectations, the failure models from which the intervals are derived are simulated 30 times using the procedure described in Section 3.4.1. Using the obtained failure amplitudes, the confidence values are calculated.

The results are shown in Table 3.9 for $\alpha = \gamma = 1.0$ and in Table 3.10 for $\alpha = \gamma = 0.75$. For evaluating the intervals, three metrics are listed.

Firstly, the average ratio of failure amplitudes outside their corresponding intervals is given, cf. Eq. (3.75).

$$\mu_{f \notin \mathcal{I}} = \frac{|\{f_{k,o_k} \in \mathbb{F} | f_{k,o_k} \notin \mathcal{I}_{\mathcal{FM}}(\mathbb{K}, \mathbb{O}(o_k))\}|}{|\mathbb{F}|} \tag{3.75}$$

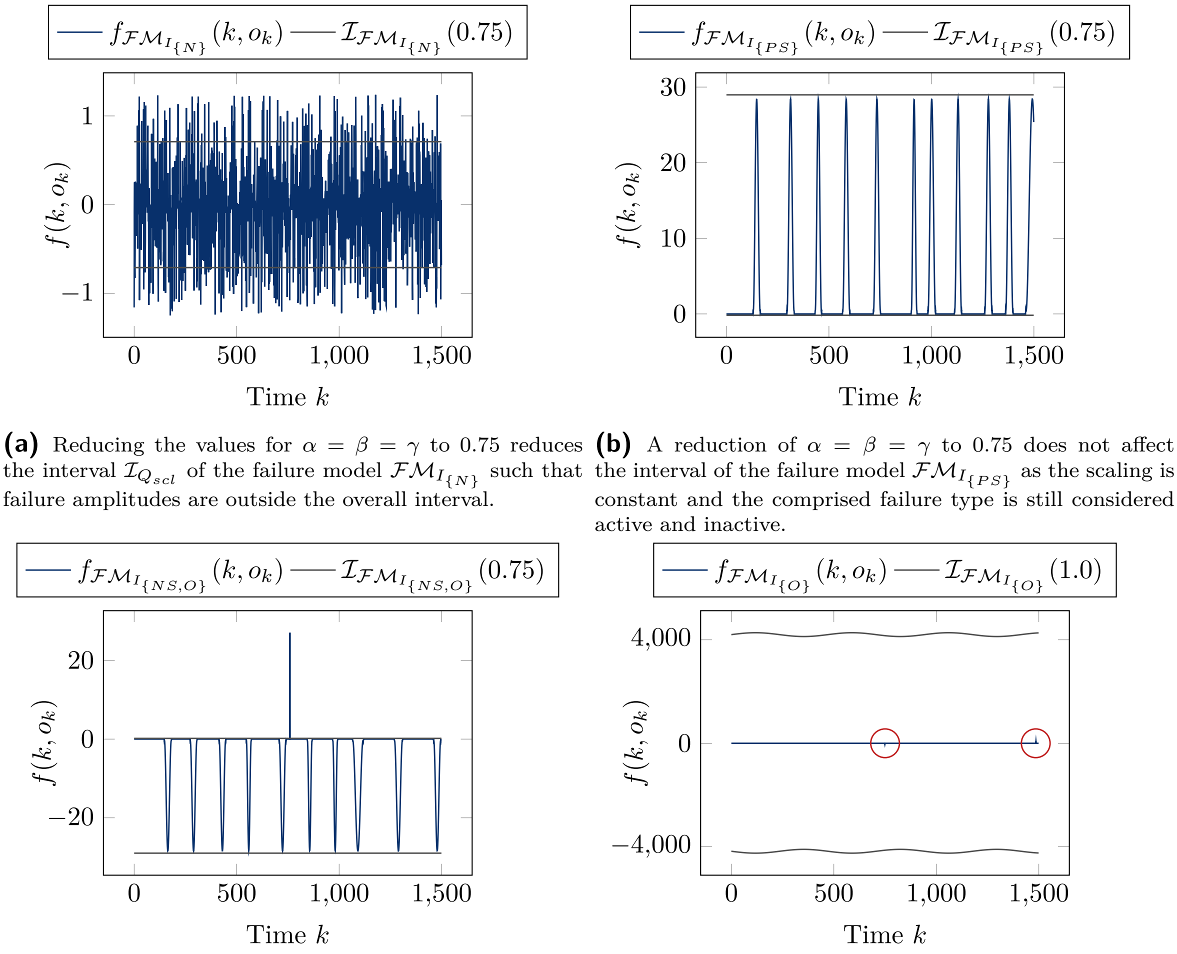
In this equation, $\mathbb{F}$ denotes the set of all failure amplitudes obtained by simulating the respective failure model $\mathcal{FM}_I$. While the numerator describes the set of failure

Tab. 3.9.: Results of verifying intervals for $\alpha = \gamma = 1.0$.

Failure Model	$\mu_{f \notin \mathcal{I}}$	$d_{\mathcal{I}}$	$range(\mathcal{I}_{\mathcal{FM}_I})$
$\mathcal{FM}_{I_{\{O\}}}$	0	0.48	8.5e+03
$\mathcal{FM}_{I_{\{N\}}}$	0	0.047	2.8
$\mathcal{FM}_{I_{\{NS\}}}$	0	0.019	29
$\mathcal{FM}_{I_{\{PS\}}}$	0	0.019	29
$\mathcal{FM}_{I_{\{NO\}}}$	0	0	0
$\mathcal{FM}_{I_{\{PO\}}}$	0	0	0
$\mathcal{FM}_{I_{\{NS,O\}}}$	0	0.49	8.6e+03
$\mathcal{FM}_{I_{\{PS,O\}}}$	0	0.49	8.6e+03
$\mathcal{FM}_{I_{\{PS,NS\}}}$	0	0.012	58
$\mathcal{FM}_{I_{\{NO,PO\}}}$	0	0	0

Tab. 3.10.: Results of verifying intervals for $\alpha = \gamma = 0.75$.

Failure Model	$\mu_{f \notin \mathcal{I}}$	$d_{\mathcal{I}}$	$range(\mathcal{I}_{\mathcal{FM}_I})$
$\mathcal{FM}_{I_{\{O\}}}$	0.00098	-1.2e+27	0
$\mathcal{FM}_{I_{\{N\}}}$	0.2	-0.38	1.4
$\mathcal{FM}_{I_{\{NS\}}}$	0	0.019	29
$\mathcal{FM}_{I_{\{PS\}}}$	0	0.019	29
$\mathcal{FM}_{I_{\{NO\}}}$	0	0	0
$\mathcal{FM}_{I_{\{PO\}}}$	0	0	0
$\mathcal{FM}_{I_{\{NS,O\}}}$	0.00064	-4.1	29
$\mathcal{FM}_{I_{\{PS,O\}}}$	0.00084	-4.1	29
$\mathcal{FM}_{I_{\{PS,NS\}}}$	0	0.012	58
$\mathcal{FM}_{I_{\{NO,PO\}}}$	0	0	0



(a) Reducing the values for $\alpha = \beta = \gamma$ to 0.75 reduces the interval $\mathcal{I}_{Q_{scl}}$ of the failure model $\mathcal{FM}_{I_{\{N\}}}$ such that failure amplitudes are outside the overall interval.

(b) A reduction of $\alpha = \beta = \gamma$ to 0.75 does not affect the interval of the failure model $\mathcal{FM}_{I_{\{PS\}}}$ as the scaling is constant and the comprised failure type is still considered active and inactive.

(c) For $\alpha = \beta = \gamma = 0.75$ the Outlier failure type contained in $\mathcal{FM}_{I_{\{NS,O\}}}$ is considered to always be inactive. Therefore, only the interval of the Spike failure type is considered and occurrences of the Outlier failure type may cause failure amplitudes outside the interval.

(d) Due to the high degree of the Outlier's quantile function in $\mathcal{FM}_{I_{\{O\}}}$, the Lipschitz constant required for determining the interval is high, causing an overestimation of the failure amplitudes for $\alpha = \beta = \gamma = 1.0$. The value-correlations in the failure type's σ_{scl} function are visible in the correspondingly changing interval borders.

Fig. 3.24.: Comparing determined interval borders with failure amplitudes of corresponding failure models.

amplitudes that are outside their corresponding intervals, the denominator states the overall number of failure amplitudes, that is, $4.5 \cdot 10^4$. In other words, if this ratio is greater than zero, failure amplitudes outside the extracted intervals are observed.

The second column states the minimal distance to the interval borders and is given in Eq. (3.66).

Finally, for reference, the maximal range of the calculated intervals is given in the third column.

As expected, Table 3.9 shows that the average ratio $\mu_{f \notin \mathcal{I}}$ is indeed for all failure models zero. Therefore, no failure amplitudes outside the calculated intervals are observed. Moreover, only small distances $d_{\mathcal{I}}$ are obtained for failure models $\mathcal{FM}_{I_{\{N\}}}$-$\mathcal{FM}_{I_{\{PO\}}}$ and $\mathcal{FM}_{I_{\{PS,NS\}}}$-$\mathcal{FM}_{I_{\{NO,PO\}}}$. These indicate that the intervals are close to the simulated failure amplitudes and are thereby not drastically overestimated. Contrarily, these value are increased for failure models comprising the Outlier failure type ($\mathcal{FM}_{I_{\{O\}}}$,$\mathcal{FM}_{I_{\{NS,O\}}}$,$\mathcal{FM}_{I_{\{PS,O\}}}$). Implied by the increased range value and as shown in Fig. 3.24d, the interval extracted from this failure type is drastically overestimating the actual failure amplitudes, which range between $f(k, o_k) \in [-20, 140]$. The reason for this overestimation is the Lipschitz constant $L_{Q_{scl}} = 39.7$ of the scaling distribution's quantile function combined with a step size of $K_S = 167$ in the time domain. Firstly, the Lipschitz constant causes the interval of the quantile function to be $[-4.9, 4.9]$, which overestimates the true interval (cf. Fig. 3.23). Secondly, the failure type's linear value-correlated scaling function $\sigma_{scl}(k, o_k)$ has a Lipschitz constant of $L_{\sigma_{scl}} = 10.0$ which is multiplied by $0.5 \cdot K_S$ to generate the upper and lower bound of the function's interval, cf. Section 3.2.1. As both intervals are multiplied, the resulting interval strongly overestimates the true failure amplitudes. This problem affects the intervals $\mathcal{I}_{\mathcal{FM}_{I_{\{O\}}}}$,$\mathcal{I}_{\mathcal{FM}_{I_{\{NS,O\}}}}$,$\mathcal{I}_{\mathcal{FM}_{I_{\{PS,O\}}}}$.

To counteract the effect, either the step size K_S has to be reduced or the Lipschitz constant of the employed functions.

On the other hand, Fig. 3.24d shows that the high resolution of $O_s = 1.7^{-6}$ used to discretize the value domain resulted in fine-grained intervals which follow the value-correlation of the failure type's standard deviation closely. The sinusoidal curve is a direct consequence of the chosen reference signal.

In contrast to the Outlier failure type, the range values stated for failure models $\mathcal{FM}_{I_{\{NO\}}}$, $\mathcal{FM}_{I_{\{PO\}}}$, and $\mathcal{FM}_{I_{\{NO,PO\}}}$ are zero. This has two reasons. In case of $\mathcal{FM}_{I_{\{NO\}}}$ and $\mathcal{FM}_{I_{\{PO\}}}$, the failure amplitudes are constant ±30, which means that there is no variation resulting in an interval having only one value. For $\mathcal{FM}_{I_{\{NO,PO\}}}$, the reason is different. The failure model combines the negative and positive variants of the constant Offsets. Due to the summation in Eq. (3.17), the failure amplitudes cancel each other out. This is possible only because both failure types are constantly active.

In contrast, failure model $\mathcal{FM}_{I_{\{PS,NS\}}}$ combines positive and negative Spikes. While both are scaled exactly complementary to each other, it can not be guaranteed that both are always active at the same time. Therefore, the range of the resulting interval doubles and the failure types can not cancel out each other.

Reducing the values for α and γ to 0.75 produces the results listed in Table 3.10. In contrast to the previous intervals, mean values $\mu_{f \notin \mathcal{I}}$ greater than zero are now observed. This is due to the reduced range of the calculated intervals.

For instance, the interval extracted from the Gaussian Noise ($\mathcal{FM}_{I_{\{N\}}}$) is reduced in its range because of its quantile function. As $\alpha = 0.75$, only a smaller portion of the represented scaling distribution is considered, resulting in an interval that does not cover all simulated failure amplitudes, cf. Fig. 3.24a.

For failure model $\mathcal{FM}_{I_{\{NS,O\}}}$, the reason for failure amplitudes outside its interval is the Outlier, cf. Fig. 3.24c. The reduced value for γ causes the assumption that the Outlier is always inactive. Therefore, its interval is zero. Contrarily, the failure type occurs and imposes the Spike failure type, causing a failure amplitude outside the interval. As a result, the minimal distance $d_{\mathcal{I}}$ is negative and therefore provides an additional indication.

Nevertheless, these failure amplitudes outside of their respective intervals were expected, which underlines that the extraction of intervals from *GFM* works properly. Therefore, confidence values for the failure models generated by the processing chain can be calculated next.

Tab. 3.11.: Confidence values according to Eq. (3.66) for $\alpha, \gamma \in [1.0, 0.75, 0.5]$.

Failure Model	$d_{\mathcal{I}}(\alpha = \gamma = 1.0)$	$d_{\mathcal{I}}(\alpha = \gamma = 0.75)$	$d_{\mathcal{I}}(\alpha = \gamma = 0.5)$
$\hat{\mathcal{FM}}_{\{N\}}$	-2.7	-11	-14
$\hat{\mathcal{FM}}_{\{O\}}$	0.47	0.45	0.43
$\hat{\mathcal{FM}}_{\{S\}}$	0.41	0.38	0.32
$\hat{\mathcal{FM}}_{\{A\}}$	0.23	0.2	0.17
$\hat{\mathcal{FM}}_{\{N,O\}}$	0.45	0.43	0.4
$\hat{\mathcal{FM}}_{\{N,S\}}$	0.3	0.28	0.26
$\hat{\mathcal{FM}}_{\{N,A\}}$	0.1	-0.0026	-0.07
$\hat{\mathcal{FM}}_{\{S,O\}}$	0.41	0.3	0.047
$\hat{\mathcal{FM}}_{\{O,A\}}$	0.42	0.43	0.44
$\hat{\mathcal{FM}}_{\{S,A\}}$	0.42	0.4	0.37
$\hat{\mathcal{FM}}_{\{N,S,O\}}$	0.45	0.45	0.44
$\hat{\mathcal{FM}}_{\{N,O,A\}}$	0.45	0.43	0.4
$\hat{\mathcal{FM}}_{\{N,S,A\}}$	0.4	0.38	0.36
$\hat{\mathcal{FM}}_{\{S,O,A\}}$	0.47	0.45	0.41
$\hat{\mathcal{FM}}_{\{N,S,O,A\}}$	0.45	0.44	0.44

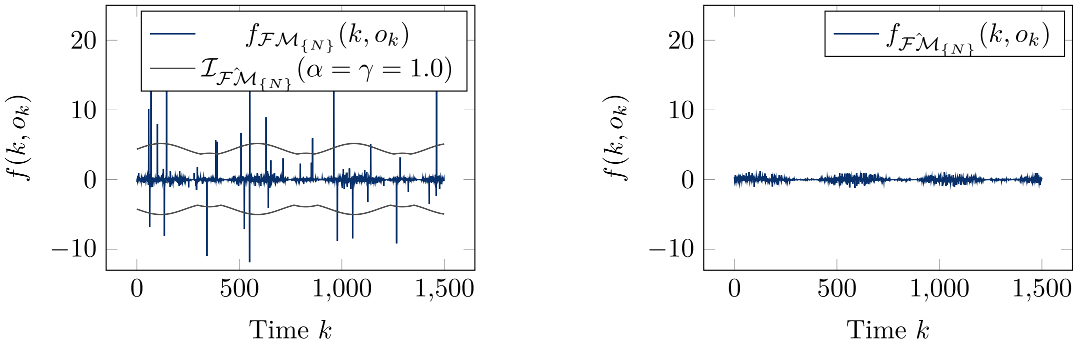
Calculating Confidence Values for Generated Failure Models

Using the original failure amplitudes $f(k, o_k)$ from which the previous two stages generated failure models, the last stage aims at providing a confidence value stating to which degree the original failure characteristics are described. With this, the confidence criterion will be addressed.

In this endeavor, this paragraph assumes values of $\alpha, \gamma \in [1.0, 0.75, 0.5]$ to calculate the minimal distance to the interval borders of the generated failure models. Similar as before, a time horizon of $\mathbb{K} = 1500$ time steps, that is, the entire length of the time series is considered, while the value domain is discretized with $O_s = 1.7^{-6}$, which results in $\mathbb{O}(o_k) = [o_k - 0.85^{-6}, o_k + 0.85^{-6}]$. The results are shown in Table 3.11.

As one can see, the minimal distance $d_{\mathcal{I}}$ generally remains on a high level for most failure models and falls below 0.3 only for some failure models even for $\alpha = \gamma = 0.5$. This indicates an overall high distance to the interval borders. On the one hand, this means that the initial failure characteristics are completely covered by those failure types. On the other hand, the range of failure amplitudes is overestimated when extracting intervals from the corresponding *GFM*.

In contrast, the minimal distance of $\hat{\mathcal{FM}}_{\{N\}}$, which contains only the Noise failure type, is not covering all failure amplitudes, even at $\alpha = \gamma = 1.0$. The reason can be seen in Fig. 3.25 where, on the left-hand side, the original failure amplitudes are



(a) Despite using $\alpha = \gamma = 1.0$, failure amplitudes of the Noise failure are not included in the interval. This is caused by the Outlier failure amplitudes, that are not represented in the automatically generated failure model.

(b) A simulation of the generated Noise failure type shows that Outliers are not modeled. In contrast, value-correlations similar to the original failure amplitudes can be observed.

Fig. 3.25.: Comparing determined interval borders with failure amplitudes of corresponding failure models.

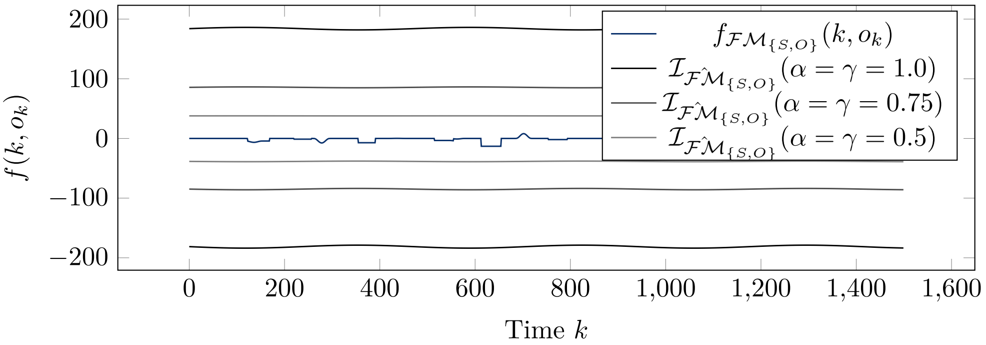
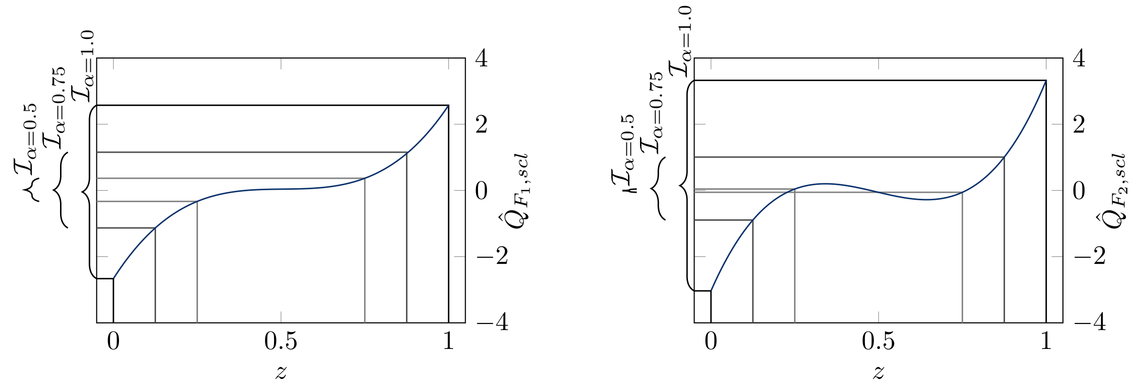


Fig. 3.26.: The intervals extracted from the automatically generated failure model successfully cover the range of failure amplitudes for $\mathcal{FM}_{\{S,O\}}$.

shown together with the interval calculated from the extracted failure model. On the right-hand side, failure amplitudes simulated according to the extracted failure model are shown. It illustrates the expectation formualted with respect to Fig. 3.22a. The failure type captures small Noise and its value-correlation, but extreme values are not represented. As such, it fails at representing the Outlier failure type originally integrated into the Noise failure type. Thus, the extracted interval is not sufficient to cover the corresponding failure amplitudes contained in the training data.

The same reason applies for $\hat{\mathcal{FM}}_{\{N,A\}}$ where the original failure amplitudes comprise not only Noise with Outliers but also occurrences of the Artificial failure type. While the extracted failure model covers the original failure amplitudes when considering $\alpha = \gamma = 1.0$, it fails to provide the same for reduced values of $\alpha, \gamma \in [0.75, 0.5]$.

The effect of reducing these values is observable by other failure models as well, as the overall minimal distance to the interval borders is decreasing correspondingly. This gradient is especially steep for $\hat{\mathcal{FM}}_{\{S,O\}}$, cf. Fig. 3.26. The reason for the steep gradient is depicted in Fig. 3.27. The quantile functions $\hat{Q}_{scl}(z)$ of the scaling distributions of the extracted failure types show similarities with a quantile function of a Gaussian distribution (cf. Fig. 3.15a) and react therefore on changes of the α parameter. A reduced value causes the intervals to collapse accordingly.



(a) Quantile function of the first failure type.

(b) Quantile function of the second failure type.

Fig. 3.27.: Quantile functions of scaling distributions of the failure types of the extracted failure model $\mathcal{FM}_7$. The intervals for levels of $\alpha \in [1.0, 0.75, 0.5]$ are visualized as well.

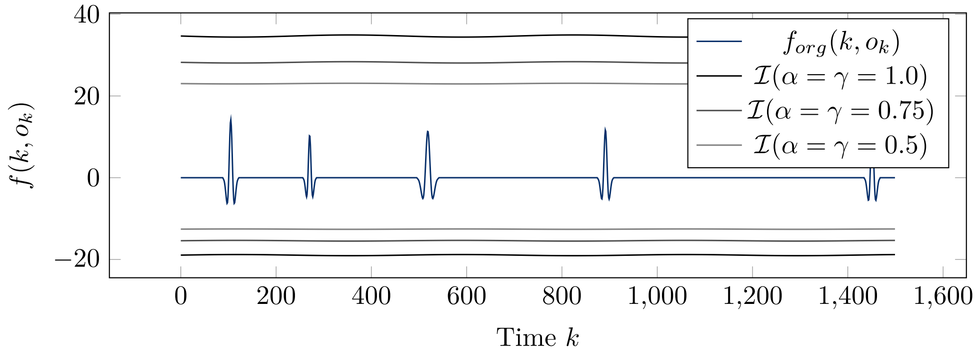


Fig. 3.28.: The intervals extracted from the automatically generated failure model successfully cover the range of failure amplitudes for $\mathcal{FM}_4$.

On the other hand, a disadvantage of using polynomials to represent quantile functions can be seen as well. The quantile function of the second failure type (cf. Fig. 3.27b) is decreasing for values of $z \in [0.34, 0.64]$. This violates the requirement of a quantile function to be always increasing.

Nevertheless, the initial intervals are overestimating the actual failure amplitudes, which is caused by increased Lipschitz constants similar to the Outlier failure type described in the previous section.

On the other hand, the example of $\hat{\mathcal{FM}}_{\{A\}}$ shows that estimations closer to the actual failure amplitudes are possible as well. While the interval shrinks in response to the adjusted values of α and γ, it does not overestimate the failure amplitudes as severely as $\hat{\mathcal{FM}}_{\{N,A\}}$.

In summary, the confidence values in Table 3.11 underline that the processing chain is capable of extracting failure models from given failure amplitudes. These, however, should be inspected, e.g. using the confidence values, regarding their actual coverage of the failure characteristics to represent. In the case of $\hat{\mathcal{FM}}_{\{N\}}$ for instance, the interpretation of each polynomial and each failure type might be leveraged to manually adjust their parameters for mitigating the over- or underestimation of failure characteristics.

3.4.5. Comparison to State-of-the-Art Models

The confidence values provided by the processing chain allow assessing the coverage of the failure characteristics represented by the produced models with respect to the initial failure amplitudes they are extracted from. This, however, does not address the performance aspect of the failure model, that is, how well the failure characteristics are represented. In the endeavor of assessing this, this section focuses on a comparison of the *GFM* to exemplary modeling techniques from the state of the art. For that, the next subsection introduces the considered comparison models while the following subsection describes the employed comparison metric. Finally, the results are presented and discussed.

Considered Models

For comparing the modeling performance of the generated failure models, exemplary state-of-the-art techniques are considered. For that, the following paragraphs introduce not only the selected models but also describe the employed procedure to find their respective parameters and simulate time series of failure amplitudes.
With respect to the reviewed categories of failure modeling approaches, the first paragraph discusses uniform and Gaussian distributions. While the former is an interpretation of an interval-based modeling approach, where a uniform distribution of failure amplitudes between minimal and maximal failure amplitudes is assumed, the latter is an approach from the category of the distribution-based models. As both of these approaches are not capable of representing time- and value-correlations, a *Long-Short Term Memory (LSTM)* network (a kind of *Artificial Neural Network (ANN)* specifically suited for time series analysis) is considered as well. In contrast to a *GFM*, an *LSTM* is a discriminative model that represents or predicts only the most likely failure amplitude and not its possible distribution. Therefore, a Gaussian Process capable of additionally representing the (co-)variance is considered as well.

Uniform Distribution A uniform distribution describes failure characteristics only by their minimal and maximal failure amplitudes observed. It assumes that the remaining failure amplitudes are distributed uniformly between those limits.
Accordingly, given time series of failure amplitudes $f(k, o_k)$, the model is trained by searching for the minimal and maximal failure amplitudes f_{max} and f_{min} respectively. Based on that, time series of the same are simulated by drawing random values z from a uniform distribution and scaling them to be in the interval $[f_{min}, f_{max}]$, cf. Eq. (3.76).

$$f(k, o_k) = z \cdot (f_{max} - f_{min}) + f_{min} \text{ with } z \in \mathbb{U}(0, 1) \tag{3.76}$$

As such, each time step of the failure amplitudes is considered independent. This means that neither time- nor value-correlations can be represented by this model.

Gaussian Distribution Opposed to the uniform distribution, the Gaussian distribution assumes that values close to its mean are more likely to occur. It represents failure amplitudes according to Eq. (3.77).

$$f(k, o_k) \sim \mathbb{N}(\mu_f, \sigma_f) \tag{3.77}$$

Similar to the uniform distribution, each time step is considered to be independent of each other which does not allow the model to represent any time- or value-correlations.
Training the model requires only to determine the mean values μ_f and the standard deviation σ_f of all failure amplitudes $f(k, o_k)$. Then, simulating the model for generating time series of $f(k, o_k)$ is achieved by drawing samples from the represented distribution at each time step independently.

***Long-Short Term Memory (LSTM)* Network** Opposed to both of the previous approaches, *LSTM* networks are capable of representing time- and value correlations. This type of neural network has recurrent connections that enable it to maintain an internal state or context representation. The output provided by an *LSTM* network does not only depend on its current input but also its history of inputs. This facilitates the network to contextualize its inputs and support its decision-making when applied to sequences. With these characteristics, *LSTM* networks are commonly used for speech recognition [94] or time series analysis [95].
In the context of failure modeling, it enables representing time- and value-correlations. For that, the model used in this work has a single *LSTM* layer with 20 cells followed by a single linear layer with 20 neurons. For training the model a vanilla *SGD* algorithm with a learning rate of $\alpha = 0.01$ and the *MAE* as a loss function is used. The weights are updated for 50 epochs.

$$f(k, o_k) = lstm(k, o_k) \tag{3.78}$$

During training, the network is presented with the current time index k as well as the value of the reference signal o_k, which are the same inputs as presented to the *GFM* models, cf. Eq. (3.78). Correspondingly, for simulating a time series of $f(k, o_k)$ given an *LSTM* network, the reference signal o_k is used together with the time index k as an input to the network.

Gaussian Process Lastly, a *Gaussian Process (GP)* is considered. A *Gaussian Process (GP)* [96] shares the advantage of the *GFM*. It is able to represent the mean as well as the (co-)variance of failure amplitudes at each time step. The central assumption is that the stochastic process to be modeled can be represented as a multi-variate Gaussian distribution, cf. Eq. (3.79).

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \sim \mathbb{N} \left(\mu(\mathbf{y}), \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix} \right) \tag{3.79}$$

with

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x_1) & k(x_n, x_2) & \cdots & k(x_n, x_n) \end{bmatrix} \tag{3.80}$$

Opposed to the previous models, a *GP* model does not need to be fitted in the sense that a vector of model parameters needs to be determined. Instead, training samples of inputs ($\mathbf{y} = [y_1, y_2, \ldots, y_n]$) and outputs ($\mathbf{x} = [x_1, x_2, \ldots, x_n]$) are used to directly construct the mean $\mu_{\mathbf{y}}$ and the covariance matrix (K). For the latter, a covariance function, also called a *kernel*, is chosen, which states the believed correlation between two inputs x_1 and x_2.

$$k(x_1, x_2) = \sigma_f^2 e^{\frac{-(x_1-x_2)^2}{2l^2}} + \sigma_n^2 \delta(x_1, x_2) \tag{3.81}$$

The idea of this kernel is to state the belief that input values x_1,x_2 that are close in the input space will have similar function values while values for x_1,x_2 that are far apart are likely to have different function values. To state this belief, this work uses the kernel of Eq. (3.81) with $l^2 = e^{2.0}$, $\sigma_f^2 = 1$, $\sigma_n^2 = e^{0.5}$. The kernel combines a radial basis function kernel (first part of the sum), with a Noise assumption (second part of the sum)[5].

Similar to the kernel, a function determining the mean $\mu(\mathbf{y})$ can be specified. However, in this work, a constant zero is assumed. Note that this does not mean that the predictive mean is zero as well, as can be seen in Eq. (3.82).

Using the covariance function, the model can be *trained* by determining the overall covariance K from the training data. The remaining elements, that is y_*, K_*, K_{**}, K_*^T are determined when predicting failure amplitudes.

For that, the given input x_* (o_k and k in the case of modeling failures) is firstly used to generate the matrix $K_* = [k(x_*, x_1), k(x_*, x_2), \ldots, k(x_*, x_n)]$ and the element $K_{**} = k(x_*, x_*)$. Secondly, a prediction is made by leveraging the Bayes theorem, which allows transforming the marginal distribution given in Eq. (3.79) into the conditional distribution Eq. (3.82).

$$y_* | \mathbf{y} \sim \mathbb{N}(K_* K^{-1} \mathbf{y}, K_{**} - K_* K^{-1} K_*^T) \tag{3.82}$$

Eq. (3.82) enables not only to predict the failure amplitude having the highest probability but also the corresponding variance.

$$f(k, o_k) \sim \mathbb{N}(y_*, K_{**} - K_* K^{-1} K_*^T) \tag{3.83}$$

Therefore, simulating a time series of failure amplitudes using a *GP* model is done by sampling from the given distribution, cf. Eq. (3.83)

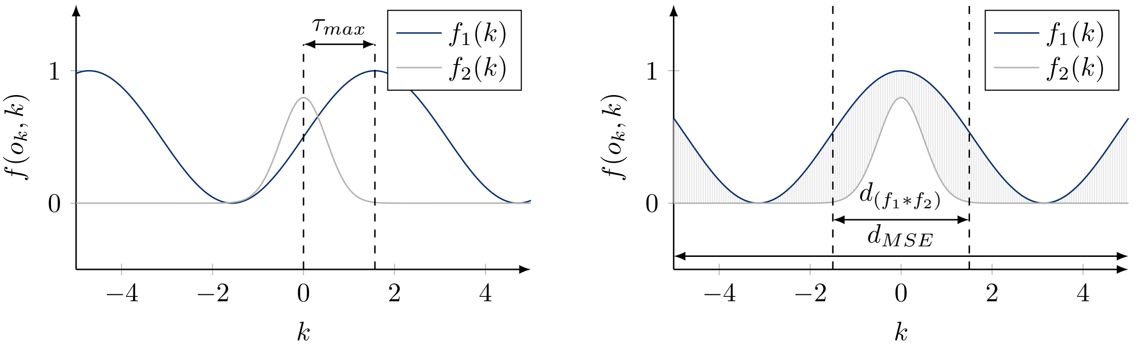
Evaluation Metric

The presented types of models approach representing time series in different ways and can therefore not be compared directly. Instead, the stochastic nature of failure amplitudes is leveraged, which enables interpreting each model as a random variable. The goal is to compare the distribution of these random variables. In that endeavor, the random variables are sampled, that is, failure amplitudes are generated, and the resulting sample sets are compared.

In other words, each model is simulated N_{sim} times to generate failure amplitudes $f(k, o_k)$. The set of generated series of failure amplitudes then represents the distribution of the random variable, that is, the model.

Comparing the models therefore boils down to comparing their sets of simulated failure amplitudes. For that, the Wasserstein distance (more specifically the *Earth Mover's Distance (EMD)*) can be employed as a distance metric [97]. Its application, howerver,

[5] See [96] for more information on combining kernel



(a) Using cross-correlation of function f_1 and f_2 to align them in time.

(b) Difference between using cross-correlation and *Mean Squared Error (MSE)* to compare functions. The cross-correlation $d_{(f_1*f_2)}$ takes only time steps into account at which both functions are not zero. The *MSE* can consider all time steps but does not reflect temporal pattern.

Fig. 3.29.: Elements taken into account to form the ground distance.

is twofold. Firstly, a ground distance used to compare individual samples of each distribution has to be defined. Building on that, the Wasserstein distance can be applied.

Correspondingly, the next paragraph introduces a measure for comparing two time series of failure amplitudes with each other (the ground distance) before the following paragraph discusses its usage in the calculation of the Wasserstein distance.

Comparing Time-Series of Failure Amplitudes The first step in applying the Wasserstein distance is to specify an underlying distance measure that compares two samples of the considered random variables. In this case, failure amplitudes $f_{\mathcal{FM}}(k, o_k)$ of the original failure model and failure amplitudes $f_{\hat{\mathcal{FM}}}(k, o_k)$ of a comparison model are considered.

At this point, it is known that these signals comprise stochastic and deterministic parts. As a consequence, it is not sufficient to either compare solely stochastic moments of two signals as this would neglect the presence of temporal patterns or to compare signals directly time step by time step as this neglects the stochastic aspects. To address this challenge, a distance metric based on the cross-correlation (addressing the stochastic part) and the well-known *Mean Squared Error (MSE)* is defined.

The idea is visualized in Fig. 3.29. On the one hand, temporal patterns are anticipated in the signals. However, their occurrences are random causing varying times between two consecutive patterns. To address this randomness, the best temporal alignment of two signals f_1 and f_2 are calculated using cross-correlation in a first step cf. Fig. 3.29a. The cross-correlation of two signals is the area of the combination of both signals with respect to a displacement τ, cf. Eq. (3.84).

$$(f_1(k) * f_2(k))(\tau) = \sum f_1(k) \cdot f_2(k + \tau) \tag{3.84}$$

$$\tau_{max} = \underset{\tau \in K}{\operatorname{argmax}} \left((f_1 * f_2)(\tau)\right) \tag{3.85}$$

By calculating the cross-correlation and determining the value of τ that maximizes it (cf. Eq. (3.85)), a temporal offset, by which one signal has to be shifted for optimal correlation with the other, can be found. As this shift already represents a difference in both signals, it is used as the first part of the ground distance, cf. Eq. (3.86).

$$d_\tau = 0.5 \cdot \frac{\left(\frac{K}{2} - \tau_{max}\right)^2}{\frac{K^2}{4}} + 0.5 \tag{3.86}$$

Here, K denotes the length of both signals (equal length of $f1$ and $f2$ is assumed as both will be simulations of corresponding failure models). d_τ is therefore a value in $[0.5, 1]$ where 0.5 represents an offset of $\tau_{max} = 0$ and 1.0 represents a maximal offset of $\tau_{max} = \frac{K}{2}$.

In a second step, the difference of the aligned signals $\hat{f}_1$ and $\hat{f}_2$ is calculated using the *MSE*, cf. Eq. (3.87).

$$d_{mse}(\hat{f}_1, \hat{f}_2) = \frac{1}{\hat{K}} \sum_{k \in [0, \hat{K}]} \left(\hat{f}_1(k) - \hat{f}_2(k)\right)^2 \tag{3.87}$$

$\hat{K}$ denotes the adjusted length of both signals, that is, the remaining time steps at which both signals overlap after temporally aligning them. The idea of choosing *MSE* is depicted in Fig. 3.29b. The figure shows the aligned signals and highlights the area between them. Indicated by the vertical dashed lines, a cost $d_{(f_1 * f_2)}$ based on the correlation of both signals would focus on time steps at which both are non-zero. This is caused by the multiplicative combination of both signals. This, however, neglects possible differences at other time steps. In contrast, the *MSE* takes these time steps into account as well and is therefore used here. Combining both produces the ground distance used for applying the Wasserstein distance in the following paragraph,cf. Eq. (3.88).

$$d_w(f_1, f_2) = d_{mse}(\hat{f}_1, \hat{f}_2) \cdot d_\tau(\tau_{max}) \tag{3.88}$$

Earth Mover's Distance (EMD) Having a ground distance $d_w(f_1, f_2)$ to compare two samples with each other, the *Earth Mover's Distance (EMD)* can be applied to calculate the distance between two sample sets $\mathcal{M}_1 = \{(w_{f_{1_1}}, f_{1_1}), \ldots, (w_{f_{1_{N_{sim}}}}, f_{1_{N_{sim}}})\}$ and $\mathcal{M}_2 = \{(w_{f_{2_1}}, f_{2_1}), \ldots, (w_{f_{2_{N_{sim}}}}, f_{2_{N_{sim}}})\}$ obtained from their models M_1 and M_2. Essentially, it measures the minimal cost required to transform the distribution of one into the other. While $w_{f_{1_i}}$ and $w_{f_{2_j}}$ represent the weight of each sample, f_{1_i} and f_{2_j} are the samples, that is, the simulated time series of failure amplitudes in this case. To transform M_1 into M_2, the weights have to be *transported* to match the corresponding weights of the target distribution. Therefore, the weights $w_{f_{2_i}}$ can also be seen as demands that have to be fulfilled by supplies provided by $w_{f_{1_j}}$.

Due to this analogy, the problem of calculating the *EMD* is often associated with the problem of optimal transportation and can be depicted as a directed graph Fig. 3.30. The set of suppliers $\mathcal{M}_1$ is the samples of the first model where the weight determines the capacity of supplies. The set of demands $\mathcal{M}_2$ is the samples of the second model where the weight determines the requested supplies. To move supply from sample i to sample j the distance between both samples has to be overcome. This distance is given by the ground distance $d_w(f_1, f_2)$ defined in the previous paragraph.

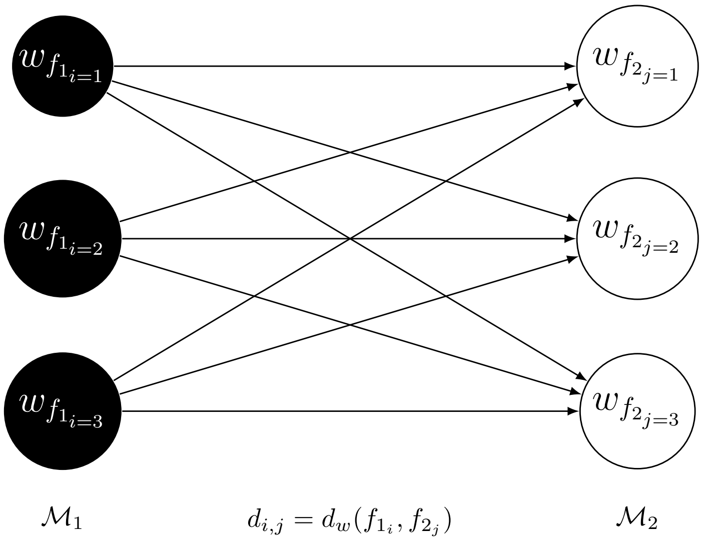


Fig. 3.30.: Visualizing the problem of calculating *EMD* as a directed graph connecting demands with supplies. The weight of each sample is indicated by the size of the nodes. Figure motivated by [97].

The *EMD* is calculated by finding the *minimum cost flow* of the directed graph. As the weight of each sample is equal to $\frac{1}{N_{sim}}$, this boils down to selecting the edges fulfilling all demands with minimal accumulated costs. Mathematically, this is a minimization problem, Eq. (3.89) [97].

$$\min \sum_{i\in\{1,\ldots,|\mathcal{M}_1|\}} \sum_{j\in\{1,\ldots,|\mathcal{M}_2|\}} w_{i,j} d_w(f_{1_i}, f_{2_j}) \tag{3.89}$$

subject to the constraints:

$$w_{i,j} \geq 0 \qquad i \in \{1,\ldots,|\mathcal{M}_1|\}, j \in \{1,\ldots,|\mathcal{M}_2|\} \tag{3.90}$$

$$\sum_{i\in\{1,\ldots,|\mathcal{M}_1|\}} w_{i,j} = w_{f_{2_j}} \qquad j \in \{1,\ldots,|\mathcal{M}_2|\} \tag{3.91}$$

$$\sum_{j\in\{1,\ldots,|\mathcal{M}_2|\}} w_{i,j} \leq w_{f_{1_i}} \qquad i \in \{1,\ldots,|\mathcal{M}_1|\} \tag{3.92}$$

The task is to find a *flow* $\mathbf{W} = [w_{i,j}]$ such that the costs are minimal (cf. Eq. (3.89)). This is constrained such that

1. supply is transferred only from suppliers to demanders (cf. Eq. (3.90)),
2. all demands are fulfilled (cf. Eq. (3.91)),
3. and the capacity of what can be supplied is not exceeded (cf. Eq. (3.92)).

By solving the minimization problem, a minimal cost flow is produced. This is used to calculate the final *EMD* as the work normalized by the total flow, cf. Eq. (3.93) [97].

$$EMD(\mathcal{M}_1, \mathcal{M}_2) = \frac{\sum_{i\in\{1,\ldots,|\mathcal{M}_1|\}} \sum_{j\in\{1,\ldots,|\mathcal{M}_2|\}} w_{i,j} d_w(f_{1_i}, f_{2_j})}{\sum_{i\in\{1,\ldots,|\mathcal{M}_1|\}} \sum_{j\in\{1,\ldots,|\mathcal{M}_2|\}} w_{i,j}} \tag{3.93}$$

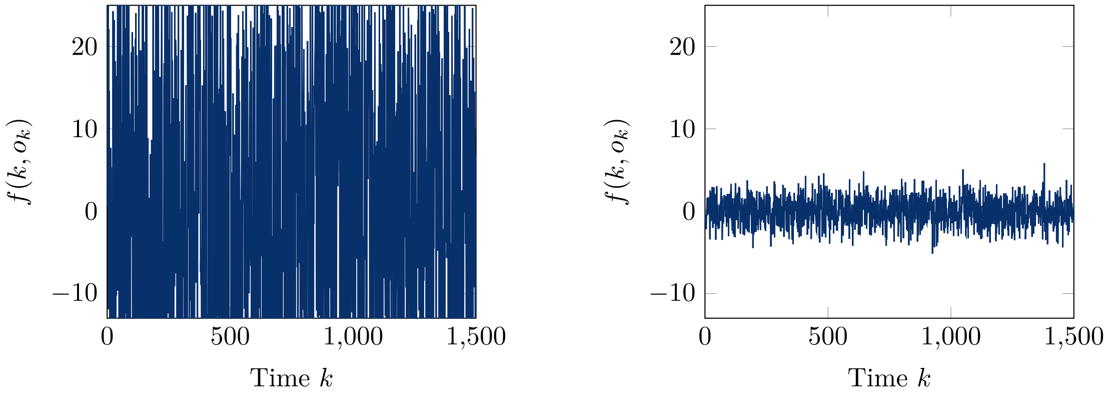
Tab. 3.12.: Comparing failure models generated by the processing chain with state-of-the-art modeling techniques. The gray-colored cells indicate the minimal values achieved by the approaches of the respective row.

Failure Model	EMD_{GFM}	EMD_{GP}	EMD_{LSTM}	$EMD_{Gaussian}$	$EMD_{Uniform}$
$\mathcal{FM}_{\{N\}}$	0.002	0.002	0.003	0.003	0.155
$\mathcal{FM}_{\{O\}}$	0.022	0.042	0.050	0.055	0.161
$\mathcal{FM}_{\{S\}}$	0.007	0.013	0.017	0.022	0.157
$\mathcal{FM}_{\{A\}}$	0.006	0.007	0.008	0.012	0.127
$\mathcal{FM}_{\{N,O\}}$	0.011	0.023	0.026	0.029	0.132
$\mathcal{FM}_{\{N,S\}}$	0.003	0.004	0.005	0.006	0.157
$\mathcal{FM}_{\{N,A\}}$	0.004	0.003	0.004	0.006	0.156
$\mathcal{FM}_{\{S,O\}}$	0.022	0.039	0.050	0.052	0.147
$\mathcal{FM}_{\{O,A\}}$	0.020	0.036	0.044	0.050	0.172
$\mathcal{FM}_{\{S,A\}}$	0.012	0.013	0.016	0.019	0.135
$\mathcal{FM}_{\{N,S,O\}}$	0.015	0.022	0.030	0.034	0.165
$\mathcal{FM}_{\{N,O,A\}}$	0.013	0.022	0.028	0.035	0.144
$\mathcal{FM}_{\{N,S,A\}}$	0.005	0.005	0.006	0.008	0.152
$\mathcal{FM}_{\{S,O,A\}}$	0.019	0.038	0.044	0.050	0.150
$\mathcal{FM}_{\{N,S,O,A\}}$	0.015	0.023	0.028	0.030	0.143

Results and Discussions

To compare the considered models using the presented metric, each model was simulated for $N_{sim} = 250$ times with $K = 1500$ time steps and using the sinusoidal reference signal. Then, each of the 250 time series of failure amplitudes of the manually designed model was compared to each of the 250 time series of failure amplitudes of a comparison model (the extracted *GFM*, *GP*, *LSTM*, Gaussian distribution, uniform distribution) using the distance metric Eq. (3.88). This results in a 250 × 250 distance matrix for each considered comparison model from which the *EMD* is calculated. The final distances between the original failure model and each comparison model are stated in Table 3.12.

As expected when reviewing the state of the art on failure modeling approaches, Gaussian and Uniform models are advantageous due to their simplicity but neglect time- and value correlations causing an overall reduced performance. The reason is twofold and can be seen, for instance, by comparing the failure amplitudes simulated by both models aiming at representing $\mathcal{FM}_{\{N\}}$, cf. Fig. 3.31. The original failure amplitudes include value-correlations and Outlier (cf. Fig. 3.25a). Neither the Gaussian distribution nor the uniform distribution are capable of representing correlations and therefore



(a) As the uniform distribution assumes equal probability for all failure amplitudes between the observed minimum and maximum, a more sever characteristics is represented. Moreover, time- and value-correlations are not described.

(b) The range of failure amplitudes represented by the Gaussian distribution is comparable to the original failure model but time- and value-correlations are missing.

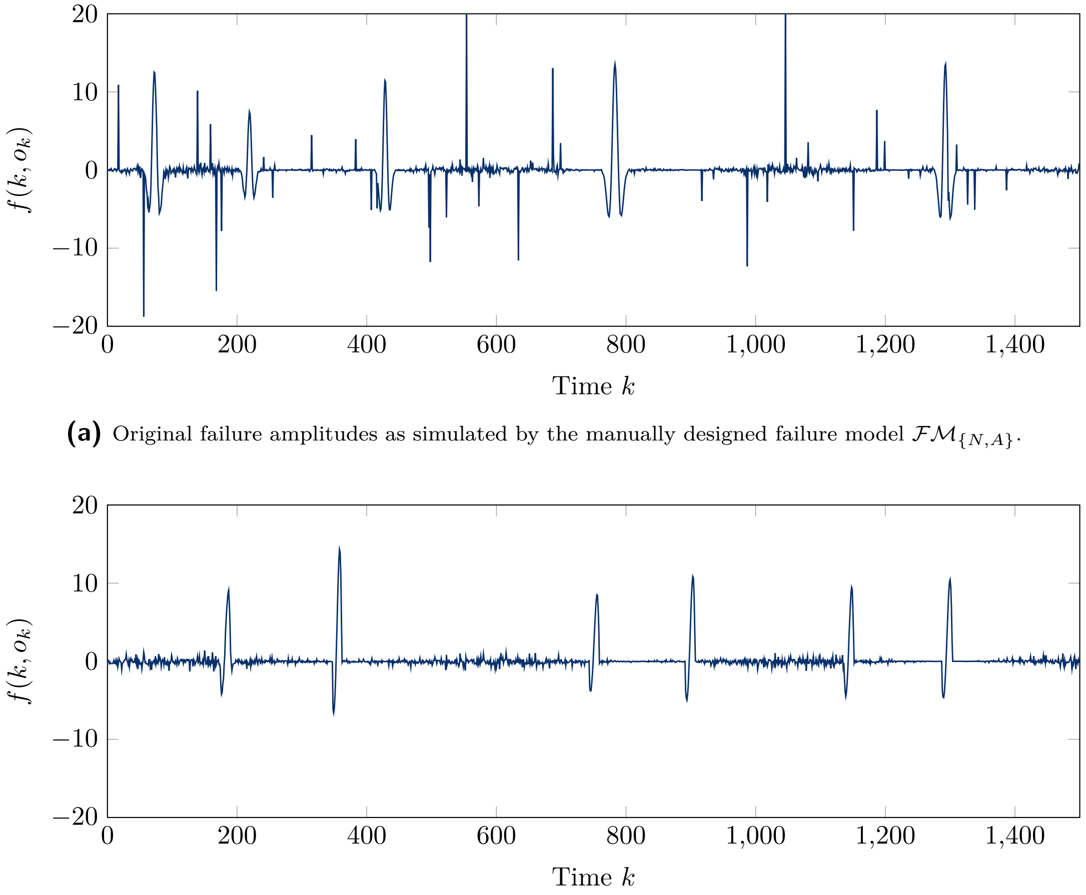
Fig. 3.31.: Simulation of $\hat{\mathcal{FM}}_{\{N\}}$ as represented by the Gaussian and Uniform failure model.

fail at simulating those. Moreover, in the case of the uniform distribution, equal occurrence probability for all failure amplitudes is presumed. Therefore, the Outliers of the original failure amplitudes cause the overestimation of the occurrence of high failure amplitudes. Consequently, the *MSE* between the simulations of the uniform distribution and the manually designed failure model is increased.

In contrast, the *GP* model does not suffer from these problems and achieves the second best results. For its representation of the failure model $\mathcal{FM}_{\{N,A\}}$, it obtains the best evaluation value. This is as the *GP* performs well when representing uncertainties. An example of original failure amplitudes is shown in Fig. 3.32a. As one can see, Noise prevails while short occurrences of the Artificial failure type introduce deterministic patterns only temporarily. In Fig. 3.33a one can see that the corresponding *GP* model successfully represents the Noise as uncertainty and succeeds in simulating Artificial-like patterns at the beginning of the time series. However, similar to the *GFM* shown in Fig. 3.32b, the *GP* model fails at modeling Outlier occurrences.

Another disadvantage of the *GP* model becomes apparent when comparing two consecutive simulations, cf. Fig. 3.33. The model directly depends on observed failure amplitudes and represents uncertainty as the remaining variance. By providing a failure amplitude for each reference value and each time step, the remaining variance is small. As a result, simulations do not show relevant variations. On the other hand, this could be counteracted by choosing appropriate kernel functions and kernel parameters.

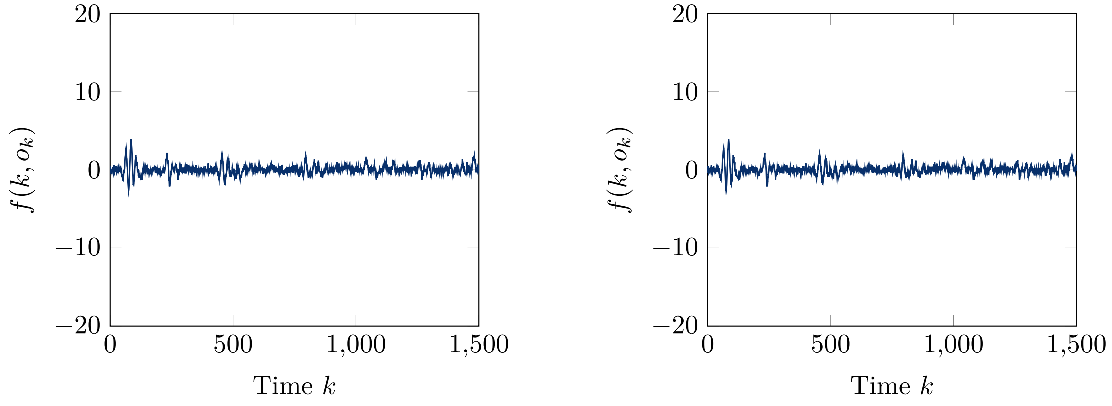
The *LSTM* experiences similar problems. It achieves acceptable performance values regarding the employed evaluation metric, but examining exemplary simulations reveals its shortcomings for modeling stochastics. As one can see in Fig. 3.34d, only a constant value is modeled. This is because the occurrence of specific failure amplitudes (the desired output of the network) does not directly depend on the time k and reference o_k, which are the inputs to the network. As discussed during the review of state-of-the-art models, this relation can be deterministic but is often reported to be stochastic. Therefore, correlations that can be learned by the network are not available. As a consequence, the model's parameters are adapted during training such that the error



(a) Original failure amplitudes as simulated by the manually designed failure model $\mathcal{FM}_{\{N,A\}}$.

(b) While Outlier failure amplitudes are not covered by the generated *GFM*, value-correlations of Noise occurrences are represented. Similarly, the Artificial failure pattern is appropriately approximated which results in a comparable range of failure amplitudes.

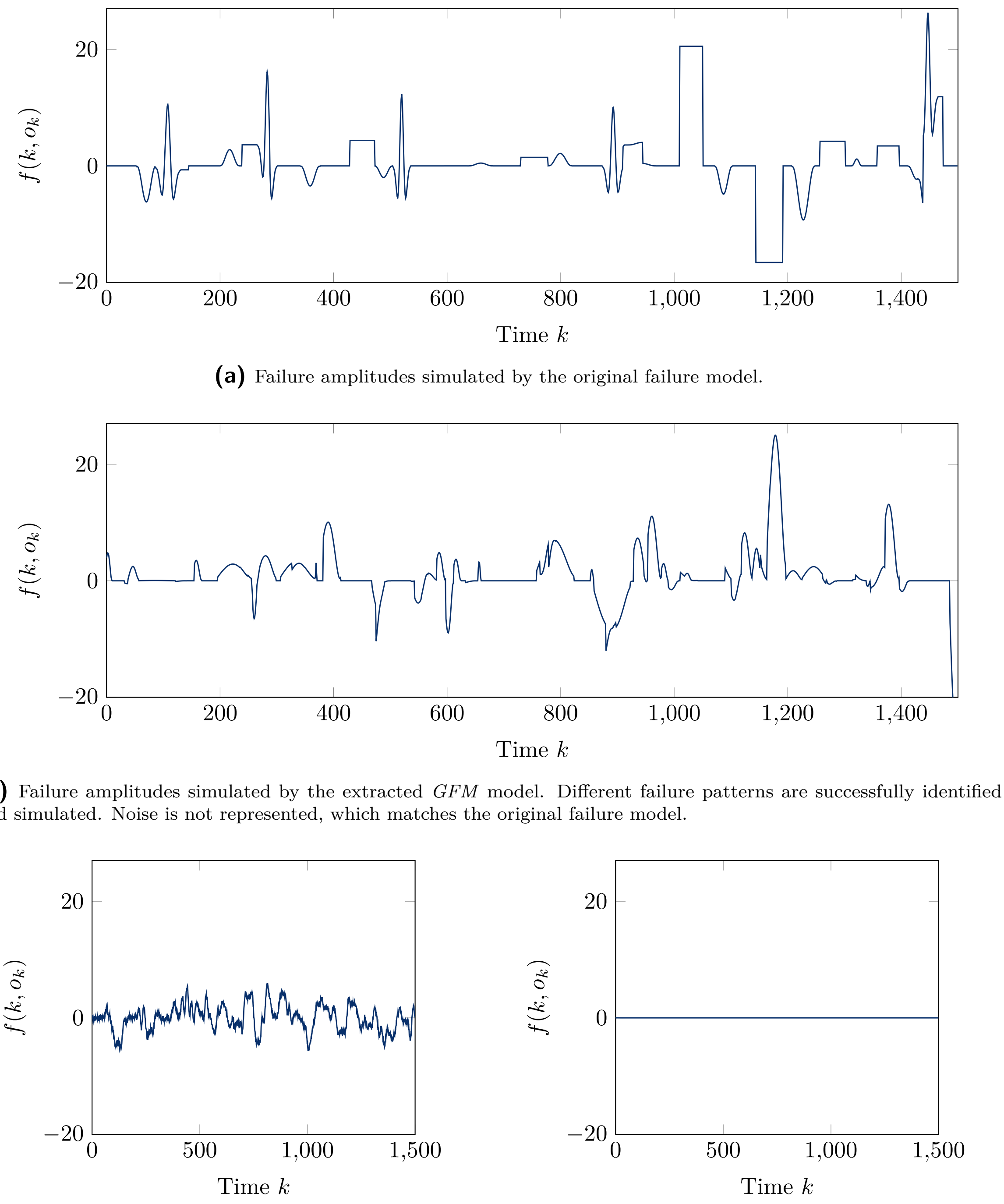
Fig. 3.32.: Comparing the model $\hat{\mathcal{FM}}_{\{N,A\}}$ represented as a *GFM* with the original failure amplitudes.



(a) Simulation one.

(b) Simulation two.

Fig. 3.33.: Two simulations of $\mathcal{FM}_{\{A\}}$ using the *GP* model. Only limited variations are observable between consecutive simulations as the remaining variance is limited due to the high number of training data points.



(a) Failure amplitudes simulated by the original failure model.

(b) Failure amplitudes simulated by the extracted *GFM* model. Different failure patterns are successfully identified and simulated. Noise is not represented, which matches the original failure model.

(c) Failure amplitudes simulated by the *GP* model. As remaining uncertainties are represented, Noise is introduced in simulations as well.

(d) Failure amplitudes simulated by the *LSTM* model. As correlations between the inputs and outputs are missing, only a constant is represented by the network.

Fig. 3.34.: Comparing simulations of $\mathcal{FM}_{\{S,O,A\}}$ obtained using the corresponding *GFM*, *GP*, and *LSTM* model with the original failure amplitudes.

is minimal on average, that is, a constant close to zero.

Opposed to the *LSTM*, the *GFM* successfully identifies the patterns and models their stochastic occurrences, cf. Fig. 3.34b. Although specific patterns, such as Offsets and the Artificial pattern, can not be identified perfectly, approximations that resemble the overall signal are modeled. Moreover, similar to the original failure amplitudes shown in Fig. 3.34a, the simulations provided by the extracted failure model do not exhibit Noise failures, cf. Fig. 3.34b. The *GP* model, on the other hand, does represent remaining uncertainties as a variance causing Noise failures to be simulated.

This underlines the overall evaluation given in Table 3.12. The *GP*, *LSTM*, and *GFM* models provide close approximations of the failure characteristics regarding the employed distance measure. However, only the *GFM* successfully represents variations in these failure characteristics and thereby achieves the overall best distance values.

It needs to be noted that a single parameterization was used to generate the failure models here. The performance of individual failure models could be improved by using parameterizations adjusted to the individual datasets.

Jäger *et al.* [31], for instance, identified the failure characteristics of a lane detection algorithm and compared the modeling performance of a *GFM* with an *LSTM* similarly to the evaluation presented here. They showed that in this use case the *LSTM* could provide similar performance to the employed *GFM* model.

3.5. Summary

In the endeavor of fulfilling Objective 1.2, this chapter introduced the concept of a *Generic Failure Model (GFM)*. According to the predefined criteria, it enables to represent failure characteristics of shared data unambiguously by using a mathematical representation. Building upon the idea that a failure model is nothing but a set of failure types, each of these is represented by a deterministic failure pattern and a scaling, activation, and deactivation distribution. To represent time- and value-correlations, each of these distributions comprises not only a normalized quantile function but also mean and standard deviation functions stating the distribution's dependency on the current time k and value o_k. Finally, each of the defined functions is approximated by a polynomial.

After a discussion of the fulfillment of the predefined criteria, where it was noted that the confidence criterion was yet to be fulfilled, the conversion of a *GFM* to an interval-based representation was introduced. This underlines not only the fulfillment of the comparability criterion but also lays the ground for the definition of a confidence measure. For that, a processing chain facilitating the automatic extraction of a *GFM* from given failure amplitudes was presented. In its last stage, a confidence measure was defined which converts the extracted *GFM* to an interval and measures the minimal distance between the original failure amplitudes and the interval borders. Thereby, the confidence states to what extent the observed failure characteristics is covered.

To evaluate both aspects, the *GFM* and the proposed processing chain, an artificial set of failure models were designed. Simulating these produced failure amplitudes from which a *GFM* could be extracted by means of the processing chain. Finally, the modeling performance of the *GFM* could be compared to state-of-the-art approaches.

The evaluation, therefore, showed that (i) due to its clarity, it is possible to manually design failure types, (ii) due to the processing chain, it is possible to automatically extract suitable failure models from given failure amplitudes, (iii) and the modeling performance of the *GFM* is advantageous to state-of-the-art models when it comes to representing stochastic signals with limited correlations to independent variables.

4. Region of Safety

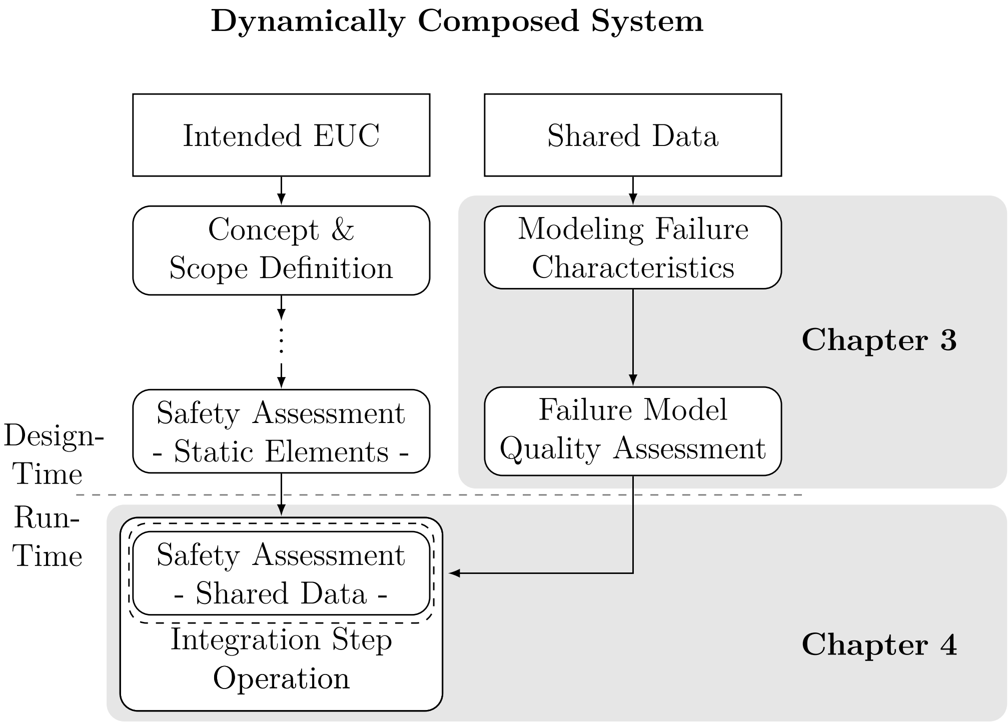

Dynamically Composed System
Intended EUC
Shared Data
Concept & Scope Definition
Modeling Failure Characteristics
Chapter 3
Safety Assessment - Static Elements -
Failure Model Quality Assessment
Design-Time
Run-Time
Safety Assessment - Shared Data -
Integration Step Operation
Chapter 4

Fig. 4.1.: Simplified safety process from Fig. 1.9 showing the components addressed in this chapter.

The definition of the *GFM* addressed Objective 1.2 and is, therefore, a mandatory prerequisite in this work's overall goal to guarantee the safety of a dynamically composed system when using shared data, cf. Fig. 4.1. In this endeavor, the focus of this chapter is on fulfilling Objective 1.1.

Taking the criteria defined in Section 1.3.1 into account, an approach capable of analyzing a failure model of shared data at a system's run-time to provide guarantees on whether or not a safety function adheres to its required performance when using shared data is in question. Moreover, the approach has to be designed in such a way that it can be shown at design-time that the decision provided at run-time is functionally correct.

In Section 2.1.3 it was noted that these approaches can be found at a functional level but are missing for a technical level. At this level, the approach of *Region of Attraction (RoA)* is most promising as it was shown in the literature to provide safety guarantees during reinforcement learning. Thereby, it fulfills the criteria of run-time certification and functional correctness. Building upon the concept's ability to be executed at a system's run-time, as it was shown in [45] already, allows focusing on the central criterion of assessing failure models of shared data, which is not satisfied yet.

On the one hand, this is because general system models do neither assume shared data nor failures thereof. On the other hand, the requirements on *RoA* are overly strict and can not be met when dealing with uncertainties[1] such as failures of shared data.
Thus, this chapter builds upon the idea of *RoA* and extends it to the concept of *Region of Safety (RoS)*. For that, Section 4.1 introduces an extended system model that facilitates specifying versatile sources of uncertainty and most importantly failures of shared data. This enables taking these into account during *RoA* estimation as well. A corresponding evaluation at the beginning of Section 4.2 shows the shortcomings of *RoA* when handling uncertainties and motivates that its overly restrictive requirements on stability are not necessary for guaranteeing safety. Therefore the section extends the concept and introduces *RoS* where the focus is shifted from stability to safety. To illustrate the approach, the example of the inverted pendulum is introduced and used for evaluating the concept in Section 4.3. The chapter is concluded with a summary in Section 4.4.

4.1. Explicitly Modeling Uncertainties for Dynamically Composed Systems

Common approaches to estimating *RoA* do not consider uncertainties [98]. Neither regarding the system model nor regarding obtained and processed (sensory) data.
To be applicable to safe reinforcement learning, Berkenkamp *et al.* [45] assume an initial system model which is extended with a model of uncertainty U_{model}, cf. Eq. (2.6). The idea is that a design-time model neglects or oversimplifies real-world influences. This results in differences between the behavior of the controlled system in the real-world and the model used by the controller. Learning these differences and modeling them using U_{model} at run-time enables representing these uncertainties.
However, they are presumed to be deterministic (for instance, result from linearizing the system model). In contrast, failures of shared data are presumably stochastic, which contradicts this modeling approach. Moreover, shared data are only one source of uncertainty. Others, for instance failures of internal sensors or disturbances of the system's environment, have to be taken into consideration during a safety assessment as well. More specifically, each of these sources has to be represented independently to allow evaluating different combinations. In the end, the combination of a failure model of shared data with the remaining sources of uncertainty has to be evaluated during a run-time safety assessment. Therefore, in this section an extended system model that separately addresses different sources of uncertainty is presented.

[1] The term "uncertainty" is used here to address the general concept of being not certain about a value or parameter. Therefore, failures of sensors or shared data are assumed to be one aspect of uncertainty, but disturbances originating in a system's environment or an actuator's inability to execute a given command precisely are other aspects as well.

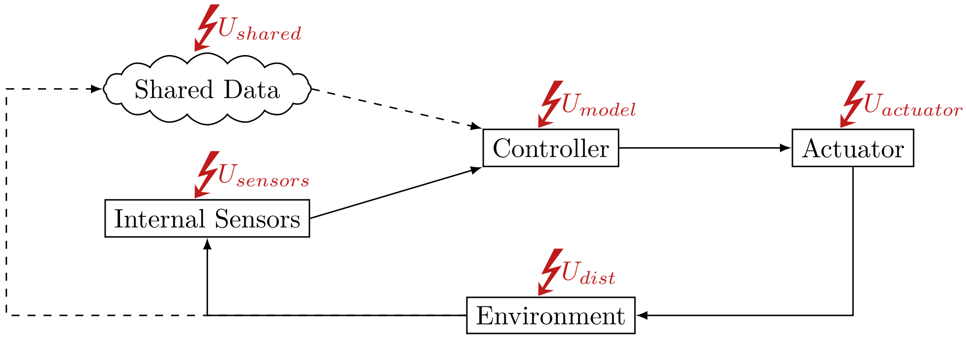


Fig. 4.2.: Defining sources of uncertainty using an abstract control loop.

In that endeavor, the abstract control loop is considered again and five sources of uncertainty are allocated to its elements, cf. Fig. 4.2. They are mathematically represented within the extended system model in Eqs. (4.1) to (4.7).

$$\dot{\mathbf{x}}(t) = f^*_\pi(\mathbf{x}(t)) = f(\mathbf{x}(t), \hat{\mathbf{u}}(t)) + U_{model}(\mathbf{x}(t), \hat{\mathbf{u}}(t), t) + U_{dist}(\mathbf{x}(t), t) \tag{4.1}$$

$$\hat{\mathbf{u}}(t) = U_{actuator}(\mathbf{u}(t), t) \tag{4.2}$$

$$\mathbf{u}(t) = \pi(\hat{\mathbf{o}}_i(t), \hat{\mathbf{o}}_s(t)) \tag{4.3}$$

$$\hat{\mathbf{o}}_i(t) = \mathbf{o}_i(t) + U_{sensor}(\mathbf{o}_i(t), t) \tag{4.4}$$

$$\mathbf{o}_i(t) = s_i(\mathbf{x}(t)) \tag{4.5}$$

$$\hat{\mathbf{o}}_s(t) = \mathbf{o}_s(t) + U_{shared}(\mathbf{o}_s(t), t) \tag{4.6}$$

$$\mathbf{o}_s(t) = s_s(\mathbf{x}(t)) \tag{4.7}$$

In its center it is assumed that a system is given as a system of *ODEs*, that is $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$. It describes the state change $\dot{\mathbf{x}}$ over time given the control actions $\mathbf{u} \in \mathbb{R}^m$ for each state $\mathbf{x} \in \mathbb{R}^n$. As described, for instance, by Berkenkamp *et al.* [45] this deterministic model may not represent the real world due to uncertainties regarding its parameters or due to linearizing it. Therefore, the actual state change $\dot{\mathbf{x}}$ is not only provided by the system model but is also imposed by model uncertainties U_{model}. They describe the difference between the assumption the controller makes on the system under control and the actual real-world system.

Next to model uncertainties, disturbances U_{dist} originated in the environment of a system may introduce uncertainty regarding the actual state change. Consider, for instance, frontal gusts at stormy weather which influence the velocity of a vehicle or surface conditions that change the friction of wheels on the ground. Such uncertainties may depend on the current state of the system but can be of random nature as well. Opposed to model uncertainties, however, they are external to the system, cf. Fig. 4.2.

While both, the model uncertainty U_{model} and disturbances U_{dist}, directly influence the state change (cf. Eq. (4.1)), actuator faults cf. Eq. (4.2) are posing an indirect source of uncertainty. For instance, actuator faults may cause a loss of effectiveness, stuck failures, or complete failure [99], [100]. Another example of actuator uncertainties can be saturation. It describes an actuator's inability to follow the target action $\mathbf{u}$ due to physical constraints. The set of failures affecting a system's actuators can be modeled using $U_{actuator}$, which causes the control action $\mathbf{u}$ to be transformed into $\hat{\mathbf{u}}$, cf. Eq. (4.2).

The initial control actions $\mathbf{u}$ in turn are provided by the system's control policy. This policy evaluates provided observations to determine an appropriate action $\mathbf{u}$, cf. Eq. (4.3).

For dynamically composed systems, these observations may be provided by internal sensors s_i or shared data s_s. While internal sensors provide observations $\mathbf{o}_i$, shared data provides external observations $\mathbf{o}_s$. From a modeling perspective, both of which are observations of the current state $\mathbf{x}(t)$, cf. Eqs. (4.5) and (4.7). Therefore, shared data may be considered as an external sensor to a system as well.

As discussed in Chapter 2, observations provided by internal sensors and shared data, are subject to failures. Therefore, the theoretically correct observations $\mathbf{o}_i$ and $\mathbf{o}_s$ are imposed by failures $U_{sensor}(\mathbf{o}_i(t), t)$ and $U_{shared}(\mathbf{o}_i(t), t)$. These can be time- and value-correlated.

As sensors and shared data provide observations on which the controller can calculate control actions, the control loop is closed.

In summary, five sources of uncertainty (U_{model}, U_{dist}, $U_{actuator}$, U_{sensor}, and U_{shared}) are identified and can be addressed separately in the given system model. To reflect their stochastic nature, they are considered to be random variables. A consequence of this assumption is that the differential equation becomes a *Stochastic Differential Equation (SDE)*[101], that is, the state change $\dot{\mathbf{x}}$ is a random variable itself.

4.2. Regions of Safety for Dynamically Composed Systems

The extended system model presented in the last section enables not only to apply the approach of *RoA* for analyzing the effect of failures of shared data on a controller's stability but also the effect of other sources of uncertainty.

For assessing the stability of a controller, the approach of *RoA* builds on Lemma 2.1. It states that, given a suitable *Control Lyapunov Function (CLF)*, the *RoA* is a set of states for which a controller is guaranteed to provide stabilizing control actions that drive the system under consideration closer to the targeted stability point. This guarantee of asymptotic stability requires that the action calculated by a controller always minimizes the value of the *CLF*. However, this requirement may not be met by a system under consideration in presence of uncertainty.

To exemplify this, the next subsection firstly introduces the use of intervals to enable estimating a controller's *RoA* when dealing with uncertainty before the methodology is applied to the inverted pendulum problem where only sensor failures are considered. It can be observed that even minor uncertainties render the requirement of an *RoA* to be unfulfilled. Through simulation, on the other hand, it can be seen that the system remains at safe states nonetheless. Motivated by that, Section 4.2.2 extends the idea of *RoA* and introduces the concept of *RoS*. This concept relaxes the initial requirement on stability and focuses on showing that a controller only chooses actions such that the system under consideration does not leave a set of safe states, an *RoS*. Finally, the last subsection discusses how this approach addresses the criteria defined in Section 1.3.1 and thereby fulfills Objective 1.1.

4.2.1. Estimating Regions of Attraction in Presence of Uncertainty

The goal of this subsection is to examine the effect uncertainties have on estimating a controller's *RoA*. For that, Lemma 2.1 is to be applied to the problem of the inverted pendulum. However, as the uncertainties are represented by random variables, the state change $\dot{\mathbf{x}}$ predicted by the system model is a random variable itself. As this causes the gradient of the *CLF* ($\dot{V}(\mathbf{x})$) to be a random variable as well, Eq. (2.5) (requirement of *RoA*) can not be evaluated directly. It is an inequality comparing a scalar with a threshold.

Consequently, the next paragraph briefly describes how the random variable $\dot{V}(\mathbf{x})$ is converted to an interval and used to apply the concept of *RoA* in presence of uncertainty. Having this prerequisite in place, the problem of the inverted pendulum is introduced before the *RoA* of its controller is determined and discussed. The effect of uncertainties on the resulting *RoA* is analyzed and compared to simulations of the system.

Using Intervals for Estimating Regions of Attraction

The introduced system model $f^*_\pi(\mathbf{x})$ defined the random variables U_{model}, U_{dist}, $U_{actuator}$, U_{sensor}, and U_{shared} for representing different sources of uncertainty, cf. Eqs. (4.1) to (4.7). According to this changed system model, the central requirement for *RoA* stated in Eq. (2.5) is updated to Eq. (4.8).

$$\dot{V}(\mathbf{x}) = \frac{\partial V(\mathbf{x})}{\partial t} = \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}}{\partial t} = \frac{\partial V(\mathbf{x})}{\partial \mathbf{x}} \cdot f^*_\pi(\mathbf{x}) < -L_{\dot{V}} \cdot \tau \tag{4.8}$$

Here it can be seen that if $\partial\mathbf{x}/\partial t$ becomes a random variable, $\dot{V}(\mathbf{x})$ becomes a random variable as well. As a consequence, the requirement of Eq. (4.8) may hold with a certain probability, which is given by the gradients *Cumulative Distribution Function (CDF)*. Therefore, Eq. (4.8) can be rewritten as follows.

$$P(\dot{V}(\mathbf{x}) < -L_{\dot{V}} \cdot \tau) \geq P_{RoA} \tag{4.9}$$

The probability of the gradient $\dot{V}(\mathbf{x})$ being less than the threshold $-L_{\dot{V}} \cdot \tau$ has to exceed a predefined probability of P_{RoA}. It essentially states how certain one can be about the controller being able to minimize the *CLF* at state $\mathbf{x}$. This directly relates to the likeliness of the controller being able to stabilize the system, that is, driving it closer to the stability point. Considering this as the safety function to be carried out by the controller, P_{RoA} reflects the safety performance the controller is required to provide. Therefore, the value has to be set in accordance with applicable safety standards, for instance the IEC 61508 and its *SILs*, [20].

However, as noted in Challenge 1.3, specifying the required safety performance of a component using shared data is challenging due to the missing knowledge about the failure characteristics at design-time. Although being possibly overly restrictive, one can take up on the argumentation in Section 1.3 and require the highest safety performance. This results in $P_{RoA} = 1$ and entails that all values of the distribution of $\dot{V}(\mathbf{x})$ have to be less than the threshold. This is used to simplify Eq. (4.9) and derive Eq. (4.10).

$$\max(\dot{V}(\mathbf{x})) < -L_{\dot{V}} \cdot \tau \tag{4.10}$$

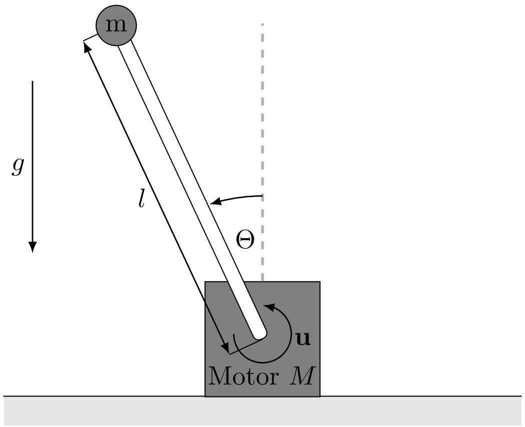


Fig. 4.3.: Schematic representation of the inverted pendulum system.

Instead of knowing about the distribution of $\dot{V}(\mathbf{x})$ it is sufficient to determine its maximal value, which is the upper bound of the distribution's interval of values ($\dot{V}(\mathbf{x}) \in \mathcal{I}_{\dot{V}(\mathbf{x})} = [\min(\dot{V}(\mathbf{x})), \max(\dot{V}(\mathbf{x}))]$). Thus, evaluating the fulfillment of Eq. (4.10) at a given state $\mathbf{x}$ requires to determine $\mathcal{I}_{\dot{V}(\mathbf{x})}$.

This means that not the entire distribution has to be determined but only the interval from which the maximal value will be known. On the one hand, this simplifies the calculation. On the other hand, this means that the worst-case uncertainties are always assumed.

In the endeavor of determining the interval, two options are available. Firstly, interval arithmetic can be used, [102]. Starting with the sources of uncertainties from which the initial intervals have to be extracted (e.g. confidence intervals of the specified distributions), interval arithmetic can be used to track them through the system model and ultimately provide $\mathcal{I}_{\dot{V}(\mathbf{x})}$. While this is computationally inexpensive as it requires to evaluate the system model at state $\mathbf{x}$ only once, the *dependency problem* [103] of interval arithmetic causes $\mathcal{I}_{\dot{V}(\mathbf{x})}$ to overestimate the actual range of $\dot{V}(\mathbf{x})$. As a consequence, Eq. (4.10) might be incorrectly evaluated as unfulfilled.

A second option is a sampling-based approach, where the initial intervals are tracked through the system model by evaluating it at different points. This work uses a mixture of both. While the initial intervals of failure amplitudes of sensory and shared data are tracked to obtain the interval of state changes using a sampling-based method (cf. Eq. (4.1)), the final interval of $\mathcal{I}_{\dot{V}(\mathbf{x})}$ is obtained through interval arithmetic (cf. Eq. (4.10)). In that way, computational efficiency and the effect of the dependency problem are balanced.

The Inverted Pendulum

With the adaptation of the previous paragraph, systems having different sources of uncertainty can be analyzed and their *RoA* can be estimated. In the endeavor of examining the effect of uncertainties on the estimated *RoA*, the inverted pendulum is considered here, cf. Fig. 4.3. It is a recurring example in control theory that appears in different versions in the literature, e.g. mounted on a mobile cart [104]–[106] or a quadrotor aerial vehicle [107].

This work assumes a simplified version where a pole of length l with mass m is mounted on a motor which allows applying a torque $\mathbf{u}$ to the pole. The control goal for π is

Tab. 4.1.: Parameters of the inverted pendulum.

Parameter	Value	Description
l	$0.5\,\mathrm{m}$	Length of the pole
m	$0.15\,\mathrm{kg}$	Mass of the pendulum located at the poles end
$\mu_{friction}$	$0.05\,\mathrm{N}$	Friction at the poles mounting point
g	$9.81\,\mathrm{m\,s^{-2}}$	Standard gravity constant
$p_{pendulum}$	5.0	P-Gain of the P-controller

to balance the pole in an upright position for which a maximal deviation of $|\Theta|\leq 0.75\,\mathrm{rad}$ has to be maintained. This top-level safety goal stems from the limitation of the considered motor, which can exert a maximal torque sufficient to counteract gravitational forces only for those angles. Greater angles will therefore result in the pendulum falling, which has to be prevented. The dynamics of the system is given by Eq. (4.11) and its parameters are listed in Table 4.1.

$$\ddot{\Theta}(t) = \frac{mglsin(\Theta(t)) - \mu\dot{\Theta}(t) + u(t)}{ml^2} \tag{4.11}$$

Eq. (4.11) states the acceleration of the pendulum's angle depending on the current control input $u(t) = \pi(\mathbf{x}(t))$ where $\mathbf{x} = [\Theta\ \dot{\Theta}]^T \in \mathcal{X} \subseteq \mathbb{R}^2$ is the current state. The control input is provided according to Eq. (4.12).

$$\pi(x) = \pi(\Theta, \dot{\Theta}) = -p_{pendulum} \cdot (\Theta + \dot{\Theta}) \tag{4.12}$$

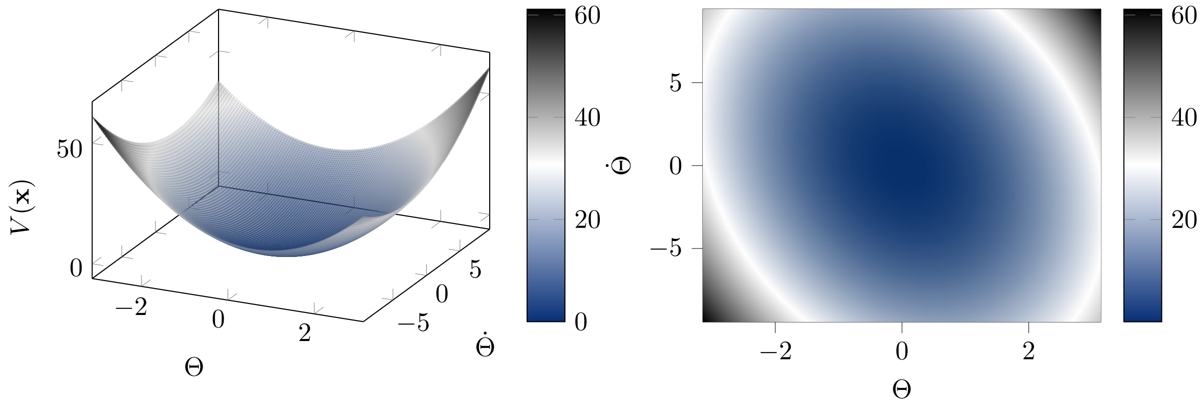
Although more complex control strategies are possible, a P-controller is considered here for the sake of simplicity. The controller reacts linearly to the sum of the pendulum's angle and angular velocity. It generates the effect that the calculated control action takes the current movement of the pendulum into account. For instance, a negative angle is counteracted by an already positive velocity, which means that the pendulum's movement towards the stability point is respected for calculating the corresponding control action. Conversely, a negative angle paired with a negative angular velocity indicates that the displacement is increasing and therefore increases the reaction of the controller.

The control action provided by the controller is, however, imposed by the actuator failure model, cf. Eq. (4.2). Here, it describes the motor's inability to exert a torque greater than 0.5 N, cf. Eq. (4.13).

$$U_{actuator}(u(t), t) = \max(\min(u(t), 0.5), -0.5) \tag{4.13}$$

This inability is modeled as a saturation and limits $\hat{u}(t) \in [-0.5\,\mathrm{N}, 0.5\,\mathrm{N}]$.

Apart from the actuator failure model, no model uncertainties or disturbances are considered. For the sake of this example, only observation failures of internal sensors, that is, U_{sensor} are considered. As their effect on calculating an *RoA* is the subject of the next paragraph, they are not defined here.



(a) 3D plot of $V_{pendulum}(x)$.

(b) Heatmap of $V_{pendulum}(x)$

Fig. 4.4.: Visualizing the *CLF* for estimating the *RoA* of the inverted pendulum controller.

However, for determining an *RoA*, a suitable *CLF* is missing. Although multiple candidates will be evaluated in Section 4.3, a first candidate is introduced here already to motivate the discussion, cf. Eq. (4.14).

$$V_{pendulum}(x) = w_{\Theta} \cdot \Theta^2 + w_{comb} \cdot (\Theta + \dot{\Theta})^2 \tag{4.14}$$

Following the idea of a cost function, the squared value of Θ is considered to form a function that is monotonically increasing in each direction from the global minimum at $\mathbf{x} = 0$. Additionally, the squared of the additive combination of the angle Θ and the pendulum's angular velocity $\dot{\Theta}$ is considered again. Similar to its use for defining the control policy, the combination is used here to assess the displacement combined with the moving direction. A moving direction bringing the pendulum closer to the origin is favorable over a moving direction increasing the displacement. The final *CLF* function $V_{pendulum}(x)$ is displayed in Fig. 4.4.

The Effect of Uncertainty on Estimating Regions of Attraction

The previously introduced problem of the inverted pendulum shall be used to exemplify the effect of considering uncertainties during estimating a controller's *RoA*. In this endeavor, this subsection firstly defines the sensor failures as the only source of uncertainty considered here. Secondly, the approach of *RoA* is applied and the results are discussed.

Although the extended system model presented in Section 4.1 facilitates defining a failure model for shared data, only internal sensors that fully observe the state space ($s_i(\mathbf{x}) = \mathbf{x}$) are assumed for this scenario. Thus, the model of shared data provides an empty set of observations, that is, $s_s(\mathbf{x}) = \emptyset$, cf. Eqs. (4.5) and (4.7). Moreover, it is assumed that only observations provided by internal sensors are subject to uncertainties. For the sake of simplicity, it is assumed that the Noise failure type described in Section 3.4.1 models the failure characteristics of these. The failure type is extended to two-dimensional data by presuming independence between both dimensions, cf. Eq. (4.15).

$$U_{sensor_N} \sim Q_{Noise}\left(\mu = \mathbf{0}_{2\times 1}, \Sigma = \mathbf{I}_{2\times 2} \cdot 0.0125\right) \tag{4.15}$$

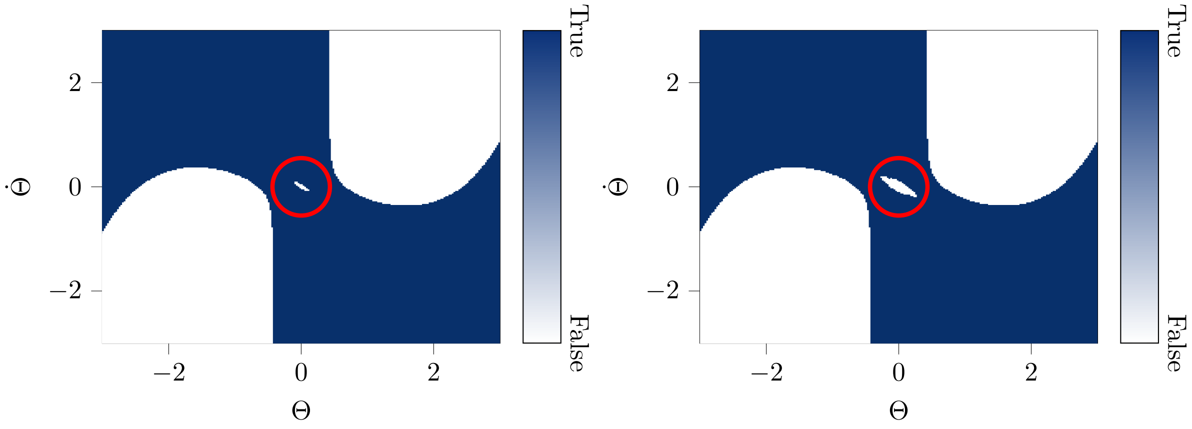
Tab. 4.2.: Definition of the state space considered for the inverted pendulum.

Parameter	Interval	Discretization
Θ	$[-\pi\,\mathrm{rad}, \pi\,\mathrm{rad}]$	$\Theta_S = 0.025\,\mathrm{rad}$
$\dot{\Theta}$	$[-3\pi\,\mathrm{rad\,s^{-1}}, 3\pi\,\mathrm{rad\,s^{-1}}]$	$\dot{\Theta}_S = 0.025\,\mathrm{rad\,s^{-1}}$
t	$[0\,\mathrm{s}, 10\,\mathrm{s}]$	$\tau = 0.025\,\mathrm{s}$

Note that, in contrast to the one-dimensional case, no time- or value-correlation is assumed here.

Having the uncertainties affecting the system defined, the corresponding *RoA* can be estimated. For that, the state space and its discretization as listed in Table 4.2 are assumed. While the range of Θ from $-\pi$ rad to π rad fully describes its possible values, the limits of the angular velocity $\dot{\Theta}$ are estimated from the system configuration. Using an energy-based approach, the maximal velocity of the pendulum during falling from $\Theta = 0$ to $\Theta = -\pi$ can be calculated. To ensure coverage of all relevant states, this velocity is rounded up to $\pm 3\pi\,\mathrm{rad\,s^{-1}}$. Finally, for simulating the pendulum and converting the employed *GFMs* to intervals a time horizon of $T = 10\,\mathrm{s}$ is considered. All three dimensions (Θ,$\dot{\Theta}$, and t) are discretized using 0.025 which balances computational effort with the resolution of the state space.

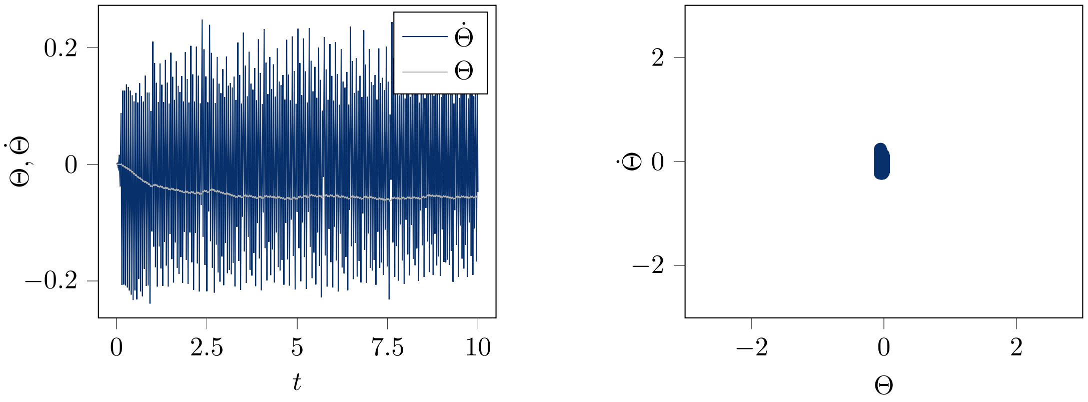
For each state $\mathbf{x}$, intervals of the considered sensor failure model are extracted using the approach of Section 3.2. The intervals cover the time horizon T as well as the value domain $\mathcal{I}_{\mathbf{x}} = [\mathbf{x} - 0.0125, \mathbf{x} + 0.0125]$. Therefore, by considering all states over the considered state space, the extracted intervals cover the continuous range of states. Based on these intervals, the maximal gradient $\max(\dot{V}(\mathbf{x}))$ is calculated and evaluated according to Eq. (4.10).



(a) Considering the ideal case of no uncertainties.

(b) Considering Noise failures affecting sensor observations of Θ and $\dot{\Theta}$.

Fig. 4.5.: Categorizing states of the inverted pendulum problem according to their fulfillment of Eq. (4.10).



(a) Visualization of Θ and $\dot{\Theta}$ over time t.

(b) States of the inverted pendulum during simulation.

Fig. 4.6.: Simulation of inverted pendulum with Noise sensor failure.

Whether or not a state fulfills the requirement is illustrated in Fig. 4.5. White areas correspond to states violating the requirement and blue areas indicate the opposite. While Fig. 4.5a considers the ideal case, that is, no uncertainties affecting the inverted pendulum, Fig. 4.5b uses Eq. (4.15) to model sensor failures.

In general, one can see that, according to the estimated *RoA*, the controller is able to stabilize the pendulum for angles of approx. $\Theta = \pm 0.5\,\text{rad}$. In case the angular velocity counteracts the displacement, greater angles are possible as well. In contrast, at states at which the angular velocity increases the displacement, stabilization can not be provided and the corresponding states are colored white.

Thus, as blue states surround the stability point at $\mathbf{x} = [0\ 0]^T$, the area is eligible to be an *RoA*. However, as one can see in Fig. 4.5a already, only seven states (highlighted by the red circle) centered at the stability point do not fulfill Eq. (4.10). Therefore, Lemma 2.1, which requires all states of an *RoA* to fulfill Eq. (4.10), can not be satisfied and no *RoA* can be estimated. Similarly, when considering Noise failures in Fig. 4.5b, the region of states not fulfilling Eq. (4.10) increases and thereby prevents estimating a valid *RoA* as well.

In both cases, the states for which Eq. (4.10) is not satisfied entail that asymptotic stability can not be guaranteed, that is, that the pendulum is always moving towards the stability point.

This, however, is not required for safety. Instead, the system needs to be guaranteed to not enter states violating safety requirements. In the case of the pendulum, this means that states with $|\Theta| \geq 0.75\,\text{rad}$ should not be entered. This requirement differs from asymptotic stability. While asymptotic stability requires that the system always moves towards the stability point (also when in safe states), safety requires only that a set of safe states is not left.

In fact, simulating the inverted pendulum underlines this assumption, cf. Fig. 4.6. In Fig. 4.6a one can see that the angle stabilizes at approx. $\Theta = -0.05\,\text{rad}$. Moreover, the pendulum does not leave the region of $\Theta \in [-0.063\,\text{rad}, 0\,\text{rad}]$ and $\dot{\Theta} \in [-0.24\,\text{rad}\,\text{s}^{-1}, 0.25\,\text{rad}\,\text{s}^{-1}]$, which can also be seen in Fig. 4.6b. Therefore, the inverted pendulum does not to leave the set of safe states and consequently does not violate its safety requirement despite its controller being confronted with observation failures.

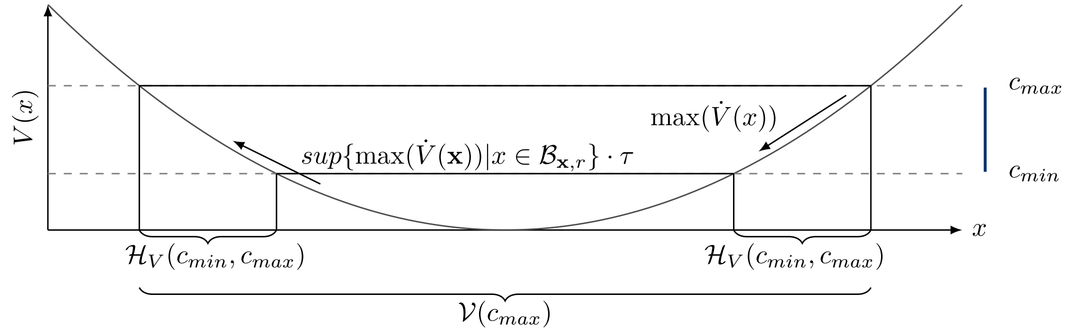

$V(x)$
$\max(\dot{V}(x))$
$sup\{\max(\dot{V}(\mathbf{x}))|x \in \mathcal{B}_{\mathbf{x},r}\} \cdot \tau$
c_{max}
c_{min}
x
$\mathcal{H}_V(c_{min}, c_{max})$
$\mathcal{H}_V(c_{min}, c_{max})$
$\mathcal{V}(c_{max})$

Fig. 4.7.: Schematic representation of requirements for hull states and uncertain states comprising a *Region of Safety (RoS)*.

4.2.2. Introducing the Concept of Region of Safety

The previous subsection showed that the requirement of asymptotic stability as posed by *RoA* is overly restrictive when it comes to safety. Instead, the idea of showing that a system will not leave its set of safe states emerged. In the endeavor of formalizing this idea to address Objective 1.1, that is, to propose a run-time safety assessment method satisfying the predefined criteria of Section 1.3.1, the next paragraph builds on Lemma 2.1 and introduces the concept of *RoS*. Afterward, an algorithmic approach for estimating a controller's *RoS* is presented.

The Theorem of Region of Safety

Simulating the inverted pendulum indicated that asymptotic stability is not required to show that a system remains within a safe set of states. Instead, it is sufficient to guarantee that the controller emits actions driving the system into the inner region of the safe set for states at its border.

In other words, it needs to be shown that the controller will always choose a control action such that the system remains in the safe set when it is close to leaving it and that the system will not leave the safe set without passing such stabilizing states. These two requirements are formalized in the following theorem, which builds upon the idea of *RoA*. An initial version of the theorem was stated in Jäger *et al.* [32].

Theorem 4.1 ***(Region of Safety)*** *Let $f^*_\pi(\mathbf{x})$ be a controlled, stochastic system according to Eq.* (4.1). *Let $V(\mathbf{x})$ be the system's Control Lyapunov Function (CLF) defined on the state space $\mathcal{X}$ and $L_{\dot{V}}$ the Lipschitz constant of its gradient function. Furthermore, let $\mathcal{B}_{\mathbf{x},r} = \{p \in \mathcal{X} | d(\mathbf{x}, p) \leq r\}$ be the local neighborhood of a state $\mathbf{x}$ with respect to a distance metric $d(\mathbf{x}, p)$ and distance $r = \max(\{x_s \in \mathbf{X}_s\})$ on a discrete state space $\mathcal{X}_\tau \subseteq \mathbb{R}^m$ with discretization $\mathbf{X}_S \in \mathbb{R}^m_{>0}$. With the bounded level set $\mathcal{H}_V(c_{min}, c_{max}) = \{\mathbf{x} \in \mathcal{X}_\tau | V(\mathbf{x}) \leq c_{max} \wedge V(\mathbf{x}) \geq c_{min}\}$ defined on the discrete state space, a system is said to be safe according to the specified CLF for all states $\mathbf{x} \in \mathcal{V}(c_{max}) = \{\mathbf{x} \in \mathcal{X} | V(\mathbf{x}) \leq c_{max}\}$ and an initial state $\mathbf{x}_0 \in \mathcal{V}(c_{max})$ if*

$$\max(\dot{V}(\mathbf{x})) < -L_{\dot{V}} \cdot \tau \quad \forall \mathbf{x} \in \mathcal{H}_V(c_{min}, c_{max}) \tag{4.16}$$

$$V(\mathbf{x}) + sup\{\max(\dot{V}(\mathbf{x}))|x \in \mathcal{B}_{\mathbf{x},r}\} \cdot \tau < c_{max} \quad \forall \mathbf{x} \in \{\mathbf{x} \in \mathcal{X}_\tau | V(\mathbf{x}) < c_{min}\} \tag{4.17}$$

As $V(\mathbf{x})$ is guaranteed to be differentiable due to its definition (cf. Section 2.1.2), $L_{\dot{V}}$ exist. Based on that the theorem formalizes the previously linguistically expressed requirements by defining Eq. (4.16) and Eq. (4.17). Fig. 4.7 visualizes the theorem.

Eq. (4.16) adopts the condition of Eq. (4.10) for guaranteeing asymptotic stability of the system for states surrounding the stability point. Thus, this condition is called the *RoA condition* in the following. Note that not all but only states in $\mathcal{H}_V(c_{min}, c_{max})$ have to satisfy the condition. These states are also referred to as *stabilizing states* and are colored blue (if not stated otherwise) in diagrams, for instance as in Fig. 4.5. The existence of these states facilitates guaranteeing that the controller chooses appropriate actions $\mathbf{u}$ to drive the system into the center of the region towards the stability point. Therefore, they prevent the system from leaving the *RoS*.

As it was shown in the previous section, however, this condition can not be fulfilled by states close to the stability point when considering uncertainties. For those states, that is, states having a *CLF* value of $V(\mathbf{x}) < c_{min}$, stability is uncertain, which is why they are also referred to as *uncertain states.* While it can not be shown that the controller will drive the system closer to its stability point for those states, it can be shown that the controller mitigates the effect of uncertainties and that the system evolves to stabilizing states at maximum. This is formalized in Eq. (4.17)[2]. It leverages the maximal gradient of the *CLF* to guarantee that its function value will be less than $V(\mathbf{x}) < c_{max}$ after the time horizon τ.

For that, the supremum is determined locally, that is, on the neighborhood $\mathcal{B}_{\mathbf{x},r}$ of state $\mathbf{x}$ according to the distance metric d. It, therefore, encodes the locally expected uncertainties and the resulting maximal gradient. Following this gradient over time τ guarantees that the system either remains in the region of uncertain states or is driven to the region of stabilizing states (indicated by the end of the arrow in Fig. 4.7). As it is guaranteed that the system will be driven towards the stability point at those states, it is guaranteed that the system does not leave its *Region of Safety (RoS).*

As a consequence of both conditions, assuming a system starts with a state $\mathbf{x}_0 \in \mathcal{V}(c_{max})$, it may alternate between stabilizing and uncertain states, but is guaranteed to never leave the set $\mathcal{V}(c_{max})$.

Another consequence of the theorem is that it essentially divides the state space into safe and unsafe states by defining c_{max} according to a *CLF*. From a safety perspective, the *CLF* can therefore be interpreted as an abstract measure of risk. It evaluates a given state $\mathbf{x}$ and assigns an abstract risk value which increases towards states that violate safety requirements. However, as the components of *probability of an event* and its *consequences* are not as clearly stated as required by the definition of risk provided by the IEC 61508 (cf. Definition 1.7), it is referred to as *criticality* in the following.

Nevertheless, for a *CLF* to inform about the criticality of a state with respect to the safety of a system, monotonicity is not required. While the function should, in general, be increasing, to reflect increased criticality of states, it would be overly restrictive to require the same for the entire function. As such, for estimating an *RoS* the employed *CLF* is not required to provide monotonicity in the following.

It needs to be noted that this contradicts the definition of a *CLF* in the literature, for instance [45]. In contrast, the requirement for a single global minimum which is at the targeted stability point remains.

[2] Here the theorem is corrected compared to Jäger *et al.* [32]

Algorithmic Estimation of a Region of Safety

Algorithm 1 Pseudo-code for estimating a controller's *RoS* according to Theorem 4.1.

```
function EstimateRoS($\mathcal{X}_\tau$, $\tau$, $V(\mathbf{x})$, $\lambda_c$)
    $c_{max} \leftarrow \max_{\mathbf{x} \in \mathcal{X}_\tau}(V(\mathbf{x}))$
    $c_{min} \leftarrow c_{max} \cdot (1.0 - \lambda_c)$
    while $(c_{max} > 0)$ do
        $\mathcal{X}_{RoS} \leftarrow \{\mathbf{x} \in \mathcal{X}_\tau | V(\mathbf{x}) \leq c_{max}\}$
        for all $\mathbf{x} \in \mathcal{X}_{RoS}$ do
            if $(V(\mathbf{x}) \geq c_{min}) \wedge (\max(\dot{V}(\mathbf{x})) \geq -L_{\dot{V}} \cdot \tau)$ then    ▷ See Eq. (4.16)
                $\mathcal{X}_{RoS} \leftarrow \emptyset$
            if $(V(\mathbf{x}) < c_{min}) \wedge (V(\mathbf{x}) + sup\{\max(\dot{V}(\mathbf{x}))|x \in \mathcal{B}_{\mathbf{x},r}\} \cdot \tau > c_{max})$ then
                $\mathcal{X}_{RoS} \leftarrow \emptyset$    ▷ See Eq. (4.17)
        if $(\mathcal{X}_{RoS} \neq \emptyset)$ then
            return $\mathcal{X}_{RoS}$
        $c_{max} \leftarrow c_{min}$
        $c_{min} \leftarrow c_{max} \cdot (1.0 - \lambda_c)$
    return $\emptyset$
```

The theorem introduced in the last paragraph defined the requirements that have to be fulfilled by a set of states to form a controller's *RoS*. For estimating the same according to a given system model, the theorem is transformed into an algorithm in pseudo-code, cf. Algorithm 1. Similar to the theorem, an initial version of this algorithm was given in Jäger *et al.* [32] but is adjusted to the changed theorem here.

Following Theorem 4.1, the algorithm aims at estimating the maximal set of states satisfying Eqs. (4.16) and (4.17). It, therefore, requires the system model $f^*_\pi(\mathbf{x})$, the *CLF* $V(\mathbf{x})$, the discrete state space $\mathcal{X}_\tau$, and the time constant τ as inputs.

Additionally, the algorithm defines the parameters λ_c for deriving the values of c_{max} and c_{min}. While c_{max} is initialized as the maximal *CLF* value over the specified state space, c_{min} has to be set to differentiate between stabilizing and uncertain states. For that, λ_c is used. Depending on the shape of the used *CLF*, λ_c can be seen as a measure for the ratio between stabilizing and uncertain states. The higher its value, the more stabilizing states are required to form the *RoS*.

After initializing c_{max} and c_{min}, the set $\mathcal{X}_{RoS}$ forming an *RoS* candidate is generated. Each of its states is checked to either fulfill Eq. (4.16) or Eq. (4.17) depending on its *CLF* value. If all states adhere to their respective conditions, the set is shown to be an *RoS* and is returned by the algorithm. Otherwise, c_{max} and c_{min} are reduced to generate the next candidate. This procedure is repeated until either a valid *RoS* is found or c_{max} is zero, in which case the empty set is returned to indicate that no *RoS* could be found.

Note that, the calculation of the required Lipschitz constants, as well as of the supremum of the maximal gradient of $V(\mathbf{x})$, the sampling-based approach as described in Section 3.2.1 is used within this work.

4.2.3. Discussion on the Fulfillment of the Predefined Criteria

In this section, the concept of *Region of Safety (RoS)* is introduced as an approach to fulfilling Objective 1.1, that is, providing run-time safety assessment for using shared data in safety-critical control systems. To be applicable in that sense, Section 1.3.1 defined criteria to be fulfilled by a suitable approach.

The most important criterion asks for the ability to analyze a failure model of shared data and is satisfied by the presented approach through the usage of the proposed system model (cf. Section 4.1). It facilitates specifying a failure model of shared data independently from other sources of uncertainty but allows analyzing them in combination.

For this analysis, the concept of *RoS* is proposed. It builds upon the idea of *RoA* but respects the uncertainty introduced by, for instance, failures of shared data. The algorithm introduced together with Theorem 4.1 facilitates estimating a controller's *RoS* at run-time. It either provides a set of states for which it is guaranteed that the system will not leave the set or returns the empty set. While in the former case it is shown that safety will be maintained when using the shared data, the usage has to be rejected for the latter. It is precisely this binary decision which the *Run-Time Certification* criterion asks for and which is thereby fulfilled by the approach.

Moreover, the guarantee for safety is based on the definition of a *Control Lyapunov Function (CLF)*, which can interpreted as a measure of risk here. This function is defined at the system's design-time and does not directly depend on the failure characteristics of (shared) data. This enables an in-depth analysis of the *CLF* function already at design-time where it can be shown that the function appropriately assesses the risk each state poses to the safety of the overall system. At run-time, this function is not changed or adjusted, but only evaluated regarding its value and gradient to reflect the implications entailed by uncertainties, for instance, of shared data. By showing that a set of states generates gradients fulfilling the conditions of Theorem 4.1, it is assessed whether or not the employed controller, that is, the safety function, sufficiently reduces the risk, that is, adheres to its assigned safety performance.

Thus, while the *CLF* is key to performing a safety assessment at run-time, its definition and the algorithm for evaluating it can be examined at design-time to guarantee functional correctness. This renders the corresponding criterion of *Functional Correctness* to be fulfilled by the approach as well.

4.3. Evaluating the Concept of Region of Safety

The discussion on the predefined criteria underlined the applicability of the concept of *RoS* qualitatively. This shall be complemented by a quantitative evaluation in this section.

In this endeavor, the next subsection revisits the inverted pendulum to introduce the scenario and the considered uncertainties. Based on these considerations, Section 4.3.2 starts with designing a suitable *CLF* for the system in question. Examining the *RoS* for different candidates shows that the guarantee provided by Theorem 4.1 directly depends on the suitability of the chosen function. However, the value of λ_c, the parameter to be defined for applying Algorithm 1 for estimating *RoS*, affects the results and its validity as well. Therefore, the effect of this parameter and an approach on how to choose it are

Tab. 4.3.: Parameterization of failure types considered to form varying failure models for examining the estimation of *RoS* of the inverted pendulum.

Failure Type	Scaling	Activation	Deactivation
Noise	Q_{Noise} $(\mu = 0, \sigma = 0.005)$	0	1
Positive Spike	$0.025 \cdot \max(0, \Theta, \dot{\Theta}) + 0.01 \cdot t$	$\mathbb{N}(1.35, 0.15)$	$\mathbb{N}(0.75, 0.08)$
Negative Spike	$-0.025 \cdot \max(0, \Theta, \dot{\Theta}) - 0.01 \cdot t$	$\mathbb{N}(1.35, 0.15)$	$\mathbb{N}(0.75, 0.08)$
Constant Positive Offset	0.025	0	∞
Constant Negative Offset	-0.025	0	∞
Offset	$\mathbb{N}(0, 0.025)$	$\mathbb{N}(7.5, 0.15)$	$\mathbb{N}(0.75, 0.08)$

analyzed in Section 4.3.3. Finally, both of these investigations prepare for evaluating the effect of the considered uncertainties on the resulting *RoS* in Section 4.3.1. It will be shown that the estimated set of states is not left by the pendulum when confronted with the specified failure characteristics, which underlines not only the concept but facilitates using it as a run-time safety assessment method in dynamically composed systems and thereby fulfills Objective 1.1.

4.3.1. Defining the Scenario and Considered Uncertainties

For evaluating the concept of *RoS* the example of the inverted pendulum introduced in Section 4.2.1 is revisited. To examine the effect of different uncertainties, however, additional failure characteristics for U_{sensor} are defined. Similar to the previous chapter, Table 4.3 lists the considered failure types and their template distributions from which varying failure models are to be defined. For the sake of simplicity, both dimensions, that is, failure amplitudes of Θ and $\dot{\Theta}$, are assumed to be of the same magnitude. Therefore, it is refrained from stating the parameterization for each dimension individually.

For representing the failure models and failure types, the concept of *GFM* is used. This allows reusing the definitions provided in Section 3.4.1 and Section 3.4.4. The Noise failure type, for instance, leverages the already defined quantile function which incorporated Outliers (cf. Fig. 3.16). On the other hand, no value-correlations are considered this time. These are considered for the Spike failure types whose failure pattern remains the same, cf. Fig. 3.14c. Similar to the evaluation of the interval extraction method (cf. Section 3.4.4), two versions with complementary scaling distributions are considered. However, only positive values of Θ and $\dot{\Theta}$ are affected while the magnitude is time- and value-correlated at the same time. This shall enable examining whether or not these correlations are reflected in the resulting *RoS*.

Following a similar idea, constant positive and negative Offsets are considered again. As shown in Section 3.4.4, the individual failure types should cause uncertainties within

Tab. 4.4.: Failure models considered for the inverted pendulum scenario.

Failure Model	Failure Types	Failure Model	Failure Types
$\mathcal{FM}_{\{N\}}$	$\{F_{Noise}\}$	$\mathcal{FM}_{\{O\}}$	$\{F_{Offset}\}$
$\mathcal{FM}_{\{NO\}}$	$\{F_{Negative\ Offset}\}$	$\mathcal{FM}_{\{N,O\}}$	$\{F_{Noise}, F_{Offset}\}$
$\mathcal{FM}_{\{PO\}}$	$\{F_{Positive\ Offset}\}$	$\mathcal{FM}_{\{PO,NO\}}$	$\{F_{Positive\ Offset}, F_{Negative\ Offset}\}$
$\mathcal{FM}_{\{NS\}}$	$\{F_{Negative\ Spike}\}$	$\mathcal{FM}_{\{PS,NS\}}$	$\{F_{Positive\ Spike}, F_{Negative\ Spike}\}$
$\mathcal{FM}_{\{PS\}}$	$\{F_{Positive\ Spike}\}$	$\mathcal{FM}_{\{N,PO,NO\}}$	$\{F_{Noise}, F_{Positive\ Offset}, F_{Negative\ Offset}\}$

the resulting *RoS* while they are expected to cancel out each other when considered in combination.

Finally, the Offset failure type resembles an uncorrelated version of the Offset failure type presented in Section 3.4.1. In contrast to the other failure types, its scaling values are Gaussian distributed with a standard deviation of 0.025 and are therefore not deterministic.

The individual failure types are grouped into different failure models which are listed in Table 4.4.

Next to considering the failure types separately, the combination of Noise and Offset failure types are taken into account to analyze the effect of increased failure amplitudes. The combination of constant positive and negative Offsets, however, facilitates examining the assumption that failure types cancel each other out. It is assumed that the same is not true for Spike failure types in correspondence to the results of Section 3.4.4. Finally, $\mathcal{FM}_{\{N,PO,NO\}}$ should provide the same results as considering only the Noise failure type and is therefore used to ensure consistency within the results.

4.3.2. Designing a Control Lyapunov Function

With the definition of the failure models that are to be considered for estimating different *RoS*, the scenario and the system model are defined. Based on this, the *CLF* has to be defined before Algorithm 1 can be applied to estimate an *RoS*.

In the context of *RoS*, the *CLF* assesses the criticality of a state $\mathbf{x}$ with respect to the safety of the system under consideration. For that, the function has to be positive semi-definite and continuously differentiable with a global minimum at $V(\mathbf{x}_0) = 0$ where $\mathbf{x}_0 = [0\ 0]^T$. It is assumed that the risk posed to the system's safety is minimal at this point while criticality increases relative to the distance to this point.

Consequently, a controller intends to drive the system closer to its stability point and thereby minimizes the function over time. This is leveraged by Theorem 4.1, which builds upon the correspondingly negative gradient of a specified *CLF*. As this directly relates to the decision on whether or not a set of states is deemed safe, the *CLF* ultimately encodes safety constraints to be fulfilled by the system. Therefore, choosing an appropriate function at design-time is mandatory for being able to guarantee safety at run-time.

In this endeavor, this subsection investigates the effect different Lyapunov function candidates have on the estimation of *RoS* and derives aspects that have to be considered when designing an appropriate *CLF*. For that, the next paragraph discusses candidate functions that are evaluated regarding their suitability in the following paragraph.

Control Lyapunov Function Candidates

Designing an appropriate *CLF* is an active field of research where no general method exists yet [44]. Therefore, two different options are considered here from which 9 candidates are derived, cf. Table 4.5.

On the one hand, an approach using the energy within the system as a measure of criticality is used. Similar approaches were reported to result in appropriate *CLFs*, for instance in [108]. On the other hand, the distance of a state $\mathbf{x}$ from the targeted stability point $\mathbf{x}_0$ can be interpreted as a measure of criticality. Hence, distance-based function candidates are considered as well.

The idea of energy-based *CLF* is that a system is stable if it reaches a point of minimum energy. At that point, energy has to be provided for the system to leave the stability point and therefore the system will naturally return to its stability point once the additional energy is not provided.

Using this thinking, a *CLF* candidate for the inverted pendulum can be designed by considering its potential and kinetic energy, cf. Eqs. (4.18) to (4.20).

$$E_{kin} = \frac{1}{2} ml^2 \dot{\Theta}^2 \tag{4.18}$$

$$E_{pot} = mgl \cdot cos(\Theta) \tag{4.19}$$

$$V_9(\Theta, \dot{\Theta}) = \frac{1}{2} ml^2 \dot{\Theta}^2 + mgl(1 - cos(\Theta)) \tag{4.20}$$

The kinetic energy of the inverted pendulum is given by Eq. (4.18) and solely depends on the angular velocity $\dot{\Theta}$. It increases when the pendulum is in motion but has its global minimum at $\dot{\Theta} = 0\,\mathrm{rad\,s^{-1}}$. Additionally, the potential energy is given by Eq. (4.19). Using the cosine of Θ and multiplying it with the length of the pendulum provides the height of the pendulum. Consequently, the potential energy has a global minimum at $\Theta = -\pi\,\mathrm{rad}$ and a maximum at $\Theta = 0\,\mathrm{rad}$. For that reason, the cosine value is inverted and shifted in Eq. (4.20). On the one hand, this ensures that the global minimum is at $\mathbf{x} = \mathbf{x}_0 = [\Theta = 0\,\mathrm{rad}\ \dot{\Theta} = 0\,\mathrm{rad\,s^{-1}}]^T$. On the other hand, it underlines that designing an appropriate *CLF* depends on the system at hand and can not be considered as an isolated activity.

The resulting function is visualized in Fig. 4.8. It combines both energies to measure the criticality of a state by the amount of *energy* in the system it relates to[3]. Furthermore, the function is continuously differentiable and positive semi-definite and thereby fulfills the mathematical requirements of a *CLF*.

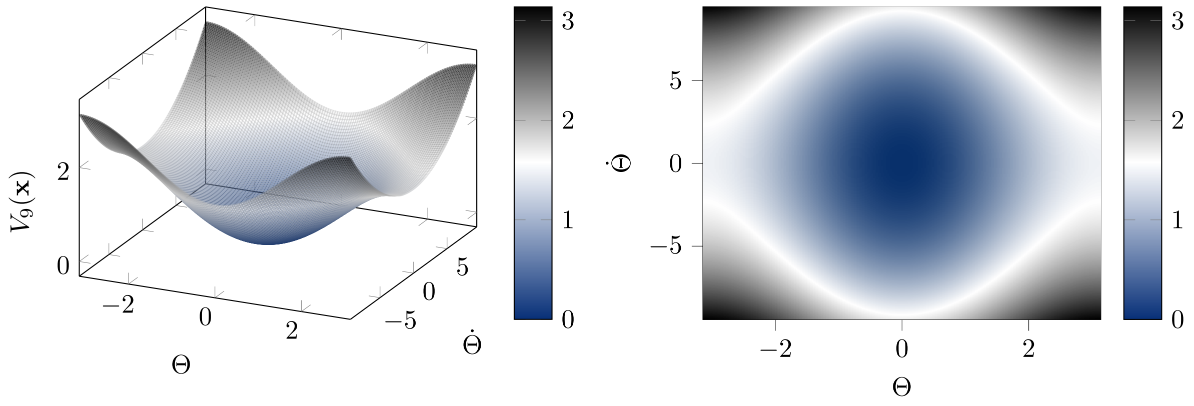
The manual adaptations required by the energy-based approach were necessary to define the *CLF* candidate in such a way that the function value is increasing relative to the distance of state $\mathbf{x}$ from the envisioned stability point $\mathbf{x}_0$. To investigate this direction of designing *CLF* functions, Eq. (4.21) is considered next.

$$V(\Theta, \dot{\Theta}) = w_\Theta \Theta^2 + w_{\dot{\Theta}} \dot{\Theta}^2 + w_{\Theta+\dot{\Theta}} (\Theta + \dot{\Theta})^2 \tag{4.21}$$

[3] Energy is written italic here to state the fact that the final value does not state the actual energy but is only based on the concepts of energy.

Tab. 4.5.: Configurations of *CLF* candidates.

ID	w_Θ	$w_{\dot{\Theta}}$	$w_{\Theta+\dot{\Theta}}$	ID	w_Θ	$w_{\dot{\Theta}}$	$w_{\Theta+\dot{\Theta}}$
①	1.0	0.0	0.0	⑤	2.0	0.0	1.0
②	0.0	1.0	0.0	⑥	2.0	1.0	1.0
③	0.0	0.0	1.0	⑦	1.0	0.0	0.2
④	2.0	1.0	0.0	⑧	3.0	0.0	0.2
⑨	$\frac{1}{2}ml^2\dot{\Theta}^2 + mgl(1 - cos(\Theta))$						



(a) Surface plot of the energy-based *CLF*.

(b) Heatmap of the energy-based *CLF*.

Fig. 4.8.: Plotting the function value of the energy-based *CLF* candidate ⑨.

Similar as before, the function has a global minimum at $\mathbf{x}_0$ while its value is monotonically increasing in each direction from that point, which ensures that the function is positive semi-definite. Furthermore, the function is three times differentiable and thereby Lipschitz continuous, which is a requirement to apply Theorem 4.1.

Apart from these mathematical requirements, the *CLF* builds upon the idea of Euclidean distance (the square root is not taken to simplify the expression) to express the criticality of a state. Taking the central safety requirement of the pendulum remaining in an upward position into account, the criticality of states increases with their value of Θ. For that, the squared value of Θ is used.

However, not only the angle but also the angular velocity has to be considered. In general, high angular velocities cause the pendulum to leave its equilibrium state. Therefore, as before, the squared value of $\dot{\Theta}$ is added as well.

Finally, as discussed for the control strategy in Section 4.2.1 already, angular velocities can lower or increase the criticality of a state depending on the current angle. A state having a positive displacement ($\Theta \geq 0\,\mathrm{rad}$) but a negative angular velocity ($\dot{\Theta} \leq 0\,\mathrm{rad\,s^{-1}}$) is less critical than a positive displacement combined with a positive angular velocity (or vice versa). This fact is assessed by the last element of the function, $(\Theta + \dot{\Theta})^2$.

Each of these aspects is weighted by their corresponding coefficients (w_Θ, $w_{\dot{\Theta}}$, and

$w_{\Theta+\dot{\Theta}}$). Using these, different versions of this function can be analyzed. The variations considered here are stated in Table 4.5.

While configurations 1-3 are chosen to analyze the effect of the individual components of the Lyapunov function candidate, configurations 4-6 are used to compare the effect of including the combinations of Θ and $\dot{\Theta}$. Finally, configurations 7 and 8 optimize the weights. Configuration 9 resembles the *CLF* obtained using the energy approach in Eq. (4.20).

Evaluating Control Lyapunov Functions

The defined *CLF* candidates shall be evaluated regarding their appropriateness for estimating the *RoS* of the inverted pendulum. As this requires taking multiple perspectives into consideration, the employed evaluation procedure and applied metrics are described in the next paragraph before the results stated in Table 4.6 are discussed.

Procedure To evaluate the presented candidates, they are used to estimate *RoS* according to Algorithm 1. For that, no uncertainties were considered and the parameter λ_c was set to 0.7. It was refrained from assuming uncertainties at this point to examine the implications of different *CLF*. Furthermore, this enables using a conservative value for λ_c. Note that a value of $\lambda_c = 1.0$ is not possible as this would effectively turn the *RoS* estimation into an *RoA* estimation, for which it was shown in Section 4.2.1 that no valid sets are found.

To assess the resulting *RoS*, different aspects are considered. First and foremost, the question of whether or not a system is correctly classified as safe has to be answered. In that endeavor, each state $\mathbf{x} \in \hat{\mathcal{V}}(c_{max})$ is considered as a starting state for a simulation of the inverted pendulum. Using the Runge-Kutta method with orders 4/5 [109], the system Eq. (4.1) is solved with a maximal time step of $\tau = 0.025\,\mathrm{s}$ while the time step is adapted to maintain an integration error of 10^{-6}. Opposed to that, the control policy is sampled with an exact period of $\tau = 0.025\,\mathrm{s}$ to emulate real-time constraints. For each time step, it is checked whether the safety requirement is maintained ($|\Theta| \leq \frac{\pi}{2}$). If the system violates the requirement at any time step, the *RoS* associated with the starting state is considered unsafe, resulting in a *False* value in the right most column of Table 4.6.

However, an *RoS* should not only enable maintaining safety but also guarantee that the system does not leave the corresponding set of states. For that, the maximal distance $d_{\hat{\mathcal{V}}(c_{max})}$ (Euclidean distance) from this set is calculated during simulation. Due to the discretization of the state space, distances of $d_{\hat{\mathcal{V}}(c_{max})} \leq \sqrt{2 \cdot 0.025^2} \cdot 0.5 = 0.0177$ are acceptable. Distances greater than this threshold, however, indicate that the guarantee is violated.

Both of these metrics analyze the resulting *RoS* regarding the safety aspects. In contrast, the performance aspect, that is, how well the true *RoS* is estimated, has to be considered as well. For that, the following three metrics are applied, cf. Eqs. (4.22) to (4.24).

$$r_{\mathcal{X}_\tau} = \frac{\left|\hat{\mathcal{V}}(c_{max})\right|}{|\mathcal{X}_\tau|} \tag{4.22}$$

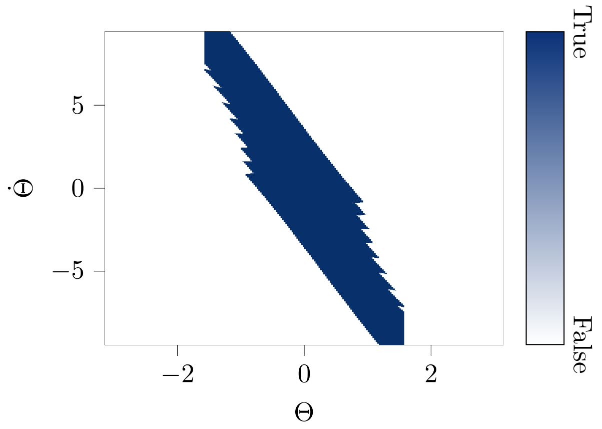


Fig. 4.9.: True *RoS* of the inverted pendulum when assuming no uncertainties. This set was found through simulation. Each of the considered states was considered as a starting state for simulating the inverted pendulum with the considered control policy. Only if the pendulum remained in an upright position, the state was accepted as part of the *RoS*.

Eq. (4.22) defines $r_{\mathcal{X}_\tau}$ as the ratio of the number of states in the estimated *RoS* $\hat{\mathcal{V}}(c_{max})$ to the overall number of states. In that way, the relative size can be assessed.

While it is favorable to cover as many states as possible, it remains a question of safety. Therefore, the estimated *RoS* has to be compared to the set of states for which the system is truly safe. For that, the true *RoS* is determined by assuming each state $\mathbf{x}$ of the state space $\mathcal{X}_\tau$ as a starting state for the simulation. If the control policy is able to drive the inverted pendulum to the targeted stability point, the starting state is considered as part of the true *RoS* $\mathcal{V}(c_{max})$.

The result is shown in Fig. 4.9. As the motor can apply only a limited torque on the inverted pendulum, displacements of approx. $|\Theta|\geq 0.8\,\text{rad}$ can be counteracted only if the pendulum is already in a motion directed towards the stability point.

$$r_{\mathcal{V}(c_{max})} = \frac{\left|\hat{\mathcal{V}}(c_{max}) \cap \mathcal{V}(c_{max})\right|}{|\mathcal{V}(c_{max})|} \tag{4.23}$$

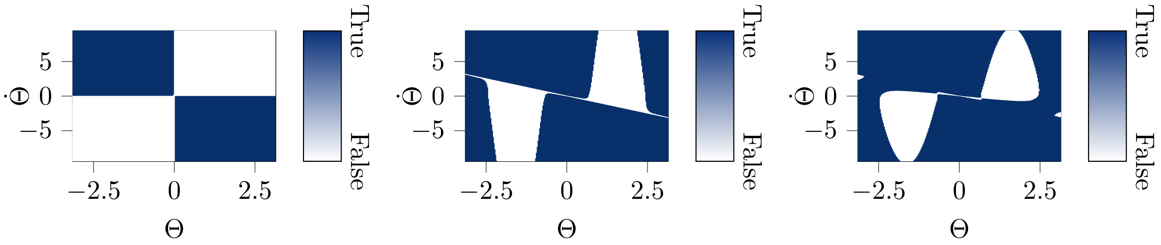
Based on the true *RoS*, Eq. (4.23) defines $r_{\mathcal{V}(c_{max})}$ as the ratio of states of the estimated *RoS* $\hat{\mathcal{V}}(c_{max})$ that are within the true *RoS* over the overall number of states in the true *RoS*. This enables assessing how conservative the estimation of *RoS* is. In general, the goal is to cover as many states of the true *RoS* as possible, that is, achieve a value of $r_{\mathcal{V}(c_{max})}$ close to one.

$$r_{\notin\mathcal{V}(c_{max})} = \frac{\left|\left\{\mathbf{x}|\mathbf{x} \in \hat{\mathcal{V}}(c_{max}) \wedge \mathbf{x} \notin V(c_{max})\right\}\right|}{|\mathcal{V}(c_{max})|} \tag{4.24}$$

Eq. (4.24) complements the examination of the estimated and true *RoS* by defining $r_{\notin\mathcal{V}(c_{max})}$ as the ratio of states that are within the estimated *RoS* but not within the true *RoS* over the number of states in the true *RoS*. As any value greater than zero indicates that the estimated *RoS* contains states from which the inverted pendulum will not return to its stability point but possibly violate its safety requirement, zero is the value to be aimed at for this metric.

Tab. 4.6.: Evaluation of *CLF* candidates.

ID	$r_{\mathcal{X}_\tau}$	$r_{\mathcal{V}(c_{max})}$	$r_{\notin\mathcal{V}(c_{max})}$	$\max(d_{\hat{\mathcal{V}}(c_{max})})$	**Is Safe?**
①	0.00000	0.00000	0.00000	0.00000	False
②	1.00000	1.00000	5.33524	0.00942	False
③	0.00000	0.00000	0.00000	0.00000	False
④	0.00000	0.00000	0.00000	0.00000	False
⑤	0.00000	0.00000	0.00000	0.00000	False
⑥	0.00000	0.00000	0.00000	0.00000	False
⑦	0.03146	0.19929	0.00000	0.01516	True
⑧	0.02802	0.17751	0.00000	0.00052	True
⑨	0.00012	0.00077	0.00000	0.12811	True



(a) Results for candidate ①.

(b) Results for candidate ③.

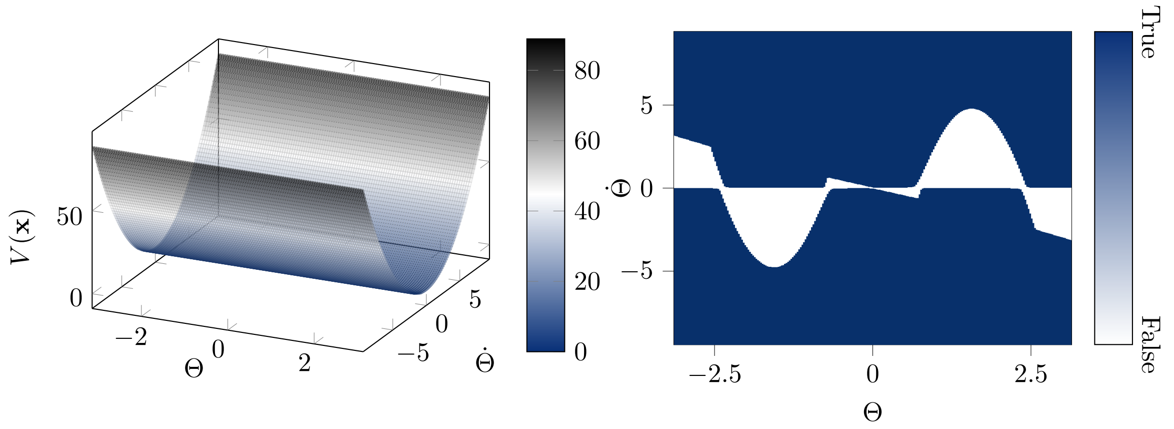
(c) Results for candidate ⑥.

Fig. 4.10.: Fulfillment of the *RoA* condition (cf. Eq. (4.16)) by candidates ①, ③, and ⑥.

Results With the aforementioned method, each of the proposed *CLF* candidates is evaluated. The results are listed in Table 4.6.

The candidates ①, ③, ④, ⑤, and ⑥ do not allow estimating an *RoS*. This is caused by the controller's inability to minimize the corresponding functions over time. As a consequence, no connected set of states forming a valid hull $\mathcal{H}_V(c_{min}, c_{max})$ can be found. This is illustrated in Fig. 4.10 which exemplarily shows the fulfillment of the *RoA* condition (cf. Eq. (4.16)) by the individual states of the state space. In the case of candidate ①, which assesses only the squared displacement Θ, negative gradients of the *CLF* can be obtained only for states having angular velocities driving the inverted pendulum towards the stability point. Thus, according to this estimate, the control policy does not affect the stability of the inverted pendulum.

This is caused by a missing indirection. While the controller determines a torque u which results in an increased or decreased value for $\dot{\Theta}$, the *CLF* takes only Θ into account. Therefore, the effect the control policy has is not reflected in the gradient. As



(a) Surface of the *CLF*. **(b)** Fulfillment of the *RoA* condition by the states.

Fig. 4.11.: Visualization of *CLF* candidate ②.

a consequence, states not fulfilling the *RoA* condition are within each possible set of hull states $\mathcal{H}_V(c_{min}, c_{max})$ and thereby prevent the estimation of an *RoS*. Thereby, this example underlines that the *CLF* and the applied control policy have to be aligned with each other, that is, consider the same state space and state changes.

The described phenomenon is observable for ③ and ⑥ as well. For candidate ③, the white area indicating states not fulfilling the condition divides the blue area completely. As a consequence no level set of the *CLF* can be found to form the set of stabilizing states as it will always contain states not fulfilling the *RoA* condition. While the white areas forming around $|\Theta| \in [0.7, 2.3]$ correctly identify states where the torque that can be provided by the motor is not sufficient to stabilize the system, a line of white states additionally divides the state space, crossing the origin. This line indicates $\Theta + \dot{\Theta} = 0$, which is precisely the *error* the control policy acts on, cf. Eq. (4.12). As only a simplistic P-controller was chosen, no control action is applied when the inverted pendulum is at those states, which entails that the gravitational force is not counteracted. Consequently, the inverted pendulum moves downwards, resulting in $\dot{V}(\mathbf{x}) > 0$. Although mitigated by candidate ⑥, the same reason applies.

In contrast to these candidates, *CLF* ② considers the entire state space as the controller's *RoS*, as indicated by the value of $r_{\mathcal{X}_\tau}$. While this entails that all states of the true *RoS* are within the estimated *RoS*, unsafe states are included as well. It has to be noted that the distance $d_{\hat{\mathcal{V}}(c_{max})}$ indicates that the pendulum leaves the estimated *RoS* as the simulation is not restricted to the defined state space.

As a consequence of considering the entire state space as the estimated *RoS*, the system's safety is not maintained during all simulations. The overestimation of the *RoS* is caused by the *CLF*'s focus on $\dot{\Theta}$, as can be seen in Fig. 4.11. Generally, the control policy aims at minimizing the angular velocity and therefore applies a conversely directed torque. This suffices for high magnitudes of angular velocities to generate a negative gradient on the *CLF*, cf. Fig. 4.11b. However, for states where the angular velocity is already zero, the control policy determines a torque increasing the velocity in order to bring the pendulum towards an angle of $\Theta = 0$. This goal is not reflected in candidate

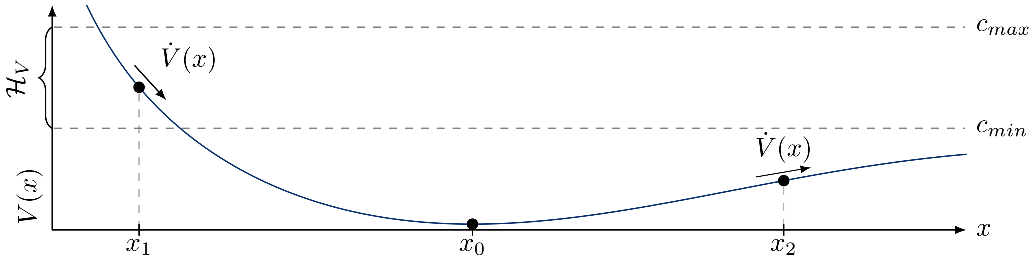
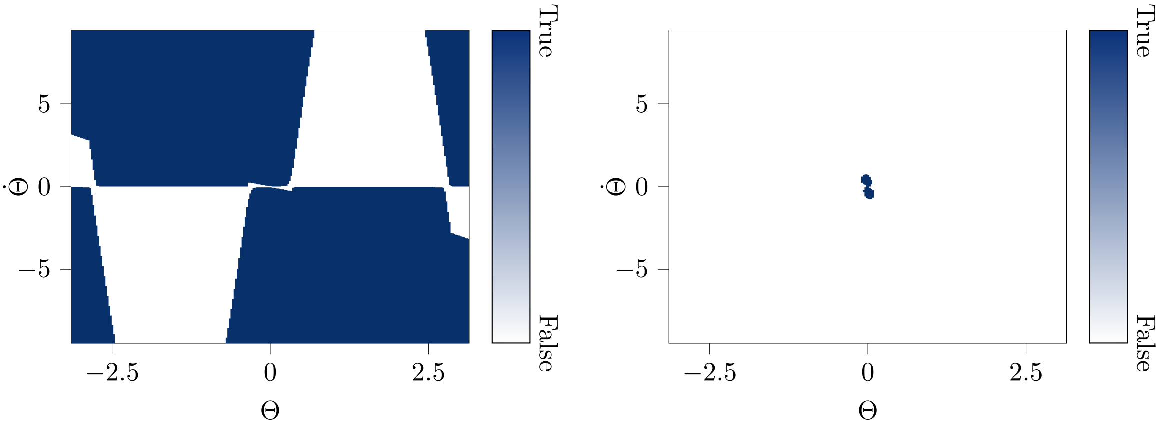


Fig. 4.12.: Visualizing the effect of an asymmetric *CLF*.



(a) Fulfillment of the *RoA* condition (cf. Eq. (4.16)) by states according to *CLF* ⑨.

(b) Fulfillment of the *RoS* condition (cf. Eq. (4.17)) by states according to *CLF* ⑨.

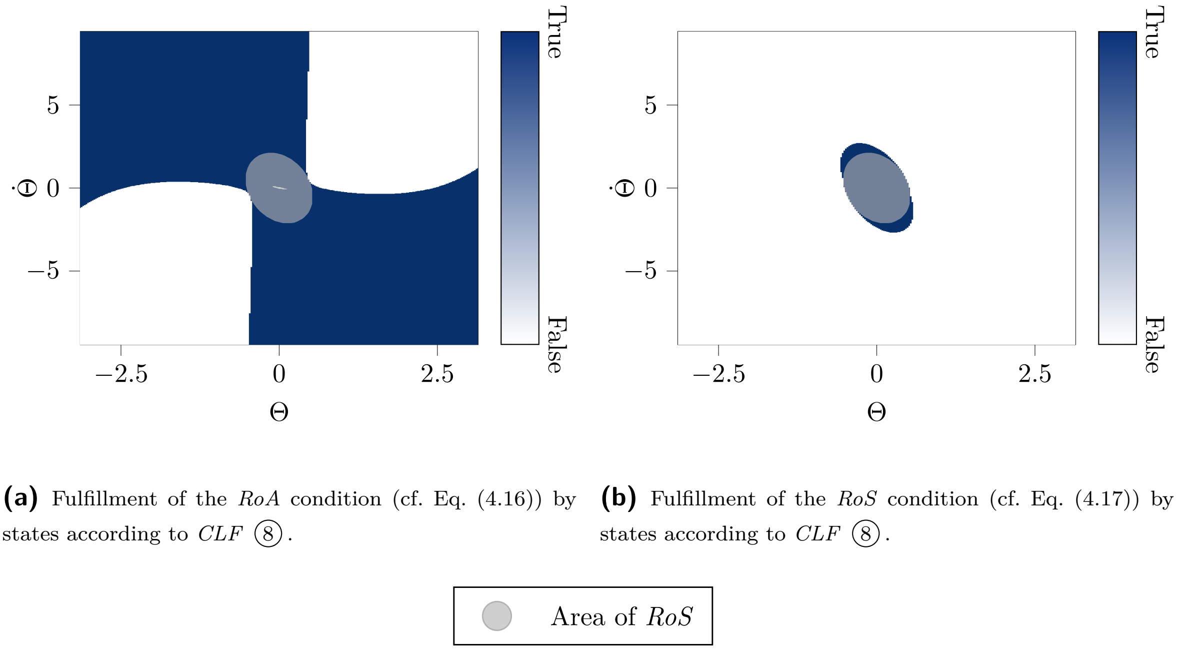
Fig. 4.13.: Visualization of *CLF* candidate ⑨.

②, which causes the white area along $\dot{\Theta}$ approx. 0. This, however, does not prevent an *RoS* to be estimated due to the asymmetry of the *CLF*, cf. Fig. 4.11a.

For the sake of the argument, the problem is illustrated by an exemplary curve considering only one dimension in Fig. 4.12. Note that this example has no relation to the inverted pendulum but serves solely the purpose of illustrating the general problem. The displayed function does not increase symmetrically when starting at x_0. While it increases drastically for values $x \leq x_0$ it increases only moderately for $x \geq x_0$. Additionally, it is assumed that the employed control policy produces negative gradients for states of the former case but can not minimize the function for states of the latter case (see x_1 and x_2 respectively). Applying Algorithm 1 can result in an invalid *RoS* as indicated by c_{max} and c_{min}. While the set of stabilizing states that form $\mathcal{H}_V$ fulfill Eq. (4.16), the system may evolves towards unsafe states when moving towards $x \geq x_0$. In this sense, $\mathcal{H}_V$ does not *encompass* all uncertain states as it does not form a closed, convex hull around them (with respect to the state space).

Regarding ② and the inverted pendulum, this is the case as only $\dot{\Theta}$ is considered by the *CLF* but not Θ. As a consequence, stabilizing states are found for angular velocities of high magnitudes, which enables the estimation of an *RoS* and causes uncertain states to fulfill Eq. (4.17) due to the high value of c_{max}.

In contrast, only candidates ⑦, ⑧, and ⑨ enable estimating *RoS* that indeed



(a) Fulfillment of the *RoA* condition (cf. Eq. (4.16)) by states according to *CLF* ⑧.

(b) Fulfillment of the *RoS* condition (cf. Eq. (4.17)) by states according to *CLF* ⑧.

(c) Legend for *RoS* visualizations, e.g. Figs. 4.14a and 4.14b.

Fig. 4.14.: Visualization of *CLF* candidate ⑧.

solely contain states for which the system's safety is maintained. Candidate ⑨ produces the smallest set of states which covers 0.012 % of the overall state space and 0.077 % of the true *RoS*. Due to this small set, which covers states surrounding the origin of the state space, the pendulum maintains the safety requirement in all simulations. However, as indicated by $d_{\hat{\mathcal{V}}(c_{max})} = 0.12811 >> 0.017$, during these simulations, the pendulum leaves the estimated *RoS* and thereby violates the guarantee that shall be provided. Similar to the previous example, the stabilizing states are found for increased *CLF* values but do not encompass contained uncertain states completely. It can be seen in Fig. 4.13b that white areas divide the state space. The estimation of the invalid *RoS* is not prevented by the *RoS* condition either, due to the discretization of the state space. However, increasing the discretization could prevent the estimation of an *RoS* as the number of uncertain states would increase and the corresponding *RoS* condition would not be fulfilled.

Therefore, candidates ⑦ and ⑧ remain. Both of them provide valid *RoS*. While candidate ⑦ covers a greater number of states of the true *RoS*, its distance $d_{\hat{\mathcal{V}}(c_{max})}$ is increased. It does not exceed the threshold of 0.017, but due to its closeness, further work focuses on candidate ⑧. The fulfillment of the *RoA* and *RoS* conditions by the corresponding states is visualized in Fig. 4.14. In contrast to the previous examples, the area of blue states indicates that the fulfillment of the *RoA* condition is connected. This enables forming a set of stabilizing states that encompasses the remaining uncertain states surrounding the stability point.

For these uncertain states, Fig. 4.14b shows that the inverted pendulum will not evolve to a state $V(\mathbf{x}) > c_{max}$ within a time horizon of τ as they fulfill the *RoS* condition. Moreover, it can be seen that the blue area even extends beyond the estimated *RoS*, which is indicated by the gray overlay. This is a consequence of considering the maximum gradient of the *CLF* in the *RoS* condition in Eq. (4.17), which can be negative. The gradient is multiplied by the considered time horizon τ and added to the *CLF* value at the considered state to compare the resulting value with c_{max}, cf. Eq. (4.17). Given a negative gradient with a sufficient magnitude, the fulfillment of the *RoS* condition for states outside the actual level set representing the *RoS* is possible. Thus, these blue-colored states indicate that the controller is able to provide control actions driving the system into the estimated *RoS* despite being outside of it. These are not part of the *RoS*, however, as these or other states having the same *CLF* value are not fulfilling the *RoA* condition.

Nevertheless, the fulfillment of the *RoS* condition by all uncertain states underline that the inverted pendulum should always remain within the estimated *RoS*. This is further supported by the value of $d_{\hat{\mathcal{V}}(c_{max})}$, which is smaller than $\sqrt{2 \cdot 0.025^2} \cdot 0.5 = 0.0177$, that is, half the euclidean distance of the chosen discretization. As such, it is shown that the distance by which the inverted pendulum leaves the estimated *RoS* is caused by the discretization of the state space.

4.3.3. Determining an Appropriate Value for λ_c

The last section discussed different *CLF* candidates and deemed ⑧ as most suitable. Moreover, aspects that have to be taken into consideration while designing a *CLF* were derived from invalid candidates. For instance, ② underlined that asymmetric functions may result in overestimating the *RoS*. This was visualized in Fig. 4.12, where it was shown that the calculated set of stabilizing states misses encompassing the uncertain states. Another solution, next to adjusting the *CLF*, is to adapt the parameter of λ_c. For ②, an increased value of λ_c would have prevented the estimation of a *RoS*. Although the candidate would remain inappropriate for the inverted pendulum, it motivates to examine the value of λ_c and how to set it.

In this endeavor, the next paragraph briefly discusses the procedure to evaluate the effect of λ_c before the following paragraph discusses the results and determines the correct value for the inverted pendulum.

Procedure For examining the effect of λ_c, the parameter is set to values $\lambda_c \in [0, 1]$ and *RoS* are estimated accordingly. For each *RoS*, six parameters are extracted.

To generate general indications on the validity of the estimated *RoS*, the result of the simulation-based check of the safety requirement presented in the previous evaluation is stated along with the maximal distance $d_{\hat{\mathcal{V}}(c_{max})}$ again. These parameters are complemented with metrics to assess the set of hull states $\mathcal{H}_V(c_{min}, c_{max})$ calculated for each *RoS*, cf. Eqs. (4.25) to (4.27) and (4.30).

$$r_{\mathcal{H}} = \frac{|\mathcal{H}_V(c_{min}, c_{max})|}{|\hat{\mathcal{V}}(c_{max})|} \tag{4.25}$$

$r_{\mathcal{H}}$ defined in Eq. (4.25) states the ratio of stabilizing states to the overall number of states in the estimated *RoS*. It thereby allows assessing the quantity of stabilizing states.

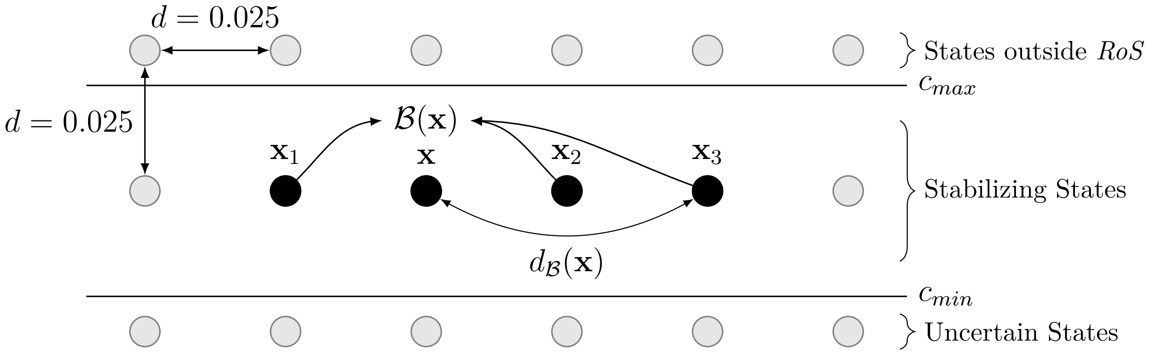


Fig. 4.15.: Schematic illustration of Eq. (4.29). The gray and black-colored circles indicate states of the two-dimensional state space. While states colored black are associated to the considered set $\mathcal{B}(\mathbf{x})$, gray-colored states are not considered by the same. Separated by the thresholds of c_{min} and c_{max}, uncertain states, stabilizing states, and states outside the RoS are distinguished. Finally, the value of $d_{\mathcal{B}}(\mathbf{x})$ is nothing but the greatest distance of any state $\mathbf{x}_n \in \mathcal{B}(\mathbf{x})$ to $\mathbf{x}$.

However, this is not sufficient to assess whether or not these states encompass the uncertain states. For that, the parameters $d_{\mathcal{H}_V}$ and d_{RoS} are defined in Eq. (4.26) and Eq. (4.27).

$$d_{\mathcal{H}_V} = \max_{\mathbf{x}_1,\mathbf{x}_2 \in \mathcal{H}_V(c_{min},c_{max})} (\|\mathbf{x}_1 - \mathbf{x}_2\|) \tag{4.26}$$

$d_{\mathcal{H}_V}$ states the maximal distance between all stabilizing states. In case the stabilizing states form a cluster encompassing only parts of the uncertain states, this distance is reduced. To further assess this distance, d_{RoS} is calculated.

$$d_{RoS} = \max_{\mathbf{x}_1,\mathbf{x}_2 \in \hat{\mathcal{V}}(c_{max})} (\|\mathbf{x}_1 - \mathbf{x}_2\|) \tag{4.27}$$

d_{RoS} states the maximal distance between all states of the estimated RoS. In general, the stabilizing states should form a hull surrounding the RoS. Thus, a valid value of λ_c should entail $d_{\mathcal{H}_V} = d_{RoS}$.

However, this does not allow assessing whether or not the stabilizing states in $\mathcal{H}_V(c_{min}, c_{max})$ are sufficiently close to each other, that is, whether or not the convex hull is closed.

$$\begin{aligned}\mathcal{B}(\mathbf{x}) = \{&\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 | \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathcal{H}_V(c_{min}, c_{max}) \\ &\wedge \|\mathbf{x} - \hat{\mathbf{x}}\| \geq \|\mathbf{x}_1 - \hat{\mathbf{x}}\| \geq \|\mathbf{x}_2 - \hat{\mathbf{x}}\| \geq \|\mathbf{x}_3 - \hat{\mathbf{x}}\| \\ &\forall \hat{\mathbf{x}} \in \mathcal{H}_V(c_{min}, c_{max}) \setminus \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}\}\end{aligned} \tag{4.28}$$

$$d_{\mathcal{B}}(\mathbf{x}) = \max_{\hat{\mathbf{x}} \in \mathcal{B}(\mathbf{x})} (\|\mathbf{x} - \hat{\mathbf{x}}\|) \tag{4.29}$$

$$d_{max_{\mathcal{B}}} = \max_{\mathbf{x} \in \mathcal{H}_V(c_{min},c_{max})} (d_{\mathcal{B}}(\mathbf{x})) \tag{4.30}$$

To assess this, the maximal distance to the three closest neighbors is calculated for each state $\mathbf{x}$ in the set of hull states, cf. Eqs. (4.28) to (4.30). The principles of Eqs. (4.28) and (4.29) are illustrated in Fig. 4.15.

The set $\mathcal{B}(\mathbf{x})$ contains nothing but the three stabilizing states $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ that are closest to the state $\mathbf{x}$ in $\mathcal{H}_V(c_{min}, c_{max})$. Based on that, Eq. (4.29) determines the maximal distance between $\mathbf{x}$ and its three closest neighboring states. By determining the maximal value of these distances over all stabilizing states in Eq. (4.30) a value assessing the *connectivity* of the stabilizing states is provided.

This value should be balanced in accordance with the discretization of the state space. With regard to the inverted pendulum and taking the value discretization of 0.025 for Θ and $\dot{\Theta}$ into account, the optimal value for $d_{max_{\mathcal{B}}}$ is 0.05, as visualized in Fig. 4.15.

This value indicates, that the neighboring states of each stabilizing state are forming a line at maximum as the values of c_{min} and c_{max} are sufficiently close while the *RoA* condition for each stabilizing state is fulfilled.

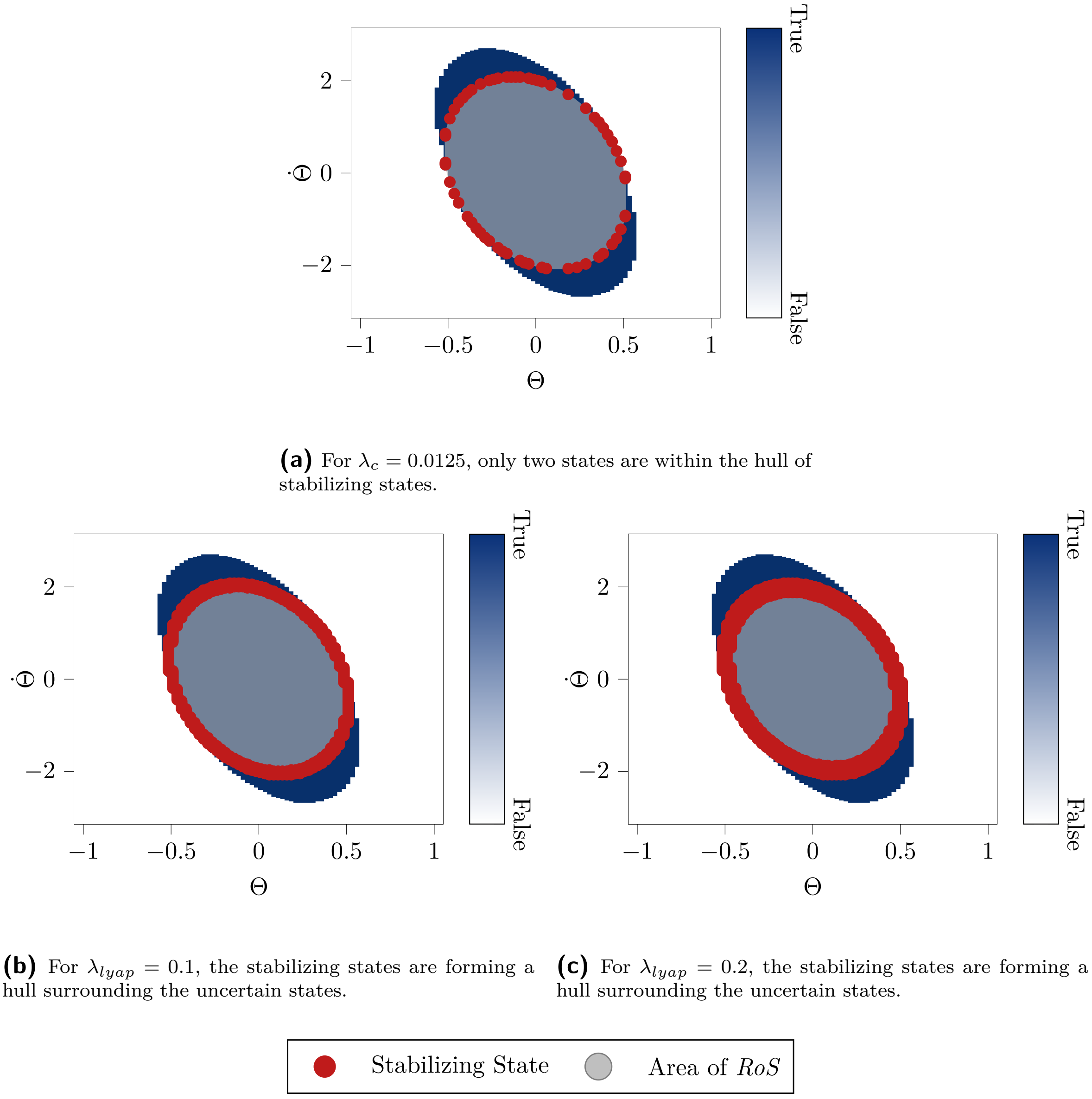
Values of $d_{max_{\mathcal{B}}} > 0.05$, however, indicate that the stabilizing states are not sufficiently connected, that is, the convex hull is not *closed.* This means that state trajectories along which the system may leave the estimated *RoS* are possible, which violates the guarantee an *RoS* shall provide.

On the other hand, values $d_{max_{\mathcal{B}}} < 0.05$ indicate that the set of stabilizing states is denser than required. This may occur if the range between c_{min} and c_{max} is increased (e.g. by larger values of λ_c) such that the stabilizing states do not form a line of singular states (as illustrated in Fig. 4.15) but form a line of two or more states. This may result in overly restrictive *RoS* estimations.

Results Table 4.7 summarizes the results by stating the aforementioned metrics. The first row states the values obtained for $\lambda_c = 1.0$. This value entails that all states within the estimated *RoS* have to fulfill the *RoA* condition. As was shown in Section 4.2.1, this is not the case for the inverted pendulum due to the simplicity of the employed control policy and resulting uncertain states surrounding the stability point. Consequently, no *RoS* is estimated.

In contrast, valid *RoS* are obtained for values $\lambda_c \leq 0.9$. Both corresponding metrics, the distance $d_{\hat{\mathcal{V}}(c_{max})}$ and the simulation-based safety check, return positive. Furthermore, as is to be expected, the number of states in the hull $\mathcal{H}_V(c_{min}, c_{max})$ reduces in proportion to the specified value of λ_c. For values of $\lambda_c \geq 0.0125$, the condition $d_{\mathcal{H}_V} = d_{RoS}$ holds, meaning that the stabilizing states indeed encompass the estimated *RoS*.

However, the stabilizing states are not sufficiently connected, as indicated by $d_{max_{\mathcal{B}}} = 0.50559$. This can be seen in Fig. 4.16a as well where the stabilizing states are colored red. At approx. $|\Theta| = 0.5$ it becomes clear that the distances between these are increased and thereby trajectories of the system leaving the *RoS* may be possible. According to Table 4.7 the value of λ_c has to be increased to 0.1 for all stabilizing states to be sufficiently connected. This is supported by Fig. 4.16b where the red-colored stabilizing states successfully encompass all states of the estimated *RoS*. Moreover, by comparing Fig. 4.16b and Fig. 4.16c it becomes clear that increasing λ_c to 0.2 is not necessary here as it increases the number of states required to form $\mathcal{H}_V(c_{min}, c_{max})$. Thereby, the estimation of *RoS* becomes overly restrictive. The same is underlined by the value of $d_{max_{\mathcal{B}}}$ for $\lambda_c = 0.2$, which decreases to 0.03536 and thus supports this conclusion. Consequently, for examining the effect of uncertainties on the estimation of *RoS* a value of $\lambda_c = 0.1$ is applied.



(a) For $\lambda_c = 0.0125$, only two states are within the hull of stabilizing states.

(b) For $\lambda_{lyap} = 0.1$, the stabilizing states are forming a hull surrounding the uncertain states.

(c) For $\lambda_{lyap} = 0.2$, the stabilizing states are forming a hull surrounding the uncertain states.

(d) Legend for *RoS* and stabilizing states visualizations, e.g. Figs. 4.16a to 4.16c.

Fig. 4.16.: Analysis of different values of λ_c. The blue area indicates states that fulfill the *RoS* condition while the transparent, gray overlay illustrates the states of the estimated *RoS*. The red states correspond to the set $\mathcal{H}_V(c_{min}, c_{max})$ of the respective *RoS*.

Tab. 4.7.: Evaluating the effect of λ_c on the estimated RoS.

λ_c	$r_{\mathcal{H}}$	$d_{\mathcal{H}_V}$	d_{RoS}	$d_{max_{\mathcal{B}}}$	$\max(d_{\hat{\mathcal{V}}(c_{max})})$	**Is Safe?**
1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	False
0.90000	0.89983	4.16473	4.16473	0.03536	0.00052	True
0.80000	0.80154	4.16473	4.16473	0.03536	0.00052	True
0.70000	0.70174	4.16473	4.16473	0.03536	0.00052	True
0.60000	0.60064	4.16473	4.16473	0.03536	0.00052	True
0.50000	0.49934	4.16473	4.16473	0.03536	0.00052	True
0.40000	0.40105	4.16473	4.16473	0.03536	0.00052	True
0.30000	0.30088	4.16473	4.16473	0.03536	0.00052	True
0.20000	0.20015	4.16473	4.16473	0.03536	0.00052	True
0.10000	0.09886	4.16473	4.16473	0.05000	0.00052	True
0.05000	0.05271	4.16473	4.16473	0.12748	0.00052	True
0.02500	0.02476	4.16473	4.16473	0.27613	0.00052	True
0.01250	0.01201	4.16473	4.16473	0.50559	0.00052	True
0.00625	0.00759	4.36212	4.36212	0.98011	0.00258	True
0.00000	0.00029	0.35089	4.76681	0.35089	0.00531	True

4.3.4. The Effect of Varying Sensor Failures on Regions of Safety

With the *CLF* function and the λ_c value defined, Algorithm 1 can be used to estimate the *RoS* of the inverted pendulum in presence of uncertainties, which is the goal of this subsection. As the uncertainties, more specifically, failure characteristics described in Section 4.3.1 are represented by *GFMs*, however, appropriate values of α and γ need to be determined before valid *RoS* can be estimated. Therefore, the next paragraph applies the evaluation approach described in Section 3.4.4 to derive the appropriate values before the effect of different uncertainties on the estimated *RoS* is discussed in the following paragraph.

Determining the Values of α and γ

The representation of failure characteristics by a *GFM* provides detailed information about the impairments to anticipate for sensor observations. To be applicable during the estimation of *RoS*, however, intervals stating the minimal and maximal failure amplitudes are required. Thus, the approach of Section 3.2 is applied to convert a *GFM* to an interval valid for a given time horizon $\mathbb{K}$ and domain $\mathbb{O}$.

While information about the time- and value-correlations of the failure characteristics

Tab. 4.8.: Choosing α and γ values for interval calculation employed during *RoS* estimation.

Failure Model	$d_{\mathcal{I}}(\alpha = \gamma = 1.0)$	$d_{\mathcal{I}}(\alpha = \gamma = 0.95)$	$d_{\mathcal{I}}(\alpha = \gamma = 0.75)$
$\mathcal{FM}_N$	0.24	0.15	-0.13
$\mathcal{FM}_O$	0.076	0.032	-6.7e+23
$\mathcal{FM}_{NO}$	0	0	0
$\mathcal{FM}_{PO}$	0	0	0
$\mathcal{FM}_{NS}$	0.1	0.1	0.1
$\mathcal{FM}_{PS}$	0.1	0.1	0.1

represented by the *GFM* is lost during the conversion process, specifying $\mathbb{K}$ and $\mathbb{O}$ with respect to the discretization of the state space $\mathbf{X}_s$ enables limiting the loss. Thus, for estimating an *RoS* the domain $\mathbb{O}$ is set to $[\mathbf{x} - 0.5\mathbf{X}_s, \mathbf{x} + 0.5\mathbf{X}_s]$ while $\mathbb{K}$ is set to cover the considered time horizon ($\mathbb{K} = [0\,\mathrm{s}, 10.0\,\mathrm{s}]$ for the inverted pendulum). Although information about possible time-correlations of failure amplitudes is therefore lost as the entire time horizon has to be considered, value-correlations are retained depending on the discretization of the state space.

Central to the conversion are the parameters α (Eq. (3.21)) and γ (Eq. (3.45)), which determine the range of the extracted intervals. On the one hand, the greater the range of the intervals, the more severe are the represented uncertainties which result in a greater set of uncertain states in the estimated *RoS*. Therefore, from the perspective of *RoS*, small intervals are favorable. On the other hand, the intervals have to represent the minimal and maximal failure amplitudes that may occur and can therefore not underestimate the range as this may result in an *RoS* not taking the required uncertainties into account. These perspectives are balanced by adjusting the parameters α and γ.

In the endeavor of determining the optimal value for the inverted pendulum and the failure models defined in Section 4.3.1, intervals for each failure type are extracted for values of $\alpha, \gamma \in \{1.0, 0.95, 0.75\}$ and compared to failure amplitudes observed during simulation of these. Using the minimal distance to the interval borders (Eq. (3.66)), the over- or underestimation of the actual failure amplitudes is assessed. Table 4.8 lists the results. While the failure amplitudes of the positive and negative constant Offsets can be determined precisely as they are purely deterministic, the intervals extracted from positive and negative Spike failures are always overestimating their magnitude. This is due to the time- and value-correlations of their scaling distribution which causes non-zero Lipschitz constants during the sampling-based interval extraction. In contrast, the intermittent Offset and the Noise failure type comprise non-deterministic scaling distributions. Therefore, by reducing the values of α and γ, the range of failure amplitudes is reduced. For values of $\alpha = \gamma = 0.75$, the interval underestimates their magnitude, which means that values of $\alpha = \gamma = 0.95$ have to be considered. As this value balances the tightness of the extracted intervals with the safety aspect of overestimating the range of possible failure amplitudes, the value will be used for analyzing the effect of different failure models on the estimation of *RoS*.

Tab. 4.9.: Analyzing the effect of different failure models on the estimation of *RoS*.

Failure Model	$r_{\mathcal{X}_\tau}$	$r_{\mathcal{H}}$	$\max(d_{\hat{\mathcal{V}}(c_{max})})$	**Is Safe?**
$\mathcal{FM}_{\{N\}}$	0.028	0.099	0.001	True
$\mathcal{FM}_{\{O\}}$	0.028	0.099	0.001	True
$\mathcal{FM}_{\{N,O\}}$	0.000	0.000	0.000	False
$\mathcal{FM}_{\{NO\}}$	0.028	0.099	0.000	True
$\mathcal{FM}_{\{PO\}}$	0.028	0.099	0.005	True
$\mathcal{FM}_{\{NS\}}$	0.028	0.099	0.001	True
$\mathcal{FM}_{\{PS\}}$	0.028	0.099	0.001	True
$\mathcal{FM}_{\{NO,PO\}}$	0.028	0.099	0.001	True
$\mathcal{FM}_{\{NS,PS\}}$	0.000	0.000	0.000	False
$\mathcal{FM}_{\{N,NO,PO\}}$	0.028	0.099	0.003	True

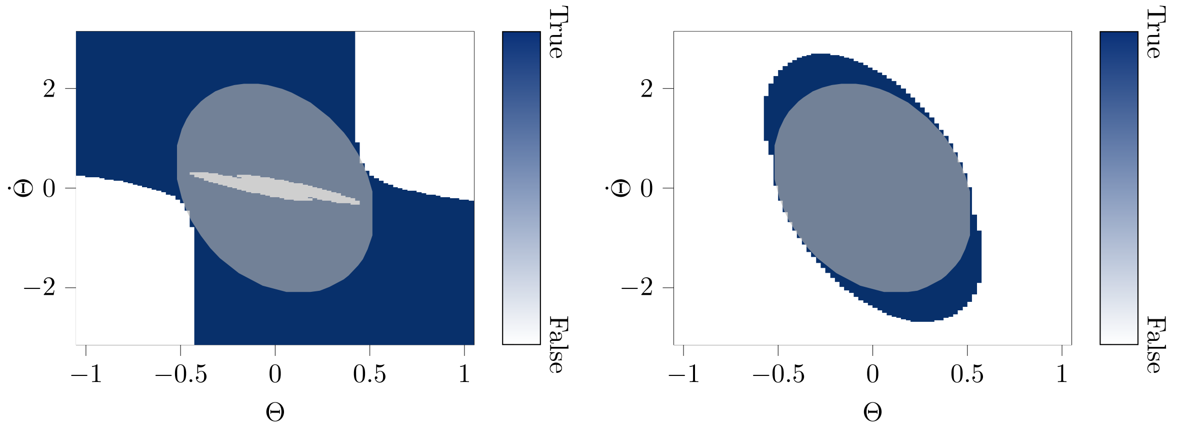
Analyzing Regions of Safety

With the values of $\alpha = \gamma = 0.95$ set, the effect of the considered failure models (cf. Table 4.4) on the estimated *RoS* can be examined. Similar as before, the next paragraph describes the metrics employed for analyzing these while the following paragraph discusses the results.

Procedure The central idea of *RoS* is to estimate the set of states for which it can be guaranteed that the system will adhere to its safety requirements and that it will not leave this set. In the endeavor of evaluating whether or not this is the case for the estimated *RoS* when considering the failure models, the simulation-based check of the safety requirement is employed once again. This time, each state within the estimated *RoS* was considered as a starting state of $N_{sim} = 20$ simulations. In this way, the stochastic nature of the considered failure models is taken into account. Moreover, the maximal distance $\max(d_{\hat{\mathcal{V}}(c_{max})})$ of states observed during these simulations to the estimated *RoS* is calculated again. Finally, the ratios $r_{\mathcal{H}}$ (cf. Eq. (4.25)) and $r_{\mathcal{X}_\tau}$ (cf. Eq. (4.22)) are reconsidered as well to assess the estimated *RoS* itself.

Results Table 4.9 states the obtained results according to these evaluation metrics. Apart from $\mathcal{FM}_{\{N,O\}}$ and $\mathcal{FM}_{\{NS,PS\}}$, *RoS* could be estimated successfully for the controller when considering any of the failure models. Despite their modeled failure characteristics affecting the simulated sensor observations, the safety requirement was maintained during the simulation and the maximal distance $\max(d_{\hat{\mathcal{V}}(c_{max})}) = 0.005$ is less than 0.017[4] and thereby underlines that the pendulum indeed remained within

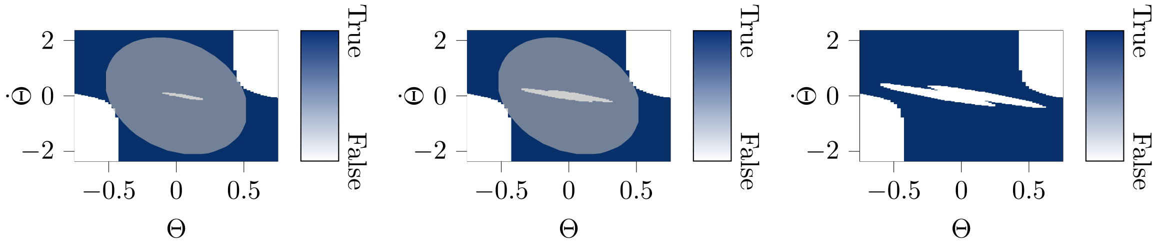
[4] $\sqrt{2 \cdot 0.025^2} \cdot 0.5 = 0.017$ is half of the Euclidean distance between to neighboring states in the specified state space.



(a) Fulfillment of *RoA* condition when considering $\mathcal{FM}_{\{O\}}$.

(b) Fulfillment of *RoS* condition when considering $\mathcal{FM}_{\{O\}}$.

Fig. 4.17.: Examining the effect of $\mathcal{FM}_{\{O\}}$ on the fulfillment of the *RoA* and *RoS* condition at individual states and the resulting *RoS*. The legend is given in Fig. 4.16d.



(a) $\mathcal{FM}_{\{NO\}}$

(b) $\mathcal{FM}_{\{PS\}}$

(c) $\mathcal{FM}_{\{NS,PS\}}$

Fig. 4.18.: Comparing the effect of failure models $\mathcal{FM}_{\{NO\}}$, $\mathcal{FM}_{\{PS\}}$, and $\mathcal{FM}_{\{PS,NS\}}$ on the fulfillment of the *RoA* condition and the estimated *RoS*. The legend is given in Fig. 4.16d.

the estimated *RoS*. Therefore, the guarantee provided by Theorem 4.1 is successfully supported by the results.

The fulfillment of the *RoA* and *RoS* conditions obtained for the example of the intermitten Offset (cf. Fig. 4.17) is in line with these results. Although Fig. 4.17a shows that the controller is unable to provide stabilizing control actions for states surrounding the stability point, it is demonstrated by the fulfillment of the *RoS* condition (cf. Fig. 4.17b) that the system will not leave the overall *RoS*. Therefore, estimating the same is possible despite the uncertain states.

Similar to the *CLF* candidate ⑧ discussed in association with Fig. 4.14, the blue area of states fulfilling the *RoS* condition extends beyond the estimated *RoS*.

Examining the examples of $\mathcal{FM}_{\{NO\}}$, $\mathcal{FM}_{\{PS\}}$, and $\mathcal{FM}_{\{PS,NS\}}$ enables comparing successful and unsuccessful estimations of *RoS*. In Fig. 4.18a, the effects of constant negative Offsets are visualized regarding the fulfillment of the *RoA* condition. As one can see, the area of uncertain states within the *RoS* is shifted only slightly. Therefore, the direction of the magnitude of failure amplitudes does not seem to affect the estimation of the *RoS*. However, comparing these results to $\mathcal{FM}_{\{PS\}}$ shows that the magnitude itself affects the region of uncertain states. The time- and value-correlations of $\mathcal{FM}_{\{PS\}}$ cause overall increased failure amplitudes which in turn cause the controller to calculate inappropriate control actions for a greater set of states surrounding the stability point. In other words, the area of uncertain states increases. Consequently, for $\mathcal{FM}_{\{NS,PS\}}$, where the most severe failure amplitudes are to be expected, stability of all states in $\mathcal{H}_V(c_{min}, c_{max})$ can not be shown anymore. Therefore, the *RoA* condition of Theorem 4.1 is not fulfilled and no *RoS* can be estimated.

Moreover, from the comparison of $\mathcal{FM}_{\{NO\}}$, $\mathcal{FM}_{\{PS\}}$, and $\mathcal{FM}_{\{PS,NS\}}$ it can be seen that the considered failure models affect only states near the stability point but do not affect the fulfillment of the *RoA* condition for states of approx. $|\Theta| \geq 0.45\,\text{rad}$. This is caused by the control policy. Incorrect sensor observations implied by the considered failure models cause the controller to calculate a control action based on an incorrect error value (cf. Eqs. (4.3) and (4.12)). In the worst case, this means that the control policy perceives a state $\hat{\mathbf{x}}$ which requires a less rigorous action than the actual state $\mathbf{x}$ would require. In contrast, for states with approx. $|\Theta| \geq 0.45\,\text{rad}$, both, the incorrectly perceived state $\hat{\mathbf{x}}$ and the actual state $\mathbf{x}$ require the maximal control action for stabilizing the pendulum. Therefore, the sensor failures do not affect the calculated control action.

This is precisely the reason why the values of $r_{\mathcal{H}}$ and $r_{\mathcal{X}_\tau}$ are either zero in case no *RoS* could be estimated or 0.099 and 0.028 for all other failure models. If a sufficient set of stabilizing states can be found, the estimated *RoS* is limited only by states for which the pendulum can not be stabilized due to the limited torque provided by the motor, not because of the considered sensor failures.

In summary, using Algorithm 1, *RoS* can be estimated for the inverted pendulum. In case the failure characteristics described by the considered *GFM* are tolerable by the employed control policy, the estimated *RoS* guarantees that the inverted pendulum remains within the set of states while the algorithm returns the empty set if the specified characteristics are not tolerable.

4.4. Summary

This chapter started in the endeavor of fulfilling Objective 1.1, which asks for a run-time safety assessment method applicable to dynamically composed systems. For that, it builds upon the discussion on approaches available in the literature, cf. Section 2.1.3. While it was found that methods aiming at a system's functional level are available, the same are missing at a system's technical level where shared data is processed to derive control actions. Solely the approach of *Region of Attraction (RoA)* was found to be applicable at run-time, but does not cover the use of shared data.

Serving as a starting point, the chapter firstly extended the system model to facilitate representing different sources of uncertainty, cf. Section 4.1. Using this model for a preliminary evaluation of the inverted pendulum problem revealed that estimating a controller's *RoA* is not possible while considering sensor failures. This is due to its requirement of asymptotic stability which can not be satisfied for states near the targeted stability point. However, safety does not require asymptotic stability but a guarantee that a system does not evolve to an unsafe state.

With this shift of perspective in mind, Theorem 4.1 was proposed. Building upon the idea of stabilizing states as provided by the concept of *RoA*, the theorem takes the possibility of states for which stability is uncertain into account. The provided *Region of Safety (RoS)*, therefore, guarantees that the system under consideration will not leave the specified set of states under a given control policy and assumed uncertainties. By utilizing the extended system model, not only failure characteristics of shared data can be taken into account but also environmental disturbances, model uncertainties, failures of internal sensors, and actuator failures. Therefore, varying combinations of those variables can be examined as well. Finally, Algorithm 1 is presented as a means of estimating an *RoS*.

The presented concept was analyzed qualitatively in a first step by reviewing it regarding the predefined criteria of Section 1.3.1. In a second step, the already introduced example of the inverted pendulum was used to evaluate it and examine the effect of different uncertainties on estimated *RoS*.

The evaluation revealed challenges in setting up the estimation of *RoS* for a system under consideration. Firstly, different aspects have to be taken into account while constructing an appropriate *CLF* to assess the criticality of states. Not only does it need to be ensured that all relevant parameters are reflected, but the *CLF* has to be aligned with the employed control policy as well. For the inverted pendulum, this means that the angular velocity $\dot{\Theta}$ has to be respected as it is the variable that is influenced by the controller. Opposed to that, the main criterion for the system's safety is the pendulums angle Θ. Moreover, it was shown that the choice of λ_c, a parameter of Algorithm 1, affects the validity of the estimated *RoS* as well.

Nevertheless, the evaluation showed that by constructing an appropriate *CLF* and determining a valid value for λ_c, the safety of the system under consideration can be assessed successfully. On the one hand, *RoS* were successfully estimated for failure models stating failure characteristics tolerable by the defined control policy. On the other hand, failure characteristics, for which the safety of the system could not be shown with sufficient confidence, were rejected.

Therefore, by executing these calculations at run-time, a certification of whether or not a safety function can adhere to its required safety performance is possible. This underlines the fulfillment of the predefined criterion of *Run-Time Certification*. Moreover, the design of an appropriate *CLF* and the parameterization of the corresponding algorithm are done at a system's design-time. As such, sufficient confidence can be gathered that the *CLF* reflects the criticality of states and is aligned with the employed control policy. In other words, the elements ultimately deciding on whether to use shared data or not can be shown to be functionally correct at design-time such that the result determined at run-time is valid. This fulfills the eponymous criterion of *functional correctness*.

As the analysis of a shared failure model is facilitated by the extended system model, it is shown that the concept of *RoS* satisfies the predefined criteria and fulfills Objective 1.1.

5. Evaluation and Integration

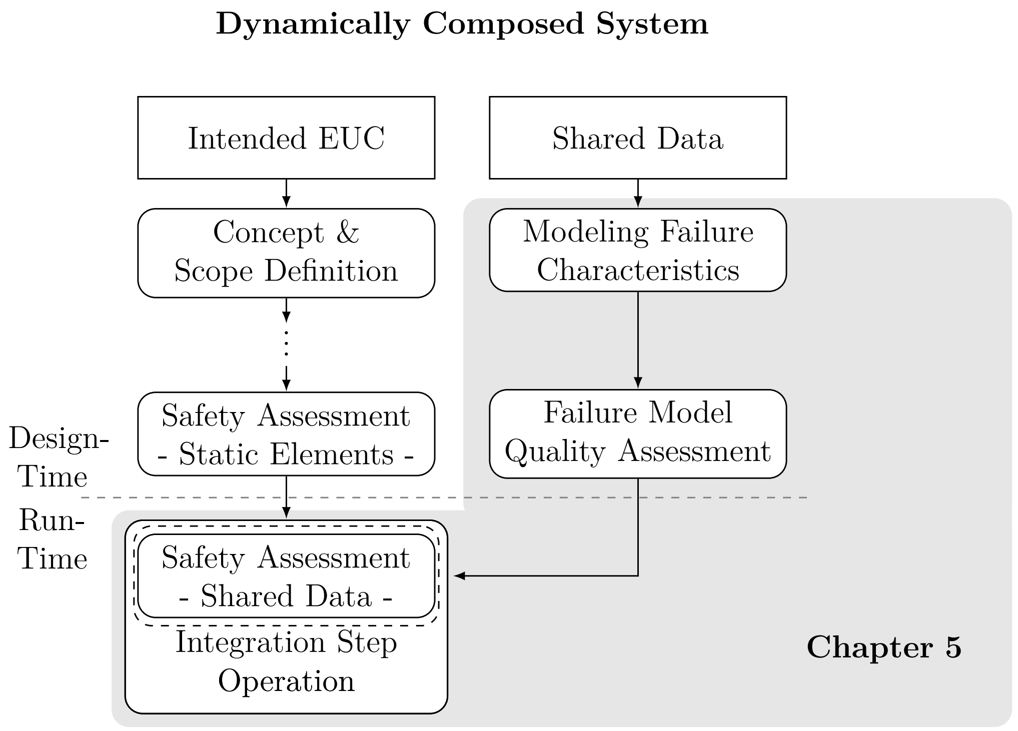

Dynamically Composed System
Intended EUC
Shared Data
Concept & Scope Definition
Modeling Failure Characteristics
Safety Assessment - Static Elements -
Failure Model Quality Assessment
Design-Time
Run-Time
Safety Assessment - Shared Data -
Integration Step Operation
Chapter 5

Fig. 5.1.: Simplified safety process from Fig. 1.9 showing the objective addressed in this chapter.

The previous chapters introduced the main concepts of this thesis, the *Generic Failure Model (GFM)* and *Region of Safety (RoS)*. The former addresses Objective 1.2 and is thereby a prerequisite for the fulfillment of Objective 1.1, which is addressed by the latter. Together, both concepts shall enable a run-time safety assessment for handling shared data in safety-critical control systems.

Opposed to this goal, the previous evaluations either focused only on the *GFM* (cf. Section 3.4) or did not consider shared data (cf. Section 4.3). Thus, this chapter aims at evaluating both concepts in combination, cf. Fig. 5.1. Specifically, it aims at (i) evaluating *RoS* as a run-time safety assessment method in dynamically composed systems where failure characteristics of shared data are represented by a *GFM* and (ii) connecting both concepts to approaches providing run-time safety assessment at the functional level, that is, to the idea of *Level of Service (LoS)*.

For that, Section 5.1 takes upon the initial discussion in Chapter 1 and derives the use case of a delivery robot navigating in a smart warehouse where it shares its position data with a second robot to avoid collisions. This clarifies the allocation of concerns. While the *GFM* will be used to represent failure characteristics of the shared data, the concept of *RoS* will be used to examine the same for assessing whether or not the employed collision avoidance policy tolerates the specified uncertainties and maintains the robot's safety. However, as a robot can (in 2D) bypass an obstacle either on its left or on its right side, this policy has two stability points. This contradicts the application of *RoS*, which supports only one stability point identified by the global minimum of the applied *CLF*. Thus, a use-case-specific fusion strategy is proposed that considers the *RoS* estimated for each stability point in separation and generates a fused *RoS*.

Afterward, the first step towards implementing the use case is to define the failure model of shared data. Assuming that the robot's position is provided by camera-based marker detection, Section 5.2 discusses the setup used to obtain real-world observations $\hat{\mathbf{o}}_k$ along with reference data $\mathbf{o}_k$ from which failure amplitudes $f(k, \mathbf{o}_k)$ can be calculated. These are examined by the processing chain presented in Section 3.3 to generate a *GFM*, which is manually adjusted to improve its performance. The failure model is then assumed to state the failure characteristics of shared data.

As such, it is used in Section 5.3 to estimate the *RoS* of the defined collision avoidance policy. For that, a *CLF* applicable to both of its stability points is defined first. Afterward, it is shown that the run-time safety assessment based on fused *RoS* successfully enables maintaining the safety of the robots when sharing their position data. Moreover, it facilitates choosing an appropriate parameterization of the employed control strategy at run-time to adapt to the provided quality of shared data. This is shown in a final experiment which shows how the concept of *RoS* can be used to realize the idea of *LoS*. Finally, the chapter is summarized and concluded in Section 5.4.

5.1. Multi-Robot Collision Avoidance – Reconsidering the Smart Warehouse Use Case

In Section 1.1, the paradigm of *Industry 4.0* was examined and its impact on statically composed systems (cf. Definition 1.1) becoming dynamically composed systems (cf. Definition 1.2) was reviewed in light of currently employed safety processes. Resulting in the objectives of this Thesis, the discussion evolved around the example of a smart warehouse where mobile robots organize and manage the storage system. Instead of humans covering long distances to fulfill orders or replenish the warehouse's racks, mobile robots move the racks to dedicated picking or replenishing stations where humans obtain or store items in the racks.

In the endeavor of further automating this process, the idea emerges to accept delivery of goods to the warehouse by autonomous mobile robots as well. This entails, however, that such a delivery robot temporarily integrates with the automated warehouse. It may be dynamically instructed by the warehouse to place its goods at different goal positions, depending on the type of transported goods and the current storage organization. While this is resource-efficient from the perspective of the warehouse, it requires the delivery robot to safely operate in an area shared with other mobile robots. Most importantly, it would be required that the robots do not collide with each other, which could be supported by sharing their position data.

In accordance with IEC 61508, such a collision avoidance algorithm is considered a safety function. More specifically, a safety function relying on shared data. As a safety function, it has to be shown that it adheres to its assigned safety performance. Following the discussion in Section 1.3, *SIL* 4 is assumed here.

Therefore, the use case of a collision avoidance controller employed for robots sharing their position data precisely resembles the use case the concepts of this thesis were designed for and is therefore used to evaluate the same.

For that, the assumptions made for the scenario are discussed in the next subsection before two experimental designs aligned with this chapter's goals are derived in Section 5.1.2. The first experiment, referred to as the *circle scenario*, is designed to examine whether the *RoS*-based run-time safety assessment indeed facilitates maintaining the safety of the dynamically composed system. The second experiment, referred to as the *navigation scenario*, is designed according to the motivating use case and shows that the idea of *LoS* can be realized by building on the concept of *RoS*. Having the experiment in place, Section 5.1.3 introduces the kinematic model for the assumed robots. Based on that, the control policy for driving a robot to its target position while avoiding collisions is discussed in Section 5.1.4. It is complemented by a discussion on how to model assumptions made about the behavior of a second robot in Section 5.1.5. Building upon the defined control policy, the section is concluded by defining the use-case-specific strategy for fusing *RoS* estimated for separate stability points.

5.1.1. Assumptions

For deriving the *circle scenario* and the *navigation scenario*, the presented use case has to be refined. For that, assumptions about the scenarios, the robots, and the data they share are made.

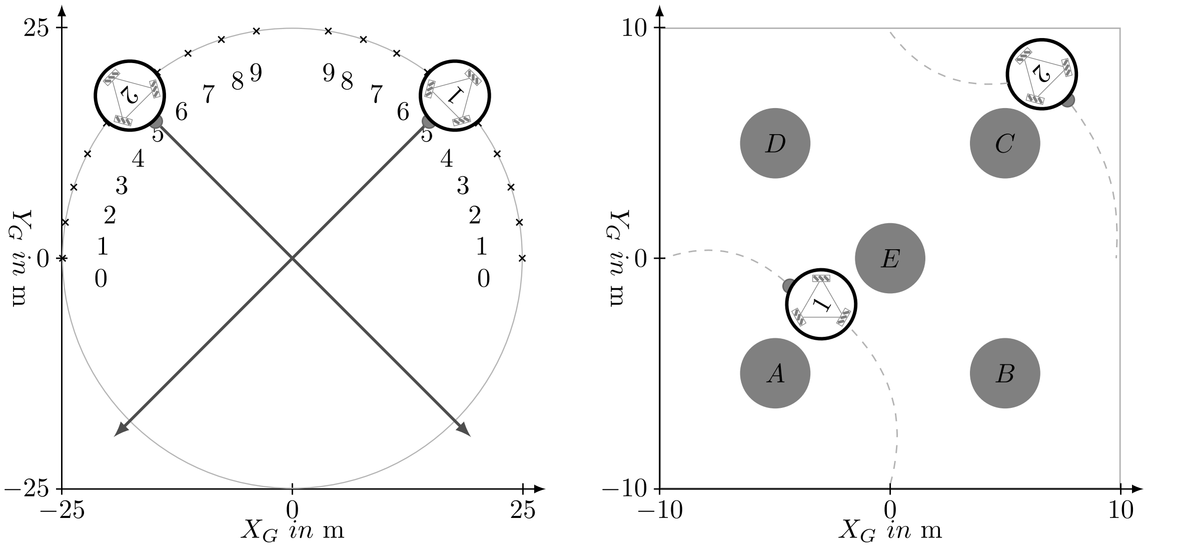
Firstly, the number of robots operating in a shared area is assumed to be exactly two while up to five obstacles are assumed. The shape of both, robots and obstacles, is abstracted to a circle of radius $r_s = 0.1\,\mathrm{m}$. The robots are assumed to have the same platform and follow the same control and collision avoidance strategy, thus, they are assumed to be identical.

While the obstacles are assumed to be static, that is, have a fixed position, the robots are assumed to have holonomic drives enabling omnidirectional kinematics. Such kinematics are commonly used in mobile robotic platforms ([110], [111]) as they facilitate the robot to rotate and move in any direction at any time.

As neither the obstacles nor the positions of the robots are known at design-time, it is assumed that both are sensed at run-time. More specifically, each robot senses its position and the position of the nearest obstacle using its internal sensors, which are modeled using $s_i(\mathbf{x})$, cf. Eq. (4.5). Contrarily, the position of the other robot is provided by sharing the position data between the robots, that is, by shared data modeled by $s_s(\mathbf{x})$, cf. Eq. (4.7).

Next to the continuously changing position data, the robots additionally are assumed to share metadata about their maximal translational and rotational velocities, as well as the distance both robots aim at maintaining to each other. These metadata inform about the behavior of the robots when engaging in a collision avoidance maneuver and enable modeling the same.

As these metadata do not represent changing information but describe the general specification of the robotic system, it is assumed to be free of failures or uncertainties. Similarly, it is assumed that observations provided by internal sensors are not affected by any failure characteristics as well. This is as current system development and safety processes (cf. Chapter 1) are capable of taking failure characteristics of static system



(a) Schematic illustration of the circle scenario in which two robots are placed on a virtual circle and assigned target positions such that the trajectories intersect and collisions are provoked.

(b) Schematic illustration of the navigation scenario where two robots operate in a shared area with static obstacles.

Fig. 5.2.: Representations of the navigation and circle scenario.

components (i.e. internal sensors) into account and enable tuning processing chains such that their remaining uncertainties are negligible with respect to the scope of this work.

Contrarily, the central concepts to be evaluated in this chapter focus on shared data. Thus, failure characteristics are assumed to affect position data shared between robots.

Finally, it is assumed that robots do not intentionally share incorrect data, that is, byzantine errors are excluded from these considerations [112]. This is as the focus of this work is on safety and thereby excludes security considerations aiming at the correctness and trustworthiness of shared data.

5.1.2. Design of the Circle and Navigation Scenarios

The previously formulated assumptions form the basis for the evaluation scenarios of this chapter. Both are visualized in Fig. 5.2 and are aligned with the initially formulated goals. While the circle scenario aims at evaluating whether or not a run-time safety assessment can be realized using the concept of *RoS*, the navigation scenario shall underline how the concept can be integrated with the state-of-the-art approach of *LoS*. With these goals in mind, the following paragraphs briefly introduce both scenarios.

The Circle Scenario

The central challenge addressed by this Thesis is to ensure safety despite using shared data in safety functions. For that, the concepts of *GFM* and *RoS* are presented. With regard to the use case discussed in this chapter, the safety function using shared data is the collision avoidance strategy. Consequently, the first evaluation scenario, called the *circle scenario*, aims at examining the effectiveness of the concepts in guaranteeing the safety performance of this safety function.

For that, the scenario assumes two robots placed on a virtual circle, cf. Fig. 5.2a. Each robot is assigned a target position at the opposite side such that their trajectories intersect. By varying the starting positions of the robots, different incident angles are provoked, facilitating the examination of different situations with potential collisions.

According to the presented concepts, the controller's RoS will be estimated with respect to its parameters and the specified failure model of shared data. A non-empty, valid RoS will indicate a safe parameterization while the estimation of an empty set indicates unsafe parameters.

When parameterizing the collision avoidance strategies with values of D_{min} deemed safe by this RoS-based analysis, it is expected that collisions are successfully prevented. Contrarily, by setting D_{min} to values deemed unsafe, collisions are provoked and are expected to be observed during simulation.

The Navigation Scenario

The circle scenario provides empirical evidence that the collision avoidance controller indeed maintains the safety of the system when using shared data as long as a valid RoS can be estimated. As the concept of RoS targets run-time safety assessment at the technical level, that is, a safety function implemented as a control system, integration with state-of-the-art approaches aiming at the functional level is the next step. To underline that this is possible with the presented concept, the *navigation scenario* aims at integrating the RoS estimation with the idea of LoS.

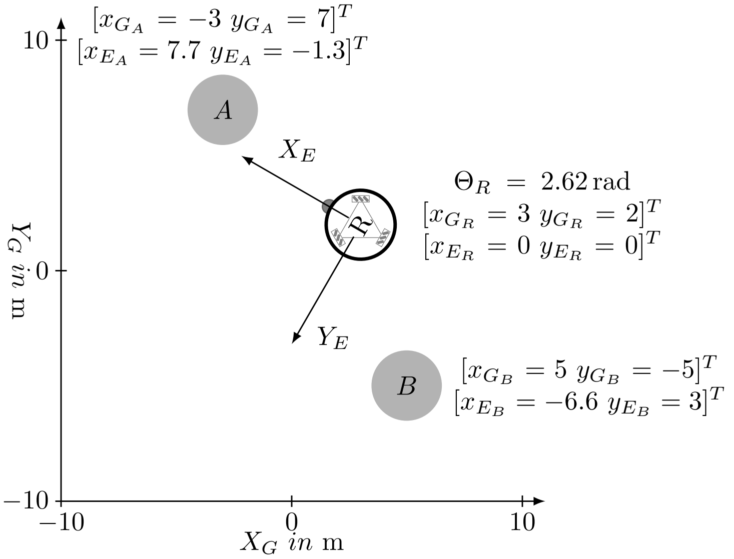
For that, the navigation scenario is motivated by the use case of the smart warehouse and assumes two robots navigating in a shared area with static obstacles. As illustrated in Fig. 5.2b, five obstacles (A-E) are considered. The robots are assigned random target positions within the rectangular area, requiring them to prevent collisions with each other and obstacles simultaneously. However, the robots are allowed to move outside the area for this sake as well.

During simulation, different failure characteristics affecting the shared data are assumed. In correspondence to these, the robots are required to adapt their minimal distance D_{min} to each other. This adaptation will be realized using the concept of LoS.

Assuming different, predefined values for D_{min} (each representing a single LoS), the concept of RoS is used to analyze whether or not safety can be maintained for each value. Then, depending on the failure model of the shared data, the minimal value for D_{min}, for which safety can be guaranteed, is chosen. Through simulation, it shall be shown that the safety of the system is maintained when applying the lowest possible value of D_{min} for which a valid RoS can be estimated.

5.1.3. Kinematics

In the endeavor of executing the defined scenarios, the next step is the definition of a system model in accordance with Eq. (4.1) for applying Algorithm 1 and estimating RoS. For that, this section introduces the kinematics of the considered robots, that is, $f(\mathbf{x}(t), \hat{\mathbf{u}}(t))$ and $U_{actuator}(\mathbf{u}(t), t)$.



$[x_{G_A} = -3\ y_{G_A} = 7]^T$
$[x_{E_A} = 7.7\ y_{E_A} = -1.3]^T$
A
X_E
$\Theta_R\ =\ 2.62\,\mathrm{rad}$
$[x_{G_R} = 3\ y_{G_R} = 2]^T$
$[x_{E_R} = 0\ y_{E_R} = 0]^T$
R
Y_E
B
$[x_{G_B} = 5\ y_{G_B} = -5]^T$
$[x_{E_B} = -6.6\ y_{E_B} = 3]^T$
10
0
−10
−10
0
10
Y_G *in* m
X_G *in* m

Fig. 5.3.: Defining global and local coordinate system of robots. The subscript G denotes coordinates in the global coordinate system while the subscript E denotes coordinates in the local (ego) frame.

Starting with their movement in the global coordinate system (cf. Fig. 5.3) in the next subsection, the kinematics describing the movement of obstacles in the local coordinate system of a robot are introduced next. Stating the problem of collision avoidance in the local coordinate system will not only simplify defining the collision avoidance strategy but also the definition of a *CLF* and estimating an *RoS* respectively.

Global Kinematics

In accordance with the previously stated assumptions, the robots are assumed to have omnidirectional kinematics. Different mechanics for achieving these are possible, for instance using Mecanum wheels [113] in a three-wheeled robot [111]. However, the dynamics and specific mechanics are not within the scope of this thesis. Therefore, the robot is abstracted as a point mass in a 2D space, having a position of $[x_{G_R}, y_{G_R}]^T$ in the global coordinate system and an orientation of Θ_{G_R} Using the resulting state definition $\mathbf{x}_{G_R} = [x_{G_R}\ y_{G_R}\ \Theta_{G_R}]^T$, the kinematic model $\dot{\mathbf{x}}_{G_R} = f(\mathbf{x}, \hat{\mathbf{u}})$ for a robot is given in Eqs. (5.1) to (5.3). Note that the subscript G_R indicates that the position/state change of the robot is given with respect to the global coordinate system.

$$\dot{x}_{G_R} = \hat{u}_x \tag{5.1}$$

$$\dot{y}_{G_R} = \hat{u}_y \tag{5.2}$$

$$\dot{\Theta}_{G_R} = \hat{u}_\Theta \tag{5.3}$$

By neglecting the specific dynamics entailed by the mechanical components of the robot and focusing on its motions, the system model is simplified and takes only the control actions $\hat{\mathbf{u}}$ realized by the actuators into account.

Note that $\hat{\mathbf{u}}$ represents only the actions that are actually realized by the actuators, that is, the motion that could be achieved despite their inabilities, inaccuracies, and imprecisions. In correspondence with Eq. (4.1), these are modeled by the failure model of the employed actuators. Similar to the example of the inverted pendulum, the actuator failure model $U_{actuator}$ considered here limits the translational and rotational velocities, cf. Eqs. (5.4) to (5.6).

$$\hat{u}_x = \begin{cases} \frac{u_x}{\|[u_x \; u_y]^T\|} U_{T_{max}}, & \text{for } \left\|[u_x \; u_y]^T\right\| \geq U_{T_{max}} \\ u_x, & \text{else} \end{cases} \tag{5.4}$$

$$\hat{u}_y = \begin{cases} \frac{u_y}{\|[u_x \; u_y]^T\|} U_{T_{max}}, & \text{for } \left\|[u_x \; u_y]^T\right\| \geq U_{T_{max}} \\ u_y, & \text{else} \end{cases} \tag{5.5}$$

$$\hat{u}_\Theta = \min(\max(\hat{u_\Theta}, -U_{R_{max}}), U_{R_{max}}) \tag{5.6}$$

In other words, the failure model represents a saturation of both velocities. While the rotational velocity is saturated using a min-max operation (cf. Eq. (5.6)), the translational velocity is saturated by scaling both directional scalars u_x and u_y. Thus, the direction of the resulting translational vector is maintained but its magnitude is limited by $U_{T_{max}}$.

Local Kinematics

The previously defined kinematics state the motion of a robot considered as a point mass in the global coordinate system. This allows simulating one or more robots and their trajectories with respect to the employed control actions and is, therefore, a basic requirement for evaluating the presented concepts.

However, in the endeavor of defining a collision avoidance policy with which the simulated robots can navigate safely, a change of perspective is helpful. More specifically, as planning collision avoidance in the global coordinate system requires the control policy to predict not only the movement of the robot but also the relative movement of the obstacles in its vicinity, reformulating the problem in the robot's local coordinate system (also called the robot's ego coordinate system) simplifies the task. The robot is assumed to be at the coordinate system's origin, that is, $\mathbf{x}_{E_R} = [x_{E_R} = 0 \; y_{E_R} = 0]^T$ while the obstacles are described by their position relative to the robot. Note that now, the subscript E_R indicates that the position of the robot is given with respect to the local coordinate system.

For the sake of illustration, Fig. 5.3 compares both perspectives. For that, the position data of both obstacles A and B as well as of the robot R are stated for the global coordinate system (subscript G) and for the local coordinate system (subscript E).

For modeling the kinematics of the robot the changed perspective has to be taken into account: the robot always remains in the origin of the coordinate system. Thus, instead of stating the state change of the robot, the local kinematics state the change $\dot{\mathbf{x}}_{E_O} = [\dot{x}_{E_O} \; \dot{y}_{E_O}]^T$ of the position of the considered obstacle, cf. Eqs. (5.7) to (5.10).

$$\Theta_{E_O} = atan2(y_{E_O}, x_{E_O}) \tag{5.7}$$

$$\dot{\Theta}_{E_O} = \left\|[x_{E_O} \; y_{E_O}]^T\right\| \hat{u}_\Theta \tag{5.8}$$

$$\dot{x}_{E_O} = -\sin(\Theta_{E_O})\dot{\Theta}_{E_O} - \hat{u}_x \tag{5.9}$$

$$\dot{y}_{E_O} = \cos(\Theta_{E_O})\dot{\Theta}_{E_O} - \hat{u}_y \tag{5.10}$$

Eq. (5.7) and Eq. (5.8) state the angle and angular velocity of the obstacle in relation to the robot as a result of the robot's rotation $\hat{u}_\Theta$. Based on these, Eq. (5.9) and

Tab. 5.1.: Overview of configuration parameters of the robots' control policy.

Parameter	Value	Description
K_P	2	P-Value for the P-controller of the goal-finding strategy
K_C	2	P-Value for the P-controller keeping the robot on a circular trajectory while driving around an obstacle
K_O	1	P-Value for the P-controller keeping the distance between the robot and the obstacle at D_{min}
ϕ	0.75π	Minimal Angle between the obstacle and goal before collision avoidance can be disengaged
D_{min}		Minimal distance to keep between robot and obstacle during collision avoidance

Eq. (5.10) derive the position change of the obstacle and take the movement of the robot into account.

The first advantage of using the local coordinate system here is the reduction of the state space. Instead of having three variables, only the position, that is, x_E and y_E are required.

5.1.4. Control Policy

Based on the kinematics defined in the robot's local coordinate system, the control strategy for a single robot can be defined. It comprises a goal-finding strategy to reach the robot's goal position $[x_{G_g}\ y_{G_g}\ \Theta_{G_g}]^T$ and the collision avoidance strategy employed for bypassing obstacles on the way, cf. Eq. (5.11).

$$\mathbf{u} = \pi(\hat{\mathbf{o}}_i, \hat{\mathbf{o}}_s) = \begin{cases} \pi_{goal}(\hat{\mathbf{o}}_i), & \text{if } \mathit{engaged} = 0 \\ \pi_{collision}(\hat{\mathbf{o}}_i, \hat{\mathbf{o}}_s), & \text{if } \mathit{engaged} = 1 \end{cases} \tag{5.11}$$

As indicated by the binary variable *engaged* $\in \{0, 1\}$, the overall control policy acts according to the goal-finding strategy π_{goal} as long as collision avoidance is not engaged. Consequently, in case an obstacle or other robot is within the robot's vicinity, collision avoidance $\pi_{collision}$ is engaged.

To realize this switching as well as for calculating the control actions according to both strategies, internal and external sensor observations, that is, $\hat{\mathbf{o}}_i$ and $\hat{\mathbf{o}}_s$ are required.

Correspondingly, the next subsection briefly describes the sensor models providing the observations. Afterward, the goal-finding strategy and the collision avoidance strategies are discussed.

Within this discussion, R_1 denotes the ego robot, that is, the robot executing the control policy, while R_2 denotes the robot sharing its position data.

Sensor Model

Before the control policy can be defined, its inputs have to be specified. In this endeavor, the observations $\hat{\mathbf{o}}_i$ and $\hat{\mathbf{o}}_s$ need to be described.
Starting with the internal sensors, the previously formulated assumption is leveraged. As the focus of this thesis is on shared data, a simplified model is assumed for the robot's internal sensor system. It directly provides position observations of the robot's nearest obstacle ($\mathbf{x}_{E_{OS}}$) as well as an estimate of the robot's pose $\mathbf{x}_{R_1}$ in the global coordinate system, cf. Eq. (5.12).

$$\mathbf{o}_i = [\mathbf{x}_{E_{OS}}\ \mathbf{x}_{G_R}]^T = [x_{E_{OS}}\ y_{E_{OS}}\ x_{G_{R_1}}\ y_{G_{R_1}}\ \Theta_{G_{R_1}}]^T = s_i(\mathbf{x}) \tag{5.12}$$

$$\hat{\mathbf{o}}_i = \mathbf{o}_i \tag{5.13}$$

Moreover, it is assumed that these observations are perfect, that is, $U_{sensor}(\mathbf{o}) = 0$, cf. Eq. (5.13).
In contrast, observations provided through shared data are assumed to be incorrect and imprecise.

$$\mathbf{o}_s = [x_{G_{R_2}}\ y_{G_{R_2}}]^T = s_s(\mathbf{x}) \tag{5.14}$$

$$\hat{\mathbf{o}}_s = \mathbf{o}_s + U_{shared}(\mathbf{o}_s, t) \tag{5.15}$$

The information shared by the second robot is its position data $x_{G_{R_2}}$ and $y_{G_{R_2}}$ in the global coordinate system. It is assumed that the failure model $U_{shared}(\mathbf{o}_s, t)$ is shared along and analyzed during run-time safety assessment. The failure model in this scenario will be discussed in detail in Section 5.2.

Goal-Finding Strategy

From a performance perspective, the overall goal of the robot at state $\mathbf{x}_{G_R}$ is to navigate to its goal position $[x_{G_g}\ y_{G_g}\ \Theta_{G_g}]^T$. Depending on the environment and distance the robot has to travel versatile algorithms can be employed for solving this task. One of the first algorithms to be proposed was the so-called *Bug algorithms* [114]. Originated in maze solving algorithms, they are rule-based algorithms to navigate a robot to its target position without the need for a map of the environment. Motivated by these algorithms, a P-controller is employed in this work, cf. Eqs. (5.16) to (5.18).

$$\begin{bmatrix} x_e \\ y_e \end{bmatrix} = \begin{bmatrix} \cos(\Theta_{G_{R_1}}) & -\sin(\Theta_{G_{R_1}}) \\ \sin(\Theta_{G_{R_1}}) & \cos(\Theta_{G_{R_1}}) \end{bmatrix} \cdot \begin{bmatrix} x_{G_g} - x_{G_{R_1}} \\ y_{G_g} - y_{G_{R_1}} \end{bmatrix} \tag{5.16}$$

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} x_e \\ y_e \end{bmatrix} \cdot K_P \tag{5.17}$$

$$u_\Theta = (\Theta_{G_g} - \Theta_{G_{R_1}}) \cdot K_P \tag{5.18}$$

$$\mathbf{u}_{goal} = [u_{x_{goal}}\ u_{y_{goal}}\ u_{\Theta_{goal}}]^T = \pi_{goal}(\mathbf{o}_i) \tag{5.19}$$

Using the position estimate $\mathbf{x}_{G_R}$ provided by the internal sensors, the translational difference between the robot's current position and the goal position is calculated and transformed into the local coordinate system, cf. Eq. (5.16). The error is then multiplied

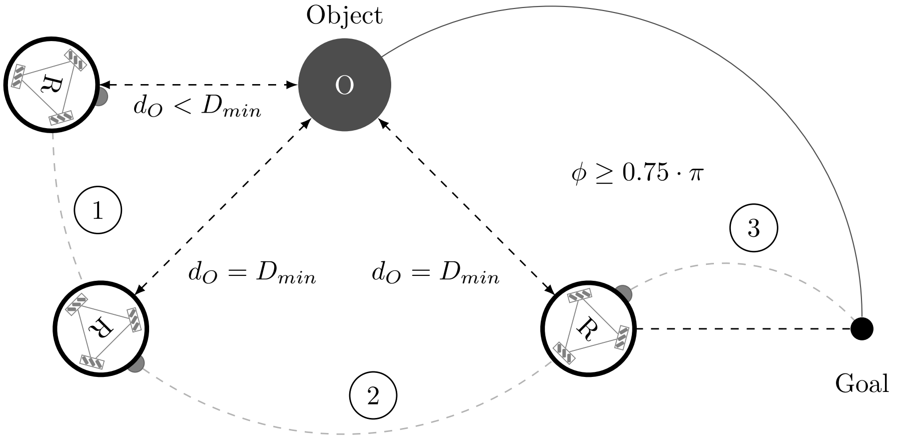


Fig. 5.4.: Phases of the collision avoidance strategy.

by the P-Gain K_P to derive the translational velocities u_x and u_y given in the local coordinate system. Similarly, the difference in the current and targeted orientation is combined with the P-Gain K_P to obtain the rotational velocity u_Θ. Together, these velocities form the control action $\mathbf{u}_{goal}$ provided by the first part of the control policy, the goal-finding strategy $\pi_{goal}(\hat{\mathbf{o}}_i)$.

Collision Avoidance

The control actions $\mathbf{u}_{goal}$ provided by the goal-finding strategy are applicable only if no obstacle is in the robot's vicinity. Contrarily, the considered use case of a smart warehouse, in which robots operate in a shared area, requires avoiding collision with these as well as with obstacles. Therefore, a collision avoidance strategy $\pi_{collision}$ has to be defined, cf. Eq. (5.20).

$$\mathbf{u}_{collision} = [u_x \; u_y \; u_\Theta]^T = \pi_{collision}(\hat{\mathbf{o}}_i, \hat{\mathbf{o}}_s) \tag{5.20}$$

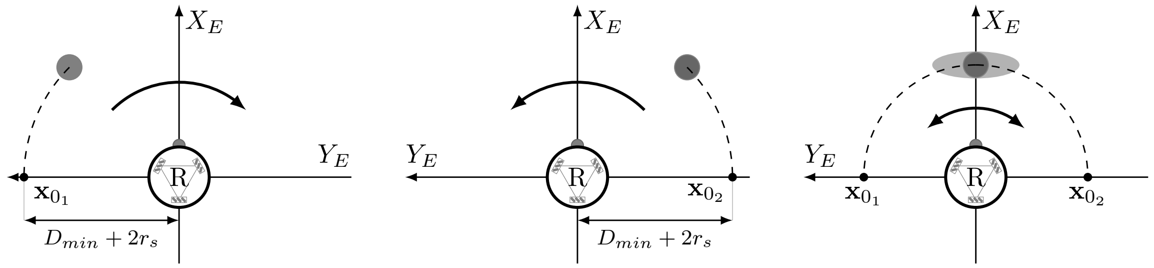
As this strategy has to prevent collisions with robots and obstacles alike, it starts with examining internal sensor observations $\hat{\mathbf{o}}_i$ and shared data $\hat{\mathbf{o}}_s$ to determine the closest object (robot or obstacle). In that endeavor, the shared position data of the robot is transformed into the local coordinate system, cf. Eq. (5.21).

$$\mathbf{x}_{E_{R_2}} = \begin{bmatrix} x_{E_{R_2}} \\ y_{E_{R_2}} \end{bmatrix} = \begin{bmatrix} \cos(\Theta_{G_{R_1}}) & -\sin(\Theta_{G_{R_1}}) \\ \sin(\Theta_{G_{R_1}}) & \cos(\Theta_{G_{R_1}}) \end{bmatrix} \cdot \begin{bmatrix} x_{G_{R_2}} - x_{G_{R_1}} \\ y_{G_{R_2}} - y_{G_{R_1}} \end{bmatrix} \tag{5.21}$$

Building upon the transformed position, the euclidean norm of the position vectors $\mathbf{x}_{E_{OS}}$ and $\mathbf{x}_{E_{R_2}}$ is used for determining the closest object, Eq. (5.22).

$$\mathbf{x}_{E_O} = \begin{cases} \mathbf{x}_{E_{OS}}, & \text{if } \|\mathbf{x}_{E_{OS}}\| < \left\|\mathbf{x}_{E_{R_2}}\right\| \\ \mathbf{x}_{E_{R_2}}, & \text{else} \end{cases} \tag{5.22}$$

Acting only with respect to the closest object $\mathbf{x}_{E_O}$ enables defining a simplified collision avoidance strategy. For that, three different phases are distinguished and are discussed in the following paragraphs. Fig. 5.4 provides an overview and references the corresponding phases.



(a) Turning right when the obstacle is on the left side.

(b) Turning left when the obstacle is on the right side.

(c) Uncertainty causing switching between turning directions.

Fig. 5.5.: Using the objects Y_{E_O} coordinate to decide the turning direction. When the value is close to zero, failures affecting the position data may cause an incorrect turning direction.

Phase 1: Engaging Collision Avoidance The first phase starts with the decision on engaging the collision avoidance. This decision is made based on the robot's distance to the obstacle d_O, cf. Eqs. (5.23) and (5.24).

$$\mathbf{d} = \begin{bmatrix} x_{E_O} \\ y_{E_O} \end{bmatrix} - \begin{bmatrix} \cos(\Theta_{G_{R_1}}) & -\sin(\Theta_{G_{R_1}}) \\ \sin(\Theta_{G_{R_1}}) & \cos(\Theta_{G_{R_1}}) \end{bmatrix} \cdot \begin{bmatrix} x_{G_{R_1}} \\ y_{G_{R_1}} \end{bmatrix} \tag{5.23}$$

$$d_O = \|\mathbf{d}\|^2 - 2r_s \tag{5.24}$$

As opposed to the obstacle's position, which is provided in the robot's local coordinate system already, the robot's position has to be transformed into the same before the difference can be calculated. The final distance d_O leverages the assumptions that (i) obstacles and robots are of circular shape (ii) and have the same radius of r_s.

If the distance d_O is less than the minimal distance D_{min}, the collision avoidance applies, that is, *engage* = 1, cf. Eq. (5.11). Consequently, the control actions provided by $\pi_{goal}(\mathbf{o}_i)$ are overwritten in order to maintain the distance D_{min} to the obstacle.

In that endeavor, the difference between the actual and the targeted distance is determined. Using this error to scale the vector directed towards the object to avoid results in control actions that cause the robot to move away or towards the object, cf. Eq. (5.25).

$$\begin{bmatrix} u_{x_{collision}} \\ u_{y_{collision}} \end{bmatrix} = \frac{\hat{\mathbf{x}}_{E_O}}{\|\hat{\mathbf{x}}_{E_O}\|} \cdot K_O \cdot (d_O - D_{min}) \tag{5.25}$$

With this linear relation to the error d_O, the control policy resembles a P-controller.

Besides the translational movement, the direction in which to turn has to be decided to determine u_Θ and prepare to bypass the object on a circular trajectory. For that, three relevant situations are displayed in Fig. 5.5, where the object to bypass is considered in the robot's local coordinate system.

For the situations displayed in Fig. 5.5a and Fig. 5.5b the position of the object relative to the robot is sufficient to decide on a turning direction. In case y_{E_O} is positive, the object is on the robot's left side and the robot should turn right. Similarly, in case y_{E_O} is negative, the object is on the robot's right side and the robot should turn left.

However, considering the situation displayed in Fig. 5.5c where the object is in front of the robot and its position data $\hat{\mathbf{x}}_s$ is impaired by observation failures, deciding on a turning direction using the value of y_{E_O} could result in an alternation. Deciding at each time step whether to turn left or right could cause the robot to switch between both options regularly and thereby (in the worst case) collide with the object.

To prevent this situation, the turning direction is fixed depending on the sign of y_{E_O} when collision avoidance is engaged. That is, the first time d_O is less than D_{min}, the sign of y_{E_O} is used to determine the turning direction $\Psi \in \{-1, 1\}$ which is not changed until collision avoidance is disengaged in phase three.

Depending on the turning direction, the collision avoidance strategy aims at stabilizing the coordinates of the object in the robot's local coordinate system at $\mathbf{x}_{0_1}$ or $\mathbf{x}_{0_2}$, that is, on the left or right side of the robot. For that, the targeted angular velocity u_Θ is calculated according to Eq. (5.26).

$$u_{\Theta_{collision}} = atan2(-1.0 \cdot \Psi(D_{min} + 2r_s) \cdot x_{E_O}, \Psi(D_{min} + 2r_s) \cdot y_{E_O}) \cdot K_O \tag{5.26}$$

Once again, the radius of the robot and the object in question have to be taken into account to determine the difference between the targeted orientation and the current orientation of the robot relative to the object to bypass. Finally, the same gain parameter K_O is used such that this control policy resembles a P-controller as well.

Phase 2: Bypassing the Object As a result of phase one, the distance between the robot and the object to bypass is stabilized at $d_O = D_{min}$ and the direction in which to turn Ψ is determined. Thus, the object is brought into the stability point $\mathbf{x}_0 = \mathbf{x}_{E_O} = [x_{E_O} = 0 \; y_{E_O} = \Psi(D_{min} + 2r_s)]^T$, that is, $\mathbf{x}_{0_1}$ or $\mathbf{x}_{0_2}$.

The goal of the second phase is to maintain the stability point while bypassing the object on a circular trajectory. For that, the robot moves forward with half of the maximal linear speed $U_{T_{max}}$ while its angular velocity is calculated in accordance with the trajectory to follow, cf. Eq. (5.29).

$$u_{x_{collision}} = 0.5 \cdot U_{T_{max}} \tag{5.27}$$

$$u_{y_{collision}} = 0 \tag{5.28}$$

$$u_{\Theta_{collision}} = \frac{u_x}{D_{min} + 2r_s} + K_C \cdot (d_O - D_{min}) \tag{5.29}$$

Note that the error in the distance to the object d_O is considered again to stabilize the same during phase two by applying a control strategy similar to a P-controller as well.

Phase 3: Disengaging Collision Avoidance The first and second phases of the collision avoidance strategy ensure that the robot bypasses the object on a circular trajectory while maintaining a safe distance of $d_O = D_{min}$. To fulfill its actual goal, that is, to navigate to its goal position, the collision avoidance has to be disengaged in a safe manner. For that, the third phase of the collision avoidance strategy monitors the corresponding attributes and finally decides on the disengagement. This decision is based on two conditions. Firstly, the robot has to have a safe distance to the object, that is, $d_O \geq D_{min}$. Although this should be provided through phases one and two, the condition could be unfulfilled at the beginning of the third phase due to the employed P-controller. Its simplistic control strategy can cause uncertainties in maintaining the control goal and thereby requires to monitor the condition before disengaging.

Secondly, the robot has to move away from the object. The second condition is fulfilled when the angle ϕ between the vector of the velocities of the goal-finding strategy $\mathbf{u_{t_{goal}}} = [u_{x_{goal}} \ u_{y_{goal}}]^T$ and the vector representing the object's position in the robot's local coordinate system is greater or equal to $0.75 \cdot \pi$, that is, Eq. (5.31) is fulfilled.

$$\phi = |atan2(u_{x_{goal}} \cdot y_{E_O} - u_{y_{goal}} \cdot x_{E_O}, u_{x_{goal}} \cdot x_{E_O} + u_{y_{goal}} \cdot y_{E_O})| \tag{5.30}$$

$$\phi \geq 0.75 \cdot \pi \tag{5.31}$$

Once the condition is fulfilled, the collision avoidance is disengaged, that is, $engaged = 0$, cf. Eq. (5.11). This means that $\mathbf{u}_{goal}$ is returned by the control policy $\pi(\hat{\mathbf{o}}_i, \hat{\mathbf{o}}_s)$.

5.1.5. Intention Modeling by Model Uncertainty

The defined control policy enables robots to bypass objects with static positions, that is, no movement. Opposed to that, the use case envisions areas shared by robots. Their movement contradicts the model assumption made for designing the control policy and results in a model uncertainty U_{model}, cf. Eq. (4.1). This model uncertainty has to be taken into account for assessing the safety of the control policy and is therefore specified in this section. More precisely, as the safety will be assessed by estimating and analyzing the controller's RoS, the model uncertainty has to be specified such that it applies to objects in general, that is, simultaneously to robots to be bypassed as well as obstacles. Otherwise, an estimated RoS would be valid only for avoiding collisions with either of them and subsequent distinctions between the objects to avoid collisions with would be required.

As the approach for modeling the uncertainty therefore addresses objects in general, that is, including robots, they have to be assumed to move in any direction. This results in an uncertainty around their perceived position, which is indicated by the light-gray area in Fig. 5.6. Note that the figure displays the idea for two different quadrants to exemplify the idea while the estimation of RoS will assume only a single object.

This light-gray area is initially associated with the center of the quadrants (dark-gray-colored rectangle). It follows the assumptions of Section 5.1.1, which states that robots are assumed to engage in collision avoidance as well and share corresponding metadata.

Firstly, the robots share their maximal translational velocity $v_{T_{max}}$, which is used to determine the magnitude of uncertainty, that is, the size of the light-gray area in Fig. 5.6. Secondly, the robots share the minimal distance D_{min} they aim at maintaining during collision avoidance. Using this value, the uncertainty can be restricted as an object is likely to move away from the robot (denoted with R) once it undercuts this distance. This assumption is expressed by f_r, which limits the light-gray area in in Fig. 5.6. Finally, the assumption has to be aligned with the actual position of the object (dark-gray-colored circle). For that, the limited uncertainty is rotated by ν in Fig. 5.6.

In the endeavor of formalizing the described uncertainty model U_{model}, one starts with defining the worst-case assumption using the maximal translational velocity $v_{T_{max}}$, cf. Eq. (5.32).

$$\mathbf{w} = -sgn(\mathbf{x}_{E_O}) \cdot v_{T_{max}} \tag{5.32}$$

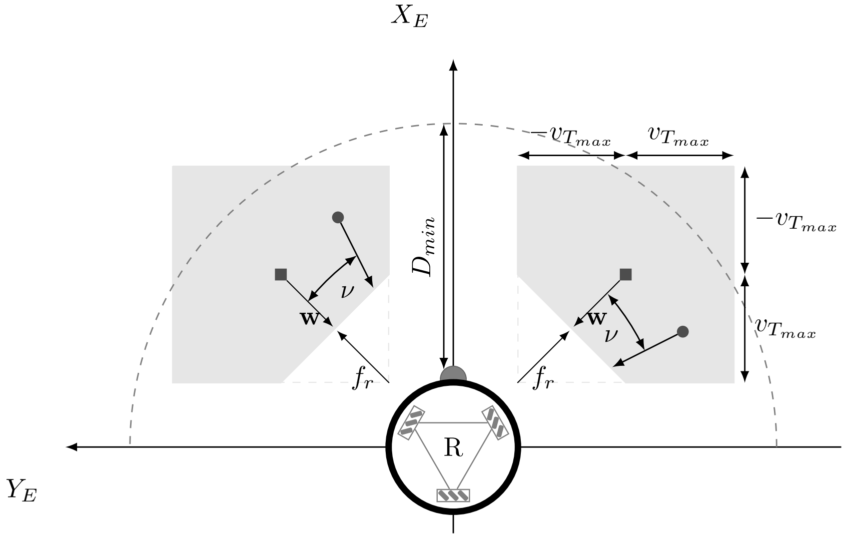


Fig. 5.6.: Limiting model uncertainty by making assumptions about the possible movements of an object with which collisions are to be avoided. The robot R in the origin of the local coordinate system aims at avoiding collisions with the object (dark-gray-colored circle). Two objects are shown here solely to illustrate the principle regarding two different quadrants. For the object, its type (obstacle or robot) decides whether it will move or not. Thus, an uncertainty about in which direct the object will move relative to the robot R results and is indicated by the light-gray-colored area. Starting at the center of the quadrant, the area is firstly limited in its extension towards the robot R by assuming that, if the object is a robot, the robot will engage in collision avoidance and limit its velocity towards R. Moreover, the area is aligned with the actual position of the object, that is, the area is rotated according to the object's position. This is indicated by ν.

Using the signum function on each element of the object's position vector $\mathbf{x}_{E_O}$, it takes the quadrant in which the object is relative to the robot into account and derives the worst-case moving vector $\mathbf{w}$ for the object. The vector is illustrated in Fig. 5.6, starting at the center of the quadrants. For aligning the worst-case assumption with the actual direction of the object, the angular displacement ν is calculated according to Eq. (5.33) and used to rotate $\mathbf{w}$ to obtain $\hat{\mathbf{w}}$ in Eq. (5.34).

$$\nu = atan2(x_{E_O}w_y - y_{E_O}w_x, -w_x x_{E_O} - w_y y_{E_O}) \tag{5.33}$$

$$\mathbf{w}' = \begin{bmatrix} \cos(\nu) & -\sin(\nu) \\ \sin(\nu) & \cos(\nu) \end{bmatrix} \cdot \begin{bmatrix} w_x \\ w_y \end{bmatrix} \tag{5.34}$$

For now, the resulting vector $\mathbf{w}'$ states that an object at $\mathbf{x}_{E_O}$ may apply its maximal translational velocity in any direction. To limit this worst-case uncertainty using the assumptions detailed previously, a factor f_r is calculated, cf. Eq. (5.35).

$$f_r = \max(\min(1.0, 1 - (D_{min} - d_O)), 0.0) \tag{5.35}$$

$$\hat{\mathbf{w}} = \frac{\mathbf{w}'}{\|\mathbf{w}'\|} \cdot f_r \tag{5.36}$$

Assuming that the other robot is adhering to the minimal distance D_{min} it communicated, any distance d_O undercutting this threshold results in a reduction in the maximal translational velocity directed towards the coordinate system's origin, that is, the robot R. Here, a linear reduction is assumed. Moreover, by limiting f_r to values between zero

and one, the worst-case assumption is at maximum restricted to no movement, meaning that the case of the object being a static obstacle is always included.
Using the restricted worst-case assumption, the final intervals of possible translational velocities are constructed, cf. Eqs. (5.37) to (5.41).

$$\mathcal{W}_x = \{\hat{w}_x, -sgn(\hat{w}_x) \cdot v_{T_{max}}\} \quad (5.37) \qquad \mathcal{W}_y = \{\hat{w}_y, -sgn(\hat{w}_y) \cdot v_{T_{max}}\} \quad (5.39)$$

$$\mathcal{I}_{model_x} = [\min(\mathcal{W}_x), \max(\mathcal{W}_x)] \quad (5.38) \qquad \mathcal{I}_{model_y} = [\min(\mathcal{W}_y), \max(\mathcal{W}_y)] \quad (5.40)$$

$$U_{model} = \begin{bmatrix} \mathcal{I}_{model_x} \\ \mathcal{I}_{model_y} \end{bmatrix} \tag{5.41}$$

At first, two sets stating the minimal and maximal translational velocity in x_E and y_E are generated, cf. Eqs. (5.37) to (5.39). The signum function is used once again to ensure that uncertainty originating in the movement of the object is restricted only towards the robot, that is, towards the coordinate system's origin. From both sets, the minimal and maximal values are determined to form corresponding intervals. The final uncertainty model U_{model} assumes that the velocities specified by the intervals $\mathcal{I}_{model_x}$ and $\mathcal{I}_{model_y}$ are equally likely, that is, it assumes a uniform distribution.

5.1.6. Fusing Regions of Safety of Multiple Stability Points

The definition of the control policy, as well as the discussion on representing model uncertainties for the presented use case, showed that two stability points exist. A robot may bypass an object at its left or right side.

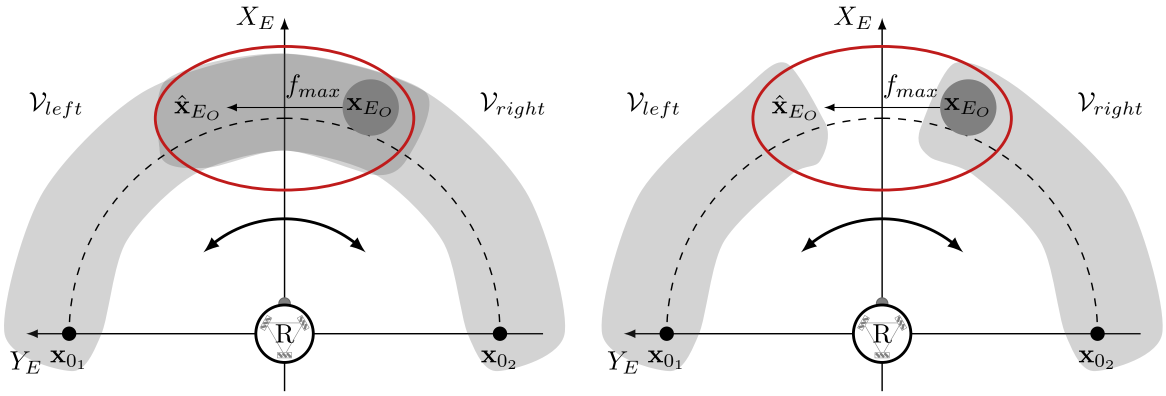
This, however, contradicts the estimation of *RoS* according to Algorithm 1, which presumes a control system and a *CLF* with a single stability point. To overcome this limitation and apply the concept of *RoS* to the collision avoidance strategy nonetheless, this subsection aims at defining a fusion strategy. The individual *RoS* resulting from applying Algorithm 1 to each stability point separately shall be fused to generate a final *RoS* covering both stability points.

The approach for defining a use-case-specific fusion strategy is depicted schematically in Fig. 5.7. In its center is the requirement of adhering to the criteria of Section 1.3.1. More specifically, the criterion of *Functional Correctness*, which asks for the validity of the run-time safety assessment's result. While the evaluation using the inverted pendulum showed that, if provided with an appropriate *CLF*, a controller's *RoS* is either valid or an empty set, the same property has to be maintained for the fused *RoS*.

For that, the idea is to guarantee that the decision on the turning direction does not influence the safety performance. In other words, the fusion strategy should only provide an *RoS* if it includes both stability points. Thereby, it shall guarantee safety for bypassing the object at both, its left and right side. Otherwise, an empty set shall be returned.

From that idea, two cases are derived. Firstly, either the *RoS* associated with $\mathbf{x}_{0_1}$ or $\mathbf{x}_{0_2}$ is an empty set. In this case, the fused *RoS* is the empty set as well.

Secondly, valid estimates are returned for both stability points, that is, one *RoS* encompassing $\mathbf{x}_{0_2}$ and another *RoS* encompassing $\mathbf{x}_{0_1}$. This case is depicted in Fig. 5.7 and requires to examine the individual *RoS* more closely.



(a) Valid combination of individual *RoS*. The safety does not depend on the turning direction chosen by the collision avoidance controller as both individual *RoS* overlap sufficiently at the critical decision point around $y_{E_O} = 0$.

(b) Invalid combination of individual *RoS*. If the object to bypass is sufficiently left or right of the robot, safety will be maintained. If the object appears in front of the robot, it can not be guaranteed that the robot reacts timely or chooses the correct turning direction.

Fig. 5.7.: Visualization of a valid and invalid combination of *RoS* estimated for both stability points.

According to the initial idea, the fused *RoS* shall be valid for both turning directions. This is to prevent the controller from choosing a single turning direction for which safety can not be guaranteed. Consequently, the focus for fusing the *RoS* is on the process of deciding the turning direction. As described in Section 5.1.4, this decision depends on the sign of the y_{E_O} coordinate of the object to avoid the collision with. Thus, while failure amplitudes impairing the shared position data will not affect this decision if y_{E_O} is either sufficiently positive or negative, they may change the decision if y_{E_O} is close to zero, that is, when the object is in front of the robot.

The goal of the fusion strategy has to be to ensure that the incorrectly perceived object does not cause the controller to choose an unsafe turning direction. This situation is exemplarily displayed in Fig. 5.7b. While the object, denoted with $\mathbf{o}_s$ and depicted with a dark-gray-colored circle, is at the robots right side, the failure amplitude f_{max} may impair the shared data such that the robot perceives the object to be at position $\hat{\mathbf{o}}_s$. This position, however, is at the robots left side and therefore causes the controller to choose the corresponding turning direction.

A consequence of this decision is, that the *RoS* $\mathcal{V}_{left}$ estimated for the stability point $\mathbf{x}_{0_1}$ applies. In other words, the controller is guaranteed to safely bypass the object only of the object's true position $\mathbf{o}_s$ is within the estimated *RoS* associated with $\mathbf{x}_{0_1}$. For Fig. 5.7b, this is not the case as the dark-gray-colored circle is within the right *RoS* $\mathcal{V}_{right}$ but not within the left *RoS*.

Contrarily, if both *RoS*, $\mathcal{V}_{left}$ and $\mathcal{V}_{right}$, would overlap such as depicted in Fig. 5.7a, the controller would be guaranteed to provide safety for the robot nonetheless. In this situation, the turning direction is still chosen incorrectly but as the true object position $\mathbf{o}_s$ is now included in the left *RoS*, the safety is guaranteed.

This concept is formalized in Eq. (5.42), which states the fusion strategy applied here.

$$\mathcal{V}_{fused}(\mathcal{V}_{left}, \mathcal{V}_{right}) = \begin{cases} \mathcal{V}_{left} \cup \mathcal{V}_{right}, & \text{for} \quad \min_{y_{E_O} \in \mathcal{V}_{left}} y_{E_O} \geq \min(\mathcal{I}_{f_{max}}) \wedge \\ & \qquad\quad \max_{y_{E_O} \in \mathcal{V}_{right}} y_{E_O} \leq \max(\mathcal{I}_{f_{max}}) \\ \emptyset, & \text{else} \end{cases} \tag{5.42}$$

Consequently, if both *RoS* ($\mathcal{V}_{left}$ and $\mathcal{V}_{right}$) overlap with at least f_{max} (the maximal failure amplitude to be expected according to the failure model of the shared data), the fused *RoS* $\mathcal{V}_{fused}$ is formed by the union of the individual *RoS*. Otherwise, the empty set is returned. For the sake of simplicity, the overlap is measured here by considering the maximal/minimal y component for both *RoS*. This is sufficient for the considered use case, but no general validity can be claimed.

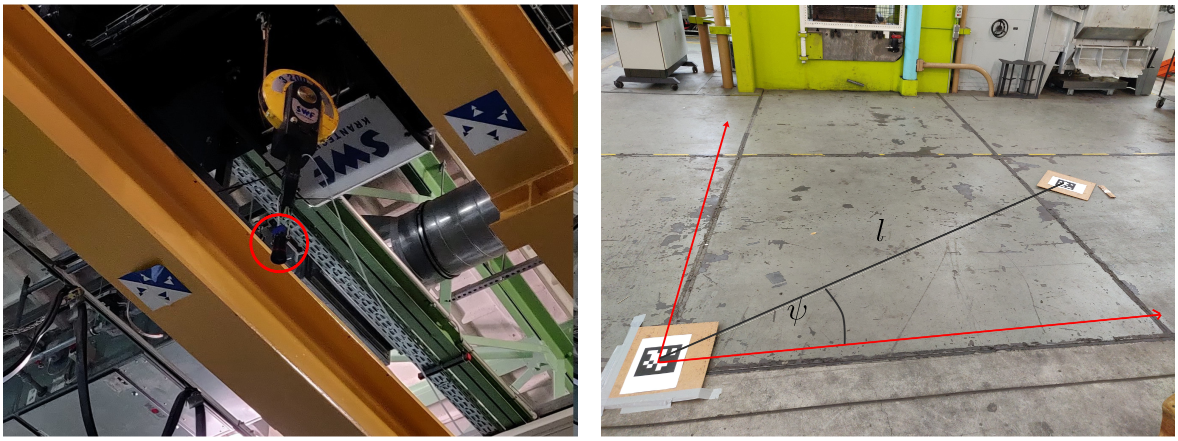
5.2. Failure Modeling for Shared Data – A Marker Detection Failure Model

The definition of the use case and the derived experimental scenarios as described in the last section provide the ability to evaluate this thesis' concepts. However, in its center is the assumption that a failure model representing failure characteristics of shared data is communicated to facilitate a run-time safety assessment. Within this chapter's use case, the shared data are position data of robots operating in a shared area. Thus, in this section, a failure model for this data is to be designed. Moreover, by designing a failure model the concept of *GFM* shall be evaluated as well. The goal is therefore to underline the applicability of the *GFM* in representing real-world failure characteristics and its ability to inform about the quality of the model using the defined confidence values.

In this endeavor, it is assumed that the robots are localized using marker detection [115]. Correspondingly, the next subsection describes the setup employed for acquiring observation and reference data. Section 5.2.2 builds upon that and discusses the generation of an initial failure model using the proposed processing chain and the manual adjustments made to optimize the result. Finally, Section 5.2.3 examines the confidence values achieved by the model according to the last stage of the processing chain.

5.2.1. Data Acquisition

For localizing robots in indoor environments different vision-based approaches have been proposed. They either assume cameras mounted on the robots themselves or static cameras installed in the robots' environment [115]. The latter is assumed in this work. The corresponding setup is described in the next subsection before the following subsection discusses the procedure applied to obtain the required data.



(a) Camera mounted on the ceiling of the experiment area.

(b) Setup for measuring failures in detected marker positions.

Fig. 5.8.: Setup for acquiring reference and observation data of marker detection framework. A camera was mounted under the ceiling (Fig. 5.8a) with which marker placed in a polar coordinate system (Fig. 5.8b) could be detected. Manually measuring the exact position and comparing it to the result of the marker detection framework provided the reference and observation data.

Setup

Corresponding to Section 3.3, reference data $\mathbf{o}_k$ as well as observations $\hat{\mathbf{o}}_k$ are required to determine the failure amplitudes $f(k, \mathbf{o}_k)$ from which the processing chain can extract a failure model. The employed setup is designed accordingly. It is visualized in Fig. 5.8. As shown in Fig. 5.8a, the camera (Matrix Vision mvBlueCOUGAR-X1012bG-POE-6211) was mounted on a gantry crane which allowed centering the camera approximately 5 m above the operation area in which two AprilTag2 [116] marker of size 0.15 m × 0.15 m were placed. The camera was configured to provide gray-scale images of size $2056px \times 1504px$, which means that each pixel covers an area of 2.4 mm × 2.6 mm.

The placement of markers is shown in Fig. 5.8b. The markers were arranged such that the first marker (lower left) denotes the coordinate system's origin while the second marker symbolizes the robot whose position is to be calculated. Using polar coordinates, the true position of the second marker in relation to the first marker is known exactly (see l and ψ in Fig. 5.8b). Together with the manually measured orientation, this data forms the reference data $\mathbf{o}_k$. Additionally, the AprilTag2 framework is used to localize both markers and obtain the position estimate $\hat{\mathbf{o}}_k = [x_{G_R}\ y_{G_R}\ \Theta_{G_R}]^T$ as their difference, which forms the observation data.

To leverage the *GFM*'s ability to represent time- and value-correlated failure characteristics, reference and observation data of different positions and orientations are required. For that, the first marker's position and orientation (the origin of the coordinate system) remained unchanged while only the second marker was moved. More specifically, the marker was placed at 96 poses resulting from three different length values $l \in \{1.1\,\text{m},\ 2.19\,\text{m},\ 3.32\,\text{m}\}$, four angles $\psi \in \{0°,\ 30°,\ 60°,\ 90°\}$ and eight orientations $\Theta \in \{-180°,\ -135°,\ -90°,\ -45°,\ 0°,\ 45°,\ 90°,\ 135°\}$. At each poses, a time series of 250 observations sampled with a frequency of 15 Hz was acquired. Therefore, an overall of 24.000 samples were obtained.

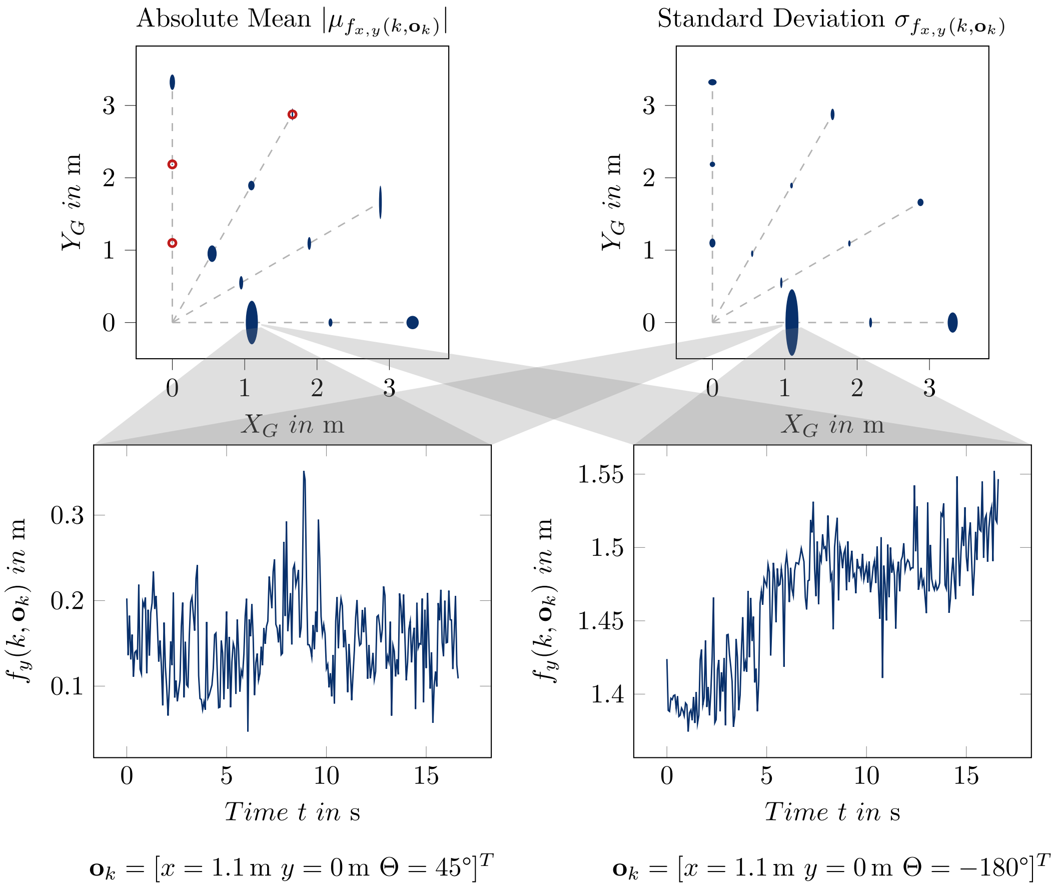


Fig. 5.9.: Visualizing the failure characteristics of the marker detection framework. The upper plots illustrate the magnitudes of the mean and standard deviations affecting the x and y components of provided pose observations. The lower diagrams show two exemplary time series of failure amplitudes in the y component obtained by subtracting the reference data from the marker detection observations, cf. Eq. (2.7). From such time series, the mean and standard deviation values of the upper diagrams are calculated.

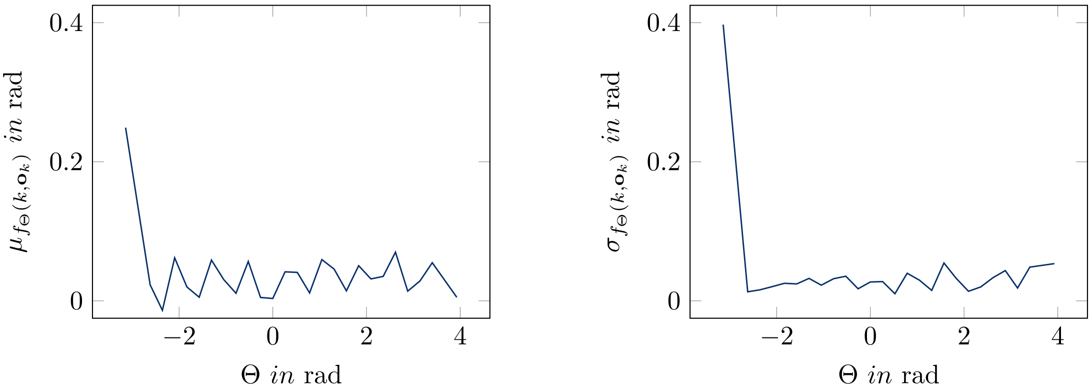
Obtained Data

The acquired data is visualized in Fig. 5.9. Utilizing the polar coordinates from which the reference data for the x and y components of the pose observations were calculated, the mean and standard deviation of the failure amplitudes (obtained by subtracting the reference data from the marker detection observations, cf. Eq. (2.7)) at these components are shown. That is, at each pose, the ellipse's magnitude in x or y directions symbolizes the mean and standard deviation of the failure amplitudes that were impairing the observations at that pose and over all eight orientations. Thus, each ellipse represents $250 \cdot 8 = 2000$ samples.

In favor of the marker detection framework, mean and standard deviations values are close to zero from some pose. Thus, to increase visibility, the sizes of the ellipses are scaled by a factor of 2 and highlighted with red circles, if necessary.

Two exemplary time series of failure amplitudes affecting the y component are visualized in the lower diagrams, each corresponding to a different Θ value but to the same pose.

When examining the mean and standard deviation values displayed in Fig. 5.9, the increased values at $[x = 1.1\,\text{m}\ y = 0\,\text{m}]^T$ are noticeable. Due to their difference to the



(a) Mean of failure amplitudes affecting the Θ component of the marker detection results with respect to the reference orientation.

(b) Standard deviation of failure amplitudes affecting the Θ component of the marker detection results with respect to the reference orientation.

Fig. 5.10.: The mean and standard deviation of failure amplitudes affecting the Θ component are displayed depending on the reference value of Θ.

values observed at other positions, it is concluded that these indicate a changed failure characteristic compared to the data's overall characteristic. This is underlined by the diagrams showing the failure amplitudes affecting the y component in the lower part of the figure. While the left diagram visualizes failure amplitudes that can be described as Noise, the right diagram shows failure amplitudes with an Offset of 1.4 m and increasing over time while imposed by similar Noise failures. Note that the increased failures occur only for $\Theta = -180°$ while failures for $\Theta = 45°$ are at a level comparable to other marker positions. As a consequence of these varying levels of failures at a single position, both, the mean and standard deviation of failures are increased for the y component.

Two reasons for the occurrence of these observations are possible. Firstly, the specific pose of $[x = 1.1\,\text{m} \; y = 0\,\text{m} \; \Theta = -180°]^T$ could have triggered a fault in the software which propagated through the marker detection framework and ultimately resulted in incorrect pose estimations. This, however, should be a deterministic behavior and should result in constant failures. In contrast, the observable Noise failures indicate that external disturbances affected the process. At the time the observations were made, clouds were blocking sunlight temporarily and caused the light conditions to change, sometimes rapidly. It is, therefore, possible that at the time where these samples were acquired, the sun might have shone directly into the hall and thereby disturbed the marker detection.

From the perspective of a real-world application, the reasons for these increased failures should be investigated in detail to stabilize marker detection and robot localization. However, this is out of scope for this work. Contrarily, in the endeavor of evaluating the concept of *GFM*, the changed failure characteristics are an opportunity to examine the concept's modeling performance.

Apart from position $[x = 1.1\,\text{m} \; y = 0\,\text{m}]^T$, the mean and standard deviation values vary only slightly. On the one hand, non-zero values of the mean indicate the presence of Offset failures while the values of the standard deviation indicate Noise failures. On the other hand, the limited variations in their magnitudes indicate that only minor time- and value-correlations are to be expected.

These assumptions are underlined by Fig. 5.10 showing the mean and standard deviation of failure amplitudes affecting the Θ component with respect to its ground truth value[1].

As one can see, the failure amplitudes are at a homogeneous level except for $\Theta = -\pi\,\mathrm{rad}$ which translates to $[x = 1.1\,\mathrm{m}\ y = 0\,\mathrm{m}\ \Theta = -180°]^T$. Thus, the increased values in mean and standard deviation noticed for the x and y component are observable in Θ as well.

5.2.2. Failure Model Generation

The initial analysis of the failure amplitudes impairing the marker detection observations indicate that three failure types are required for modeling the failure characteristics: Noise, Offset, and an Offset of increased severity representing the changed failure characteristics at $[x = 1.1\,\mathrm{m}\ y = 0\,\mathrm{m}\ \Theta = -180°]^T$. To simplify further descriptions, the latter is termed *Outlier-Offset*. In the endeavor of designing a *GFM* comprising these failure types, the next subsection discusses the application of the processing chain presented in Section 3.3 for generating an initial model that is manually tuned as described in the following subsection.

Designing an Initial Failure Model

The processing chain introduced in Section 3.3 takes the reference data $\mathbf{o}_k$ and the failure amplitudes $f(k, \mathbf{o}_k)$ into account to extract a failure model. Here, the first two stages are considered while the last stage, which assesses the quality of the generated failure model, is applied in Section 5.2.3. Consequently, the next paragraph discusses the identification stage used to obtain the failure pattern and their occurrences of the failure types. The following paragraph describes the parameterization of these and thereby the generation of the initial failure model.

Identification of Failure Types The first step for generating the initial failure model is to identify the failure types. According to the initial analysis of the obtained data, three failure types are to be identified. To that end, the process starts with the Outlier-Offset, for which only failure amplitudes at $[x = 1.1\,\mathrm{m}\ y = 0\,\mathrm{m}\ \Theta = -180°]^T$ are considered. An offset-like failure type occurring only at that position is identified.

Its occurrence is removed to obtain the failure amplitudes $\hat{f}_{y_1}(k, o_k)$ (cf. Fig. 5.11) in which the second failure type can be identified. The occurrences of this failure type resemble common Offsets and are identified for the remaining 95 poses.

By excluding its occurrences to obtain $\hat{f}_{y_2}(k, o_k)$, the identification of the last failure type is prepared. This failure type considers the remaining failure amplitudes as Noise which means that each individual time step of $\hat{f}_{y_2}(k, o_k)$ is considered as a separate occurrence. As a consequence, removing these occurrences results in an empty set of failure amplitudes and therefore concludes the identification process.

Fig. 5.11 illustrates the initial and updated failure amplitudes of the y component during the identification process. The individual time series of failure amplitudes are

[1] A scatter plot associating the magnitudes of mean and standard deviation of the failure amplitudes of the Θ component to the x,y positions can be found in Fig. C.1. It is not shown here as it requires the magnitudes of failures in the Θ component, which have the unit of rad, to be displayed in a scatter plot having units of m.

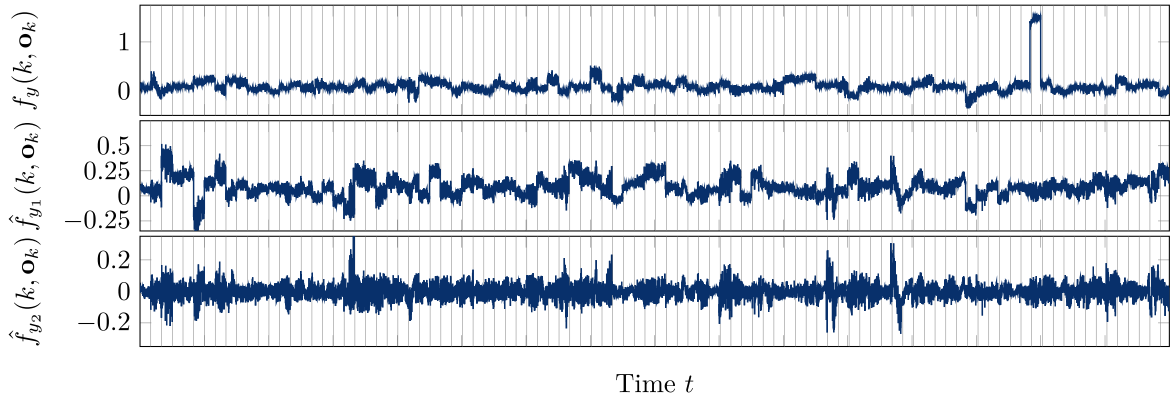


Fig. 5.11.: Intermediate results during identification of the Noise, Offset, and Outlier-Offset failure types. The time series obtained at each pose are concatenated to provide an overview on the data. The beginning and end of each time series is indicated by a vertical line. Time labels are not given to prevent giving rise to the impression of an ordering of the individual time series. The upper plot shows the initial failures in y component. The second plot displays the failure amplitudes $\hat{f}_{y_1}(k, \mathbf{o}_k)$ after the first update, that is, after identifying and removing the occurrence of the Outlier-Offset failure type. The lower plot displays the failure amplitudes $\hat{f}_{y_2}(k, \mathbf{o}_k)$ after the second update, that is after identifying and removing the occurrence of the Outlier-Offset and common Offsets.

concatenated for all poses to generate an overview of the data. To prevent misconceptions, it is refrained from providing a common time index and opted for separating individual time series by vertical bars.

As one can see, successively excluding the occurrences of the identified failure types decreases the magnitude of failure amplitudes. This indicates the successful representation of failure amplitudes by these failure types.

The parameters used for the first stage of the processing chain are listed in Table 5.2. Due to the Outlier-Offset, which has only a single occurrence, the minimal number of occurrences for a failure type to be accepted is set to $N_{\mathcal{O}} = 1$. Moreover, the individual time-series of failure amplitudes are determined from 250 consecutive observations obtained at a frequency of 15 Hz. Thus, the maximal length for a failure type to occur with was set to $K_n = 17\,\mathrm{s}$, which slightly overestimates the time horizon. Finally, as only Noise and Offset failure types are identified, only constants need to be represented

Tab. 5.2.: Parameterization of the identification stage of the processing chain for generating an initial failure model.

Parameter	Value	Description
$N_{\mathcal{O}}$	1	Number of occurrences a failure type must have at least to be accepted
K_n	$[0\,\mathrm{s}, 17\,\mathrm{s}]$	Durations, that is, scale values of *CWT* (cf. Eq. (3.47)) with which the occurrences of a failure pattern are searched for
D	0	Degree of the polynomial representing the failure pattern of an identified failure type

Tab. 5.3.: Configuration for the parameterization stage of the proposed processing chain. Note that the parameters for generating the Noise failure type's are different then the parameters for generating the remaining failure types.

Parameter	Value	Description
	Scaling Distribution	
K_W	17 s	Temporal width of the sliding window.
O_W $O_{W_{Noise}}$	$[0.15\,\text{m}\ 0.15\,\text{m}\ 0.15\,\text{rad}]^T$ $[1.5\,\text{m}\ 1.5\,\text{m}\ 1.5\,\text{rad}]^T$	Width of the sliding window in the value domain.
K_S	17 s	Temporal step size of the sliding window.
O_S $O_{S_{Noise}}$	$[0.15\,\text{m}\ 0.15\,\text{m}\ 0.15\,\text{rad}]^T$ $[0.75\,\text{m}\ 0.75\,\text{m}\ 0.75\,\text{rad}]^T$	Step size of the sliding window approach in the value domain.
	Activation and Deactivation Distribution	
K_W	17 s	Temporal width of the sliding window.
O_W	$[0.5\,\text{m}\ 0.5\,\text{m}\ \pi\,\text{rad}]^T$	Width of the sliding window approach in the value domain.
K_S	17 s	Temporal step size of the sliding window.
O_S	$[0.5\,\text{m}\ 0.5\,\text{m}\ \pi\,\text{rad}]^T$	Step size of the sliding window approach in the value domain.

by the failure patterns. Therefore, the degree of the trained polynomials is set to $D = 0$.

As a consequence, the information about the magnitude of failure amplitudes is represented solely in the scaling distribution of the individual failure types. These are derived together with the activation and deactivation distributions during the next stage of the processing chain.

Parameterizing Failure Types The goal of the parameterization stage is to extract time- and value-correlations of the scaling, activation, and deactivation distributions and represent these as polynomials. For that, the sliding window approach is employed in a first step, cf. Section 3.3.2. Its configuration is stated in Table 5.3.

The first block denotes the width and step-sizes applied to extract the training data of the scaling distributions while the second block states the same parameters for extracting the training data of the activation and deactivation distributions. Despite these individual parameters, the Noise failure type requires a specialized parameterization and is therefore stated separately where applicable.

According to the initial analysis of the data and supported by Fig. 5.11, no time-correlations are expected. Therefore, the temporal width K_W of the sliding window approach and the temporal step size K_S are set to 17 s for all distributions. As this overestimates the length of a single time series of failure amplitudes (16.6 s), the time

Tab. 5.4.: Overview of the degrees of the polynomials used to represent the individual failure types' functions. The defined values are chosen as only minor time- and value-correlations are expected. Moreover, increased complexity is anticipated only for the Noise's quantile function. Gray color cells indicate polynomials whose weights were manually adjusted after the generation of the initial failure model.

Failure Type	Polynomial	Scaling	Activation	Deactivation
Outlier-Offset	Distribution $Q(\mathbf{z})$	1	1	1
	Scale $\sigma(k, \mathbf{o}_k)$	1	1	1
	Shift $\mu(k, \mathbf{o}_k)$	1	4	4
Offset	Distribution $Q(\mathbf{z})$	1	1	1
	Scale $\sigma(k, \mathbf{o}_k)$	1	1	1
	Shift $\mu(k, \mathbf{o}_k)$	2	1	1
Noise	Distribution $Q(\mathbf{z})$	2	1	1
	Scale $\sigma(k, \mathbf{o}_k)$	1	1	1
	Shift $\mu(k, \mathbf{o}_k)$	1	1	1

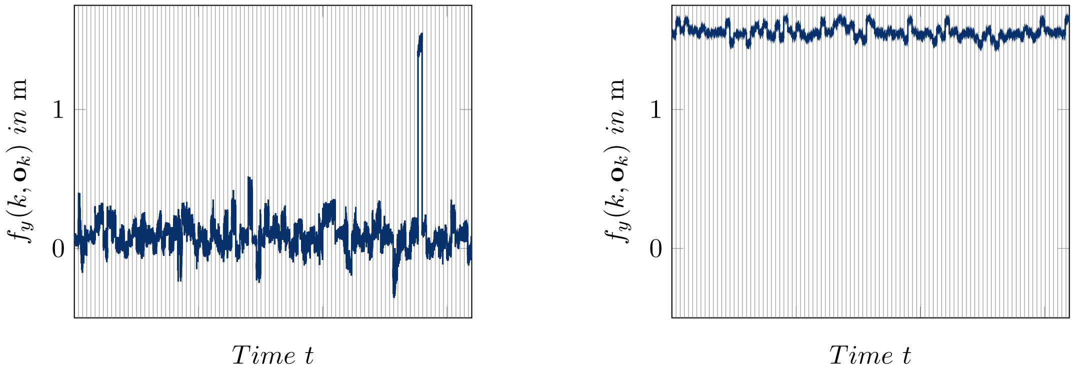
at which an occurrence of a failure type is identified is not relevant for its activation, deactivation, or scaling value. Thus, no time-correlations are represented.

While, the second and third curves in Fig. 5.11 indicate no value-correlations as well, slightly varying mean and standard deviations as shown in Fig. 5.9 contradict this assumption. Therefore, regarding the scaling distribution, a fine resolution is provided for the Offset and Outlier-Offset failure type by assuming a width of $O_W = 0.15$ and a step size of $O_S = 0.15$ for all dimensions. The values are chosen such that the windows generated during the sliding window approach are not overlapping. In that way, occurrences are considered only once, meaning that, for instance, an Offset identified at one position is not influencing the extraction of correlations at another position.

The parameterization for the Noise failure type differs in that the width parameters are increased and the step size is chosen such that the windows are overlapping. This is due to the assumption that the Noise failure type affects all positions with the same magnitude. Therefore, the scaling values are assumed to follow the same distribution for all positions.

The parameters applied to generate the training data for the activation and deactivation distributions are the same for all failure types. The width and step size parameters of the value domain are motivated by the expectation of slight value-correlations. However, as these are now influencing the activation and deactivation of the individual failure types, the resolution is reduced for the sake of computational efficiency. Similarly, no correlations are expected for Θ which is reflected by assuming $O_W = K_S = \pi$ for this component.

Using the presented parameters, the sliding window approach is applied as described in Section 3.3.2 to generate training data for fitting polynomials to represent the failure types' individual functions. To that end, the degrees of the fitted polynomials need to



(a) Initial failures of the marker detection framework in y component.

(b) Failure amplitudes in y component as simulated by the initial *GFM* produced by the processing chain.

Fig. 5.12.: Comparing original and simulated failure amplitudes for y component.

be defined first.

The assumed values are listed in Table 5.4. The degrees of the polynomials are chosen as a trade-off between minimal fitting errors and generalization performance.

For the Outlier-Offset, the scaling distribution does not require polynomials with higher degrees as the failure type has only a single occurrence. Similarly, no quantile function or scale function have to be represented for the activation or deactivation distribution. Their shift functions, however, have to be represented by polynomials of degree four to ensure that the failure type is only active at one pose.

For the Offset failure type, the degrees of all polynomials are set to one except for the polynomial representing the mean (shift) of scaling values. This function is represented by a polynomial of degree two to enable representing the anticipated value-correlations.

Contrarily, the randomness of the Noise failure type is assumed to be more complex, but static across the different poses. Thus, the quantile function is represented by a polynomial of degree two while only slight value-correlations are expected that are representable by a polynomial of degree one.

Using the defined values, the polynomials were fitted according to the generated training data and thereby the initial failure model was designed.

Manual Tuning

Using the first and second stage of the processing chain, an initial failure model is extracted from the obtained data. However, by simulating failure amplitudes using the failure model, it can be seen that the quality of this model is not sufficient, cf. Fig. 5.12. Consequently, in this subsections, adjustments of the initial failure model are described. For that, the usage of polynomials for representing the failure model's functions is leverage and their weights are manually optimized. The affected polynomials are marked by the gray-colored cells in Table 5.4.

As can be seen in Fig. 5.12b, the overall magnitude of failure amplitudes is drastically increased compared to the original failure amplitudes, cf. Fig. 5.12a. More specifically, for the y component, the value is increased by approx. 1.5 m, which is similar to the value introduced by the Outlier-Offset (cf. Fig. 5.9). This indicates that the failure type is always activated although it should be active only at pose $[x = 1.1\,\mathrm{m}\ y =$

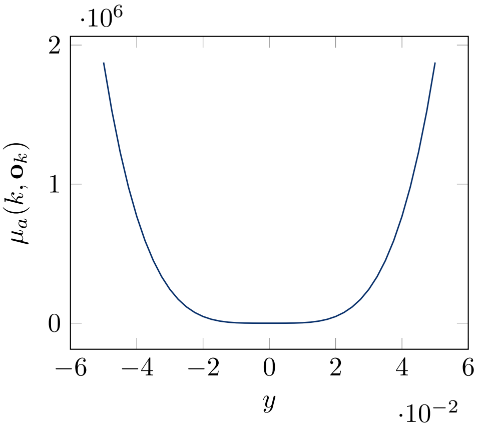
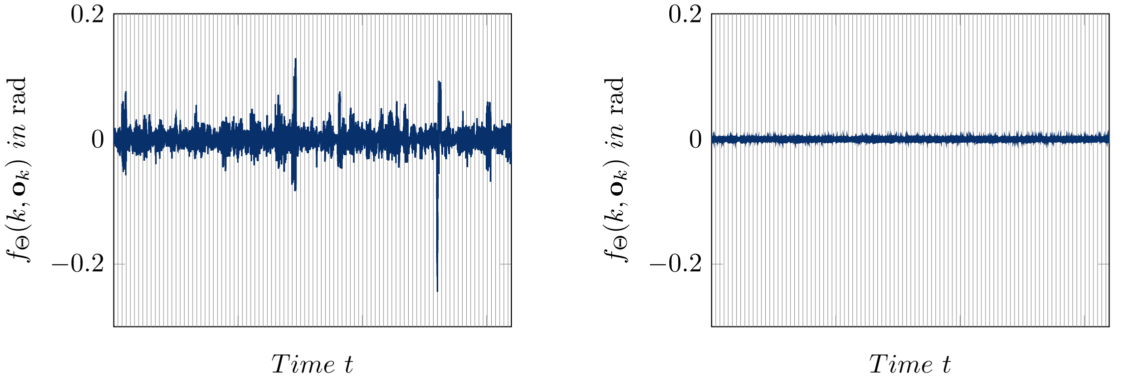


Fig. 5.13.: Shift (mean) function of the Outlier-Offset failure type's activation distribution visualized depending on y with $x = 1.1\,\mathrm{m}$, $\Theta = -180°$, and $k = 0.0\,\mathrm{s}$. By manually choosing the weights of the polynomial, it is ensured that the failure type becomes active only for $y = 0\,\mathrm{m}$ as the time between two occurrences increases drastically otherwise. The same pattern repeats for variations of x, Θ and k.



(a) Initial Failures in Θ

(b) Failures in Θ simulated by the initial *GFM* produced by the processing chain.

Fig. 5.14.: Comparing original and simulated failure amplitudes for Θ component.

$0\,\mathrm{m}\ \Theta = -180°]^T$. Therefore, the first manual adjustment is applied to the mean function $\mu_d(k, \mathbf{o}_k)$ of the failure type's activation distribution. It represents the time between two consecutive occurrences, which is not sufficiently high. Consequently, the weights of the polynomial are adjusted such that the mean is close to zero only for the specific pose of $\mathbf{o}_k = [0\,\mathrm{m}\ 1.1\,\mathrm{m}\ -180°]^T$ and drastically increases in each direction, which results in high *TBF* values for all other poses. The adjusted polynomial is visualized in Fig. 5.13 for $x = 0\,\mathrm{m}$, $y \in [-0.05\,\mathrm{m}, 0.05\,\mathrm{m}]$, $\Theta = -180°$, and $t = 0\,\mathrm{s}$.

Next to the Outlier Offset, the Noise failure has to be adjusted manually. As can be seen in Fig. 5.14 where the failure amplitudes simulated according to this failure type are compared to the original failure amplitudes, its scaling does not match the failure characteristics. To address this mismatch, the mean and standard deviation functions of its scaling distribution are adjusted. More specifically, by optimizing the constant values of the polynomials, the overall standard deviation is increased and the mean is adjusted accordingly.

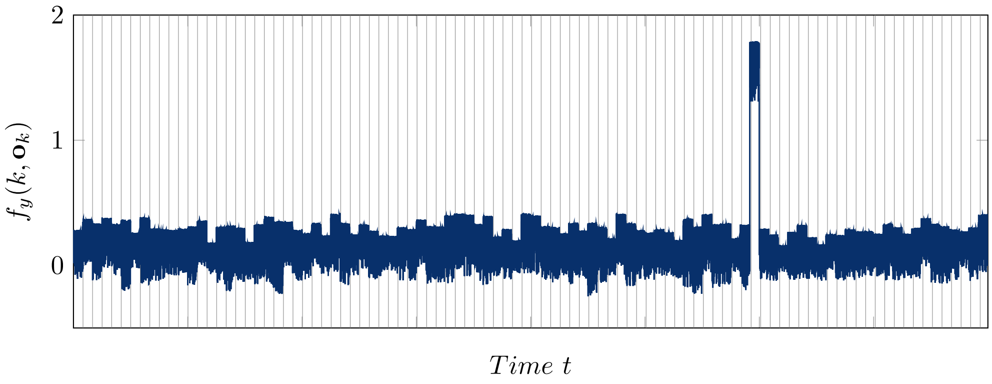


Fig. 5.15.: Simulated failure amplitudes of y component according to the final failure model.

Tab. 5.5.: Minimal distance to interval boarders for each dimension.

$\alpha = \gamma$	$d_{\mathcal{I}_x}(\alpha, \gamma)$	$d_{\mathcal{I}_y}(\alpha, \gamma)$	$d_{\mathcal{I}_\Theta}(\alpha, \gamma)$
0.95	0.02	0.011	0.031
0.85	-0.001	0.0025	0.008
0.75	-0.025	-0.05	-0.02
0.65	-0.05	-0.11	-0.05
0.5	-0.11	-0.23	-0.11

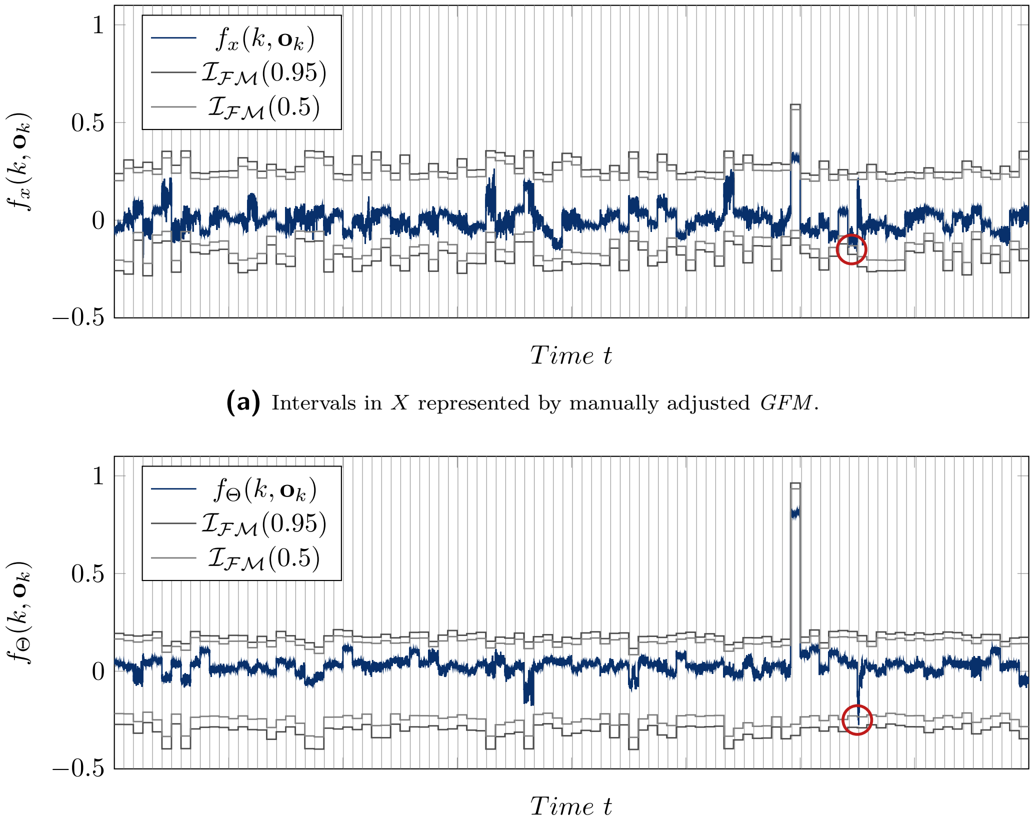
With these adjustments, the final failure model representing the failure characteristics of the shared data is generated. As can be seen by simulating the failure model (cf. Fig. 5.15), it successfully represents the Noise and Offset failures varying at similar levels while describing the changed failure characteristics at $[x = 0\,\mathrm{m}\ y = 1.1\,\mathrm{m}\ \Theta = -180°]^T$ via the specialized failure type.

5.2.3. Evaluating the Quality of the Failure Model

The last stage of the processing chain aims at assessing the quality of the designed failure model by determining the normalized distance $d_I(f(k, \mathbf{o}_k), \alpha, \gamma)$ in relation to the original failure amplitudes, cf. Eq. (3.66). Here, the values of $\alpha, \gamma \in \{0.95, 0.85, 0.75, 0.65, 0.5\}$ are assumed. Furthermore, $\alpha = \gamma$ is assumed for the sake of simplicity. The results are shown in Table 5.5, where the distance is stated for each dimension.

As it is to be expected, for values of $\alpha = \gamma = 0.95$, the distance values are positive with a minimal value of 0.011 for the y-Dimension. Therefore, the interval extracted from the final failure model successfully covers all initial failure amplitudes, which indicates that the failure characteristics are appropriately represented. Moreover, the maximal distance is 0.031 for the Θ component, which indicates that the magnitude of failures is matched and not drastically overestimated.

This is underlined by Fig. 5.16 where the initial failure amplitudes of the x and Θ components are visualized together with the interval borders for $\alpha = \gamma = 0.95$ and $\alpha = \gamma = 0.5$. In the former case, the initial failure amplitudes are always within



(a) Intervals in X represented by manually adjusted *GFM*.

(b) Intervals in Θ represented by manually adjusted *GFM*.

Fig. 5.16.: Intervals for $\alpha = \{0.95, 0.5\}$ for X and Θ dimension.

the interval borders. These are closer to the initial failure amplitudes for x while the distance is increased for Θ. However, for $\alpha = \gamma = 0.5$ failure amplitudes that are not covered by the interval are observable. These are highlighted by red circles. For instance, while Noise failures are not covered by the interval in case of the x component, a negative Outlier occurs in the Θ component. This Outlier is precisely the reason for the overall increased distance between the failure amplitudes and the interval borders for this component and indicates that an additional failure type could result in a closer approximation of the failure characteristics.

Nevertheless, the final failure model covers the original failure characteristics sufficiently close for the envisioned experiments.

5.3. Safe Handling of Shared Data in a Collision Avoidance Strategy

For this chapter's use case, it is assumed that the position data shared between robots[2] originate in the marker detection framework as described in the previous section. Therefore, its failure model describes the failure characteristics of the shared data and is consequently assumed to constitute $U_{shared}(\mathbf{o}_s, t)$ according to Eq. (4.1).

On the one hand, it thereby underlines the applicability of the *GFM* to model real-world failure characteristics. On the other hand, it is a prerequisite for estimating the *RoS* of the presented collision avoidance controller to assess whether or not it can tolerate these.

In this endeavor, the next subsection describes the configuration for estimating an *RoS* for the presented collision avoidance strategy. Firstly, a *CLF* for assessing the criticality of states with respect to the safety requirement of collision avoidance is defined. Secondly, as the *CLF* and collision avoidance is defined with respect to the robot's local coordinate system, a conversion of the shared failure model is presented. It enables the failure model stated with respect to the global coordinate system to be converted into the local coordinate system. With this conversion in place, all components of this use case are defined, which allows to estimate the controller's *RoS* in Section 5.3.2. More specifically, to not only consider a single shared failure model but to enable examining the effects of different failure models, the individual failure types of the marker-detection failure model are combined to form four different failure models. For each of these four failure models, the *RoS* estimation will provide evidence that the controller's configuration parameter D_{min} can be varied, that is, decreased, depending on the severity of the represented failure characteristics. Thus, using the circle scenario in Section 5.3.3 it will be shown that the safety of the robots sharing their position data is successfully maintained when applying the values D_{min} guaranteed to be safe by the *RoS* analysis. Furthermore, it will be shown that (i) no collision between robots sharing their position data occurs despite the considered failure characteristics, (ii) that the necessary value for D_{min} is overestimated, and (iii) severely undercutting the value indeed results in collisions. As this underlines the effectiveness of *GFM* to describe failure characteristics and of *RoS* to analyze them as part of a run-time safety assessment, Section 5.3.4 builds on that and briefly examines an approach of integrating the concepts to implement the idea of *Level of Service (LoS)*.

5.3.1. Configuration for Region of Safety Estimation

While defining the shared failure model completes the system model according to Eq. (4.1) required for estimating *RoS* (the components are listed and referenced in Table 5.6), a *CLF* for assessing the criticality of each state is missing. Thus, the next subsection describes the envisioned *CLF*.

Opposed to the shared failure model, however, the *CLF* is defined using the robot's local coordinate system. This leverages the fact that this coordinate system states the position of an object in relation to the robot, which simplifies not only calculating the distance between both but also defining the *CLF* for the task of collision avoidance.

[2] Note that, in contrast to the previous discussion, only the x and y components of the marker-detection-based pose estimation are used for the collision avoidance algorithm.

Tab. 5.6.: Overview of components of extended system model according to Eq. (4.1) defined for the multi-robot use case. Unused components are marked with *N.A.*.

Symbol	Equation	Symbol	Equation
$f^*_\pi(\mathbf{x})$	Eqs. (5.7) to (5.10)	$U_{model}(\mathbf{x}, \mathbf{u}, t)$	Eq. (5.41)
$U_{dist}(\mathbf{x}, t)$	*N.A.*	$U_{actuator}(\mathbf{u}, t)$	Eqs. (5.4) to (5.6)
$\pi(\hat{\mathbf{o}}_i, \hat{\mathbf{o}}_s)$	Eq. (5.11)	$s_i(\mathbf{x})$	Eq. (5.12)
$s_s(\mathbf{x})$	Eq. (5.14)	$U_{sensor}(\mathbf{o}, t)$	*N.A.*
$U_{shared}(\mathbf{o}, t)$	Section 5.2		

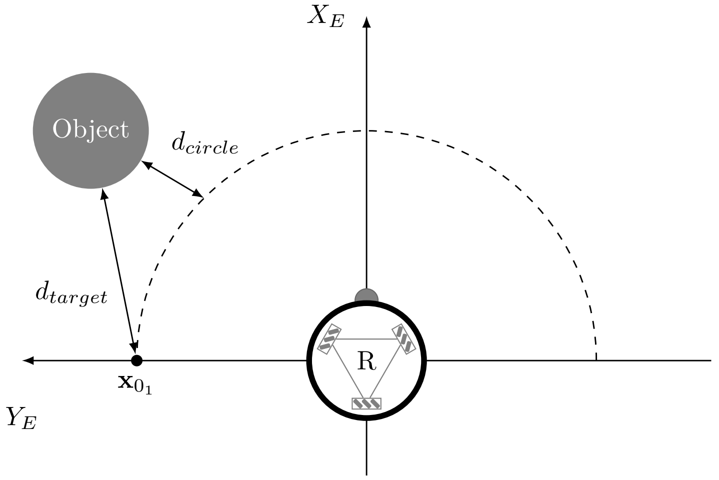


Fig. 5.17.: Aspects of the *CLF* $V_{collision}(\mathbf{x})$ for estimating *RoS* for the collision avoidance controller. Here, x_{0_1} is the considered stability point.

A disadvantage of this design choice, however, is the need for converting the shared failure model to a representation matching the local coordinate system as well. The corresponding procedure is described in the following subsection.

Control Lyapunov Function for Collision Avoidance

A valid *CLF* function applicable for estimating an *RoS* has to be continuously differentiable, positive semi-definite, and has to have a unique global minimum at $\mathbf{x}_0$. It can be thought of as a cost function informing about the criticality of a state $\mathbf{x}$.

With regard to the use case of collision avoidance, the central criterion for criticality is the distance of an object to the robot. This means that the relative position of an object to the robot is important rather than the object's global position. Therefore, the robots' local coordinate system is used to define the *CLF* here.

In that endeavor, two aspects are considered in Eq. (5.43) and are illustrated in Fig. 5.17. Both of which are described in detail in the following paragraphs.

$$V_{collision}(\mathbf{x}) = W_{circle} \cdot V_{cirlce} + W_{target} \cdot V_{target} \tag{5.43}$$

First and foremost, the euclidean distance d_O (cf. Eq. (5.24)) of an object to the robot with respect to the minimal distance D_{min} is considered. More specifically,

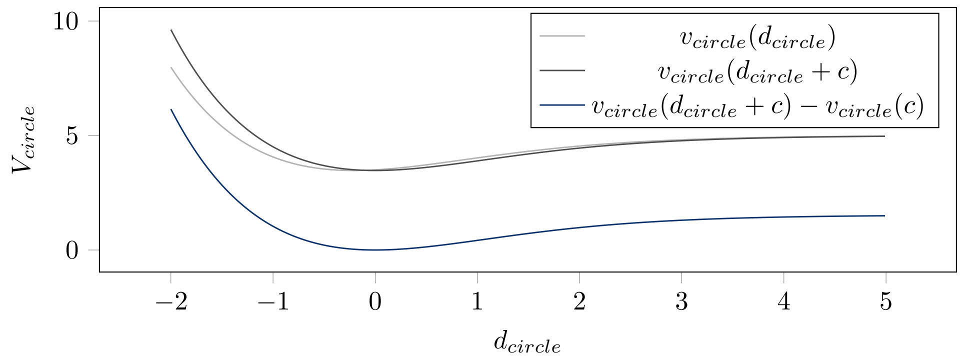


Fig. 5.18.: Assessing the criticality of obstacle distance with respect to D_{min}.

while distances $d_O > D_{min}$ are only relevant for performance considerations, values of $d_O < D_{min}$ are safety-critical. As the value of D_{min} essentially defines a circle surrounding the robot R (cf. Fig. 5.17) this aspect is encoded in V_{circle}.

The second aspect is the distance of an object from its target position $\mathbf{x}_{0_1}$ or $\mathbf{x}_{0_2}$. As dictated by the criteria of a valid *CLF*, only a single global minimum defined by the targeted stability point is possible. In the case of collision avoidance, this minimum is defined by the turning direction Ψ (from which $\mathbf{x}_{0_1}$ and $\mathbf{x}_{0_2}$ are derived) determined during the first phase of the collision avoidance strategy, cf. Section 5.1.4. Correspondingly, this aspect is encoded by V_{target}. Both aspects are defined in the following paragraphs while the parameters for weighting both aspects are listed in Table 5.7.

Defining V_{circle} The first aspect V_{circle} focuses on the difference between the distance d_O to its target value of D_{min}. Consequently, as a first step, the distance d_{circle} is defined (cf. Eq. (5.44)) and the initial value $v_{circle}(d_{circle})$ is derived (cf. Eq. (5.45)).

$$d_{circle} = d_O - D_{min} \tag{5.44}$$

$$v_{circle}(d_{circle}) = \left(e^{d_{circle}} + \frac{W_{V_{max}}}{1 + e^{-d_{circle}}} \right) \tag{5.45}$$

Following the previously presented reasoning, the value of $v_{circle}(d_{circle})$ increases drastically for negative values of d_{circle}, but converges to $W_{V_{max}}$ for positive values of d_{circle}. It thereby reflects that distances of d_O undercutting D_{min} are safety-critical, but values of $d_O > D_{min}$ are performance-relevant. These can be seen in Fig. 5.18 as well.

Moreover, it can be seen that the value of $v_{circle}(d_{circle})$ is shifted. On the one hand, the minimum is not at $d_{circle} = 0$, on the other hand, the curve does not achieve a value of $v_{circle}(d_{circle}) = 0$. With respect to the *CLF* to be defined, this would result in a global minimum at a position different to $\mathbf{x}_{0_1}$ or $\mathbf{x}_{0_2}$. As the function can therefore not be used for a *CLF*, these aspects are corrected by Eq. (5.46) and Eq. (5.47).

$$c = \ln\left(\frac{\sqrt{W_{V_{max}}} + 1}{W_{V_{max}} - 1} \right) \tag{5.46}$$

$$V_{circle} = v_{circle}(d_{circle} + c) - v_{circle}(c) \tag{5.47}$$

c is a correction factor to be added to the distance d_{circle} such that the minimum of v_{circle} is at $d_{circle} = 0$. The factor is determined by taking the first derivative of

Tab. 5.7.: Parameters of the *CLF* for estimating *RoS* of the collision avoidance controller.

Parameter	Value	Description
W_{circle}	200.0	Weighting of circle component V_{circle} (cf. Eq. (5.47)).
W_{target}	5.0	Weighting of target component V_{target} (cf. Eq. (5.48)).
$W_{V_{max}}$	5	Maximal value the circle component converges to when obstacle is outside the circle (cf. Eq. (5.45)).

Eq. (5.45), setting it to zero and solving the equation. In Eq. (5.47), the offset at the minimum of v_{circle} is removed to obtain the final value V_{circle}. The curve is colored blue in Fig. 5.18.

Defining V_{target} While V_{circle} assesses the criticality of a state $\mathbf{x}$ with respect to the robot's distance to the obstacle, V_{target} aims at assessing the obstacle's distance to its target position within the local coordinate system. In that endeavor, the euclidean distance of the obstacle from the target position is used, cf. Eq. (5.48).

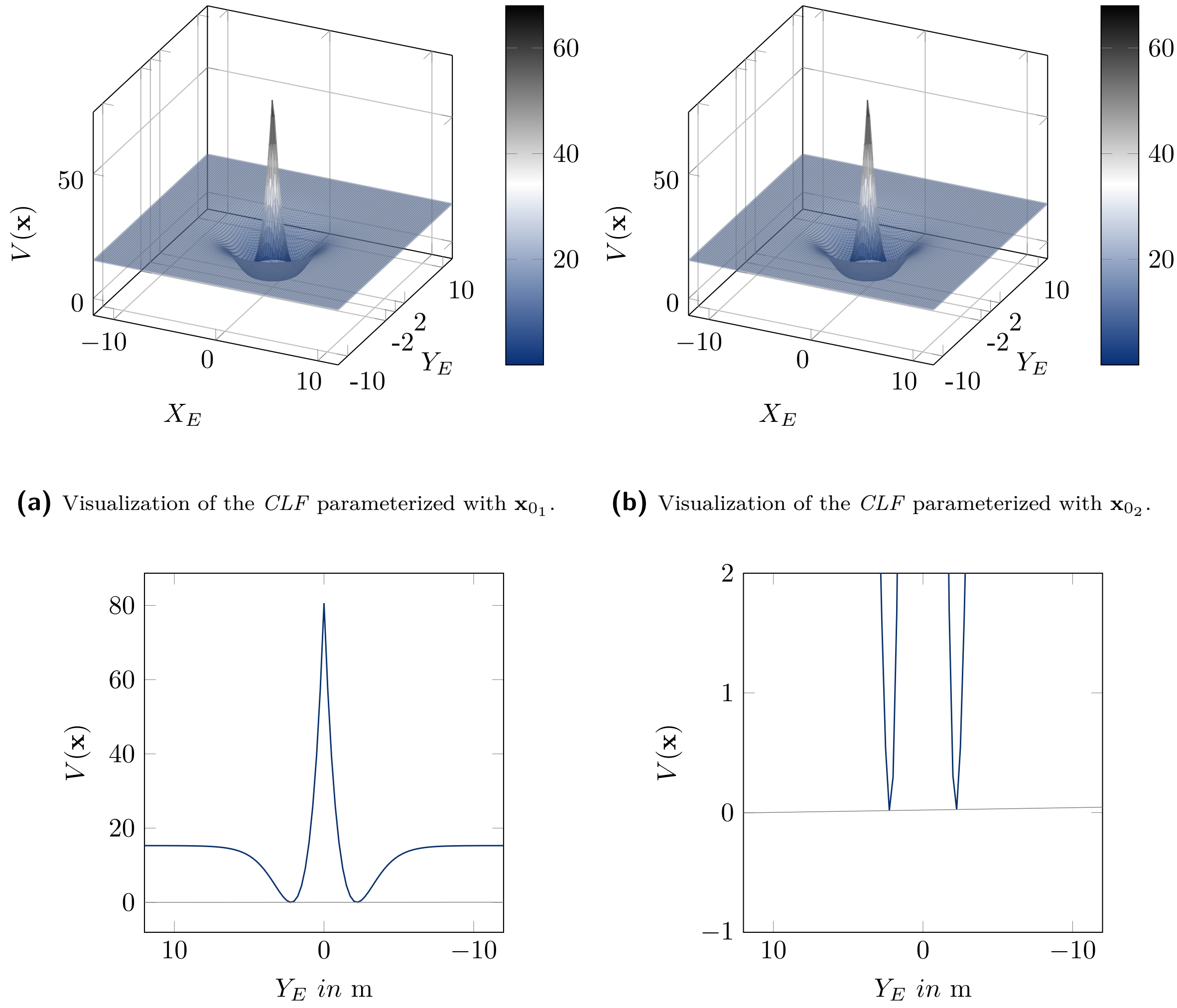
$$V_{target} = ||\mathbf{x} - \mathbf{x}_0|| \tag{5.48}$$

Here, $\mathbf{x}_0 \in \{\mathbf{x}_{0_1}, \mathbf{x}_{0_2}\}$ denotes the stability point depending on the turning direction chosen in the first phase of the collision avoidance strategy. As a consequence, the defined *CLF* can be applied to estimate an *RoS* for either one of these stability points depending on its parameterization.

Parameters of the *CLF* Both elements, V_{circle} and V_{target}, are weighted by their corresponding factors (W_{circle}, W_{target}) and summed to form the *CLF* for estimating *RoS* of the collision avoidance strategy, cf. Eq. (5.43). The parameters are listed in Table 5.7. They were determined using a similar procedure as described in Section 4.3.2, which is omitted here for the sake of brevity.

As the aspect encoded by V_{circle} focuses on the criticality of the state $\mathbf{x}$, its weight W_{circle} is drastically increased compared to the weight W_{target} of V_{target} whose focus is on the control performance. This shows that safety is the main attribute of concern and is therefore in line with the objective of the envisioned run-time safety assessment.

Due to the increased value of W_{circle}, the component V_{circle} generates a circular shape when considering the surface of the *CLF* in the two-dimensional state space. This is shown in Fig. 5.19 where the *CLF* is visualized for both stability points. The illustrations of their respective surfaces in Fig. 5.19a and Fig. 5.19b show that the impact of W_{target} is minor. Thus, in Fig. 5.19c, where the value of the *CLF* considering $\mathbf{x}_{0_1}$ is shown for $X_E = 0$, two minima are visible in accordance with the stability points $\mathbf{x}_{0_1}$ and $\mathbf{x}_{0_2}$. The limited but existing effect of W_{target}, however, causes $\mathbf{x}_{0_1}$ to be assigned a value of $V(\mathbf{x}_{0_1}) = 0$ while $\mathbf{x}_{0_2}$ is assigned a value of $V(\mathbf{x}_{0_2}) = 0.026$. This is visible in Fig. 5.19d where a gray-colored line connects both stability points and thereby visualizes the gradient generated by V_{target} which ultimately causes the difference in the *CLF* values. Therefore, the requirement of a single minimum is maintained and a valid *CLF* is defined.



(a) Visualization of the *CLF* parameterized with $\mathbf{x}_{0_1}$.

(b) Visualization of the *CLF* parameterized with $\mathbf{x}_{0_2}$.

(c) Visualization of the *CLF* parameterized with $\mathbf{x}_{0_1}$ but considering only $X_E = 0$.

(d) Visualization of the *CLF* parameterized with $\mathbf{x}_{0_1}$ but considering only $X_E = 0$. The visualization is restricted to $Y_E \in [-1, 2]$ to show the increase of $V(\mathbf{x})$ from $\mathbf{x}_{0_1}$ to $\mathbf{x}_{0_2}$.

Fig. 5.19.: Visualization of the *CLF* configured with $D_{min} = 2.0\,\mathrm{m}$ employed to estimate an *RoS* for the collision avoidance controller.

Converting the Shared Failure Model

For defining the *CLF*, the local coordinate system was used. As a consequence, estimating an *RoS* for either of the stability points requires using the local coordinate system as well. In contrast to this, the failure characteristics modeled by the shared failure model build upon the approach of marker detection and therefore assume the global coordinate system. Thus, for using it during the estimation of *RoS*, it has to be converted.

For that, one has to take the perspectives of the different coordinate systems into account. While the local coordinate system only states the spatial relation a robot and an object have to each other, that is, their relative position, the global coordinate system embeds this with respect to an independent point of reference, the origin of the coordinate system.

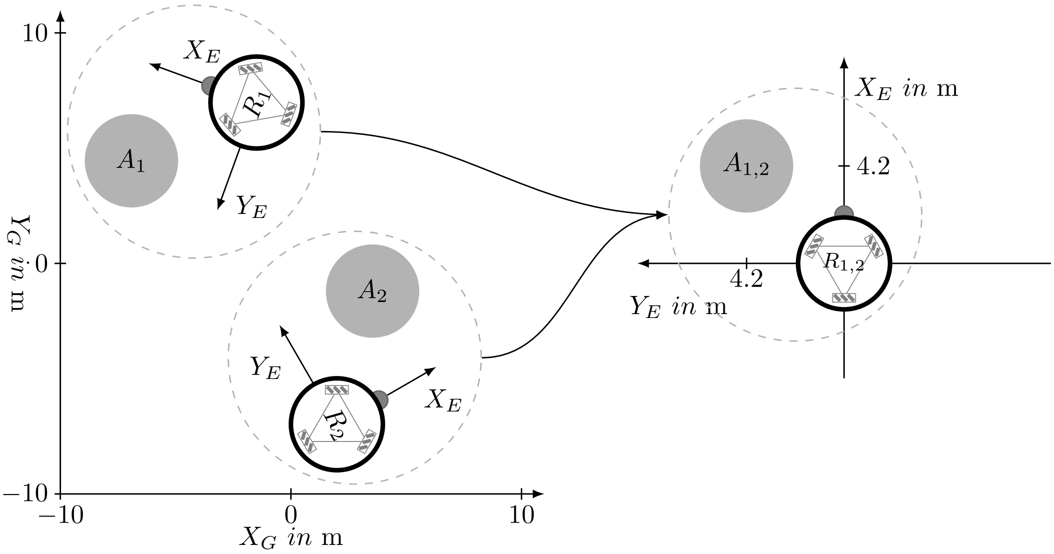


Fig. 5.20.: The figure schematically illustrates how two distinct poses of a robot (R_1 and R_2) and an object (A_1 and A_2) in the global coordinate system are mapped to the same position in the robot's local coordinate system ($R_{1,2}$ and $A_{1,2}$) during the transformation.

This gives rise to ambiguity. Multiple positions of a robot and an object are distinct regarding the global coordinate system but are mapped to the same position in the robot's local coordinate system. This is illustrated in Fig. 5.20, where R_1 and R_2, as well as A_1 and A_2, denote two positions of a robot and an object. Both positions are mapped to the same position in the robot's local coordinate system and can not be distinguished anymore.

Thus, converting the failure model from the global to the local coordinate system has to respect this change of perspective, more importantly, the resulting ambiguity. With respect to the local coordinate system, the failure model has to represent all failure characteristics possibly impairing a specific, relative position between the robot and an object. The shared failure model specifying the failure characteristics in the global coordinate system, however, models these depending on the specific location of the object. In other words, as the shared failure model encodes value-correlations, the failure characteristics depend on the global position of the object. Keeping in mind that different, global positions may result in the same, local position, the minimal and maximal failure amplitudes to anticipate for the entire considered (global) area have to be determined to state the failure model with respect to the local coordinate system.

More specifically, the maximal, absolute failure amplitude to anticipate for the x and y component would have to be determined to form the converted failure model. However, as this results in a symmetric failure model, the effect on the RoS estimates would be similar for all considered failure characteristics, as will be shown in Section 5.3.2. For the sake of analyzing the effects of varying failure characteristics on the estimation of RoS, the minimal and maximal failure amplitudes are considered to form the converted failure model, cf. Eqs. (5.49) and (5.51).

$$\mathcal{I}_x(\mathcal{FM}_G) = \left[\min_{\mathbf{o}_s \in \mathcal{O}_s, k \in \mathbb{K}} \left(f_x(k, \mathbf{o}_s)\right), \max_{\mathbf{o}_s \in \mathcal{O}_s, k \in \mathbb{K}} \left(f_x(k, \mathbf{o}_s)\right)\right] \tag{5.49}$$

$$\mathcal{I}_y(\mathcal{FM}_G) = \left[\min_{\mathbf{o}_s \in \mathcal{O}_s, k \in \mathbb{K}} \left(f_y(k, \mathbf{o}_s)\right), \max_{\mathbf{o}_s \in \mathcal{O}_s, k \in \mathbb{K}} \left(f_y(k, \mathbf{o}_s)\right)\right] \tag{5.50}$$

$$\mathcal{FM}_E = [\mathcal{I}_x(\mathcal{FM}_G)\ \mathcal{I}_y(\mathcal{FM}_G)]^T \tag{5.51}$$

Using the space of shared observations $\mathcal{O}_s$, that is, possible values shared by another robot, and the time horizon $\mathbb{K}$, the minimal and maximal failure amplitudes affecting the x and y components are determined from the shared failure model $\mathcal{FM}_G$ stating the failure characteristics with respect to the global coordinate system. As a result, the converted failure model $\mathcal{FM}_E$ states the minimal and maximal deviations to be expected for these components for the shared operation area. Θ is not required anymore, as the orientation of the robot to avoid the collision with is neither considered for the collision avoidance strategy nor for the CLF.

A downside of this approach to converting the shared failure model is that encoded time- and value-correlations are lost. In contrast, it would be possible to consider not the entire shared operation area at once but to divide it into subareas. The same conversion process could be applied for these smaller subareas and thereby enable retaining the time- and value-correlations partially. However, this is out of the scope of this thesis and is left for future work.

Tab. 5.8.: Definition of the state space considered for the robotic use case.

Parameter	Interval	Discretization
x_{E_O}	$[-12\,\mathrm{m}, 12\,\mathrm{m}]$	$x_S = 0.25\,\mathrm{m}$
y_{E_O}	$[-12\,\mathrm{m}, 12\,\mathrm{m}]$	$y_S = 0.25\,\mathrm{m}$
t	$[0\,\mathrm{s}, 17\,\mathrm{s}]$	$\tau = 0.07\,\mathrm{s}$

Tab. 5.9.: Failure models derived from shared failure model to examine the effect of different levels of uncertainty. The minimal distances D_{min} are determined by applying the *RoS* analysis.

Failure Model	Failure Types	D_{min}
$\mathcal{FM}_{\{\emptyset\}}$	$\emptyset$	$2.0\,\mathrm{m}$
$\mathcal{FM}_{\{N\}}$	$\{F_{Noise}\}$	$3.0\,\mathrm{m}$
$\mathcal{FM}_{\{N,O\}}$	$\{F_{Noise}, F_{Offset}\}$	$4.5\,\mathrm{m}$
$\mathcal{FM}_{\{N,O,OO\}}$	$\{F_{Noise}, F_{Offset}, F_{Outlier-Offset}\}$	$9.5\,\mathrm{m}$

5.3.2. Estimating Regions of Safety

Having the *CLF* and the fusion strategy defined, the *RoS* of the employed collision avoidance controller can be estimated. For that, the next subsection discusses the parameters assumed here before the results are discussed.

Parameters

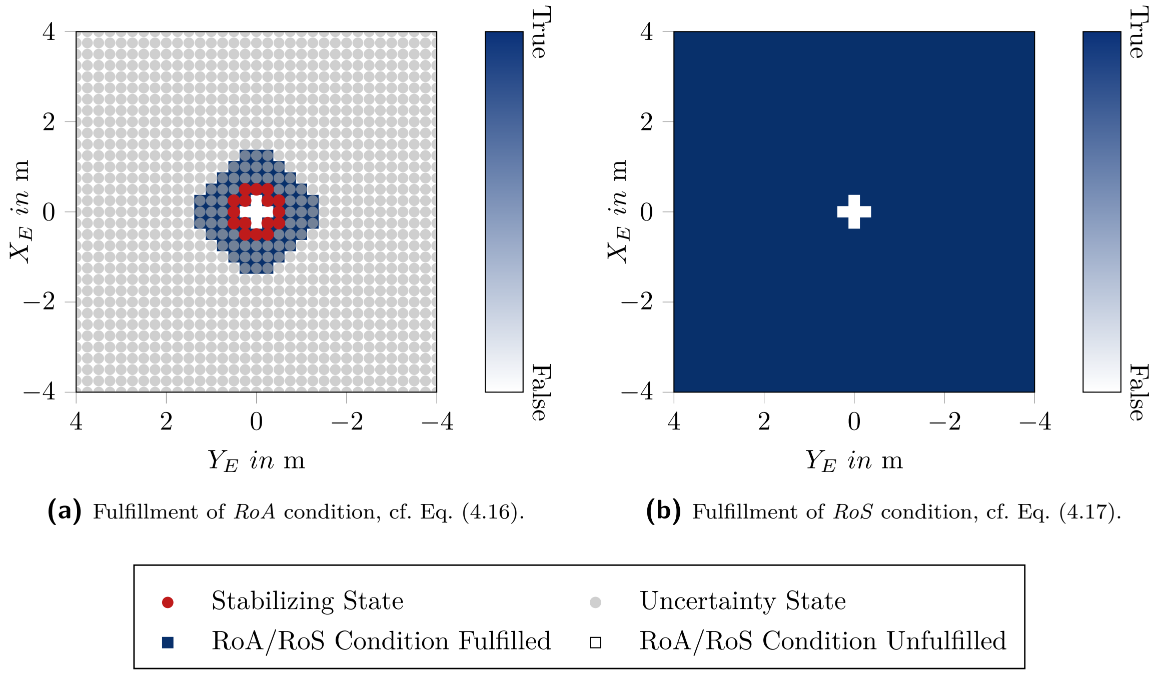
Table 5.8 details the state space configuration $\mathcal{X}_\tau$ and the considered time horizon $\mathbb{K}$ used during *RoS* estimation. The limits of the local coordinate system are chosen such that the robot can maintain a safe distance to the object that is to be bypassed even when confronted with sever failure characteristics. The time horizon, on the other hand, is chosen according to the length of samples used to extract the shared failure model in Section 5.2. Similarly, for converting the shared failure model to intervals stating the minimal and maximal failure amplitudes, $\alpha = \gamma = 0.95$ was used according to the confidence values stated in Table 5.5. For these, it is ensured that the true failure characteristics are covered by the calculated intervals. Finally, using the same procedure as in Section 4.3.3 the value of $\lambda_c = 0.35$ was determined.

Using these parameters, Algorithm 1 can be applied to estimate *RoS*. To analyze the effect of different failure characteristics, the failure types of the shared failure model are regrouped to form additional failure models, cf. Table 5.9. Next to $\mathcal{FM}_{\{N,O,OO\}}$, which is the shared failure model as described in Section 5.2, the failure model $\mathcal{FM}_{\{N\}}$ considering only the Noise failures and $\mathcal{FM}_{\{N,O\}}$ considering the Noise and Offset failures are assumed as these describe the prevailing failure characteristics. Finally, considering $\mathcal{FM}_{\{\emptyset\}}$ enables examining the theoretic case of sharing ideal data without uncertainty as well. The results are described in the following.

Results

Using the previously described parameters, an *RoS* for the collision avoidance controller can be estimated with respect to each of the considered failure models. More specifically, for each stability point an *RoS* is estimated and both are fused to generate a final *RoS* according to Section 5.1.6. The final *RoS* informs about whether or not the controller can maintain a safe distance to an object.
The main parameter of the control strategy influencing this decision is the minimal distance D_{min}. In the endeavor of determining the minimal value of D_{min} resulting in a valid *RoS*, distances starting at 10 m down to 0.5 m with a step size of 0.5 m where assumed. The minimal values resulting in non-empty *RoS* estimates are stated in Table 5.9. As is to be expected, the required distance generally increases with the severity of the assumed failure characteristics. Starting with $D_{min} = 2.0\,\mathrm{m}$ when considering no failures impairing the shared data, a maximum of $D_{min} = 9.5\,\mathrm{m}$ is required for the collision avoidance controller to handle the failure characteristics represented by $\mathcal{FM}_{\{N,O,OO\}}$.
The *RoS* estimated for the individual configurations are examined in the following paragraphs in detail.

Regions of Safety without Uncertainty To establish a baseline, the estimated *RoS* for the ideal case, that is, $\mathcal{FM}_{\{\emptyset\}}$ are examined in this paragraph. As stated in Table 5.9, the minimal distance for which a valid *RoS* can be obtained is $D_{min} = 2.0\,\mathrm{m}$.
The corresponding *RoS* estimated for x_{0_1} is shown in Fig. 5.21. Note that it does *not* show the fused *RoS*. While the blue-colored states indicate the fulfillment of the *RoA* or *RoS* condition respectively, white-colored states indicate the opposite.
Moreover, in Fig. 5.21a the transparently, gray-colored states visualize the actual *RoS* which include the states of $\mathcal{H}_V(c_{min}, c_{max})$ colored red.
Keeping in mind that the states are visualized within the robot's local coordinate system, which means that the robot is always located in the origin, one can see that these stabilizing states completely surround the robot. This means that the value of λ_c is chosen appropriately. Furthermore, as required by Theorem 4.1, the stabilizing states guarantee that the *CLF* is minimized over time at that positions. Thus, as the chosen *CLF* utilizes d_O as the central attribute for assessing a state's criticality (cf. Eq. (5.43)), it can be concluded that the controller successfully avoids collisions by increasing the distance to the object.
For white-colored states, however, this can not be guaranteed. To ensure safety nonetheless, the *RoS* condition has to apply. Its fulfillment is visualized in Fig. 5.21b, where one can see that only states outside the estimated *RoS* are colored white. This means that the system can not evolve to a state $\mathbf{x}$ with a value $V(\mathbf{x})$ greater than c_{max}. Considering that the maximal values of the chosen *CLF* occur for states $\mathbf{x}$ close to the origin (cf. Fig. 5.19), this means that the system will not evolve to a state where the distance between the robot and the obstacle becomes critical but at most to a state where it is guaranteed by the stabilizing states that the controller will increase the distance again.



(a) Fulfillment of *RoA* condition, cf. Eq. (4.16).

(b) Fulfillment of *RoS* condition, cf. Eq. (4.17).

(c) Legend for *RoA* and *RoS* illustrations, e.g. Figs. 5.21a and 5.21b.

Fig. 5.21.: Fulfillment of *RoA* and *RoS* condition by a subset of the considered state space for $D_{min} = 2.0\,\mathrm{m}$, $\mathbf{x}_{0_1}$ and $\mathcal{FM}_{\{\emptyset\}}$.

Due to the centrality of the distance, the estimated *RoS* forms a circle around the origin of the local coordinate system, as can be seen in Fig. 5.21. As a consequence of this symmetry, the *RoS* estimated for x_{0_1} does not only include the stability point x_{0_2} as well but consists of the exactly same states as the *RoS* estimated for x_{0_2}

On the one hand, it follows that the condition of Eq. (5.42) for fusing the *RoS* is fulfilled and the fused *RoS* comprises the same states. On the other hand, it brings the necessity of a fusion strategy into question.

To examine the latter in more detail, Fig. 5.22 clarifies the relation between the estimated *RoS* and the defined *CLF*. Considering the case of $X_E = 0\,\mathrm{m}$, the figure displays the values of the *CLF* with respect to Y_E. Additionally, the set of stabilizing states is indicated by red-colored parts of the curve.

As one can see, these states are found for increased values of the *CLF*. At these states, solely the distance between the robot and the object is assessed but the control goal of bringing the object into position x_{0_1} is represented merely minimally. This is in line with the goal of the run-time safety assessment, which asks only for the system's safety. On the other hand, as the *RoS* is defined as a level set of the *CLF*, it causes Algorithm 1 to include the second stability point of x_{0_2} as well. While this does not affect the provided guarantee of safety, it brings the applied fusion strategy into question. More precisely, it indicates that a single *RoS* could be sufficient to guarantee the safety of the second stability point as well. Contrarily, the requirement of having a single, global minimum for defining a valid *CLF* caused the necessity for a fusion strategy in the first place. Thus, these results indicate that this requirement on the definition of a valid *CLF* should be investigated in detail in future work. In fact, it is indicated that, for assessing safety, cost functions having multiple minima might be

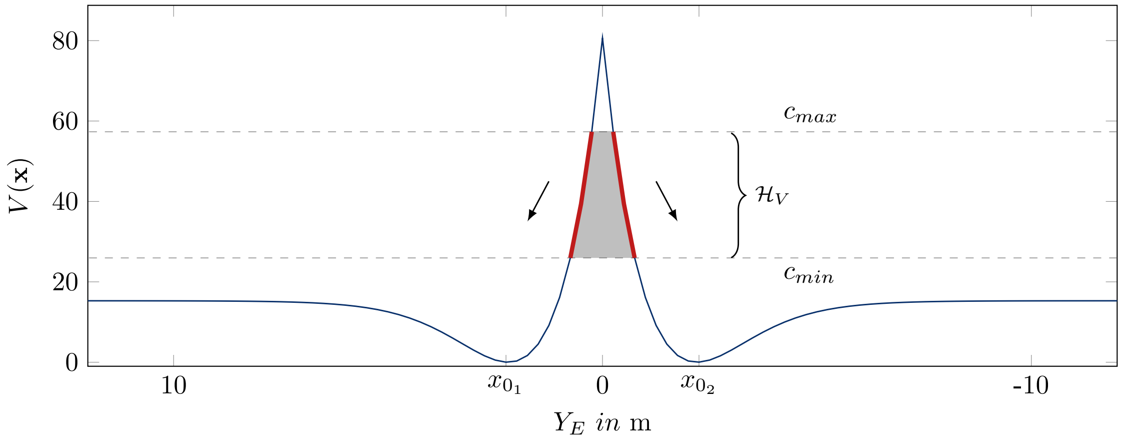


Fig. 5.22.: The figure visualizes the relation of the estimated *RoS* to the defined *CLF* by displaying the function values for $D_{min} = 2.0\,\text{m}$, x_{0_1}, and $X_E = 0\,\text{m}$. The set of stabilizing states is indicated by the gray-colored area and the red-colored part of the *CLF*'s curve. Its negative gradients are indicated by the arrows. As the resulting *RoS* is a level set of the *CLF*, it contains not only the global minimum x_{0_1} but also the **not** considered stability point of x_{0_1}.

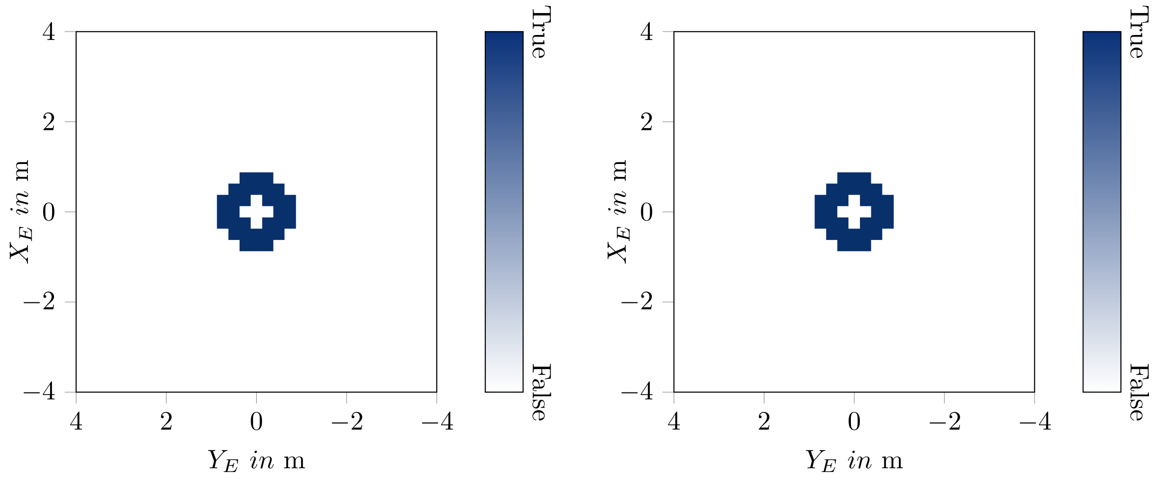
sufficient.

Nonetheless, the valid estimation of *RoS* for each stability point fulfills the condition for fusing them and therefore results in an overall, valid *RoS*. This set includes all states of the considered state space except states having $d_O < 0.35\,\text{m}$. This is in line with the sizes of the robot and the object, for which a radius of $r_s = 0.1\,\text{m}$ was assumed. Consequently, any value $d_O \leq 0.2\,\text{m}$ indicates a collision. As the considered state space is discretized by $x_S = y_S = 0.25\,\text{m}$, a value of 0.35 m is the closest possible estimation. Therefore, the collision avoidance controller is guaranteed to be safe when configured with $D_{min} = 2\text{m}$ and faced only with accurate observations.

In contrast, Fig. 5.23 shows the fulfillment of *RoA* condition for both stabilizing points but when considering $D_{min} = 1.5\,\text{m}$. Although only accurate observations are presumed and states fulfilling the *RoA* condition are surrounding the origin as well, no *RoS* is estimated. This is due to the value of λ_c. It causes not only states fulfilling the *RoA* condition to be part of $\mathcal{H}_V(c_{min}, c_{max})$ but also states that do not fulfill the same. Consequently, no *RoS* can be estimated. This indicates that, in this case, it would be possible to reduce the value of λ_c without sacrificing safety such that a lower value of D_{min} could be found.

Noise In contrast to the previous scenario, the shared data are assumed to be affected by failures. Therefore, in this paragraph, the effect of the modeled Noise failures is examined by estimating *RoS* using $\mathcal{FM}_{\{N\}}$. It is converted to a uniform failure model of $\mathbb{U}(-0.56, 0.51)$ affecting x and y components of shared data. Using the converted failure model, *RoS* were estimated for different distance values, where $D_{min} = 3.0\,\text{m}$ was found to be the smallest safe value.

Following the procedure of the previous paragraph, Fig. 5.24 shows the fulfillment of the *RoA* and *RoS* condition of the individual states for $D_{min} = 3.0\,\text{m}$. The effect of the considered failure model is visible for states surrounding the origin. As one can see, the area of states excluded from the *RoS* increases in relation to the severity of the assumed failure characteristics.



(a) Considering turning direction “left”, that is, stabilizing state at $\mathbf{x}_{0_1}$.

(b) Considering turning direction “right”, that is, stabilizing state at $\mathbf{x}_{0_2}$.

Fig. 5.23.: Fulfillment of *RoA* condition by a subset of the considered state space for $D_{min} = 1.5\,\mathrm{m}$ and $\mathcal{FM}_{\{\emptyset\}}$. Legend is given in Fig. 5.21c.

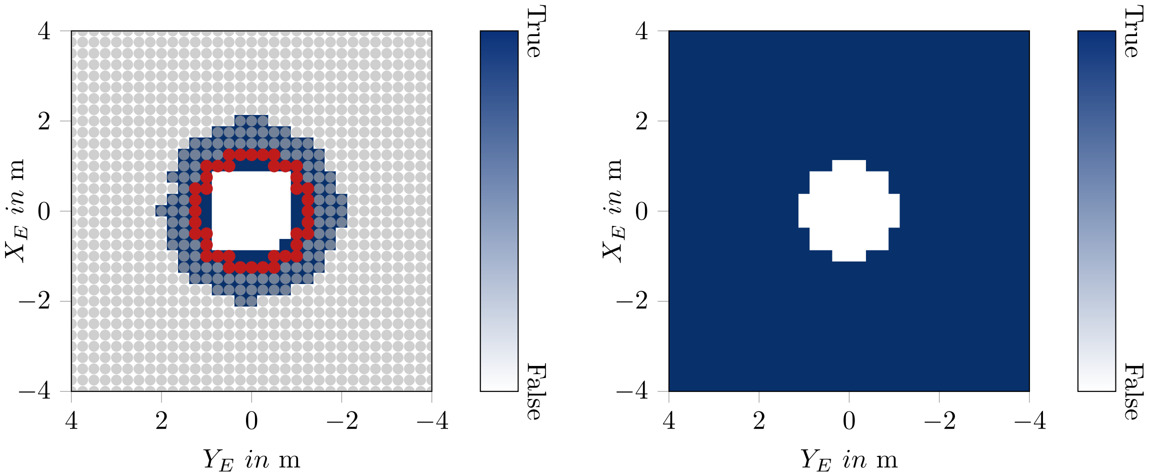
The minimal distance between the robot and the object according to the *RoS* is 1.12 m, that is, during operation, while the controller aims at maintaining a distance of $D_{min} = 3.0\,\mathrm{m}$, it is guaranteed that the distance d_O does not undercut 1.12 m due to failures of shared data. Moreover, as one can see in Fig. 5.24a, the stabilizing states fully comprise the origin. On the one hand, this underlines that λ_c is chosen appropriately once again. On the other hand, it supports the idea of investigating *CLFs* having multiple, global minima in future work.

Similar as before, considering a reduced value of $D_{min} = 2.5\,\mathrm{m}$ does not allow to estimate a valid *RoS*. Fig. 5.25 visualizes the fulfillment of the *RoA* condition by the considered states for both stability points. Compared to Fig. 5.24a the set of states fulfilling the *RoA* condition shrinks, which is caused by the reduction of D_{min} itself. Simultaneously, the set of uncertain states surrounding the stability point increases due to the considered failure characteristics. These do not only cover the origin of the states space anymore but occur in the vicinity of the considered stability point as well. This phenomenon was observed for the inverted pendulum as well, cf. Section 4.2.1.

It originates in the (almost) symmetrical failure characteristics of $\mathbb{U}(-0.56, 0.51)$. Due to their symmetry, the states not fulfilling the *RoA* condition evolves symmetrically as well. Thus, when considering only the absolute, minimal failure amplitudes to convert the failure model as discussed in Section 5.3.1, this phenomenon would have been observed for all failure characteristics.

In contrast, here, a valid set of states forming $\mathcal{H}_V(c_{min}, c_{max})$ could be possible for $\mathbf{x}_{0_1}$ but not for $\mathbf{x}_{0_2}$. As one can see, at approx. $x_{E_O} = 0.5, y_{E_O} = 1.25$ (highlighted by the red circle in Fig. 5.25b), the *RoA* condition is not fulfilled, meaning that no valid set $\mathcal{H}_V(c_{min}, c_{max})$ is possible. Consequently, no valid *RoS* can be estimated for $\mathbf{x}_{0_2}$, which entails that the fused *RoS* is the empty set as well.

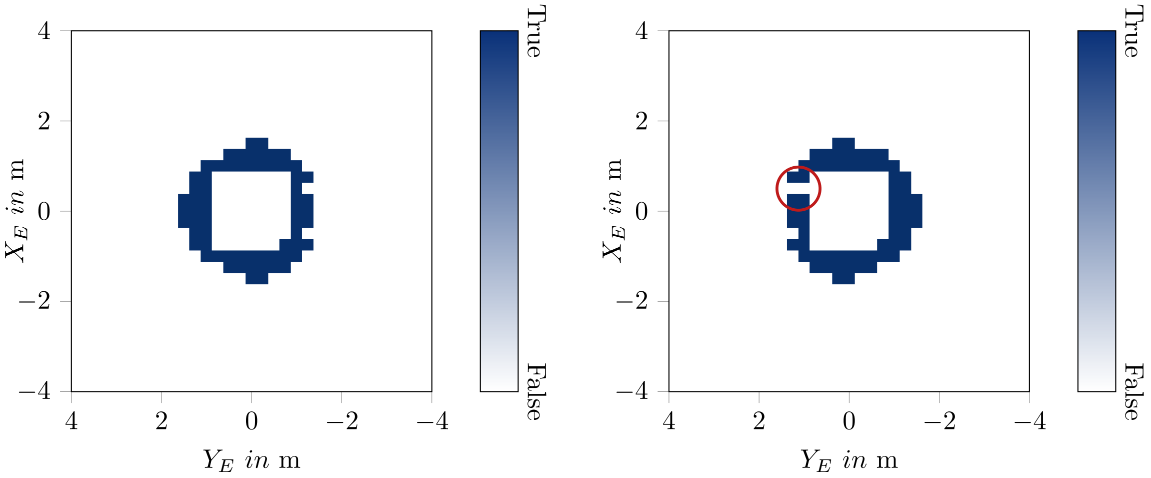
The reason for the unfulfilled *RoA* condition at this specific state is the structure of the *CLF* as well as the introduced uncertainty. In Fig. 5.25b the state $\mathbf{x}_{0_2}$ is assumed to be the targeted stability point, that is, it is assumed that the decision on the turning



(a) Fulfillment of *RoA* condition, cf. Eq. (4.16).

(b) Fulfillment of *RoS* condition, cf. Eq. (4.17).

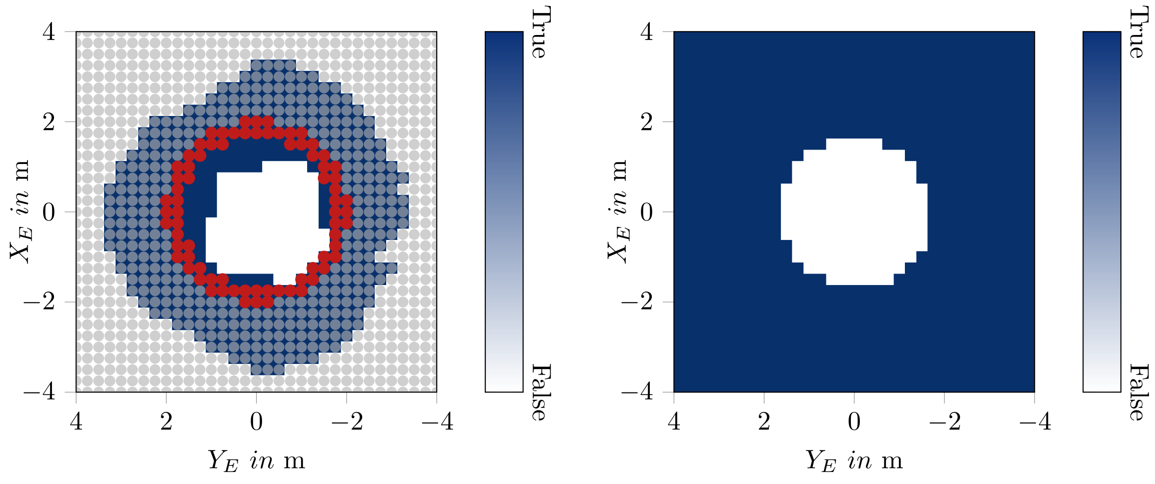
Fig. 5.24.: Fulfillment of *RoA* and *RoS* condition of states for $D_{min} = 3.0\,\mathrm{m}$, $\mathbf{x}_{0_2}$, and when considering $\mathcal{FM}_{\{N\}}$. Legend is given in Fig. 5.21c.



(a) Fulfillment of Eq. (4.16) for $D_{min} = 2.5\,\mathrm{m}$ when considering $\mathbf{x}_{0_1}$.

(b) Fulfillment of Eq. (4.16) for $D_{min} = 2.5\,\mathrm{m}$ when considering $\mathbf{x}_{0_2}$. The red circle highlights the state due to which no *RoS* can be estimated as the *RoA* condition is not fulfilled.

Fig. 5.25.: Comparison of *RoA* condition fulfillment when considering $\mathcal{FM}_{\{N\}}$ and $D_{min} = 2.5\,\mathrm{m}$ for $\mathbf{x}_{0_1}$ and $\mathbf{x}_{0_2}$. Legend is given in Fig. 5.21c.



(a) Fulfillment of *RoA* condition, cf. Eq. (4.16).

(b) Fulfillment of *RoS* condition , cf. Eq. (4.17).

Fig. 5.26.: Fulfillment of *RoA* and *RoS* condition of states when considering $\mathcal{FM}_{\{N,O\}}$ for D_{min} = 4.5 m and $\mathbf{x}_{0_1}$. Legend is given in Fig. 5.21c.

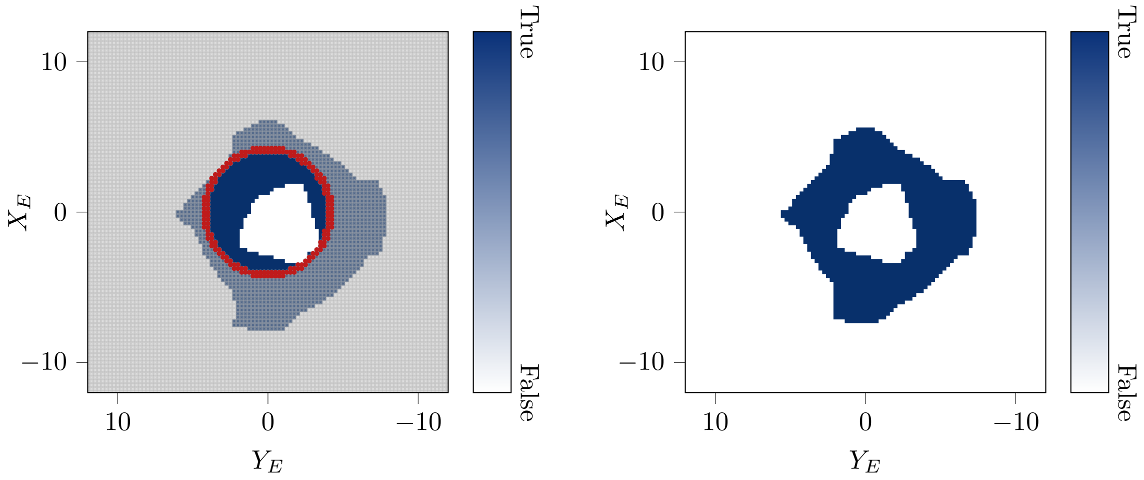
direction is made already. Consequently, the robot aims at maximizing the distance to the object according to Eq. (5.44) while turning according to Eq. (5.26). However, the minimal failure amplitude of −0.56 m causes the controller to possibly observe the state of the object in the range of $x_{E_O} \in [-0.06\,\text{m}, 1.01\,\text{m}]$. Therefore, the sign of the x component switches, resulting in diverging turning directions and ultimately causing the *CLF* to be maximized instead of being minimized in the worst case.

Increasing the minimal distance D_{min} resolves the problem as the number of stabilizing states increases, especially for greater values of d_O. For these states, the weight of W_{circle} causes the *CLF* to be sufficiently minimized by solely moving away from the object irrespectively of the chosen turning direction.

Nevertheless, the difference between $\mathbf{x}_{0_1}$ and $\mathbf{x}_{0_2}$ regarding their valid and invalid *RoS* estimations presents a contrasting example to the formulated hypothesis of allowing multiple global minima for a single *CLF* to circumvent the required fusion strategy. In contrast to the previous example in which a valid *RoS* was estimated either for both or none of the stability points, a distinction occurs in this case. This further motivates examining this direction in future work.

Noise and Offsets Considering $\mathcal{FM}_{\{N,O\}}$, which additionally takes the Offset failure type into account, increases the inequality within the failure characteristics such that the converted failure model is now $\mathbb{U}(-0.54, 1.01)$. As a consequence, the fulfillment of *RoA* condition is asymmetric with regard to the considered states as well, cf. Fig. 5.26a.

Contrarily, the fulfillment of the *RoS* condition is not affected by this asymmetry, cf. Fig. 5.26b. This is because the estimated *RoS* is a level set of the *CLF*. This means, due to the high value of W_{circle}, the *CLF* forms a circular pattern around the origin, which causes the estimated *RoS* to form circularly around the origin as well. As the *RoS* condition evaluates whether the function value of the *CLF* may evolve to a value outside the *RoS*, the excluded states are inherently not fulfilling the condition. Moreover, the stabilizing states, forming the *border* of the *RoS*, naturally



(a) Fulfillment of *RoA* condition of states for $D_{min} = 9.5\,\mathrm{m}$.

(b) Fulfillment of *RoA* condition of states for $D_{min} = 9.0\,\mathrm{m}$.

Fig. 5.27.: Comparing the fulfillment of *RoA* condition (cf. Eq. (4.16)) of states for $\mathcal{FM}_{\{N,O,OO\}}$ and $\mathbf{x}_{0_2}$. Legend is given in Fig. 5.21c.

fulfill the condition as they guarantee that the function value is minimized. Thus, the asymmetry affecting Fig. 5.26a caused by failure characteristics is not affecting the *RoS* condition.

Noise, Offsets, and Outlier Offset Finally, the failure model $\mathcal{FM}_{\{N,O,OO\}}$, which encompasses all failure types, is considered. For estimating the corresponding *RoS*, the failure model is converted to the uniform failure model of $\mathbb{U}(-0.54, 2.48)$ affecting the x and y component. Due to the increased failure amplitudes, a distance of $D_{min} = 9.5\,\mathrm{m}$ provides a valid *RoS* while $D_{min} = 9.0\,\mathrm{m}$ does not. Although stabilizing states surrounding the origin for $D_{min} = 9.0\,\mathrm{m}$, as can be seen in Fig. 5.27b, Algorithm 1 does return an empty set. This is due to the asymmetry introduced by the failure characteristics, which contradicts the circular shape dictated by the *CLF*, as can be seen in Fig. 5.27a. Further reducing D_{min}, therefore, causes the set of stabilizing states to either intersect with the outer or inner white area. In both cases, no valid set for forming the hull of stabilizing states $\mathcal{H}_V$ can be found as it would include states not fulfilling the *RoA* condition.

5.3.3. Evaluation Using the Circle Scenario

The estimation of a valid *RoS* for different values of D_{min} showed that the concepts of *GFM* and *RoS* can be combined in such a way that different failure characteristics can be analyzed as part of a run-time safety assessment. More specifically, for the assumed failure characteristics, the minimal value of D_{min}, which is deemed safe according to the RoS analysis, was calculated and is stated in Table 5.9.

In this subsection, the validity of the decisions, that is, whether or not the determined values of D_{min} are indeed maintaining the safety of the system is evaluated.

For that, the circle scenario as described in Section 5.1.2 is simulated using the parameters stated in Table 5.10. Taking the stochastic nature of the failure characteristics into

Tab. 5.10.: Parameters for simulating the circle scenario.

Parameter	Value	Description
N_{sim}	20	Number of simulations for each starting position.
τ	$0.07\,\mathrm{s}$	Control period of the controller.
T_{sim}	$150.0\,\mathrm{s}$	Maximal duration of each simulation.
r_{cirlce}	$25\,\mathrm{m}$	Radius of the virtual circle the starting positions of the robots are on.
D_{min}	$[0.1\,\mathrm{m}, 5.0\,\mathrm{m}] \cup 9.5\,\mathrm{m}$	Control parameter determining the minimal distance to keep to obstacles or other robots.

account, the scenario was simulated $N_{sim} = 20$ times for each starting state on a virtual circle with $r_{circle} = 25\,\mathrm{m}$. Each simulation covers a time horizon of $T_{sim} = 150\,\mathrm{s}$ but is stopped early if a collision is detected. In accordance with the previous estimation of RoS, the controller's control period is simulated with $\tau = 0.07\,\mathrm{s}$.

To test not only the previously calculated values of D_{min} as given in Table 5.9 but also evaluate whether or not increased values are necessary or decreased values are possible, the simulations were repeated for values of $D_{min} \in \{[0.1\,\mathrm{m}, 5.0\,\mathrm{m}] \cup 9.5\,\mathrm{m}\}$. In other words, by explicitly undercutting the values of Table 5.9 collisions are provoked.

For each simulation, the distance between both robots d_{robots} is observed. Once this value falls below zero, a collision is detected and the safety requirement is violated. An overview of the results is given in Table 5.11. These are discussed next before the results are examined in detail for each considered failure model.

Overview on Results

Table 5.11 summarizes the results of the simulations by stating the minimal observed distance d_{robots} for each of the considered failure characteristics and each value of D_{min}. Cells colored red highlight simulations resulting in collisions, that is, where safety is violated. As one can see, these occur only for values of D_{min} significantly lower than what is deemed acceptable by the previous RoS analysis (cf. Table 5.9). Contrarily, the minimally acceptable values are marked by gray-colored cells. For these, no collisions are detected.

Generally, the observed distance d_{robots} is falling below the targeted distance D_{min}, which is to be expected due to the employed P-controller. For $\mathcal{FM}_{\{\emptyset\}}$ this does not result in any violation of the safety requirement. On the one hand, this means configured with $D_{min} = 2.0\,\mathrm{m}$ the controller will indeed maintain the safety of the system. On the other hand, it indicates that this value could be reduced and shows therefore the conservatism of the approach.

Similarly, when considering $\mathcal{FM}_{\{N\}}$ negative values of d_{robots} are found only for $D_{min} = 0.1\,\mathrm{m}$ while the run-time safety assessment guarantees safety only for values $D_{min} \geq 3.0\,\mathrm{m}$. The correspondingly gray-colored cell in Table 5.11 states that a

Tab. 5.11.: The minimal distances observed during simulation when parameterizing the collision avoidance controller with decreasing values of D_{min} and confronting it with different failure characteristics. Red-colored cells indicate simulations in which the safety requirement was violated. Gray-colored cells indicate the results obtained when configuring the controller with minimal values of D_{min} deemed acceptable by the *RoS* analysis.

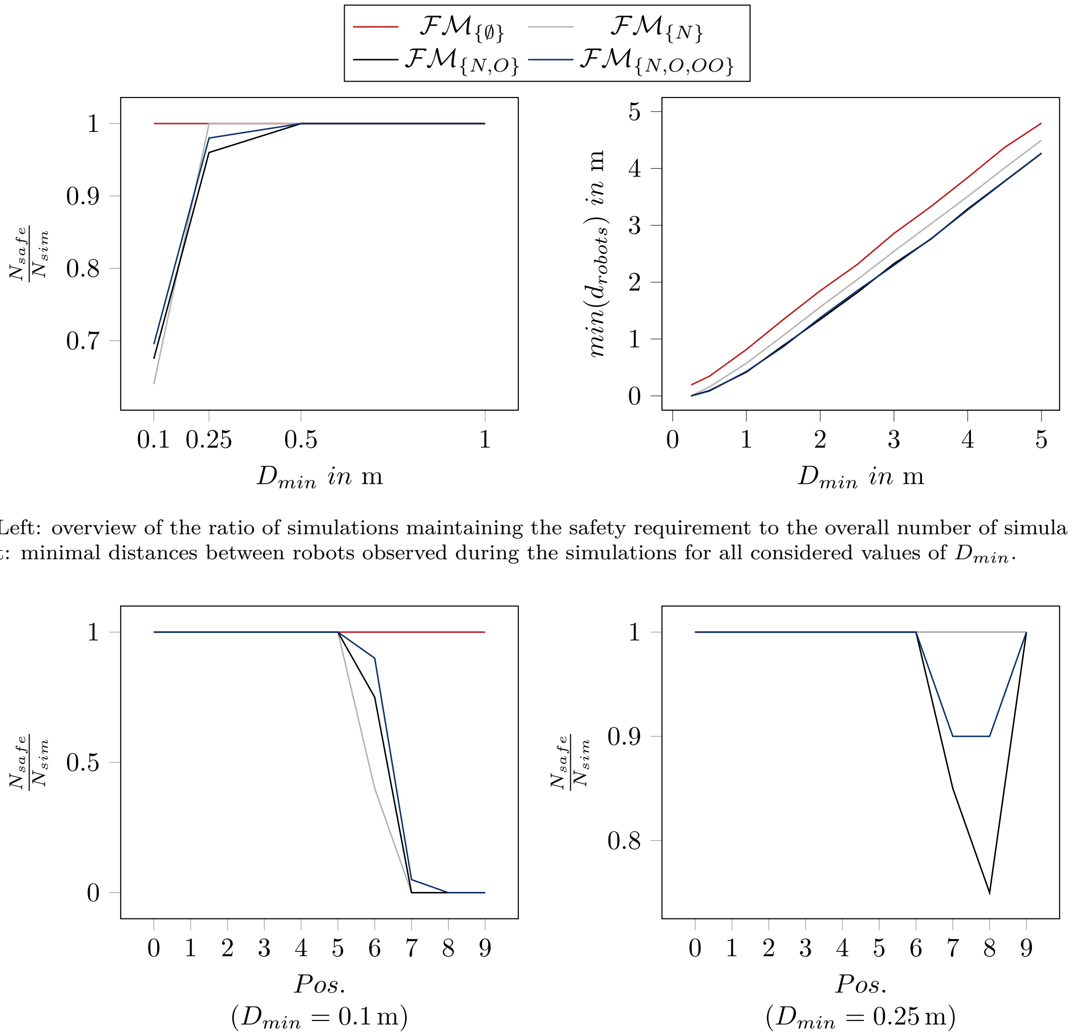
D_{min}	$\min(d_{robots})$			
	$\mathcal{FM}_{\{\emptyset\}}$	$\mathcal{FM}_{\{N\}}$	$\mathcal{FM}_{\{N,O\}}$	$\mathcal{FM}_{\{N,O,OO\}}$
0.1	0.042	-0.0008	-0.0008	-0.0008
0.25	0.193	0.001	-0.0005	-0.0004
0.5	0.347	0.159	0.095	0.087
1.0	0.817	0.575	0.419	0.430
1.5	1.343	1.063	0.887	0.865
2.0	1.849	1.562	1.344	1.374
2.5	2.309	2.043	1.815	1.848
3.0	2.857	2.543	2.327	2.297
3.5	3.329	3.025	2.756	2.763
4.0	3.840	3.507	3.288	3.271
4.5	4.37	4.009	3.776	3.772
5.0	4.78	4.496	4.269	4.265
9.5	9.336	8.89	8.646	8.682

minimal distance of $d_{robots} = 1.849\,\mathrm{m}$ is preserved at all times, which further supports the *RoS* analysis providing this value.

For $\mathcal{FM}_{\{N,O\}}$ and $\mathcal{FM}_{\{N,O,OO\}}$ the pattern repeats. In both cases, it can be shown that no collision occurs for the values of D_{min} determined by the *RoS* analysis but actual collisions occur only for values up to $D_{min} = 0.25\,\mathrm{m}$.

In contrast to the results obtained for $\mathcal{FM}_{\{\emptyset\}}$ and $\mathcal{FM}_{\{N\}}$, which undercut their respective values of D_{min} with varying magnitudes as a result of the increased failure characteristics, $\mathcal{FM}_{\{N,O\}}$ and $\mathcal{FM}_{\{N,O,OO\}}$ exhibit values at a similar level. This indicates that the additional failure type of Outlier-Offset does not has a distinct impact on the results.

This can be observed in Fig. 5.28a as well. While the left diagram shows the ratio of simulations maintaining safety over the overall number of simulations for distances of $D_{min} \leq 1.0\,\mathrm{m}$, the right diagram depicts the minimal distances of Table 5.11 with respect to the configured value of D_{min}. As one can see in the latter, the black and blue curves showing the results of $\mathcal{FM}_{\{N,O\}}$ and $\mathcal{FM}_{\{N,O,OO\}}$ cover each other due to the similar values.



(a) Left: overview of the ratio of simulations maintaining the safety requirement to the overall number of simulations. Right: minimal distances between robots observed during the simulations for all considered values of D_{min}.

(b) Illustrating the number of successful (safe) simulations over all simulations for all positions and distances $D_{min} = 0.1$ m (left) and $D_{min} = 0.25$ m (right).

Fig. 5.28.: Overview of results of circle scenario.

Tab. 5.12.: Ranges of failure amplitudes for the x and y component observed during simulations of the circle scenario.

	$\mathcal{FM}_{\{\emptyset\}}$	$\mathcal{FM}_{\{N\}}$	$\mathcal{FM}_{\{N,O\}}$	$\mathcal{FM}_{\{N,O,OO\}}$
x	$[0, 0]$	$[-0.281, 0.346]$	$[-0.295, 0.61]$	$[-0.296, 0.61]$
y	$[0, 0]$	$[-0.323, 0.271]$	$[-0.243, 0.75]$	$[-0.247, 0.753]$

The similarity of their results is originated in the activation of the individual failure types. Studying Table 5.12, which states the observed minimal and maximal failure amplitudes for each failure model and over all simulations, reveals that the ranges of $\mathcal{FM}_{\{N,O\}}$ and $\mathcal{FM}_{\{N,O,OO\}}$ are close to each other and thereby indicates that the additional failure type Outlier-Offset is not activated. This is due to the scarce occurrence of this failure type, which is limited to the specific position of $\mathbf{o}_k = [x = 1.1\,\mathrm{m}\ y = 0\,\mathrm{m}\ \Theta = -180°]^T$, cf. Section 5.2.2. As a result, the failure type is not activated during the simulations, causing the results of $\mathcal{FM}_{\{N,O\}}$ and $\mathcal{FM}_{\{N,O,OO\}}$ to not vary due to their general failure characteristics but due to the specific failure amplitudes sampled from the failure models during simulation.

The consequence can be observed in the left diagram of Fig. 5.28a as well. As both failure models impair the position data shared between robots during simulation similarly, the ratios of simulations without collisions obtain similar values.

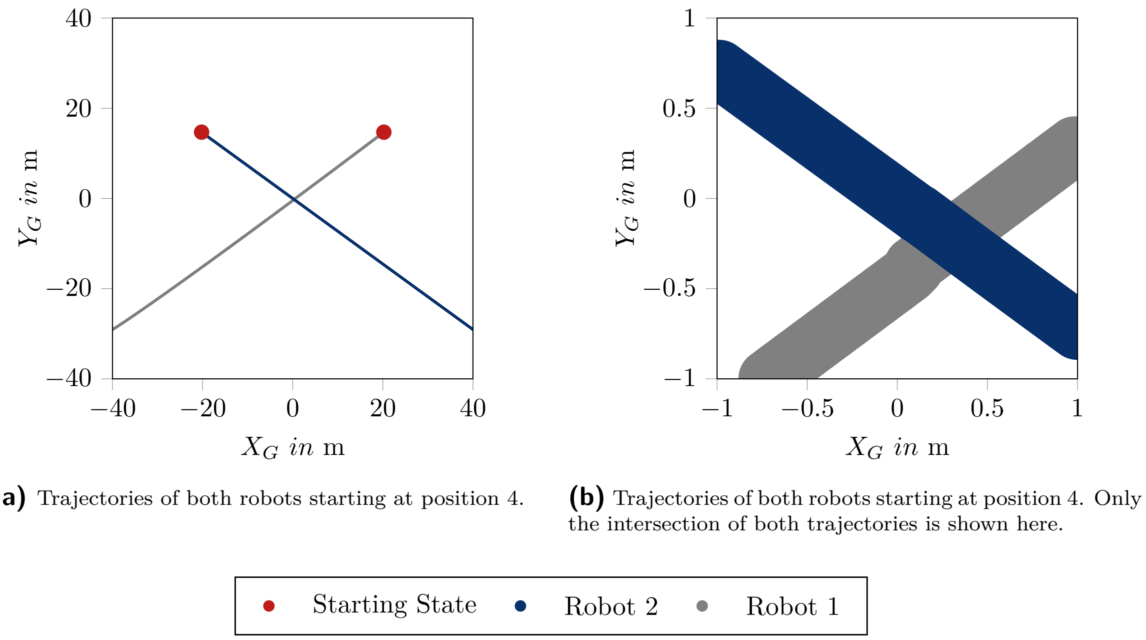
Assuming $D_{min} = 0.1\,\mathrm{m}$, one can see that $\mathcal{FM}_{\{N\}}$ causes the minimal ratio of 0.64 safe simulations, which translates to 72 simulations ending with both robots colliding. While $\mathcal{FM}_{\{N,O\}}$ causes 65 collisions (a ratio of 0.675), the failure model $\mathcal{FM}_{\{N,O,OO\}}$ causes only 61 collisions (a ratio of 0.695). As the value of $D_{min} = 0.1\,\mathrm{m}$ is covered within the ranges of failure amplitudes for all of these three failure models (cf. Table 5.12), the occurrence of collisions merely depends on the randomness with which failure amplitudes are sampled from the failure models and the starting position of the robots.

Especially the latter can be examined in Fig. 5.28b. For $D_{min} = 0.1\,\mathrm{m}$, collisions occur only for starting positions 6,7,8, and 9. As can be seen in Fig. 5.2a, these positions involve acute incident angles, that is, the robots are approaching laterally to each other. If this is combined with increased failure models and the known overshoot behavior [25] of the P-controller employed within the collision avoidance strategy, that is, its inability to slowly approach the target distance but to undercut it, collisions of both robots occur.

For $D_{min} = 0.25\,\mathrm{m}$, the likeliness of collisions is reduced. This is as the value of D_{min} is now closer to the absolute borders of the ranges of simulated failure amplitudes (cf. Table 5.12), which means that failure amplitudes causing the controller to incorrectly disengage the collision avoidance are less likely to occur. As a consequence, no collision is observed for $\mathcal{FM}_{\{N\}}$, while $\mathcal{FM}_{\{N,O\}}$ and $\mathcal{FM}_{\{N,O,OO\}}$ exhibit similar ratios of safe simulations again, cf. Fig. 5.28a. When considering the ratio with respect to the starting positions (right diagram of Fig. 5.28b), the effect of the incident angle is underlined once again.

While the incident angle resulting from starting positions 7 and 8 is such that collisions occur, the robots are starting almost in parallel for position 9 which, combined with the increased value of D_{min} prevents collisions now. This is as the overshoot behavior of the P-controller is mitigated by the lateral movement of the robots.

To examine these results in more detail, the following subsections address results for individual failure models.



(a) Trajectories of both robots starting at position 4.

(b) Trajectories of both robots starting at position 4. Only the intersection of both trajectories is shown here.

(c) Legend for trajectory visualizations, e.g. Figs. 5.29a and 5.29b.

Fig. 5.29.: Exemplary results of a simulation of the circle scenario considering the failure model $\mathcal{FM}_{\{\emptyset\}}$, parameterizing the controller with $D_{min} = 0.1\,\text{m}$, and assuming starting position 4. The legend is given in Fig. 5.29c.

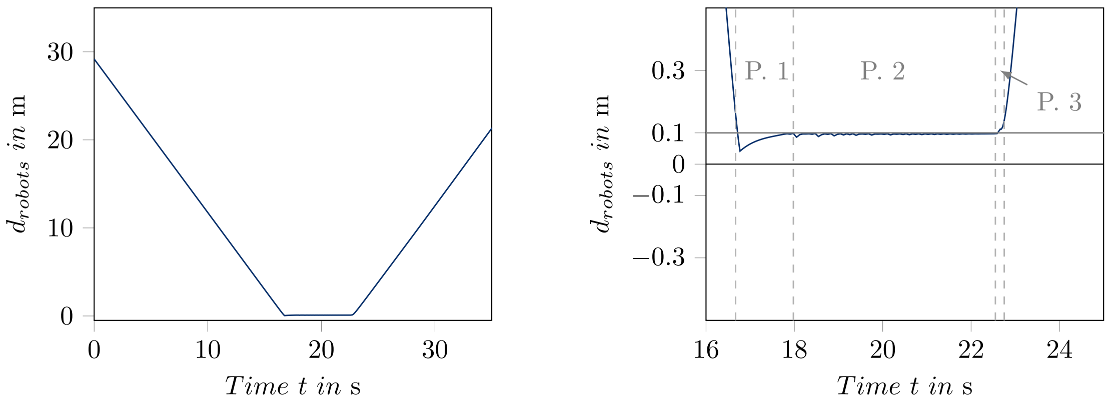
Results for $\mathcal{FM}_{\{\emptyset\}}$

For examining the results of $\mathcal{FM}_{\{\emptyset\}}$ in detail, an exemplary simulation is visualized in Fig. 5.29 and in Fig. 5.30. The former figure illustrates the trajectories for one simulation of the circle scenario configured with starting position 4 and $D_{min} = 0.1\,\text{m}$. While the starting positions are indicated by red circles, the positions of robot 1 and robot 2 are colored in gray and blue respectively. The latter figure visualizes the corresponding distances d_{robots}.

The minimal distance observed during this simulation is $d_{robots} = 0.042\,\text{m}$, which shows that the robots successfully avoid collisions with each other despite being placed on intersecting paths. As envisioned, their trajectories intersect close to the origin, as can be seen in Fig. 5.29b. Moreover, the trajectory of robot 1 shows minor deviations indicating the collision avoidance maneuver. Its successful execution is indicated by Fig. 5.30 where it can be seen that the distance d_{robots} firstly decreases but does not fall below zero and thereby states that no collision occurs.

At first, both robots are approaching each other resulting in decreasing values of d_{robots}, cf. Fig. 5.30a. Once the configured threshold of $D_{min} = 0.1\,\text{m}$ is undercut, collision avoidance is engaged and the first phase applies. This is denoted by *P. 1* in Fig. 5.30b. Similarly, *P. 2* and *P. 3* refer to the second and third phases of the collision avoidance strategy, cf. Section 5.1.4.

Agreeing with Table 5.11, the minimal value of $d_{robots} = 0.042\,\text{m}$ is reached during the first phase as the implemented P-controller can not directly stop the movement of the robot and therefore overshoots, a known behavior of P-controllers [25]. In contrast, after reaching the minimal value, the controller provides control actions to retain a distance of D_{min}.



(a) Temporal evolution of d_{robots} shown only for the relevant time horizon.

(b) Temporal evolution of d_{robots} shown for the time horizon where the collision avoidance is engaged. The gray-colored line indicates the value of $D_{min} = 0.1\,\mathrm{m}$ with which the controller was configured. P. 1, P. 2, and P. 3 denote the phases 1-3 of the collision avoidance strategy, cf. Section 5.1.4.

Fig. 5.30.: Temporal evolution of d_{robots} shown only for the relevant time horizon. The controller was configured with $D_{min} = 0.1\,\mathrm{m}$ and confronted with ideal observations, that is, $\mathcal{FM}_{\{\emptyset\}}$. As a result, the value of D_{min} is undercut due to the usage of a P-controller, but the safety is maintained as d_{robots} is above 0 at all times.

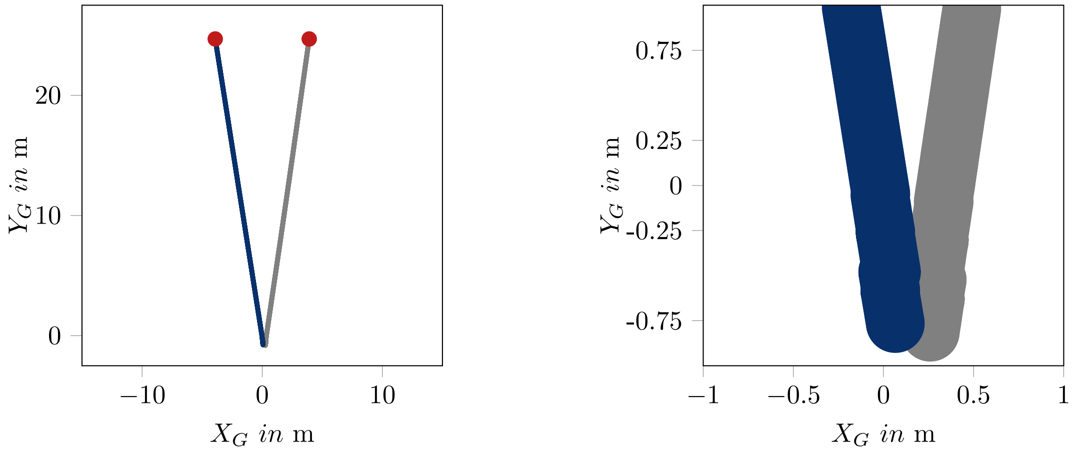
Once D_{min} is reached again, phase 2 starts, where both robots are bypassing each other on a circular trajectory. In this phase, d_{robots} remains stable at a value close to D_{min}. Here, no variations are seen as only ideal observations are assumed such that the collision avoidance strategy can be followed perfectly.

Finally, after the robots bypassed each other, phase 3 is entered in which disengagement is performed. For that, the conditions stated in Section 5.1.4 are monitored. Due to the employed P-controller and the periodicity of $\tau = 0.07\,\mathrm{s}$ with which the collision avoidance is performed, these are not directly met. Nevertheless, as no failures are influencing the systems, only a period of 0.2 s is required to fulfill the conditions and disengage the collision avoidance strategy. Thus, the robots aim at their respective target positions and start moving away from each other.

The trajectories of both robots end at their assigned target positions which are 75 m from their starting positions.

Results for $\mathcal{FM}_{\{N\}}$

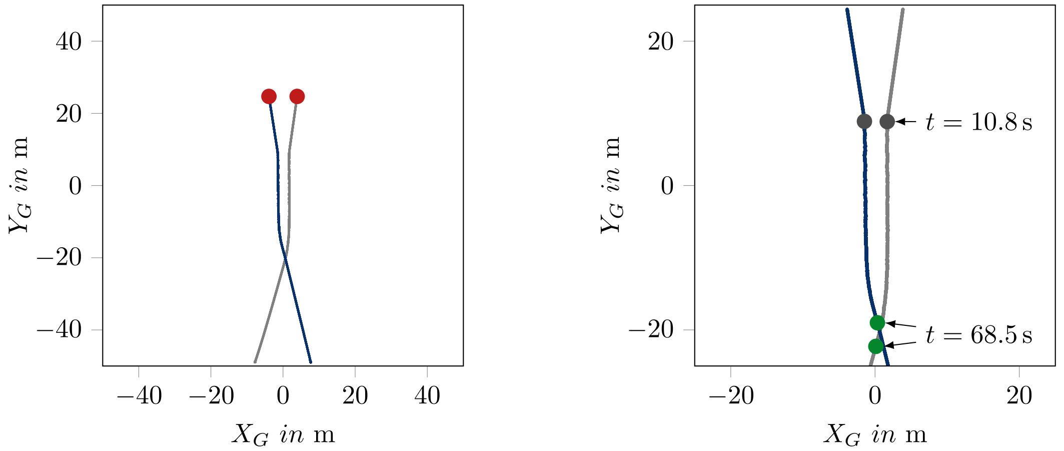
Opposed to the previous case, collisions are observed for $D_{min} = 0.1\,\mathrm{m}$ when considering $\mathcal{FM}_{\{N\}}$. Fig. 5.31 illustrates an exemplary simulation where the robots started at position 9. Thus, as shown in Fig. 5.31a their trajectories intersect with an acute incident angle. This means that deciding the turning direction, that is, deciding between $\mathbf{x}_{0_1}$ and $\mathbf{x}_{0_2}$ is not affected by the failures of shared data. However, the target distance of $D_{min} = 0.1\,\mathrm{m}$ is smaller than the range of possible failure amplitudes (cf. Table 5.12), which causes the controller to disengage collision avoidance despite the true distance being less than D_{min}. The goal-finding strategy then drives the robot towards its target position, which, by design, requires intersecting the other robot's trajectory. This, combined with the overshoot behavior of the P-controller employed by the collision avoidance strategy, causes both robots to collide.



(a) Trajectories of both robots starting at position 9.

(b) Trajectories of both robots starting at position 9. Only the intersection of both trajectories is shown here.

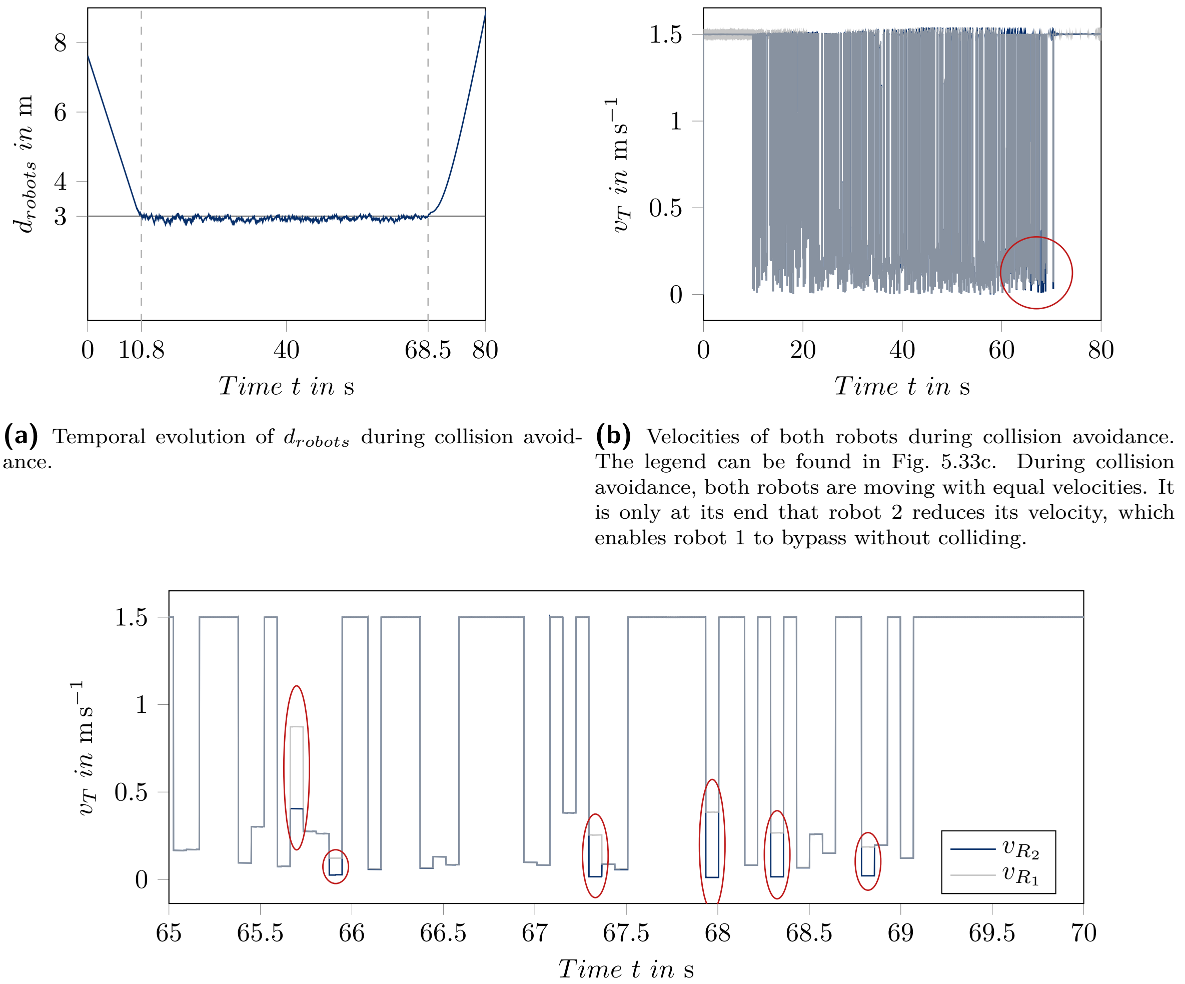
Fig. 5.31.: Exemplary results of a simulation of the circle scenario considering the failure model $\mathcal{FM}_{\{N\}}$, parameterizing the controller with $D_{min} = 0.1\,\text{m}$, and assuming starting position 9. The legend is given in Fig. 5.29c. The value of $D_{min} = 0.1\,\text{m}$ severely undercuts the value of $D_{min} = 3.0\,\text{m}$, which is deemed as the minimal safe value according to the estimated *RoS*. Consequently, the robots are colliding and the safety requirement is violated.



(a) Trajectories of both robots starting at position 9.

(b) Trajectories of both robots starting at position 9. Only the intersection of both trajectories is shown here. While the robots are having similar y coordinates when collision avoidance is engaged (dark-gray-colored circles), robot 2 progresses slower, allowing robot 1 to bypass it without colliding. Consequently, when collision avoidance is disengaged (green-colored circles), robot 2 progressed not as far as robot 1.

Fig. 5.32.: Exemplary results of a simulation of the circle scenario considering the failure model $\mathcal{FM}_{\{N\}}$, parameterizing the controller with $D_{min} = 3.0\,\text{m}$, and assuming starting position 9. The legend is given in Fig. 5.29c.



(a) Temporal evolution of d_{robots} during collision avoidance.

(b) Velocities of both robots during collision avoidance. The legend can be found in Fig. 5.33c. During collision avoidance, both robots are moving with equal velocities. It is only at its end that robot 2 reduces its velocity, which enables robot 1 to bypass without colliding.

(c) Velocities of robots in the end of phase 2 and during phase 3 of collision avoidance. Robot 2 is moving with lower velocities at times marked with red-colored ellipses causing it to fall behind robot 1 and thereby allowing it to bypass without colliding.

Fig. 5.33.: Visualization of distance d_{robots} and velocities of both robots during circle scenario. The failure model $\mathcal{FM}_{\{N\}}$ was considered while the controller was configured with $D_{min} = 3.0\,\mathrm{m}$. The robots started at starting position 9.

This deficiency is resolved when increasing D_{min} to 3.0 m. As guaranteed by the estimated RoS, the robots maintain their safety and avoid collisions with each other. Fig. 5.32 illustrates trajectories starting at the same position but with the increased distance value.

While the starting position still provokes the robot's trajectories to intersect with an acute incident angle, the increased value of D_{min} mitigates problem posed the overshoot behavior of the P-controller. Thus, at $t = 10.8\,\mathrm{s}$, the distance between both robots undercuts the threshold of $D_{min} = 3.0\,\mathrm{m}$ at which point the collision avoidance is engaged. This point is highlighted by a vertical, dashed line in Fig. 5.33a and by the dark-gray-colored circles on the trajectories of both robots in Fig. 5.32b. As one can see, both robots engage the collision avoidance strategy at similar Y_G values. From that point on, both robots drive in parallel as a consequence of their goal to maintain the configured distance and to bypass each other.

This point of equilibrium, however, is resolved by the failure amplitudes impairing the data shared by the robots and their resulting movements. More specifically, as displayed in Fig. 5.33b the randomness affecting the perceived distance between the robots cause them to accelerate and decelerate unequally. Consequently, at the end of the parallel movement (indicated by the red circle in Fig. 5.33b), robot 2 exhibits decreased velocities. This period is depicted in Fig. 5.33c where red ellipses are highlighting the relevant sections. As can be seen, robot 2 (v_{R_2}) exhibits reduced velocity values. These allow robot 1 to obtain a position in front of robot 2 and thereby bypass it without a collision.

This situation is highlighted by green circles in Fig. 5.32b indicating the positions of both robots at $t = 68.5\,\mathrm{s}$. Comparing the trajectories with the distance values d_{robots} in Fig. 5.33a, where the same time step is indicated by a vertical, dashed line, underlines that the collision avoidance successfully prevented a collision and enabled both robots to arrive at their assigned target positions.

Results for $\mathcal{FM}_{\{N,O\}}$

Adding the Offset failure increases the overall magnitude of failure amplitudes observed during simulation, cf. Table 5.12. Consequently, collisions are observable for $D_{min} = 0.25\,\mathrm{m}$ as well. Fig. 5.34 shows the trajectories of both robots starting at position 7. The robots aim at their target positions which brings them on intersecting trajectories. Due to the increased failure amplitudes, the distances they perceive to each other are impaired such that collision avoidance is not engaged. Again, this phenomenon is combined with the overshoot behavior of the P-controller and results in a collision.

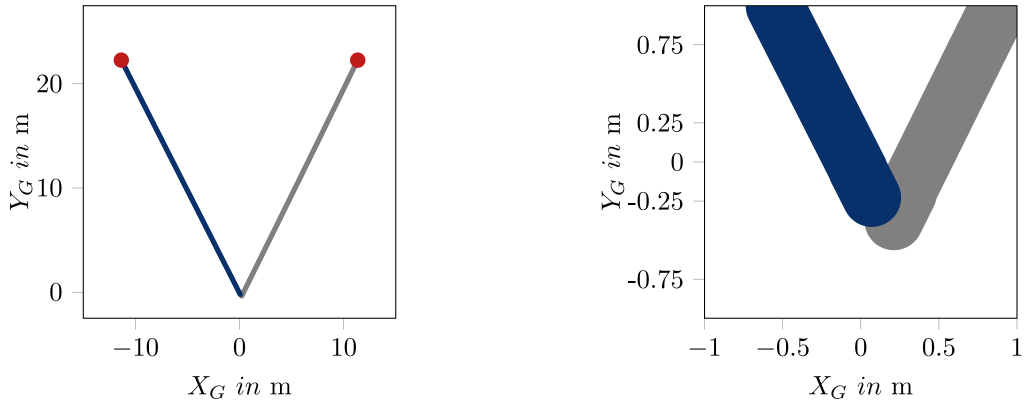
Therefore, simulations assuming the same starting position but with an increased value of $D_{min} = 4.5\,\mathrm{m}$ show that collisions are successfully avoided, cf. Fig. 5.35. As discussed in the previous example, the increased distance enables the P-controller of both robots to stabilize their respective distance and maintain the safety requirement despite its overshoot. Moreover, the incident angle, again, causes a period of equilibrium after collision avoidance is engaged. During this period, the robots are driving in parallel but exhibit different velocities, which ultimately enables robot 1 to bypass robot 2 safely.

Results for $\mathcal{FM}_{\{N,O,OO\}}$

The addition of the Outlier-Offset to the considered failure types changes the estimated *RoS* as the theoretical range of failure amplitudes increases. However, when simulating the circle scenario using $\mathcal{FM}_{\{N,O,OO\}}$, the failure type does not become active as it only occurs at the specific global pose of $[0\,\mathrm{m}\ 1.1\,\mathrm{m}\ {-180}°]^T$. Therefore, in simulation, the robots behave as when considering the previous failure model.

It is important to note here that this is the very reason for why it is not sufficient to simulate a system to prove its safety. It has to be shown that all relevant hazards (in this case the occurrence of the Outlier-Offset failure type) are addressed, which is not possible when coverage of the executed simulations can not be guaranteed. Contrarily, the *RoS*-based analysis guarantees that all *modeled* failure characteristics and the possible situations that they entail are considered.

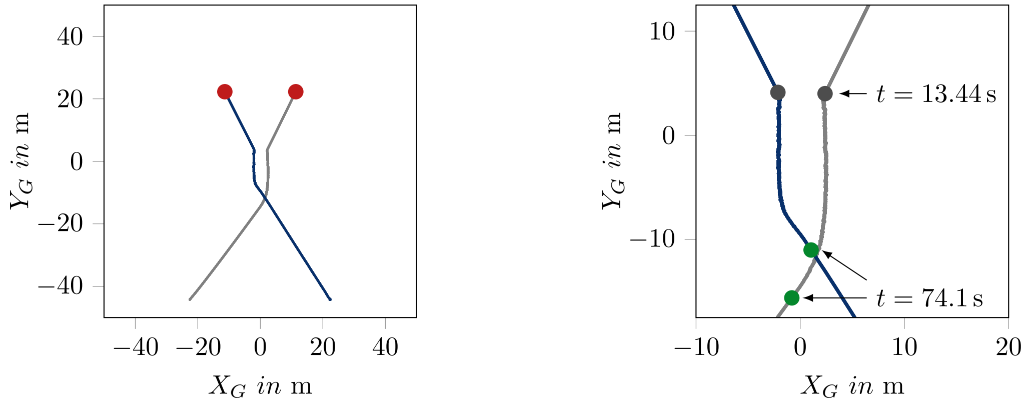
Consequently, when configured with $D_{min} = 9.5\,\mathrm{m}$, the controller is guaranteed to maintain the safety of the system. Fig. 5.36 shows a successful simulation for starting position 0. Accordingly, both robots' trajectories intersect at the origin of the global coordinate system, where it can be seen that the robots decide to bypass each other on



(a) Trajectories of both robots starting at position 7.

(b) Trajectories of both robots starting at position 7. Only the intersection of both trajectories is shown here.

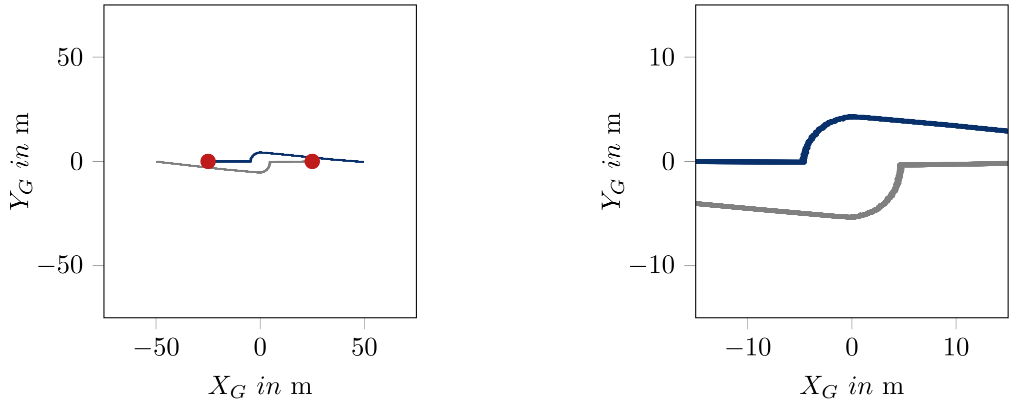
Fig. 5.34.: Exemplary results of a simulation of the circle scenario considering $\mathcal{FM}_{\{N,O\}}$, $D_{min} = 0.25\,\text{m}$, and assuming starting position 7. The legend is given in Fig. 5.29c.



(a) Trajectories of both robots starting at position 7.

(b) Trajectories of both robots starting at position 7. Only the intersection of both trajectories is shown here.

Fig. 5.35.: Exemplary results of a simulation of the circle scenario considering $\mathcal{FM}_{\{N,O\}}$, $D_{min} = 4.5\,\text{m}$, and assuming starting position 7. The legend is given in Fig. 5.29c.



(a) Trajectories of both robots starting at position 0.

(b) Trajectories of both robots starting at position 0. Only the intersection of both trajectories is shown here.

Fig. 5.36.: Exemplary results of a simulation of the circle scenario considering $\mathcal{FM}_{\{N,O,OO\}}$, $D_{min} = 9.5\,\text{m}$, and assuming starting position 0. The legend is given in Fig. 5.29c.

their right side, that is, using $\mathbf{x}_{0_2}$ as the stability point. This underlines that deciding on a turning direction in phase 1 and retaining the decision until the object is successfully bypassed prevents switching between both options, cf. Section 5.1.4.

Moreover, this example, as well as the previous examples, underlines two main results from this evaluation. Firstly, it can be shown that the safety of the robots is maintained during collision avoidance, which is in line with the guarantee provided by the *RoS* analysis. Secondly, this analysis certified only increased values of D_{min} as safe, which, as indicated by this evaluation, is conservative. Thus, the conservatism of *RoS*-based analysis should be addressed in future work.

5.3.4. Evaluation Using the Navigation Scenario

The circle scenario underlined that the concepts of *GFM* and *RoS* can be combined to implement a run-time safety assessment for a safety function that uses shared data. It provides a decision on whether the safety function will maintain its safety performance when using the shared data. However, these concepts aim only at a system's technical level and are therefore application- and implementation-specific. Contrarily, safety has to be shown for the entirety of a system, that is, considering its functional level as well. At that level, Chapter 2 showed that approaches to run-time safety assessment already exist. They require the technical level to provide safety information at run-time which can be integrated with the functional level.

To underline that the concept of *RoS* is applicable for such integration, this section aims at evaluating the same by means of an example. More specifically, the next subsection sketches out an approach to using the estimation of *RoS* to analyze the quality of shared data at run-time to select a suitable *Level of Service (LoS)* (initially proposed by the KARYON project)[40], in this case, a suitable value of D_{min}. The approach is evaluated in the following subsection using the navigation scenario. It will be shown that by adjusting the minimal distance the robots navigating in a shared operation area with static obstacles successfully maintain safety and do not collide with each other.

Conceptual Design

The KARYON project [40] addresses cooperative systems performing safety-critical functionalities. To tackle the problems of guaranteeing the safety of a system arising from uncertainties in shared data and increased complexities, a *Kernel-based ARchitecture for safetY-critical cONtrol (KARYON)* is presented. Based on the idea that a nominal system is designed with varying assumptions about its context situation, different performance levels, so-called *Level of Service (LoS)*, are provided. According to their assumptions, these are proven safe at the system's design-time.

The context assumptions may range, e.g. from weather conditions to the quality of shared data. To specifically address the latter, the *validity concept* [48] is used. It abstracts a predefined failure-type-based failure model to derive a scalar value informing an application about the confidence it can have in a sensor observation or shared data.

At run-time, available context information, such as the validity of current data, is analyzed to derive the most appropriate *LoS*. For that, a *safety manager*, as part of the *safety kernel*, compares the run-time safety information with design-time safety

Tab. 5.13.: Parameters for simulating the navigation scenario.

Parameter	Value	Description
N_{pos}	10	Number of starting positions.
N_{sim}	20	Number of simulations for each starting position.
τ	0.07 s	Control period of the controller.
T_{sim}	500.0 s	Maximal duration of each simulation.

information by means of predefined rules and selects an *LoS* accordingly. Thus, instead of restricting the system to worst-case assumptions and the corresponding minimal performance level, the risk posed by the current context is analyzed at run-time and the system performance is adjusted. As such, the approach assess the risk posed by the system's context at run-time, but relies on a safety assessment (prove of safety) executed at design-time. It thereby addresses Challenge 1.3.

This reliance enables to successfully maintain safety for systems sharing their data if they agreed on a common failure model before cooperation. However, as a consequence, changing failure characteristics are not supported. As detailed in Chapter 1, supporting the dynamic integration of shared data with varying failure characteristics necessitates assessing safety at run-time.

As shown by previous chapters and sections, this is provided by the concept of *GFM* and *RoS*. Assuming that the failure model of shared data is communicated, each available *LoS* can be analyzed by estimating its *RoS*. The result can be considered as run-time safety information and used by the safety kernel to choose the appropriate level. For instance, if the estimated *RoS* is not empty, the safety performance associated with the analyzed level will be met and safety will be maintained. Choosing the maximal level yielding a valid *RoS* allows adjusting performance while maintaining safety.

Regarding the presented collision avoidance controller, the values of D_{min} can be thought of as different *LoS* with $D_{min} = 9.5\,\text{m}$ being the lowest and $D_{min} = 2.0\,\text{m}$ being the highest level. Depending on the shared failure model, valid *RoS* are estimated only for levels corresponding to sufficiently high distance values. Moreover, the distance values stated in Table 5.9 already map the shared failure model to the appropriate level and are therefore used in this prototypical evaluation.

Evaluation Results of the Navigation Scenario

For evaluating the use-case-specific approach to integrating *RoS* and *LoS*, the navigation scenario described in Section 5.1.2 is used. By randomly generating $N_{pos} = 10$ starting positions for two robots and simulating the scenario $N_{sim} = 20$ times for each starting position, trajectories of 200 simulations are obtained. Each simulation covers 500.0 s during which a periodicity of $\tau = 0.07\,\text{s}$ is assumed for the controller.

To simulate the effect of changing failure characteristics, the schedule shown in Table 5.14 is assumed. During phases 1 and 5, no failure characteristics affecting the shared data are assumed. In phase 2 Noise and Offsets are assumed while phase 3

Tab. 5.14.: Schedule for shared failure models to evaluate the integration of *RoS* and *LoS*.

	Phase 1	Phase 2	Phase 3	Phase 4	Phase 5
Time	$[0\,\text{s}, 100\,\text{s}]$	$[100\,\text{s}, 200\,\text{s}]$	$[200\,\text{s}, 300\,\text{s}]$	$[300\,\text{s}, 400\,\text{s}]$	$[400\,\text{s}, 500\,\text{s}]$
Failure Model	$\mathcal{FM}_{\{\emptyset\}}$	$\mathcal{FM}_{\{N,O\}}$	$\mathcal{FM}_{\{N\}}$	$\mathcal{FM}_{\{N,O,OO\}}$	$\mathcal{FM}_{\{\emptyset\}}$

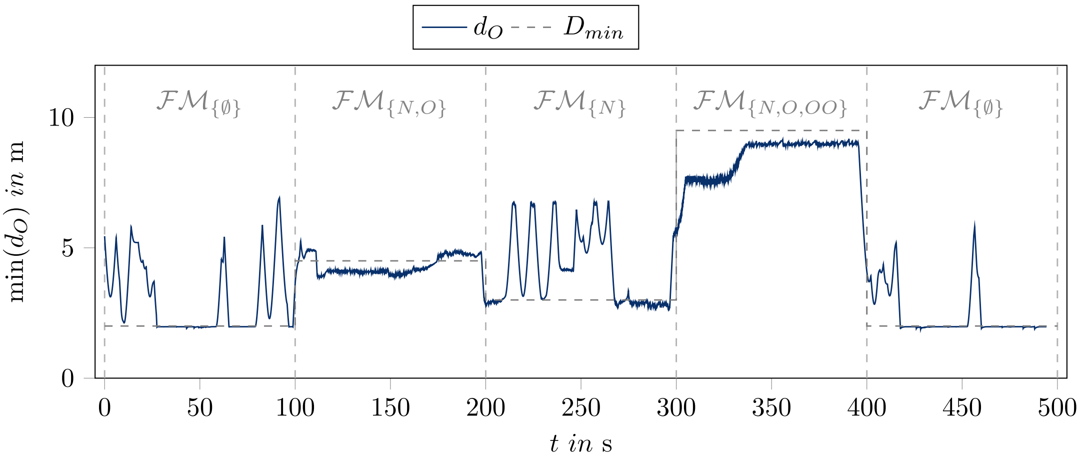


Fig. 5.37.: Minimal distances between robots and obstacles obtained by simulating the navigation scenario using starting position 0.

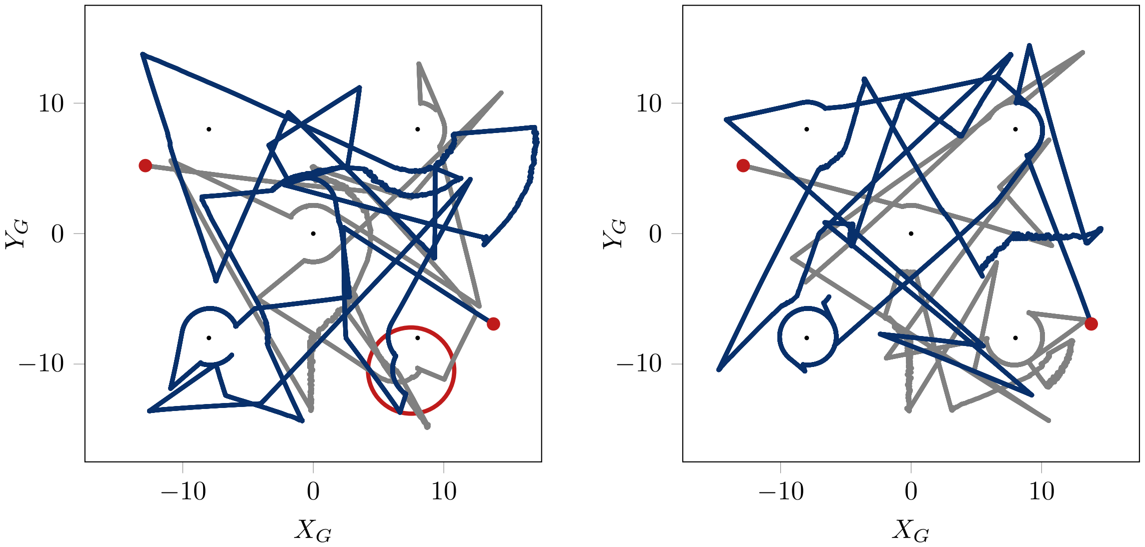
considers only Noise. Phase 4 presumes that shared data is affected by all failure types.

In correspondence to the shared failure model assumed, the controller of the robots are configured with the appropriate *LoS*, that is, the distances stated in Table 5.9. The results can be seen in Fig. 5.37, where the minimal distance d_O for both robots is stated. This means that for any of the robots, the minimal distance d_O is shown. The currently active *LoS* is shown by the gray, dashed curve.

As one can see, in phases 1 and 5 the targeted distance D_{min} is either met or exceeded by d_O. Contrarily, when considering non-empty failure characteristics, the targeted minimal distance is undercut. However, the value never falls below zero, which means that no collision occurs and safety is therefore maintained.

This is due to the successful bypassing of objects, which can be seen in the exemplary trajectories visualized in Fig. 5.38 as well. Fig. 5.38a shows the trajectories corresponding to the distance values stated in Fig. 5.37. The starting positions are marked by red circles while the objects avoided by the robots are colored black.

In both trajectories, one can see that these objects are bypassed in circular trajectories as envisioned by the second phase of the collision avoidance strategy, cf. Section 5.1.4. Moreover, circular trajectories can be observed in other areas as well. For instance in Fig. 5.38a at approx. $[x = 8\,\text{m}\ y = -10\,\text{m}]^T$. The blue trajectory indicates that the robot avoided a collision with the object and, subsequently, with the other robot. This can be seen by the gray trajectory as well, which switches from a straight line to a circular curve.



(a) First simulation of two robots of the navigation scenario starting at position 0. The legend is given in Fig. 5.29c. An exemplary situation in which both robots successfully avoided a collision with each other is highlighted by the red circle.

(b) Second simulation of two robots of the navigation scenario starting at position 0. The legend is given in Fig. 5.29c.

Fig. 5.38.: Exemplary trajectories of robots simulated in the navigation scenario.

Tab. 5.15.: Minimal distances observed over all simulations for each starting state.

Pos.	**0**	**1**	**2**	**3**	**4**
$\min(d_O)$	1.879 m	1.868 m	1.81 m	1.842 m	1.85 m
Pos.	**5**	**6**	**7**	**8**	**9**
$\min(d_O)$	1.853 m	1.846 m	1.84 m	1.827 m	1.829 m

Similar to these exemplary trajectories, the minimal distances observed over all simulations show that no collisions occur, cf. Table 5.15. While the targeted distance D_{min} is undercut in general due to the employed P-controller, sufficient safety margins are maintained due to the conservatism of the *RoS* approach. Therefore, it is shown that safety is indeed maintained in each *LoS*.

5.4. Summary

The goals of this chapter were to evaluate the concepts of *GFM* and *RoS* regarding their use in a run-time safety assessment method and to connect the approach to state-of-the-art approaches targeting the functional level.

In that endeavor, the first step was to examine this work's motivation to derive the use case of collision avoidance in multi-robot scenarios. It was assumed that two robots operating in the same area share their position data such that collisions can be prevented. Moreover, it was assumed that static obstacles are present in the operation area as well. Based on these assumptions, the experiments entitled the circle scenario and the navigation scenario were designed.

With these in mind, the employed control strategy implemented by both robots was described. As it comprised a collision avoidance strategy enabling to bypass obstacles and robots alike, it clarified that the control strategy in question features two stability points. An collision can be prevented by bypassing at an obstacle (or robots) left or right side. This, however, contradicted the assumption posed by the estimation of *RoS*, which considered only a single stability point. Thus, a use-case-specific fusion strategy was defined to obtain a single *RoS* from the estimates of the individual stability points.

In preparation of estimating the same, a failure model of the shared data was required. For that, it was assumed that robots are localized by means of marker detection. Consequently, in a real-world experiment data representing the failure characteristics of the AprilTag2 [116] framework was obtained. It was used as an input to the processing chain presented in Section 3.3 to automatically generate a preliminary failure model. While this model's performance was not sufficient for the envisioned use case, it required only minor manual adjustments. On the one hand, this showed the applicability as well as the limitations of the processing chain. On the other hand, the clarity of the *GFM* was underlined which enabled optimizing the failure model's weights manually. Finally, the quality of the adjusted failure model was assessed and represented by means of the defined confidence values, cf. Section 3.3.3. As these affirmed that the initial failure characteristics are adequately represented, the *GFMs* ability of representing real-world, multi-dimensional failure characteristics was underlined as well.

The individual failure types of the generated failure model were used to construct different failure characteristics for evaluating the *RoS* concept regarding its applicability as a run-time safety assessment method. Starting with these, the defined fusion strategy was used to estimate *RoS* for the employed collision avoidance strategy. Moreover, by adjusting the target distance D_{min} the controller aims at maintaining during collision avoidance, the minimal supported distance could be determined. Depending on the overall severity of the considered failure characteristics, this value ranged in $D_{min} \in [2.0\,\mathrm{m}, 9.5\,\mathrm{m}]$.

To evaluate that the control strategy indeed prevents collisions when configured with these values and despite the assumed failure characteristics, the circle scenario was considered. In this scenario, two robots are placed on a virtual circle and assigned target positions such that their trajectories intersect. Depending on their starting positions, varying incident angles were provoked. By simulating the scenario with varying failure characteristics it could be shown that no collisions occur when using values certified by the estimated *RoS*. On the other hand, it could be observed that collisions occur only for values of $D_{min} \leq 0.25\,\mathrm{m}$. This underlined the conservatism of the *RoS* approach.

Nevertheless, the circle scenario provided the basis on which the navigation scenario built on. Its goal is to exemplify the integration of the concepts with state-of-the-art approaches of run-time safety assessment targeting the functional level. For that, the already determined values of D_{min} were considered as *LoS*. At run-time, the shared failure model can be analyzed using the *RoS* estimation to switch between these. Simulations showed that this approach successfully maintains the safety of the systems.

Thereby, this chapter does not only show that the *GFM* and *RoS* can be combined to fulfill Objective 1.1 but also that they can be integrated into state-of-the-art approaches which supports their applicability and underlines their generality.

6. Conclusions and Future Work

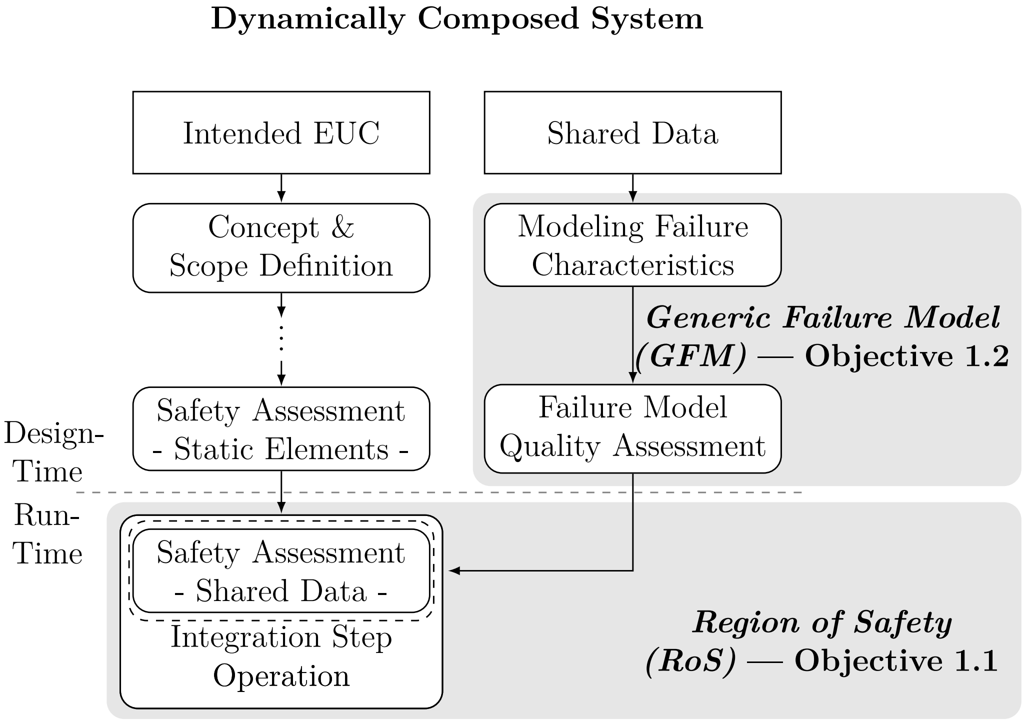

Dynamically Composed System
Intended EUC
Shared Data
Concept & Scope Definition
Modeling Failure Characteristics
Generic Failure Model (GFM) — Objective 1.2
Safety Assessment - Static Elements -
Failure Model Quality Assessment
Design-Time
Run-Time
Safety Assessment - Shared Data -
Integration Step Operation
Region of Safety (RoS) — Objective 1.1

Fig. 6.1.: Simplified safety process from Fig. 1.9 associated with the concepts and objectives of this thesis.

The previous chapter discussed the applicability of the presented concepts to dynamically composed systems, specifically with respect to collision avoidance strategies of autonomous mobile robots. It could be shown that Objective 1.1 and Objective 1.2 are fulfilled. To put this into perspective, this chapter firstly summarizes the contents of this work that lead to these fulfillments in Section 6.1 before Section 6.2 states limitations and derives possibilities for future work.
To provide an overview, the main concepts introduced by this thesis are associated with the objectives that they fulfill and the initially proposed safety process in Fig. 6.1.

6.1. Summary

The flexibility provided by the emerging paradigms of *Internet of Things (IoT)* and *CPS* combined with intelligent algorithms formed the basis of a profound industrial transformation termed *Industry 4.0*. Starting with this perspective, Chapter 1 tries to extend the applicability of these paradigms beyond individual industries. Leveraging the example of *Smart Warehouses*, a use case is defined in which autonomous delivery systems may dynamically integrate with the infrastructure on site to navigate safely and, for instance, automate replenishing of storage systems. Arising from the entailed flexibility, dynamically composed systems are defined as systems that dynamically adapt their composition to available resources at run-time. As this entails that data shared at run-time is used for potentially safety-critical applications within these systems, safety is identified as one central challenge that has yet to be solved. In that endeavor, the functional safety standard IEC 61508 [20] is considered an advocate for prevailing safety standards and examined with respect to its application to dynamically composed systems.

Caused by its general assumptions, which requires components and their characteristics to be available already at design-time and which contradicts the idea of dynamically composed systems, the analysis identified four central challenges, cf. challenges 1.1 to 1.4. Firstly, due to its openness, the *Operational Design Domain (ODD)* of a dynamically composed system can not be fully specified as neither the number of systems sharing their data nor their quality is available at design-time. Secondly, the unavailable system components can not be considered during a design-time hazard analysis, which brings its coverage into question. Thirdly, missing knowledge about the failure characteristics of data shared between systems prevents assessing the risk at design-time. Fourthly, the safety performance of a safety function using shared data can not be assessed due to the missing failure characteristics as well.

Arguing that the first three challenges are (partially) addressed in the literature, the fourth challenge is identified as unsolved and targeted by the presented work. In that endeavor, it is argued that parts of the safety assessment have to be shifted into a system's run-time as it is only then that all required resources are available. More specifically, two objectives for implementing a successful run-time safety assessment are formulated. A failure model representing the failure characteristics is needed (Objective 1.2) which can be analyzed by a run-time safety assessment (Objective 1.1). For both objectives, corresponding criteria for discussing their fulfillment are derived.

The objectives and the predefined criteria form the basis on which state-of-the-art approaches are reviewed in Chapter 2. At first, approaches to safety assessment were reviewed. For that, they were categorized according to their targeted abstraction levels. Derived from the prevailing safety standards [8], [13], the functional and technical abstraction level were considered. While the functional abstraction level comprises implementation-independent concepts, approaches of the technical level integrate use case and application-dependent solutions. Correspondingly, at a functional level, contract-based approaches are prevailing along with the approach of *Level of Service (LoS)* and the idea of a safety kernel as presented by the KARYON project [47]. These address not only safety assessment but specifically ask for its execution at run-time. Contrarily, at a technical level, such approaches can not be found. Focusing on stability, only the estimation of *Region of Attraction (RoA)*, which was shown to provide stability guarantees at a system's run-time, was found to facilitate similar functionalities. Nevertheless, none of these approaches could fulfill all required predefined criteria.

Similarly, with respect to Objective 1.2, techniques for (sensor) failure modeling were reviewed and assessed according to the predefined criteria. While interval-based failure models provide clear interpretation and unambiguous representation, their modeling capabilities are limited. Similarly, distribution-based failure models were shown to be insufficient in that regard as well. Only failure-type-based failure models, prevailing for describing sensor failure characteristics, are considered sufficient to fulfill the coverage criterion. On the other hand, these failure models are commonly defined linguistically and are thereby ambiguous. Thus, no approach fulfilling all required predefined criteria could be found either.

Consequently, Chapter 3 started with defining the *Generic Failure Model (GFM)* applicable to dynamically composed systems. It thereby addressed Objective 1.2. For that, a mathematically defined failure model is presented. It builds upon the idea of failure-type-based failure models where each failure type is represented by a temporal failure pattern, a stochastic scaling defining the magnitude of possible failure amplitudes, an activation distribution, and a deactivation distribution. All of the three distributions (scaling, activation, deactivation) are modeled by a time- and value-correlated random distribution which applies deterministic shift- and scale functions to represent correlations and uses a quantile function to represent the stochastic distribution of values. Each of these functions is finally represented by a polynomial. While this mathematical representation fulfills the central criterion of clarity, the use of multiple failure types to represent failure characteristics enables the fulfillment of the coverage criterion. Similarly, it was shown that the remaining predefined criteria are fulfilled as well.

To automate and simplify the generation of a *GFM*, a processing chain was proposed. It requires time series of failure amplitudes as well as reference data to extract and parameterize a *GFM* representing the failure characteristics. For that, three stages are executed. During the first stage, a *Continuous Wavelet Transformation (CWT)* is employed to identify the occurrences of randomly generated failure patterns, which are optimized in an iterative approach using gradient descent. During the second stage, a sliding window approach is applied to extract training data for parameterizing the individual failure types' functions, that is, fit the corresponding polynomials. During the final stage, the generated *GFM* is converted into an interval-based representation which informs about the minimal and maximal failure amplitudes. By comparing these to the initial data provided to the processing chain, confidence values stating the (over-)approximation of the true failure characteristics are calculated. These are considered as metadata and should be communicated along with the failure model to inform a receiving application about the quality of the representation.

Both, the concept of the *GFM* as well as the presented processing chain were evaluated using artificial, one-dimensional data. Next to the fulfillment of the predefined criteria, it could be shown that failure characteristics repeatedly reported in the literature can be modeled unambiguously using the *GFM*. Moreover, due to the clarity of the failure model and the use of polynomials, these can be generated manually as well as by the presented processing chain. Comparing the performance of the *GFM* to different state-of-the-art approaches from the field of time series modeling furthermore revealed its strength in representing stochastic time series. As such, the *GFM* fulfills Objective 1.2 and thereby provides a central prerequisite in the endeavor of guaranteeing safety when using shared data in safety-critical control systems.

This is addressed by Objective 1.1. In the endeavor of fulfilling it, Chapter 4 builds upon the idea of *Region of Attraction (RoA)*, which was identified as a promising approach in Chapter 2. However, to be applicable to dynamically composed systems, an extended system model enabling to specify different sources of uncertainty was required. Covering not only failure characteristics of shared data but also failures of internal sensor observations, model uncertainties, actuator failures, and environmental disturbances facilitates a fine-grained definition of uncertainties while using a uniform analysis technique.

For this, however, the estimation of *RoA* could not be used due to its insufficiency in handling uncertainties. It was shown that even minor uncertainties prevented *RoA* from being estimated.

As a consequence, Theorem 4.1 introduced the concept of *Region of Safety (RoS)*. Opposed to *RoA*, the concept focuses on safety, that is, guaranteeing that a system will not evolve to unsafe states with a given control strategy. By additionally defining Algorithm 1, *RoS* of systems could be estimated.

This was used in the preliminary evaluation of the concept, for which the example of an inverted pendulum was examined. After discussing the challenges of defining appropriate *Control Lyapunov Function (CLF)* candidates and setting the λ_c parameter of Algorithm 1, simulations showed that, if a valid *RoS* can be estimated, the system will maintain its safety and will not leave the specified *RoS*. Together with [45], where it could be shown that the approach of *RoA* can be applied at run-time, the *RoS* was shown to successfully enable a run-time safety assessment.

Nevertheless, the inverted pendulum did not consider shared data and was therefore not a dynamically composed system. Therefore, Chapter 5 aimed at providing a holistic evaluation for both concepts (*GFM* and *RoS*) as well as its integration with state-of-the-art run-time safety assessment approaches targeting the functional level. With these goals in mind, the initial discussion on smart warehousing was examined to derive the use case of collision avoidance in multi-robot scenarios. Assuming that two robots share an operating area in which they obtain their position data using a marker detection framework, the safety function of collision avoidance had to be guaranteed to meet its safety performance despite using these shared position data. On the one hand, this entailed modeling of failure characteristics of a marker detection framework, for which real-world data was used. It could be shown that the concept of *GFM* provides a suitable model even for three-dimensional data (x,y,Θ). On the other hand, bypassing an object during collision avoidance can be achieved by driving at its right or left side. This dictates the existence of two stability points, which contradicts the assumption of *CLF* and *RoS* that allow only a unique stability point to be examined at a time. Therefore, a fusion strategy for combining two separate estimates of *RoS* was defined.

With the fusion strategy in place, the *RoS* of the collision avoidance strategy could be estimated. To examine the effects of varying failure characteristics, the individual failure types of the failure model generated for the marker detection framework were grouped to form four different failure models. Only by adjusting the target value of the controller, valid *RoS* could be estimated for each of them. This means that the concept successfully distinguishes between tolerable and intolerable failure characteristics.

These results were further examined in the circle scenario, which assumes that two robots are placed at a virtual circle and configured with target positions in such a way that their trajectories intersect at specific incident angles. Simulating this scenario for varying target distances D_{min} to keep by the collision avoidance controller showed that safety was successfully maintained when configured with appropriate values. However, the simulations also indicated that the distance D_{min} could be further reduced, which was not supported by the results of *RoS*.

Nevertheless, even the estimation of *RoS* resulted in varying values of D_{min} depending on the specified failure characteristics. As such, *RoS* can not only be used to determine whether or not a safety function will provide its safety performance when using the shared data. In combination with the idea of *LoS* presented by the KARYON project [40], [47], the performance can even be adjusted.

This was shown exemplarily by the navigation scenario. It simulated two robots in a shared operation area which additionally featured static obstacles to be bypassed. Within the area, the robots had to move to randomly assigned target positions while avoiding collisions with obstacles or the other robot. Moreover, the quality of the shared data was assumed to change according to a predefined schedule. Depending on the changed failure characteristics, the minimal value of D_{min} was determined using the concept of *RoS*. Thus, over time, it could be shown that the distance had to be increased for failure characteristics featuring failure amplitudes of high magnitude while the distance could be reduced when assuming failure characteristics with failure amplitudes of lower magnitude.

With this evaluation, it was not only shown that the concepts of *GFM* and *RoS* can successfully be combined to realize a run-time safety assessment but also that they can be integrated into state-of-the-art approaches targeting the functional abstraction level. This is especially important as safety can not be guaranteed at a single level of abstraction, but has to be proven at all abstraction levels.

6.2. Limitations and Future Work

Evaluating the concepts of *GFM* and *RoS* using the robotic scenario addressed the applicability of the theoretical work presented in Chapter 3 and Chapter 4 to real-world applications. However, it also brought focus on the limitations of the current approaches. These are summarized and used to derive future work in this section.

According to the structure of this work, the next subsection discusses limitations regarding the *GFM* while shortcomings of the concept of *RoS* are discussed in the following subsection. Finally, general directions for future work on safety in dynamically composed systems are discussed.

6.2.1. Limitations and Future Work on the Generic Failure Model

The representation of failure characteristics of a marker detection framework as discussed in Section 5.2 showed that the *GFM* is able to model multi-dimensional data but simultaneously underlined existing shortcomings. Starting with limitations regarding the *GFM*, the following paragraphs also cover shortcomings of the presented processing chain and derive future work.

Representing Quantile Functions For using quantile functions to represent one and multi-dimensional distributions, the *GFM* applies the approach of standard construction. This approach is independent of the actual data to be represented but presumes an ordering of the modeled dimensions, cf. Eq. (3.8). When fitting polynomials to model the quantile function, this structure has to be represented as well. This causes inaccuracies when applied to high-dimensional data. The effect could be observed already for low-dimensional data of a marker detection framework in Chapter 5 where manual adjustments were necessary to increase the modeling performance.

Therefore, in future work, adequate experiments should firstly investigate the hypothesized origin of these inaccuracies, that is, the quantile function, and secondly research possible alternatives.

From that, criteria to be fulfilled by appropriate function approximation schemes can be derived such that increased fitting performance of quantile functions may be obtained. These criteria could encompass mathematical requirements. For instance, in the one-dimensional case, the quantile function is an increasing function. Contrarily, the current usage of polynomials does not respect this requirement. Thus, the fitted polynomials may not represent appropriate quantile functions.

Calculation of Lipschitz Constants Another limitation stems from the use of Lipschitz constants and the calculation of the same. In this work, they are calculated empirically, that is, by sampling the function in question. However, this does not guarantee that the resulting value is indeed the function's Lipschitz constant. While the evaluation results show that in both use cases, the *GFM* and the concept of *RoS*, this approximation is sufficient, future work may research alternatives, for instance as presented in [87]. Building upon the empirically determined gradients of the considered function, the authors propose to fit a Reverse Weibull distribution and use its location parameter as an estimate of the Lipschitz constant.

Fitting of Failure Patterns Next to limitations regarding the *GFM* itself, shortcomings originate in the processing chain for generating the same as well. For instance, failure patterns are learned using gradient descent in the first stage, cf. Section 3.3.1. While this enables using alternative function approximation schemes, for instance, *ANN*, it is not optimal in combination with polynomials. When using these, the training data generated through the identification of occurrences can be used to calculate the optimal weights directly, e.g., using Housholder QR decomposition. Applying such a method could increase the performance of the identification stage.

Parameterizing Failure Types The second stage of the processing chain (cf. Section 3.3.2) employs a sliding window approach to extract training data for representing time- and value-correlated random distributions of failure types. When configured correctly, the resulting polynomials represent these adequately. However, the process is sensitive to variations of these configuration parameters. The applied window size, for instance, has to be tuned such that the value- and time domains are covered sufficiently as correlations are not identified correctly otherwise.

The reason is that the number of occurrences accounted to a single window may vary, which causes unstable results. This effect should be examined in more detail. An alternative could be to introduce a dynamic window size which is adjusted according to the number of occurrences found for a failure type in general and for specific parts of the value- and time domains.

Integration with State-of-the-Art Approaches The overall idea of the *GFM* is to describe failure characteristics such that they can be shared between interacting systems. However, as concluded from the review on different run-time safety assessment techniques in Chapter 2, additional information will be required to complement the process at different abstraction levels. Schneider *et al.* [46] proposed *DDIs* to share general dependability information between systems, for instance. As this idea is similar to sharing *GFMs*, future work can be directed into the question on how to share *GFMs* as part of a *DDI*. Next to sharing a *GFM* between systems, this would enable integration with more state-of-the-art approaches.

6.2.2. Limitations and Future Work on Region of Safety

The limitations encountered for the concept of *RoS* address its conservatism (*CLF*, number of stability points, and worst-case assumption) as well as its applicability to real-world scenarios (real-time capability, integration with state-of-the-art approaches). They are discussed in the following paragraphs.

Control Lyapunov Functions The challenge of designing a suitable *CLF* is known from literature already [44]. Originated in the approach of *RoA*, this limitation applies to *RoS* as well. As no structured way exists, multiple *CLF* candidates have to be evaluated and compared to identify suitable options. This approach was taken for determining the *CLF* in Chapter 5 and was described in detail during the evaluation using the inverted pendulum as well.

In Section 4.3.2, it was derived that a *CLF* should increase symmetrically around its minimum to ensure that the calculated set of stabilizing states fully encompasses the same. Contrarily, when evaluating the concept using the collision avoidance controller, the asymmetry of the considered failure characteristics $\mathcal{FM}_{\{N,O,OO\}}$ caused the *RoA* condition to be unfulfilled for lower values of D_{min}, cf. Section 5.3.3. Thus, safety can be guaranteed using the estimation of *RoS* only for high values of $D_{min} \geq 9.5\,\mathrm{m}$.

In future work, this conservatism should be investigated and minimized to increase the applicability of the approach. For that, one direction to follow is to adjust Theorem 4.1 such that the symmetry of a *CLF* and asymmetry of considered failure characteristics do not contradict each other. Specifically, an *RoS* may not necessarily be a level set of a *CLF* function, but it should be a set of states guaranteed to not be left under a given control strategy. This means that the set of stabilizing states needs to be connected and encompass the stability point in question, but does not require to extend to a maximum value of c_{max} at all costs. In that regard, future work may aim at relaxing or removing Theorem 4.1's reliance on a *CLF*'s level set and at introducing a notion of connectivity in the set of stabilizing sets instead. This would also address the limitations associated with choosing an appropriate value for λ_c of Algorithm 1.

Number of Stability Points The idea of adjusting the theorems reliance on a *CLF* can be extended to solve another limitation encountered during Chapter 5. There, to overcome the fact that a single *RoS* can be estimated for a single stability point, a fusion strategy was proposed to handle control strategies for multiple stability points.

On the one hand, future work can focus on generalizing the presented fusion approach to achieve applicability to other use cases as well. On the other hand, the assumption about a single stability point stems from the estimation of *RoA* and its reliance on a *CLF*, which is focused on stability instead of safety. Thus, to shift focus further towards safety, one should investigate dropping the requirement of having a single stability point and should focus on defining a region of states that will not be left instead. For example, investigating the use of general cost functions with more than one global minimum could provide such functionality.

The results discussed in Section 5.3.2 underlined this hypothesis as *RoS* associated to one stability point were estimated to cover the other stability point as well. On the other hand, varying failure characteristics generated contradicting results, which leaves this as an open question for future work.

Worst-Case Assumption in Region of Safety Another reason for the conservatism of the estimated *RoS* is its worst-case assumption. For each state, an interval of minimal and maximal failure amplitudes is considered. Based on that, the estimated *RoS* provides a guarantee on how the system will behave. On the other hand, this means that even unlikely state trajectories are considered. In other words, no valid *RoS* will be estimated if a single trajectory of states exists that violates the stated requirements.

As discussed in Chapter 1, however, a system is considered safe if the risk it poses is tolerable. This means, a single, yet sufficiently unlikely state trajectory leading to unsafe states could be tolerated. By taking a probabilistic perspective, this aspect could be integrated into the concept of *RoS* and the conservatism could be further reduced. Eq. (4.9) could serve as a starting point where one could examine its relation to, for instance, *Safety Integrity Level (SIL)* and *Probability of Fails on Demand (PFD)* of the IEC 61508 standard (cf. Section 1.2.2) with respect to such a probabilistic perspective.

Moreover, with this idea in mind, reachability analysis techniques could be investigated in future work for estimating *RoS* as well.

Real-Time Capabilities The evaluations of the *RoS* presented in this work did not consider the aspect of execution time. Building upon [45], it is assumed that the approach can be executed at run-time, but real-time properties are not analyzed yet. Contrarily, for real-world applications, these properties have to be provided. Therefore, future work has to show that the reliance of *RoS* on *RoA* indeed provides the ability to adhere to real-time constraints. For that, implementations tuned towards use-case-specific scenarios are required.

Integration with State-of-the-art Approaches Similarly to the investigation of real-time capability, the applicability of the *RoS* to real-world scenarios can be supported by integrating the concept with other state-of-the-art approaches. In line with the integration of *LoS*, one can think about integrating *RoS* with the approach of *ConSert*. For instance, *RoS* can be considered a process for generating run-time evidence, which can be used to check whether specified demands and/or assumptions are fulfilled.

6.2.3. Future Work on Safety in Dynamically Composed Systems

Despite their applicability in other fields, the concepts of *RoS* and *GFM* were designed with respect to the challenge of a run-time safety assessment, that is, Challenge 1.4. However, other challenges were identified in Chapter 1 as well, which need to be addressed in future work.

Starting with the *ODD* coverage, where the question arises on how to describe the context situations in which a system will be if its composition is available only at run-time. Similar to the *GFM*, this asks for an explicit representation of *context*, that is, a way to describe situations in which a component may be employed and how these situations influence a component's nominal and failure behavior. Based on such a representation, a reasoning mechanism could take a set of components into account and derive which contextual situations have to be considered during a safety analysis. Moreover, it could be checked whether the set of available sensors or general data sources are sufficient to detect all relevant context situations.

On the other hand, such an explicit context description could enable addressing Challenge 1.2, which asks for whether or not all relevant hazards are identified during a *HARA*. Having an explicit description of each component of a system and its relevant context parameters may enable an automated analysis on which failure behaviors to expect. In the best-case scenario, it would be possible to derive new hazards for a dynamic system composition at run-time.

Consequently, a dynamic risk assessment for those hazards would be needed. On the one hand, the concept of *LoS* and the idea of a safety kernel [40], [47] provide already means to assess risks at run-time. On the other hand, this applies only to context situations identified already at design-time.

A key enabler to truly dynamically composed systems for which safety can be guaranteed at design-time and maintained at run-time are explicit descriptions of individual components with unambiguous interpretations. As shown by this work, only an unambiguous interpretation allows analyzing contents and descriptions at run-time while providing guarantees about the functional correctness of the employed methods at design-time.

Appendices

A. Defining Factors of Risk According to IEC 61508

Tab. A.1.: Components of risk according to IEC 61508 required for *SIL* determination [8]. This table complements the discussion of Section 1.2.2.

Risk Parameter		Classification	Description
Consequence	$\mathbf{C_1}$	Minor injury	The classification of consequences is oriented towards injuries and death of people and serves as an example. In correspondence with Definition 1.10, other classifications may be developed.
	$\mathbf{C_2}$	Serious injuries, death to one person.	
	$\mathbf{C_3}$	Death to several persons.	
	$\mathbf{C_4}$	Very many people killed.	
Frequency	$\mathbf{F_1}$	Rare to more often exposure.	Frequency of, and exposure time in, the hazardous zone.
	$\mathbf{F_2}$	Frequent to permanent exposure	
Possibility of avoiding the hazardous event.	$\mathbf{P_1}$	Possible under certain conditions.	This parameter takes possibilities into account that exist without implementing a safety function.
	$\mathbf{P_2}$	Almost impossible.	
Probability of the unwanted occurrence.	$\mathbf{W_1}$	Very slight probability, few unwanted occurrences are likely.	Estimates the frequency of the unwanted occurrence without implementing a safety function but considering external risk reduction facilities.
	$\mathbf{W_2}$	Slight probability, few unwanted occurrences are likely.	
	$\mathbf{W_3}$	Relatively high probability, frequent unwanted occurrences are likely.	

B. Evaluation Results for the Identification Stage

Tab. B.1.: Comparing the occurrences of identified failure types to injected occurrences of original failure type. This table complements the description and results discussed in Section 3.4.2.

$\mathcal{FM}$	$r_{\mathcal{O}}$	$\mu_{Overlap}$	μ_{scl_e}
$\mathcal{FM}_{\{N\}}$	[1.00]	[1.00]	[0.00]
$\mathcal{FM}_{\{O\}}$	[0.98]	[0.99]	[0.01]
$\mathcal{FM}_{\{S\}}$	[0.97]	[0.80]	[0.08]
$\mathcal{FM}_{\{A\}}$	[1.00]	[0.49]	[0.04]
$\mathcal{FM}_{\{N,O\}}$	[1.00, 0.88]	[1.00, 0.98]	[2.58, 0.02]
$\mathcal{FM}_{\{N,S\}}$	[1.00, 0.47]	[1.00, 0.71]	[8.40, 0.12]
$\mathcal{FM}_{\{N,A\}}$	[1.00, 1.00]	[1.00, 0.48]	[47.44, 0.02]
$\mathcal{FM}_{\{S,O\}}$	[0.63, 0.96]	[0.50, 0.95]	[2.88, 0.14]
$\mathcal{FM}_{\{O,A\}}$	[0.99, 1.00]	[0.96, 0.39]	[0.42, 0.95]
$\mathcal{FM}_{\{S,A\}}$	[0.82, 1.00]	[0.75, 0.43]	[0.51, 1.36]
$\mathcal{FM}_{\{N,S,O\}}$	[1.00, 0.13, 0.81]	[1.00, 0.40, 0.88]	[21.00, 0.52, 0.84]
$\mathcal{FM}_{\{N,O,A\}}$	[1.00, 0.78, 0.72]	[1.00, 0.93, 0.39]	[54.90, 0.15, 0.97]
$\mathcal{FM}_{\{N,S,A\}}$	[1.00, 0.50, 0.88]	[1.00, 0.73, 0.51]	[40.43, 0.30, 0.74]
$\mathcal{FM}_{\{S,O,A\}}$	[0.93, 0.86, 1.00]	[0.72, 0.35, 0.41]	[1.15, 1.05, 1.18]
$\mathcal{FM}_{\{N,S,O,A\}}$	[1.00, 0.15, 0.84, 0.76]	[1.00, 0.48, 0.85, 0.46]	[38.08, 0.50, 1.98, 0.39]

C. Overview of Failure Amplitudes of Marker Detection Results

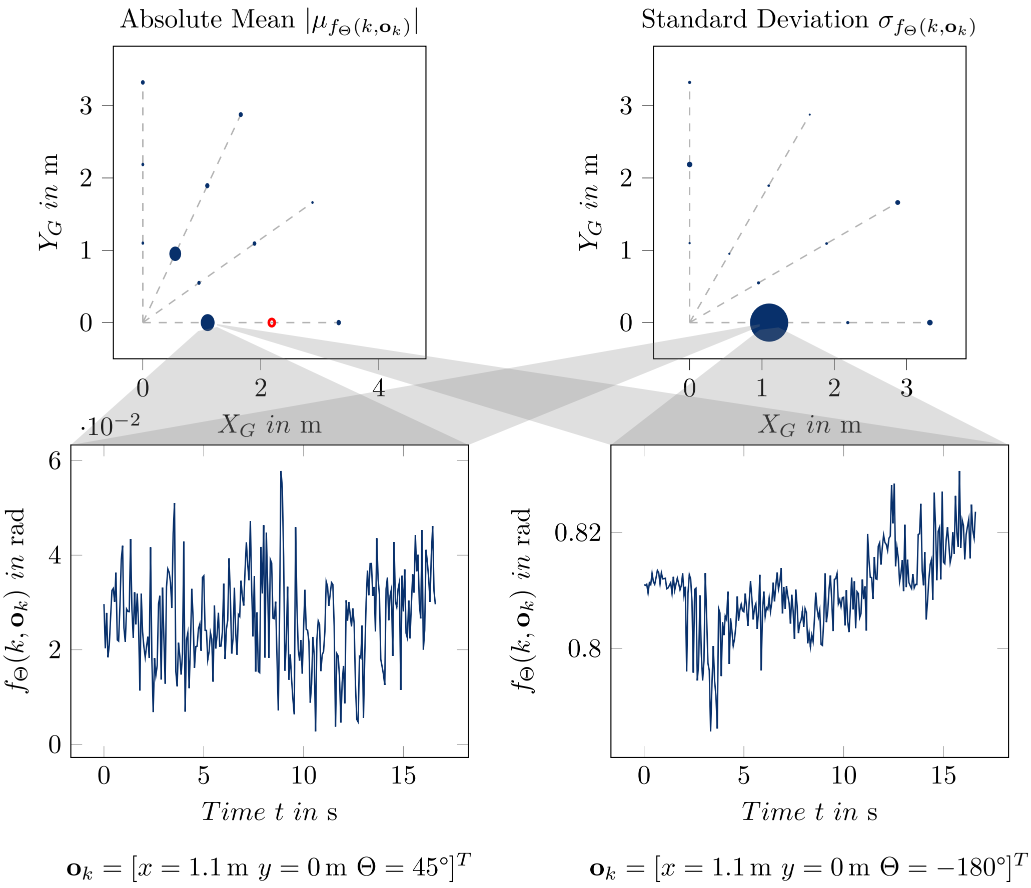


Fig. C.1.: Visualizing the failure amplitudes affecting the Θ component of the pose observations provided by the marker detection framework. Similar to Fig. 5.9, the upper plots illustrate the magnitude of the mean and standard deviations calculated from the failure amplitudes of the Θ component observed for the associated positions. However, as only Θ is considered in this case, that is, only one dimension, the radius of the circles of the upper plots indicates the magnitude. For illustrating the circles a ratio of $1\,\mathrm{m} : 2\,\mathrm{rad}$ is assumed such that the size of the circle can be displayed with respect to the x and y components. The lower diagrams show exemplary time series of failure amplitudes observed at the indicated positions.

Bibliography

[1] E. Chindenga, M. S. Scott, and C. Gurajena, “Semantics based service orchestration in iot,” in *Proceedings of the South African Institute of Computer Scientists and Information Technologists*, ser. SAICSIT ’17, Thaba ’Nchu, South Africa: Association for Computing Machinery, 2017. DOI: 10.1145/3129416.3129438.

[2] M. Milošević, M. Đurđev, D. Lukić, A. Antić, and N. Ungureanu, “Intelligent process planning for smart factory and smart manufacturing,” in *Proceedings of 5th International Conference on the Industry 4.0 Model for Advanced Manufacturing*, L. Wang, V. D. Majstorovic, D. Mourtzis, E. Carpanzano, G. Moroni, and L. M. Galantucci, Eds., Cham: Springer International Publishing, 2020, pp. 205–214.

[3] S. Aheleroff, X. Xu, R. Y. Zhong, and Y. Lu, “Digital twin as a service (dtaas) in industry 4.0: An architecture reference model,” *Advanced Engineering Informatics*, vol. 47, p. 101 225, 2021. DOI: 10.1016/j.aei.2020.101225.

[4] A. Atta, S. Abbas, M. A. Khan, G. Ahmed, and U. Farooq, “An adaptive approach: Smart traffic congestion control system,” *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 9, pp. 1012–1019, 2020. DOI: 10.1016/j.jksuci.2018.10.011.

[5] R. Mason, “Developing a profitable online grocery logistics business: Exploring innovations in ordering, fulfilment, and distribution at ocado,” in *Contemporary Operations and Logistics: Achieving Excellence in Turbulent Times*, P. Wells, Ed. Cham: Springer International Publishing, 2019, pp. 365–383. DOI: 10.1007/978-3-030-14493-7_19.

[6] A. Bolu and Ö. Korçak, “Adaptive task planning for multi-robot smart warehouse,” *IEEE Access*, vol. 9, pp. 27 346–27 358, 2021. DOI: 10.1109/ACCESS.2021.3058190.

[7] G. Jäger, S. Zug, and A. Casimiro, “Generic sensor failure modeling for cooperative systems,” *Sensors*, vol. 18, no. 3, 2018. DOI: 10.3390/s18030925.

[8] I. E. Commission, “Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES),” International Electrotechnical Commission (IEC), Geneva, CH, Standard, 2018.

[9] R. Bell, “Introduction to iec 61508,” in *ACM International Conference Proceeding Series*, vol. 162, 2006, pp. 3–12.

[10] M. Chaari, “Formalization and model-driven support of functional safety analysis,” Ph.D. dissertation, Technische Universität München, 2020.

[11] T. Meany, “Functional safety and industrie 4.0,” in *2017 28th Irish Signals and Systems Conference (ISSC)*, 2017, pp. 1–7. DOI: 10.1109/ISSC.2017.7983633.

[12] I. E. Commission, “Programmable Controllers - Functional Safety,” International Electrotechnical Commission, Geneva, CH, Standard, 2012.

[13] I. O. for Standardization, “Road vehicles - Functional safety,” International Organization for Standardization, Geneva, CH, Standard, 2018.

[14] E. C. for Standardization (CEN), “Railway applications - communication, signalling and processing systems - software for railway control and protection systems,” European Committee for Standardization (CEN), Standard, 2012.

[15] R. T. C. for Aeronautics, “Software consideration in airborne systems & equipment certification,” Radio Technical Commission for Aeronautics, Standard, 2012.

[16] I. E. Commission, “Functional safety - Safety instrumented systems for the process industry sector,” International Electrotechnical Commission, Geneva, CH, Standard, 2020.

[17] I. O. for Standardization, “Safety of machinery - Safety-related parts of control systems,” International Organization for Standardization, Geneva, CH, Standard, 2015.

[18] I. E. Commission, “Safety of machinery - Functional safety of safety-related control systems,” International Electrotechnical Commission, Geneva, CH, Standard, 2021.

[19] I. O. for Standardization, “Road vehicles - Safety of the intended functionality,” International Organization for Standardization, Geneva, CH, Standard, 2019.

[20] I. Häring, “The standard iec 61508 and its safety life cycle,” in *Technical Safety, Reliability and Resilience: Methods and Processes*. Singapore: Springer Singapore, 2021, pp. 193–207. DOI: 10.1007/978-981-33-4272-9_11.

[21] S. Mathur and S. Malik, “Advancements in the v-model,” *International Journal of Computer Applications*, vol. 1, no. 12, pp. 29–34, 2010.

[22] E. Ruijters and M. Stoelinga, “Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools,” *Computer Science Review*, vol. 15-16, pp. 29–62, 2015. DOI: 10.1016/j.cosrev.2015.03.001.

[23] A. Birolini, *Reliability & Availability of Repairable Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 169–310. DOI: 10.1007/978-3-662-54209-5_6.

[24] P. Fuchs and J. Zajíček, “Safety integrity level (sil) versus full quantitative risk value,” 2013.

[25] M. Shamsuzzoha and S. Skogestad, “The setpoint overshoot method: A simple and fast closed-loop approach for pid tuning,” *Journal of Process Control*, vol. 20, no. 10, pp. 1220–1234, 2010. DOI: 10.1016/j.jprocont.2010.08.003.

[26] R. Salay, R. Queiroz, and K. Czarnecki, “An analysis of iso 26262: Using machine learning safely in automotive software,” *arXiv preprint arXiv:1709.02435*, 2017.

[27] G. Jäger, S. Zug, T. Brade, A. Dietrich, C. Steup, C. Moewes, and A. Cretu, "Assessing neural networks for sensor fault detection," in *2014 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, 2014, pp. 70–75. DOI: `10.1109/CIVEMSA.2014.6841441`.

[28] T. Brade, G. Jäger, S. Zug, and J. Kaiser, "Sensor- and environment dependent performance adaptation for maintaining safety requirements," in *Computer Safety, Reliability, and Security*, A. Bondavalli, A. Ceccarelli, and F. Ortmeier, Eds., Cham: Springer International Publishing, 2014, pp. 46–54. DOI: `10.1007/978-3-319-10557-4_7`.

[29] G. Jaeger, T. Brade, and S. Zug, "Using failure semantics to maintain safety for dynamic composed systems," in *ARCS 2016; 29th International Conference on Architecture of Computing Systems*, 2016, pp. 1–7.

[30] J. Höbel, G. Jäger, S. Zug, and A. Wendemuth, "Towards a sensor failure-dependent performance adaptation using the validity concept," in *Computer Safety, Reliability, and Security*, S. Tonetta, E. Schoitsch, and F. Bitsch, Eds., Cham: Springer International Publishing, 2017, pp. 270–286.

[31] G. Jäger, K. Kirchheim, F. Schrödel, and S. Zug, "Multi-dimensional failure modeling for shared data in cooperative systems," *IFAC-PapersOnLine*, 2020, IFAC World Congress 2020.

[32] G. Jäger, J. Schleiss, S. Usanavasin, S. Stober, and S. Zug, "Analyzing regions of safety for handling shared data in cooperative systems," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 628–635. DOI: `10.1109/ETFA46521.2020.9211932`.

[33] G. Jäger, C. A. Mueller, M. Thosar, S. Zug, and A. Birk, "Towards robot-centric conceptual knowledge acquisition," *Robots that Learn and Reason Workshop in IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[34] M. Thosar, C. A. Mueller, G. Jäger, J. Schleiss, N. Pulugu, R. Mallikarjun Chennaboina, S. V. Rao Jeevangekar, A. Birk, M. Pfingsthorn, and S. Zug, "From multi-modal property dataset to robot-centric conceptual knowledge about household objects," *Frontiers in Robotics and AI*, vol. 8, p. 87, 2021. DOI: `10.3389/frobt.2021.476084`.

[35] M. Thosar, C. A. Mueller, G. Jaeger, M. Pfingsthorn, M. Beetz, S. Zug, and T. Mossakowski, "Substitute selection for a missing tool using robot-centric conceptual knowledge of objects," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 972–979.

[36] O. Jaradat, I. Sljivo, R. Hawkins, and I. Habli, "Modular safety cases for the assurance of industry 4.0," in *28th Safety-Critical Systems Symposium*, Feb. 2020.

[37] J. Reich, D. Schneider, I. Sorokos, Y. Papadopoulos, T. Kelly, R. Wei, E. Armengaud, and C. Kaypmaz, "Engineering of runtime safety monitors for cyber-physical systems with digital dependability identities," in *Computer Safety, Reliability, and Security*, A. Casimiro, F. Ortmeier, F. Bitsch, and P. Ferreira, Eds., Cham: Springer International Publishing, 2020, pp. 3–17.

[38] D. Schneider and M. Trapp, "Conditional safety certification of open adaptive systems," *ACM Trans. Auton. Adapt. Syst.*, vol. 8, no. 2, pp. 1–20, Jul. 2013. DOI: 10.1145/2491465.2491467.

[39] S. Müller and P. Liggesmeyer, "Dynamic safety contracts for functional cooperation of automotive systems," in *Computer Safety, Reliability, and Security*, A. Skavhaug, J. Guiochet, E. Schoitsch, and F. Bitsch, Eds., Cham: Springer International Publishing, 2016, pp. 171–182.

[40] A. Casimiro, J. Rufino, R. C. Pinto, E. Vial, E. M. Schiller, O. Morales-Ponce, and T. Petig, "A kernel-based architecture for safe cooperative vehicular functions," in *Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014)*, 2014, pp. 228–237. DOI: 10.1109/SIES.2014.6871208.

[41] J. Rushby, "Runtime certification," in *Runtime Verification*, M. Leucker, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21–35.

[42] S. Kabir, "An overview of fault tree analysis and its application in model based dependability analysis," *Expert Systems with Applications*, vol. 77, pp. 114–135, 2017. DOI: 10.1016/j.eswa.2017.01.058.

[43] R. Kianfar, P. Falcone, and J. Fredriksson, "Reachability analysis of cooperative adaptive cruise controller," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, 2012, pp. 1537–1542. DOI: 10.1109/ITSC.2012.6338839.

[44] T. Sánchez and J. A. Moreno, "A constructive lyapunov function design method for a class of homogeneous systems," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 5500–5505. DOI: 10.1109/CDC.2014.7040249.

[45] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 4661–4666. DOI: 10.1109/CDC.2016.7798979.

[46] D. Schneider, M. Trapp, Y. Papadopoulos, E. Armengaud, M. Zeller, and K. Höfig, "Wap: Digital dependability identities," in *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*, 2015, pp. 324–329. DOI: 10.1109/ISSRE.2015.7381825.

[47] A. Casimiro, J. Kaiser, E. M. Schiller, P. Costa, J. Parizi, R. Johansson, and R. Librino, "The karyon project: Predictable and safe coordination in cooperative vehicular systems," in *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*, 2013, pp. 1–12. DOI: 10.1109/DSNW.2013.6615530.

[48] T. Brade, "Failure algebra to validate sensor data," Ph.D. Thesis, Otto-von-Guericke Universität Magdeburg, Jul. 31, 2017.

[49] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 3387–3395, Jul. 2019. DOI: 10.1609/aaai.v33i01.33013387.

[50] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[51] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," *Advances in neural information processing systems*, vol. 30, 2017.

[52] E. Balaban, A. Saxena, P. Bansal, K. F. Goebel, and S. Curran, "Modeling, detection, and disambiguation of sensor faults for aerospace applications," *IEEE Sensors Journal*, vol. 9, no. 12, pp. 1907–1917, Dec. 2009. DOI: 10.1109/JSEN.2009.2030284.

[53] X. Dai, F. Qin, Z. Gao, K. Pan, and K. Busawon, "Model-based on-line sensor fault detection in wireless sensor actuator networks," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, Jul. 2015, pp. 556–561. DOI: 10.1109/INDIN.2015.7281794.

[54] J. Feng, S. Megerian, and M. Potkonjak, "Model-based calibration for sensor networks," in *SENSORS, 2003 IEEE*, vol. 2, 2003, 737–742 Vol.2. DOI: 10.1109/ICSENS.2003.1279039.

[55] S. Zug, A. Dietrich, and J. Kaiser, "Fault diagnosis in robotic and industrial systems," in *Fault Diagnosis in Robotic and Industrial Systems.* St. Franklin, Australia: Concept Press Ltd., 2012, ch. Fault-Handling in Networked Sensor Systems.

[56] S. Foix, G. Alenya, and C. Torras, "Lock-in time-of-flight (tof) cameras: A survey," *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1917–1926, Sep. 2011. DOI: 10.1109/JSEN.2010.2101060.

[57] S. Kabadayi, A. Pridgen, and C. Julien, "Virtual sensors: Abstracting data from physical sensors," in *2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks(WoWMoM'06)*, 2006, 6 pp.–592. DOI: 10.1109/WOWMOM.2006.115.

[58] Continental. "Ars 404-21." (Feb. 17, 2021), [Online]. Available: https://www.continental-automotive.com/getattachment/99443083-31fb-4345-9b84-6a02707041d4/ARS404-21_datasheet_en_170707_V07.pdf.pdf (Last Accessed Feb. 17, 2021).

[59] Hokuyo. "Urg-04lx-ug01." (Feb. 17, 2021), [Online]. Available: https://www.hokuyo-aut.jp/search/single.php?serial=166 (Last Accessed Feb. 17, 2021).

[60] R. J. Moffat, "Describing the uncertainties in experimental results," *Experimental Thermal and Fluid Science*, vol. 1, no. 1, pp. 3–17, 1988. DOI: 10.1016/0894-1777(88)90043-X.

[61] J. JCGM *et al.*, "Evaluation of measurement data—guide to the expression of uncertainty in measurement," *Int. Organ. Stand. Geneva ISBN*, vol. 50, p. 134, 2008.

[62] Y. Dodge, "Central limit theorem," in *The Concise Encyclopedia of Statistics.* New York, NY: Springer New York, 2008, pp. 66–68. DOI: 10.1007/978-0-387-32833-1_50.

[63] E. Elnahrawy and B. Nath, "Cleaning and querying noisy sensors," in *Proceedings of the 2Nd ACM International Conference on Wireless Sensor Networks and Applications*, ser. WSNA '03, San Diego, CA, USA: ACM, 2003, pp. 78–87. DOI: 10.1145/941350.941362.

[64] Y. Kim and H. Bang, "Introduction to kalman filter and its applications," *Introduction and Implementations of the Kalman Filter*, vol. 1, pp. 1–16, 2018.

[65] Y. Wang, N. Masoud, and A. Khojandi, "Real-time sensor anomaly detection and recovery in connected automated vehicle sensors," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2020. DOI: 10.1109/TITS.2020.2970295.

[66] M. A. Cooper, J. F. Raquet, and R. Patton, "Range information characterization of the hokuyo ust-20lx lidar sensor," *Photonics*, vol. 5, no. 2, 2018. DOI: 10.3390/photonics5020012.

[67] S. A. Hiremath, G. W. van der Heijden, F. K. van Evert, A. Stein, and C. J. ter Braak, "Laser range finder model for autonomous navigation of a robot in a maize field using a particle filter," *Computers and Electronics in Agriculture*, vol. 100, pp. 41–50, 2014. DOI: 10.1016/j.compag.2013.10.005.

[68] S. S. Dasika, M. P. Sama, L. F. Pampolini, and C. B. Good, "Performance validation of a multi-channel lidar sensor: Assessing the effects of target height and sensor velocity on measurement error," *Transactions of the ASABE*, vol. 62, no. 1, pp. 231–244, 2019.

[69] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava, "Sensor network data fault types," *ACM Trans. Sen. Netw.*, vol. 5, no. 3, 25:1–25:29, Jun. 2009. DOI: 10.1145/1525856.1525863.

[70] T. Muhammed and R. A. Shaikh, "An analysis of fault detection strategies in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 78, pp. 267–287, 2017. DOI: 10.1016/j.jnca.2016.10.019.

[71] M. Fagbemi, M. G. Perhinschi, and G. Al-Sinbol, "Modeling of upset sensor operation for autonomous unmanned systems applications," *International Journal of Intelligent Unmanned Systems*, 2019.

[72] L. Al Shalabi, Z. Shaaban, and B. Kasasbeh, "Data mining: A preprocessing engine," *Journal of Computer Science*, vol. 2, no. 9, pp. 735–739, 2006.

[73] W. Gilchrist, *Statistical modelling with quantile functions.* Chapman and Hall/CRC, 2000.

[74] J. H. Friedman, *The elements of statistical learning: Data mining, inference, and prediction.* springer open, 2017.

[75] R. Serfling, "Quantile functions for multivariate analysis: Approaches and applications," *Statistica Neerlandica*, vol. 56, no. 2, pp. 214–232, 2002. DOI: 10.1111/1467-9574.00195.

[76] F. Belzunce, A. Castaño, A. Olvera-Cervantes, and A. Suárez-Llorens, "Quantile curves and dependence structure for bivariate distributions," *Computational Statistics & Data Analysis*, vol. 51, no. 10, pp. 5112–5129, 2007. DOI: 10.1016/j.csda.2006.08.017.

[77] R. Y. Liu, J. M. Parelius, and K. Singh, “Multivariate analysis by data depth: descriptive statistics, graphics and inference, (with discussion and a rejoinder by Liu and Singh),” *The Annals of Statistics*, vol. 27, no. 3, pp. 783–858, 1999. DOI: 10.1214/aos/1018031260.

[78] P. Chaudhuri, “On a geometric notion of quantiles for multivariate data,” *Journal of the American Statistical Association*, vol. 91, no. 434, pp. 862–872, 1996. DOI: 10.1080/01621459.1996.10476954.

[79] P. Joslin, “Multivariate comparisons of random vectors with applications,” Ph.D. dissertation, Universidade de Murcia, May 2012, pp. 25–26.

[80] J. Fernandez-Ponce and A. Suarez-Llorens, “A multivariate dispersion ordering based on quantiles more widely separated,” *Journal of Multivariate Analysis*, vol. 85, no. 1, pp. 40–53, 2003.

[81] M. Ben-Daya, D. Ait-Kadi, S. O. Duffuaa, J. Knezevic, and A. Raouf, *Handbook of maintenance management and engineering*. Springer, 2009, vol. 7.

[82] L. D. Branges, “The stone-weierstrass theorem,” *Proceedings of the American Mathematical Society*, vol. 10, no. 5, pp. 822–824, 1959.

[83] N. E. Cotter, “The stone-weierstrass theorem and its application to neural networks,” *IEEE Transactions on Neural Networks*, vol. 1, no. 4, pp. 290–295, 1990. DOI: 10.1109/72.80265.

[84] J. Park and I. W. Sandberg, “Universal approximation using radial-basis-function networks,” *Neural computation*, vol. 3, no. 2, pp. 246–257, 1991.

[85] R. Kruse, C. Borgelt, C. Braune, S. Mostaghim, and M. Steinbrecher, *Computational intelligence: a methodological introduction*. Springer, 2016.

[86] N. Agarwal, Z. Allen-Zhu, B. Bullins, E. Hazan, and T. Ma, “Finding approximate local minima faster than gradient descent,” in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2017, Montreal, Canada: Association for Computing Machinery, 2017, pp. 1195–1199. DOI: 10.1145/3055399.3055464.

[87] G. Wood and B. Zhang, “Estimation of the lipschitz constant of a function,” *Journal of Global Optimization*, vol. 8, no. 1, pp. 91–103, 1996.

[88] T. Hickey, Q. Ju, and M. H. Van Emden, “Interval arithmetic: From principles to implementation,” *J. ACM*, vol. 48, no. 5, pp. 1038–1068, Sep. 2001. DOI: 10.1145/502102.502106.

[89] S. Mallat, *A wavelet tour of signal processing*, Third Edition, M. Stéphane, Ed. Academic Press, 2009.

[90] Y. Lei, T. Hu, G. Li, and K. Tang, *Stochastic gradient descent for nonconvex learning without bounded gradient assumptions*, 2019.

[91] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, “Automatic differentiation in machine learning: A survey,” *Journal of machine learning research*, vol. 18, 2018.

[92] S. J. Reddi, S. Kale, and S. Kumar, “On the convergence of adam and beyond,” in *International Conference on Learning Representations*, 2018.

[93] M. Daigle and I. Roychoudhury, "Qualitative event-based diagnosis: Case study on the second international diagnostic competition," 2010.

[94] H. Sak, A. Senior, and F. Beaufays, *Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition*, 2014.

[95] B. Lindemann, T. Müller, H. Vietz, N. Jazdi, and M. Weyrich, "A survey on long short-term memory networks for time series prediction," *Procedia CIRP*, vol. 99, pp. 650–655, 2021, 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 15-17 July 2020. DOI: 10.1016/j.procir.2021.03.088.

[96] P. I. Frazier, *A tutorial on bayesian optimization*, 2018.

[97] Y. Rubner, C. Tomasi, and L. Guibas, "A metric for distributions with applications to image databases," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, 1998, pp. 59–66. DOI: 10.1109/ICCV.1998.710701.

[98] F. Amato, C. Cosentino, and A. Merola, "On the region of attraction of nonlinear quadratic systems," *Automatica*, vol. 43, no. 12, pp. 2119–2123, 2007. DOI: 10.1016/j.automatica.2007.03.022.

[99] Q. Hu, B. Li, D. Wang, and E. K. Poh, "Velocity-free fault-tolerant control allocation for flexible spacecraft with redundant thrusters," *International Journal of Systems Science*, vol. 46, no. 6, pp. 976–992, 2015. DOI: 10.1080/00207721.2013.803634.

[100] R. C. Avram, X. Zhang, and J. Muse, "Quadrotor actuator fault diagnosis and accommodation using nonlinear adaptive estimators," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 6, pp. 2219–2226, 2017. DOI: 10.1109/TCST.2016.2640941.

[101] P. E. Protter, "Stochastic differential equations," in *Stochastic Integration and Differential Equations*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 249–361. DOI: 10.1007/978-3-662-10061-5_6.

[102] W. Krämer, "Generalized intervals and the dependency problem," *PAMM*, vol. 6, no. 1, pp. 683–684, 2006. DOI: 10.1002/pamm.200610322.

[103] ——, "Generalized intervals and the dependency problem," *PAMM*, vol. 6, no. 1, pp. 683–684, 2006. DOI: 10.1002/pamm.200610322.

[104] F. Grasser, A. D'arrigo, S. Colombi, and A. C. Rufer, "Joe: A mobile, inverted pendulum," *IEEE Transactions on industrial electronics*, vol. 49, no. 1, pp. 107–114, 2002.

[105] C. Aguilar-Ibanez, "A constructive lyapunov function for controlling the inverted pendulum," in *2008 American Control Conference*, 2008, pp. 5145–5149. DOI: 10.1109/ACC.2008.4587311.

[106] C. Anderson, "Learning to control an inverted pendulum using neural networks," *IEEE Control Systems Magazine*, vol. 9, no. 3, pp. 31–37, 1989. DOI: 10.1109/37.24809.

[107] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 763–770. DOI: 10.1109/ICRA.2011.5980244.

[108] T. Maeba, M. Deng, A. Yanou, and T. Henmi, "Swing-up controller design for inverted pendulum by using energy control method based on lyapunov function," in *Proceedings of the 2010 International Conference on Modelling, Identification and Control*, 2010, pp. 768–773.

[109] J. Butcher and G. Wanner, "Runge-kutta methods: Some historical notes," *Applied Numerical Mathematics*, vol. 22, no. 1, pp. 113–151, 1996, Special Issue Celebrating the Centenary of Runge-Kutta Methods. DOI: 10.1016/S0168-9274(96)00048-7.

[110] F. Tajti, G. Szayer, B. Kovács, and P. Korondi, "Robot base with holonomic drive," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 5715–5720, 2014, 19th IFAC World Congress. DOI: 10.3182/20140824-6-ZA-1003.00785.

[111] G. Indiveri, "Swedish wheeled omnidirectional mobile robots: Kinematics analysis and control," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 164–171, 2009. DOI: 10.1109/TRO.2008.2010360.

[112] V. Strobel, E. Castelló Ferrer, and M. Dorigo, "Blockchain technology secures robot swarms: A comparison of consensus protocols and their resilience to byzantine robots," *Frontiers in Robotics and AI*, vol. 7, 2020. DOI: 10.3389/frobt.2020.00054.

[113] G. Wampfler, M. Salecker, and J. Wittenburg, "Kinematics, dynamics, and control of omnidirectional vehicles with mecanum wheels," *Mechanics of Structures and Machines*, vol. 17, no. 2, pp. 165–177, 1989. DOI: 10.1080/15397738909412814.

[114] K. McGuire, G. de Croon, and K. Tuyls, "A comparative study of bug algorithms for robot navigation," *Robotics and Autonomous Systems*, vol. 121, p. 103 261, 2019. DOI: 10.1016/j.robot.2019.103261.

[115] A. Morar, A. Moldoveanu, I. Mocanu, F. Moldoveanu, I. E. Radoi, V. Asavei, A. Gradinaru, and A. Butean, "A comprehensive survey of indoor localization methods based on computer vision," *Sensors*, vol. 20, no. 9, 2020. DOI: 10.3390/s20092641.

[116] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4193–4198. DOI: 10.1109/IROS.2016.7759617.

Versicherung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.
Die Hilfe eines Promotionsberaters habe ich nicht in Anspruch genommen. Weitere Personen haben von mir keine geldwerten Leistungen für Arbeiten erhalten, die nicht als solche kenntlich gemacht worden sind. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

July 04th, 2022 M. Sc. Georg Jäger

Declaration

I hereby declare that I completed this work without any improper help from a third party and without using any aids other than those cited. All ideas derived directly or indirectly from other sources are identified as such.
I did not seek the help of a professional doctorate-consultant. Only those persons identified as having done so received any financial payment from me for any work done for me. This thesis has not previously been published in the same or a similar form in Germany or abroad.

July 04th, 2022 M. Sc. Georg Jäger