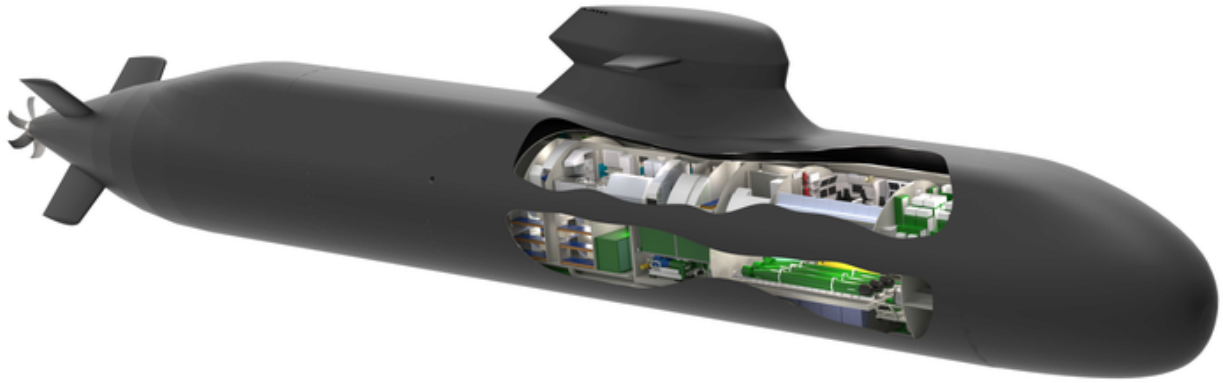
CHALMERS
UNIVERSITY OF TECHNOLOGY



On Automated Testing of Ship Control and Monitoring System in Submarine

Master's Thesis in Systems, Control and Mechatronics

Nils Hedberg

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

On Automated Testing of Ship Control and Monitoring System in Submarine

Nils Hedberg

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2024

On Automated Testing of Ship Control and Monitoring System in Submarine
Nils Hedberg

Industry Supervisor: Dr. Kim Weyns, Senior Systems Engineer at Saab Kockums
Academic Supervisor and Examiner: Dr. Martin Fabian, Professor at the Department of Electrical Engineering

Master's Thesis 2024
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of Saab Kockums's A26 submarine, the world's most modern submarine program for the Swedish Navy. Saab Kockums's A26 is a unique submarine with a proven modular design, silent long-endurance submerged performance, and excellent maneuverability in all waters. Copyright © Saab AB.

Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

On Automated Testing of Ship Control and Monitoring System in Submarine
Nils Hedberg
Department of Electrical Engineering
Chalmers University of Technology

Abstract

As technology advances, increasingly complex and autonomous engineering systems emerge. Given our daily dependence on them, these systems must work as intended. Testing is used to verify that the systems behave correctly according to given specifications. In industry, testing is often performed manually by test engineers. However, this approach is expensive, time-consuming, repetitive, and error-prone. A potential solution to these issues is test automation.

This thesis employs a design-science research methodology to investigate test automation at a case company. This resulted in the development of an automated testing system for a ship control and monitoring system in a submarine. The automated testing system consists of three different variants of test automation. The first variant involves creating scripts corresponding to manual test procedures in the current documentation. The second and third variants of the automated testing system comprise the testing approach falsification.

Optimization-based falsification involves using a simulation model to automatically identify input signals that cause a system to violate given specifications. Quantitative semantics are used to assess how close a scenario is to violating the specifications. The optimization problem includes decision variables that determine the type and shape of the generated input signals.

The proposed automated testing system has been assessed according to different measures. Combined, the automated testing system has proved that it confronts the business needs of the case company, where it enhances the productivity and quality of the testing activities. While the initial time spent on creating test scripts may be considerable, the long-term benefits become evident with saved efforts and costs, especially when most systems are tested frequently.

Keywords: Test Automation, Software Testing, Falsification, Industrial Case Study

Acknowledgements

The author of this thesis would like to send my most sincere thank you to my supervisors Kim Weyns, Joakim Davidson Truuberg, and Martin Fabian, who have throughout the study always supported me and provided feedback in prosperous times, and guidance in more challenging times.

Further, I would like to send a very big thank you to Astrid Stenholm and Peter Carlbrink, who helped me reach considerably further with my thesis.

Lastly, I would like to thank everyone at the Department of Ship Automation Submarine at Saab Kockums, not only for being incredibly helpful in this study, but they also made my stay in Malmö so much fun.

Nils Hedberg, Gothenburg, July 2023

Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

BO	Bayesian Optimization
CPS	Cyber-Physical Systems
$\mathcal{GP}$	Gaussian Processes
HCR	Hybrid-Corner-Random
HIL	Hardware-In-the-Loop
HMI	Human–Machine Interface
LCB	Lower Confidence Bound
LOX	Liquid Oxygen
LSF	Line-Search Falsification
MIL	Model-In-the-Loop
MMP	Multi Mission Portal
PLC	Programmable Logic Controller
SCMS	Ship Control and Monitoring System
SwDD	Software Design Description
SwTD	Software Test Description
SUT	System Under Test
TS	Thompson Sampling
TR	Trust Region
TuRBO	Trust Region Bayesian Optimization
VBools	Valued Booleans

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

Society is increasingly dependent on dedicated computer and software systems that assist us in nearly every aspect of day-to-day life. With advances in various technologies, we are witnessing the era of development of engineering systems with ever-increasing complexity and high levels of autonomy [1, 2, 3, 4]. Thus, we rely on the correct execution of systems' software in our daily lives. When the software does not behave as intended, the consequences can be devastating [5, 6]. Fortunately, there are numerous methods to ensure that software behaves as intended. One of these methods is testing, which is widely used today [7].

Testing involves activities to identify possible errors in software. The quality and safety of the complete system delivered to users are improved by fixing these errors. Furthermore, testing is used to validate that the software meets requirements and expectations. The traditional way of conducting testing is with a manual approach. Manual testing involves a human acting as an end-user of the system under normal and unusual circumstances. Testing entails tasks such as test case execution, comparing the actual test case result to the expected result, and summarizing the results. However, manual testing is costly, time-consuming, repetitive, tedious, and error-prone [8, 9, 10, 11]. The cost of testing is a challenge in the industry and can correspond to over 50 percent of the total development cost [10, 12].

To overcome the drawbacks of manual testing, test automation [11, 13] has been proposed as a solution. Test automation uses other tools, separate from the system under test (SUT), to automate manual testing or user actions performed in the SUT. With test automation, the testing can be completed more quickly, reliably, cheaply, frequently, and with shorter feedback cycles compared to manual testing, which results in better software [9, 11, 14, 15, 16]. However, it should be noted that while test automation is essential for improving the testing process, it is not a panacea for all testing activities. Thus, manual and automated testing can be seen as two complementary methods for assessing the correctness of systems, which can be unified to increase productivity and reliability in testing [17].

This thesis considers the problem of introducing test automation for software in systems at an industrial case company. Note, in the following, system and software testing will be used interchangeably but will always refer to software testing in systems.

1.1 Background

Testing includes methods where test data is generated from the SUT to determine whether the behavior is correct. Two of the most common testing strategies are specification-based testing and implementation-based testing. Below, both testing strategies will be introduced.

Specification-based testing is an approach where the system is viewed as a black box, and only the inputs and outputs of the system are studied. Hence, specification-based testing does not analyze the inner workings of a system. The test data is compared to a specification describing the system's expected behavior. The specification can be expressed in either natural language or mathematical notations. In automated testing, the specification must be defined unambiguously; thus, it is commonly expressed using variants of temporal logic.

Implementation-based testing is an approach where test data generation is motivated by analyzing the system's inner workings. This approach generates test data to optimize coverage criteria, such as ensuring that all source code statements are executed at least once (statement coverage) or that each Boolean subexpression has been evaluated as true and false (condition coverage). The domain of implementation-based testing includes techniques such as symbolic execution [18] and constraint solving [19].

One important quote to remember when performing testing is described by the famous computer scientist Edsger W. Dijkstra [20].

> *Program testing can be used to show the presence of bugs, but never to show their absence!*
>
> Edsger W. Dijkstra, 1970

This quote can be interpreted as meaning that there is no way to achieve complete confidence in systems' correctness solely through testing. As a result, testing can not prove (with some exceptions) that systems are correct. On the other hand, no software testing is perfect, and there is no upper limit to how thoroughly tested software can be. The associated tests increase confidence that the system will behave correctly, but there is always room for improvement.

There are alternative methods to testing for assessing the correctness of a system, e.g., formal verification [21]. Although testing is used to evaluate the correctness of a system, the propositions are not the same as those used in formal verification. In formal verification, the aim is to prove that a system is error-free. This method involves mathematical techniques to assess the functional correctness of a system. Formal verification aims to find whether a system satisfies a formal specification or not with mathematical rigor. The literature on formal verification can roughly be grouped into two branches: deductive verification [22] and model checking [23, 24].

Deductive verification is a technique where axioms and proof rules are used to prove the correctness of a system. When verifying a property, the system and the property are translated into logical formulas, and a theorem prover is used to demonstrate that the property holds for the system. However, finding good heuristics to guide automatic proof construction is difficult [25]. As a result, user

interaction is frequently required, making deductive verification a discipline that is time-consuming and requires logical reasoning skills. Thus, it must be performed by experts with extensive experience.

Model checking is a technique where an exhaustive search of the finite-state model of a system is performed in a brute-force manner to determine if some specification is true or false. The model checker, a software tool that performs the model checking, systematically examines all possible system scenarios. In this way, whether a given system model truly satisfies a specific property can be shown. However, the brute-force approach also illustrates the challenges of using model checking to evaluate the correctness of a system. The number of states in a system grows exponentially with the number of sub-systems, i.e., a state explosion problem.

Formal verification is usually impractical for industrial systems for various reasons. Many industrial systems do not have a formal model that can be used for formal verification. Also, even if formal models exist, formal verification of large systems may be intractable due to the time and space complexity of the verification algorithms. Furthermore, for many industrial applications, the SUT can only be simulated. Hence, in this thesis, the correctness of systems is assessed using testing.

1.2 Case Company and System of Analysis

The thesis was carried out in collaboration with the Swedish company Saab Kockums at their site in Malmö, Sweden. Saab Kockums has approximately 1700 employees and operates in Sweden and Singapore. The company designs, builds, and supports advanced naval systems such as surface combatants and submarines for the Swedish Armed Forces and other customers worldwide. The submarine competence places Sweden among the few nations worldwide that can produce modern and advanced submarines [26]. Saab Kockums is a business area within Saab AB that sells products to over 100 countries and operates in over 30 countries worldwide.

In June 2015, Sweden ordered two A26 submarines of the new Blekinge-class, developed and manufactured by Saab Kockums. The A26 submarines will be powered by conventional diesel-electric propulsion machinery and equipped with an air-independent Stirling propulsion system, which enhances the stealth capability. A revolutionary feature of the A26 is the Multi Mission Portal (MMP), which can be described as a torpedo tube with a diameter of 1.5 meters. Here, divers and unmanned vehicles can be transported in and out of the submarine [27]. The Blekinge-class submarines enhance all the traditional operational capabilities of a submarine and are also a robust intelligence-gathering platform within the wider defense network [28].

In this thesis, the ship control and monitoring system (SCMS) developed for the A26 submarine was chosen as the unit of analysis for test automation. The SCMS is a safety-critical system and is crucial for ensuring safe operations of the submarine. The SCMS can be likened to a spinal cord in the human body; it co-locates control and monitoring of ship systems that are distributed on board. Since the SCMS is a central part of the operational capability of the submarine and the requirements for reliability and safety are high, the SCMS needs to be thoroughly tested in collab-

oration with the distributed systems. The SCMS is based on programmable logic controllers (PLC), with software developed in the source code editor Automation Studio 4 [29] from B&R Industrial Automation.

Saab Kockums has developed a testing rig that replicates the SCMS of the A26 submarine, enabling the testing of the system. The rig is built on a platform utilizing PLCs, human-machine interfaces (HMI), and computers. Additionally, external computers can be connected to interact with the SCMS. Before this thesis, the testing of the SCMS was carried out solely through manual interaction with the system in real time via HMIs. For example, a test case can observe if a valve is closed after manipulating the circumstances in the HMI. Also, sophisticated control algorithms in the SCMS are tested using simulation-based methods.

Due to the characteristics of the naval defense industry, the project cycles span several years to decades. As a result, a significant amount of source code for integrated systems on a submarine has been developed during different phases of the project. Furthermore, several branches of source code exist that are specifically designed for various ships. These characteristics add up to difficulties in conducting the testing activities efficiently. Regression testing [30] using manual practices is impractical as a project progresses. The number of tests that must be conducted again to search for regression issues would inevitably overwhelm test engineers. By introducing automated testing for the SCMS, verification and validation activities may become more efficient.

This thesis aims to investigate test automation methods that can be applied for specification-based testing of software that runs on actual PLC hardware in the SCMS.

1.3 Related Work

The authors in [31] implement automated testing for software in a simulated communication-based train control system, a commonly used system for urban rail transit signals. The authors propose using the test automation tool Robot Framework [32] to automate the testing of the system. This tool is a generic open-source automation framework based on Python [33]. The results in [31] show that the automated testing solution is feasible for the simulated communication-based train control system. Furthermore, the automated testing solution improves the testing efficiency as well as the testing quality. However, the authors in [31] do not address the problem of test automation for software running on physical hardware that will be part of an actual end-product, nor do they address test automation for PLC-based systems. The former problem is solved in [34], [35], [36], [37], and [38], where test automation is implemented for software running on physical hardware included in actual end-products. These papers will be briefly presented below.

The authors in [34] present a solution to implement automated testing of software running on an embedded digital signal processing device using Robot Framework. The results show that test automation for this device can be realized using their solution, and it increased the speed of testing and validation of the device.

In [35], the authors present an automated testing system for software in warships. The automated testing system uses the LoadRunner Automation API to write test

sequence scripts. The results show that the main advantages of the automated testing system compared with manual testing are facilitated regression testing, reduced test time, the capability to simulate large workloads, saving human resources, and ensuring consistency and repeatability of the tests.

In [36], the authors present a solution for automated software testing in multimode terminals, which are equipment used in the telecommunication industry. Their solution is based on the tool LabWindows/CVI system [39], where the tests are scripted in the C language [40]. The authors claim that their test automation solution can perform testing quicker and more accurately than manual testing.

The authors in [37] and [38] are implementing automated testing of software in electronic control modules, which are vital components of conventional combustion engines. The test automation system consists of three main parts. The first part is the software tool NI TestStand [41], which is used for writing test sequence scripts. The second part is the digital FMET box, which is hardware that has electronically controlled relays in it. These relays enable tests related to electrical circuit continuity fault conditions, e.g., open-circuit, short-circuit to battery positive, and short-circuit to battery ground. The third part is the LUIS load box, a hardware engine simulator that provides different parameter values measured by emulated sensors. In [37], the authors conclude that the test automation system requires less manual effort and fewer man hours and is cheaper than manual testing. In [38], the test automation system delivers faster test results with reduced manual efforts and errors while saving overall cost compared to the previously used manual approach. However, [34], [35], [36], [37], and [38] have not addressed the problem of implementing test automation for PLC-based systems. This remaining problem is addressed in [42].

In [42], the authors investigate test automation tools for testing PLC programs in the Codesys IDE [43]. First, they identify CoUnit and Codesys Test Manager as the most-discussed test automation tools for Codesys. Then, they compare these tools using different criteria. Their findings imply that both test automation tools provide users with necessary automation functionalities. However, Codesys Test Manager seems more mature, has more helpful test execution features, and is more user-friendly. By contrast, CoUnit is not user-friendly, is limited in its automation features, and working with it demands structured text programming knowledge. However, the authors in [42] do not address the problem of choosing a test automation tool for testing PLC programs in Automation Studio 4. Hence, there is a lack of research regarding test automation tools applicable to systems developed in Automation Studio 4.

In [44], an approach for automated simulation-based testing grounded on fuzzy logic is presented. The proposed method uses type-2 fuzzy logic, which provides a more robust handling of data uncertainties involved in the testing process. The automated testing solution has been successfully applied to test unmanned aerial systems' perception systems. However, the proposed test automation using fuzzy logic requires domain experts with extensive experience.

The authors of [45] and [46] propose using the falsification method for automated simulation-based testing, which does not require expert knowledge to the same extent as with the fuzzy-logic solution. In [45], the falsification is applied for testing

of cyber-physical systems (CPS) [47], i.e., systems where computation, networking, and physical processes are integrated. The authors assess the performance of a simple testing strategy based on combining random inputs with inputs on the lower and upper bounds of the allowed input range. The evaluation using benchmark problems demonstrates that this combined input strategy works well. They propose using this approach as a baseline method when evaluating falsification methods. Furthermore, they propose line-search falsification (LSF) for testing CPSs, a simple gradient-free optimization method that performs optimization over a line. The authors compare the optimization methods Nelder-Mead [48] and SNOBFIT [49] to the line-search method. The comparison shows that line-search optimization improves falsification efficiency while remaining simple. Also, efforts have been made to enlighten the importance of choosing suitable quantitative semantics [50, 51] and input generator [52] when performing falsification of CPSs.

In [46], the authors investigate the falsification of CPSs using Bayesian optimization (BO). This sample-efficient method learns a surrogate function that models the relationship between a test specification evaluation and inputs. On standard benchmark problems, they use vanilla BO and two other prominent BO methods, Trust Region Bayesian optimization (TuRBO) [53] and πBO [54] and compare their performance with state-of-the-art falsification methods. As a local search approach, TuRBO is less sensitive to the complexity of the CPS and eliminates the bias for exploiting edges. In πBO, test engineers can include their prior knowledge of the falsification problem by defining where there is a higher chance of the system being falsified. Experiments show that TuRBO outperforms the other BO methods with more successful falsifications.

In [45] and [46], the falsification is performed using model-in-the-loop (MIL) simulations, i.e., no falsification using hardware-in-the-loop (HIL) simulations. Hence, there is a lack of research regarding falsification with HIL simulations.

1.4 Purpose

The purpose of the thesis is two-fold, as two main stakeholders benefit from this thesis.

First and foremost, there is the company Saab Kockums, where the main idea of the thesis originated. A significant part of the software development for A26's SCMS is spent on testing. Before this thesis, the testing was done solely through manual interaction with the system in real time via the HMI. It is both a time-consuming and costly process, which must be frequently repeated with each software update and thus many times during the submarine's life cycle. One possible way to increase productivity and quality of the testing is to implement test automation. Thus, Saab Kockums identified test automation for SCMS as a business need.

Second of all, there is the scientific purpose of the thesis. As test automation for systems based on PLCs is still relatively new, there is much to be explored. Therefore, any additional research on test automation for PLC-based systems can benefit the scientific community. Furthermore, the falsification of CPSs has been well-researched lately. However, most of the research covers testing with MIL. Hence, investigating the application of the falsification framework for testing with HIL con-

tributes to the scientific community. To this end, an industrial case study examining how test automation of PLCs and the falsification framework can be implemented in an industrial practice is a relevant approach.

1.5 Objective

This thesis aims to address the implementation of test automation for the SCMS. The four research questions below summarize this goal.

RQ1. *How could various testing tools and methods be used to automate the testing for a ship control and monitoring system?*

There are various tools and methods available for test automation. Some tools and methods are more appropriate than others for SCMS's testing environment and the purpose Saab Kockums wants to achieve with test automation. This research question aims to investigate what testing tools and methods are appropriate to incorporate in the automated testing environment for the SCMS and how these tools and methods can be implemented.

RQ2. *How does the implemented test automation solution affect the testing activities of the ship control and monitoring system?*

If the implemented test automation for the SCMS can be helpful for Saab Kockums, it remains to be proven that it can benefit the testing procedure. Automated testing only reveals failures in accordance with pre-defined logic, unlike skilled manual testers who can utilize their knowledge to prompt the system to reveal errors. Furthermore, there may be differences in the effort to develop and maintain the test automation solution for the SCMS compared to the manual practices used before this thesis. The benefits must outweigh the drawbacks. Otherwise, the test automation is of little use. Thus, it is essential to understand the strengths and weaknesses of the implemented test automation solution.

RQ3. *How can the falsification methods be improved for the testing environment in this thesis?*

Algorithms can be improved by encountering new situations, especially when the algorithms are based on heuristics. The testing environment in this thesis is new for the falsification method, both regarding the hardware and the simulation model with its corresponding input space. This research question aims to explore and present general suggestions for the falsification method that improves the testing for the problem faced in this thesis.

RQ4. *How do different test-case generation methods perform with the falsification problem in this thesis?*

Falsification can be performed using either optimization-free or optimization-

based methods. In optimization-free methods, new input parameters can be generated randomly. Optimization-based falsification generates new input parameters to find lower objective function values. The optimization method used affects the efficiency of the falsification process. This research question aims to evaluate the performance of both optimization-free and optimization-based methods and understand how they work in falsification.

1.6 Delimitations

Three factors limit the scope of the thesis. The first factor is the number of systems in the SCMS for which test automation will be implemented. The second factor is the time allocated to the iterative process in finding optimal features in the optimization-based falsification methods. The third factor is the chosen SCMS for analysis since more SCMSs are being developed at Saab Kockums but for other ships. Below, all three limiting factors will be explained.

The test automation could be implemented and evaluated for many distributed systems in the SCMS. However, the research in this thesis will be limited to only implementing test automation for two systems: the liquid oxygen (LOX) system and the hovering control system. A larger sample size would be desired to get a more profound study of the test automation for the SCMS. On the other hand, the software test specifications for different systems have similar characteristics. Thus, if the test automation is applicable for one system, it should also be applicable for other systems.

When using optimization techniques that require tuning for optimal performance, it would be beneficial to use an iterative process. For example, if the features in the falsification implementation prove ineffective, they could be revised and reapplied to the falsification problem, and any difference in results could be analyzed. However, this thesis will not consider a profound study when selecting different features for the falsification implementation. The scope is not to find the optimal features in falsifying a specific system. Instead, the scope is to give Saab Kockums suggestions on how they can automate the testing of their simulation models to find faults in the systems.

Today, at the same time as the development of A26 is taking place, the submarine HMS Halland of Gotland-class is undergoing a mid-life upgrade. This submarine also contains a SCMS where test automation could be implemented. This thesis will focus only on test automation for the testing environment built for A26's SCMS. However, the experiences and testing methods will most likely be transferable to the testing of HMS Halland.

1.7 Contributions

This thesis contributes to an automated testing system, AuTS, for Saab Kockums, facilitating test automation of the SCMS. AuTS consists of three different variants of test automation. The first variant involves writing script-based tests based on documents describing the manual test procedure to switch the test execution from

a manual to an automated approach. The second variant comprises the falsification of simulation models. The third variant covers how the functionality in the ship systems can be tested using falsification.

Moreover, this thesis contributes with three suggestions to the LSF algorithm presented in [45]. The first and second suggestions are adjustments to the heuristic function used during optimization. The third suggestion is to multiply the random direction vector by the range of the domain in each dimension. The random direction vector generates a line in the search space.

1.8 Outline

The remainder of the thesis is organized as follows: In Chapter 2, a theoretical framework is presented, providing the foundation for the implementation of test automation. Chapter 3 outlines the research approach employed in this thesis. The implementation of test automation is detailed in Chapter 4. The results of the research questions are presented in Chapter 5. In Chapter 6, a comprehensive discussion of the findings is provided. The thesis concludes with Chapter 7, offering final remarks, insights, and suggestions for future work.

2
Theory

This chapter initiates with a concise overview of submarines and the SCMS. Subsequently, it provides an introduction to submarine hydrodynamics and the hovering controller. Furthermore, a comprehensive exploration of the falsification framework is undertaken, encompassing both optimization-based and optimization-free falsification approaches. This theoretical foundation lays the foundation for the implementation of AuTS.

Note, with possibly slight abuse of denotation, the submarine will be denoted as a ship, not a boat.

2.1 Submarine Vessel

Submarines are specialized ships that can travel entirely below the water's surface. From peacekeeping to wartime activities, submarines play a part in a variety of naval operations and responsibilities in both offensive and defensive roles and thus are essential to several nations' defense strategy

The design of a submarine is complex and is observed in Figure 2.1. The submarine hull is usually based on an axisymmetric body around its longitudinal axis. Adding a fin to house items such as periscopes, snorkels, and other masts is necessary for operational purposes. Many modern submarines are propelled by a conventional propeller or a pump jet located on the longitudinal axis at the very end of the aft. To steer the submarine, the aft control surfaces include four stern rudders, commonly arranged cross-formed or X-formed.

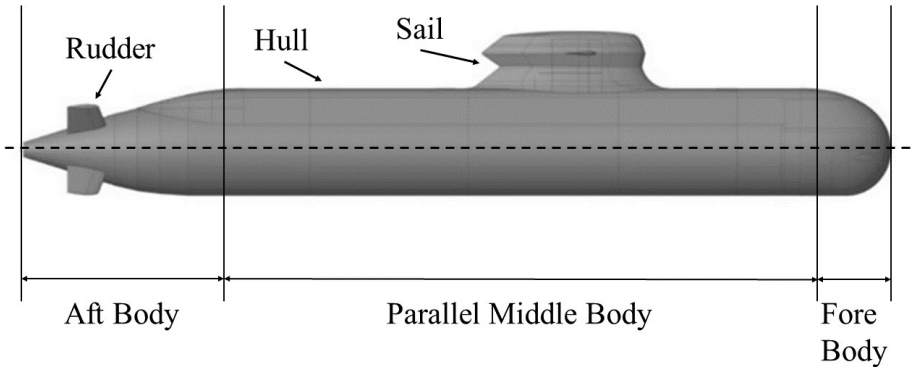


Figure 2.1: Submarine geometry.

2.1.1 Ship Control and Monitoring System

The SCMS is a system in the submarine that integrates control and monitoring of ship systems throughout the submarine, making it possible for an operator to manage the ship systems from one place. Submarines include ship systems such as LOX systems, freshwater systems, and diving and ballast systems. Control sequences, alarms, and automation are used as appropriate in the SCMS to support the operator and increase safety and efficiency.

The SCMS interfaces the ship systems through distributed Input/Output and communication nodes connected to a fault-tolerant ship-wide network. A set of PLCs performs data processing, i.e., the control and monitoring logic.

2.1.2 Dynamics of Submarine

Submarines comply with Archimedes' Principle, which states that a body immersed in a fluid experiences an upward force (buoyancy) equal to the weight of the displaced fluid. The buoyancy force must precisely balance the body's mass in the vertical plane to achieve equilibrium. The mass of a submarine will change during operations due to the use of consumables such as fuel and weapon discharge [55]. Furthermore, changes in sea water density, hull compressibility, and surface suction require the ability to change the submarine's mass to control the depth. Thus, ballast tanks are installed on a submarine to change the mass by filling and draining ballast tanks with sea water. By filling the ballast tanks, the submarine will increase its mass and thus enable a downward motion in the vertical direction. By draining the ballast tanks, the submarine enables an upward motion.

In the following, the motion model of a submarine will be presented. The motion in the vertical direction, defined as the z-direction, is based on Newton's second Law and is observed in (2.1).

$$F_z = m_{sub}\,\ddot{z}_{sub} \tag{2.1}$$

The acceleration of the submarine $\ddot{z}_{sub}$ depends on the total mass of the submarine m_{sub} and the total force F_z acting on the submarine. The free-body diagram of a submarine is observed in Figure 2.2. The force F_z consists of the force components

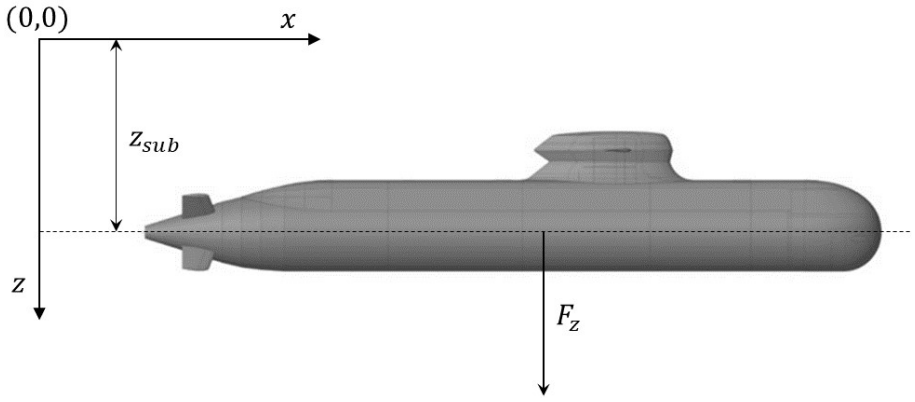


Figure 2.2: Free body diagram of a submarine.

observed in (2.2). The motion model assumes that the buoyancy force equals the

body's mass at the beginning of observation. Thus, these forces cancel each other out and are omitted from (2.2)

$$F_z = F_{load} + F_{weight\ comp} + F_{hull\ comp} + F_{density} + F_{drag} + F_{surf\ suct} + F_{MMP} \tag{2.2}$$

F_{load} is the force due to load changes that arise when loading or unloading the submarine, e.g., equipment or divers through the MMP.

$F_{weight\ comp}$ is the force due to the change of sea water volume in the ballast tanks and is defined as:

$$F_{weight\ comp} = \Delta V_{BT}\, \rho\, g \tag{2.3}$$

where ΔV_{BT} is the accumulated volume change of sea water in the ballast tanks from the beginning of the observation, ρ is the water density, and g is the gravitational acceleration. This force component models the filling and draining of ballast tanks.

$F_{hull\ comp}$ is the force caused by hull compression and is defined as:

$$F_{hull\ comp} = \chi(z_{sub} - z_{0,sub}) \tag{2.4}$$

where z_{sub} is the depth of the submarine, $z_{0,sub}$ is the initial depth of the submarine at the beginning of the observation, and χ is a constant that corresponds to the ratio of the change in buoyancy force to the change of depth. The rationale behind this force is that the deeper the submarine operates, the greater the water pressure acting on it, causing the hull to compress. This reduces the submarine's immersed volume and, as a result, reduces upward buoyancy force. Conversely, if the submarine moves closer to the sea water surface, the water pressure acting on it decreases, and thus, the immersed volume and upward buoyancy force increase. The structure of the submarine determines the magnitude of χ.

$F_{density}$ is the force caused by changes in sea water density and is defined as:

$$F_{density} = -V_{sub}\, \varepsilon\, g\, (z_{sub} - z_{0,sub}) \tag{2.5}$$

where V_{sub} is the total volume of the submarine, and ε is the sea water density gradient with respect to depth, i.e., $\varepsilon = d\rho/dz$. The density in sea water varies due to variations in salinity, temperature, and pressure [56].

F_{drag} is the force acting opposite to the relative motion of the submarine with respect to the surrounding fluid and is defined as:

$$F_{drag} = -\frac{1}{2}\, \rho\, \dot{z}_{sub}\, |\dot{z}_{sub}|\, C_D\, A_{sub} \tag{2.6}$$

where $\dot{z}_{sub}$ is the velocity of submarine in z-direction relative to the fluid, C_D is the drag coefficient of the submarine, and A_{sub} is the cross-sectional area of the submarine in the z-direction.

$F_{surf\ suct}$ is the force caused by surface suction, an upward force that increases the closer the submarine is to the water's surface. This force can significantly impact a submarine's behavior and is reported to affect the submarine even at a depth of 50 meters [55]. Surface suction can occur in calm water, but wind-generated waves enhance it. The equations Saab Kockums uses to model the surface suction are omitted from this report for confidentiality reasons.

F_{MMP} is a force to model dynamics experienced in the MMP. The equations Saab Kockums uses for modeling F_{MMP} are omitted from this report for confidentiality reasons.

2.1.3 Hovering Controller

The steering system of the A26 submarine includes a hovering controller to keep the submarine hovering at a certain depth in the water by automatically filling and draining the ballast tanks. The objective of the hovering controller is to steer the depth tracking error e to zero, defined as:

$$e = z_{sensor} - z_{sp,sub} \tag{2.7}$$

where z_{sensor} is the depth of the submarine measured by sensors, and $z_{sp,sub}$ is the setpoint for the hover depth.

The measured depth by the sensors is affected by disturbances due to waves in the sea, roll motion of the submarine, and sensor noise, i.e., noise that corresponds to random variations of sensor output unrelated to variations of sensor input. The measured depth by the sensors is defined as:

$$z_{sensor} = z_{sub} + z_{wave} + z_{roll} + z_{noise}. \tag{2.8}$$

where z_{wave} is wave disturbance, z_{roll} is roll disturbance, and z_{noise} is sensor noise. The disturbances z_{noise} and z_{roll} are defined as:

$$z_{noise} \sim \mathcal{N}\left(\mu, \sigma^2\right), \tag{2.9}$$

$$z_{roll} = A_{roll} \cdot \sin(\omega_{roll}\, t + \varphi_{roll}) \tag{2.10}$$

where μ and σ^2 are the mean and variance of Gaussian sensor noise. Also, A_{roll}, ω_{roll} and φ_{roll} are amplitude, angular frequency, and phase for a sine wave of time t, respectively. The equations modeling z_{wave} are omitted from the report for confidentiality reasons.

2.2 Falsification of Cyber-Physical Systems

Falsification [57] is a testing method that automatically checks for cases where a given system specification is not fulfilled. Simulation-based falsification can be used when a system model $\mathcal{S}$ can be simulated, and a temporal logic specification φ exists. The behavior of the actual system is approximated by $\mathcal{S}$. The expected behavior of the system is described by φ. Falsification is a black-box method that relies only on input and output traces from simulating the model. Falsification can be performed using optimization-free or optimization-based methods to generate new input signals.

It might be computationally expensive to simulate the model. Thus, it is desirable to reduce the number of simulations needed during falsification to find an example where the specification is not fulfilled. In optimization-based falsification, the objective is to reduce the number of necessary simulations by using optimization methods. The input signals for the next simulation are based on evaluating the model's output from previous simulations. Since $\mathcal{S}$ is given as a black box, the optimization is limited to gradient-free methods. In the optimization-free approach, random methods can be used to generate new input signals.

The falsification procedure is shown in Figure 2.3, which is inspired by [58]. Initially, a generator creates input signals $\mathbf{x}_i^s$ based on an input parametrization $\mathbf{x}$. A simulator generates output responses $\mathbf{x}_o^s$, simulating the system model using $\mathbf{x}_i^s$ as input. The combination of input signals $\mathbf{x}_i^s$ and output responses $\mathbf{x}_o^s$, is denoted as trace $\mathbf{x}^s$. Next, $\mathbf{x}^s$ is evaluated based on the specification φ using an objective function f^φ defined by quantitative semantics. This objective function estimates the distance to the specification being falsified. If the specification is falsified, the falsification procedure terminates. Otherwise, a parameter selector generates new input parameters $\mathbf{x}$ according to optimization-free or optimization-based methods. Then, a new iteration of the falsification procedure begins. In the following, each of the components in Figure 2.3 will be described

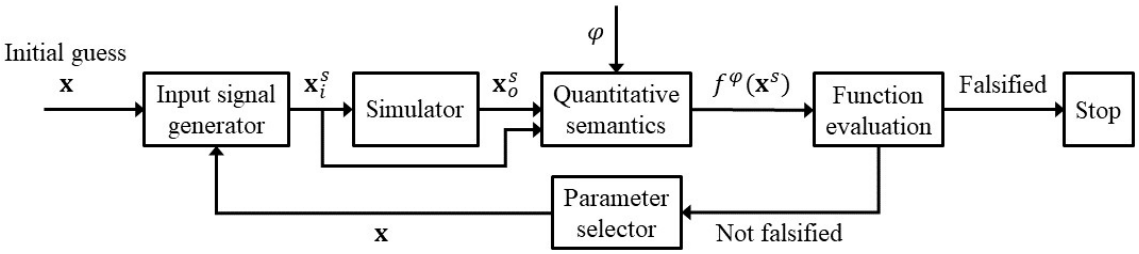


Figure 2.3: A flowchart describing the falsification process.

2.2.1 Input Signal Generator

Input signals are required for the simulation of the system model. The input signals $\mathbf{x}_i^s(k)$ are represented as functions of time k, which ranges from the start to the end of the simulation. CPS falsification problems generally involve continuous-time input signals, so the search space is of infinite dimension [57]. By considering input signals that are parameterized by a finite number of parameters $\mathbf{x}$, the complexity of the falsification problem is reduced. Thus, the input parameters $\mathbf{x}$ are used in the input signal generator to create the actual input signals $\mathbf{x}_i^s$.

In the optimization-based methods, the input parameters $\mathbf{x}$ are decision variables in the optimization algorithm. The dimensionality of the optimization problem affects the performance of the optimizer. Many decision variables are difficult for the optimizer because the search space grows rapidly with the increase in decision variables. On the other hand, reducing the dimensionality by modeling the problem with a small number of input parameters makes it difficult to generate input signals that falsify the SUT. Hence, there is a trade-off between the flexibility of the input signals generation and the dimensionality of the optimization problem [57]. Furthermore, a thorough understanding of the SUT is required to define appropriate input signals and parameters since system dynamics are often complex.

2.2.1.1 Input Parametrization

The domain of the input parameters $\mathcal{X}$, i.e., search space, is defined as:

$$\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^n \mid l_i \leq x_i \leq u_i\} \quad \text{for } i = 1, \ldots, n \tag{2.11}$$

where l_i and u_i are the lower and upper bounds for the ith input parameter, and n is the number of input parameters.

The input parameterization methods used in this thesis are constant, staircase, pulse, and sinusoidal. All four parametrization methods will be introduced below.

A constant input signal only requires one parameter, *base*, that defines the signal's value. This input signal remains constant for the entire simulation. The *base* can be any value inside or on the allowed input signal range $[l, u]$. Figure 2.4a shows an example of a constant input signal.

A staircase input signal comprises a set of equally spaced steps of equal amplitude. This signal is defined by four parameters: *amplitude*, *base*, *delay*, and *period*. Note that the input signal will be saturated to the closest boundary value in the allowed range $[l, u]$ if the value of the staircase function is outside the allowed range. Depending on the allowed ranges for the input parameters, the staircase can produce various signal types, e.g., a constant signal if $delay > T$ or a single step signal if $period > T$ and $delay < T$. Figure 2.4b shows an example of the staircase input signal.

A pulse input signal is a periodic square wave that can be defined by five parameters: $amplitude_1$, $amplitude_2$, *delay*, *period*, and *width*. These parameters correspond to the suggested pulse generator in [52], i.e., this is not the conventional way to parametrize a pulse. However, parametrizing, as proposed in [52], simplifies defining appropriate domains for the input parameters. The pulse generator can also produce a variety of signal types. Figure 2.4c shows an example of the pulse input signal.

A sinusoidal input signal is a continuous and smooth periodic wave that can be defined by four parameters: $amplitude_1$, $amplitude_2$, *delay*, and *period*. To simplify the definition of appropriate domains for the input parameters, the sinusoidal signal is parameterized similarly to the pulse generator in [52]. The conventional parameters for a sinusoidal are amplitude and midline. However, these parameters can be derived based on $amplitude_1$ and $amplitude_2$. Figure 2.4d shows an example of the sinusoidal input signal.

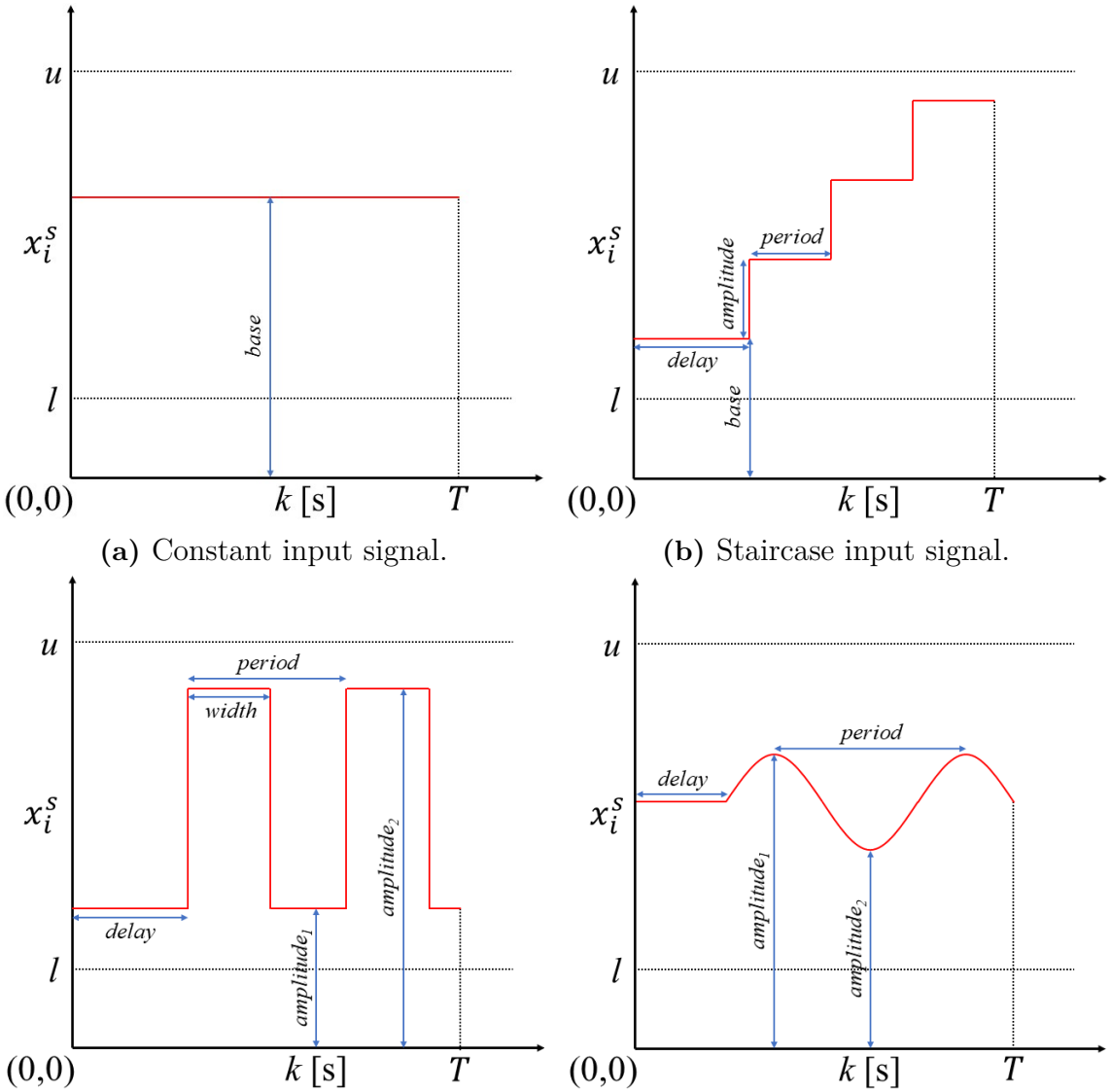
2.2.2 Simulator

The purpose of the simulator is to generate the corresponding output responses $\mathbf{x}_o^s$ from the system model using the input signals $\mathbf{x}_i^s$. The simulator with the system model $\mathcal{S}$ is observed in Figure 2.5.

2.2.3 Quantitative Semantics and Valued Booleans

To test a system with the falsification method, the expected behavior of the system must be defined and expressed mathematically in a specification φ. The specification can be formulated using different mathematical formalisms [57]. In this thesis, the specifications are formulated using the framework Valued Booleans (VBools) [58].

A VBools $\langle v, \theta \rangle$, is a Boolean value $v \in \mathbb{B}$ together with a robustness value $\theta \in \mathbb{R}_{\geq 0}$ that indicates how true or false the Boolean v is. The domain of the



(a) Constant input signal.

(b) Staircase input signal.

(c) Pulse input signal.

(d) Sinusoidal input signal.

Figure 2.4: Four parameterization techniques to generate input signals.

Figure 2.5: The simulator for system model $\mathcal{S}$.

VBools is $\mathbb{V} = \mathbb{B} \times \mathbb{R}_{\geq 0}$. The VBools' comparison operator $\leq_v$ corresponds to $\leq$ and is defined as:

$$\leq_v : \mathbb{R} \times \mathbb{R} \to \mathbb{V}$$

$$x \leq_v y = \begin{cases} (\top, y - x) & \text{if } x \leq y \\ (\bot, x - y) & \text{otherwise.} \end{cases} \tag{2.12}$$

where $\top$ and $\bot$ stand for true and false, respectively, and x and y are arguments. The difference between its arguments determines the robustness value for the comparison operator. One of the arguments must change by at least $|x-y|$ for the value of $x \leq y$ to change.

To assess a specification based on several VBools at all time instances, quantitative semantics are needed. Quantitative semantics are used to define the objective function f^{φ}. The trace $\mathbf{x}^s$ is evaluated using f^{φ} and returns a measure of the distance to the specification being falsified. This thesis uses two quantitative semantics: Max [58] and Additive [58], where both can be expressed using VBools. The logical operator AND ($\wedge$) is used for these two semantics.

2.2.3.1 Max Semantics

Using VBools, the Max-and operator $\wedge_{\text{Max}}$ is defined as:

$$\begin{aligned}
(\top, x) \wedge_{\text{Max}} (\top, y) &= (\top, \min(x, y)), \\
(\top, x) \wedge_{\text{Max}} (\bot, y) &= (\bot, y), \\
(\bot, x) \wedge_{\text{Max}} (\top, y) &= (\bot, x), \\
(\bot, x) \wedge_{\text{Max}} (\bot, y) &= (\bot, \max(x, y)).
\end{aligned} \tag{2.13}$$

The first case represents when both VBools are true. To falsify the conjunction, it is sufficient to falsify whichever of the VBools with the lowest robustness value. The second and third cases represent conjunctions of VBools where one is true, and one is false. The conjunction is false with the robustness value given by the false VBools. The fourth case represents when both VBools are false, and thus, the conjunction is false. To make the conjunction true, both VBools must become true. The conjunction's robustness value is determined by which of the VBools appears to be the most difficult to make true, i.e., have the highest robustness value.

The Max-always operator $\square_{\text{Max},[a,b]}$ over the interval $[a, b]$ is defined in terms of the operator $\wedge_{\text{Max}}$ as:

$$\square_{\text{Max},[a,b]}\, \varphi = \bigwedge_{k=a}^{b}{}_{\text{Max}}\ \varphi(k) \tag{2.14}$$

where φ is a finite sequence of VBools defined for all the discrete-time instances in $[a, b]$.

2.2.3.2 Additive Semantics

The Additive-and operator $\wedge_{+}$ is defined as:

$$\begin{aligned}
(\top, x) \wedge_{+} (\top, y) &= \left(\top, \frac{1}{\frac{1}{x} + \frac{1}{y}}\right), \\
(\top, x) \wedge_{+} (\bot, y) &= (\bot, y), \\
(\bot, x) \wedge_{+} (\top, y) &= (\bot, x), \\
(\bot, x) \wedge_{+} (\bot, y) &= (\bot, (x + y)).
\end{aligned} \tag{2.15}$$

The first case represents a conjunction in which both VBools are true. In this case, the robustness value of the conjunction is determined as $1/(1/x+1/y)$. This formula

considers VBools' robustness values and yields a value less than the maximum of x and y. The second and third cases represent conjunctions of VBools where one is true, and one is false. The conjunctions are false with a robustness value given by the false VBools. In the fourth case, both VBools are false. The conjunction is false, and the robustness value is defined as the sum of x and y.

The Additive-always operator $\square_{+,[a,b]}$ over the interval $[a, b]$ is defined in terms of the operator $\wedge_+$ as:

$$\square_{+,[a,b]}\,\varphi = \bigwedge_{k=a}^{b}{}_{+}\ \varphi(k)\#'\delta t \tag{2.16}$$

where φ is a finite sequence of VBools defined for all the discrete-time instances in $[a, b]$, δt is the simulation step time, and $\#'$ is defined as in (2.17). The robustness value is independent of the simulation time due to using $\#'$.

$$\begin{aligned}(\bot, x)\,\#'\,\delta t &= (\bot, x \cdot \delta t)\\ (\top, x)\,\#'\,\delta t &= (\top, x/\delta t)\end{aligned} \tag{2.17}$$

2.2.4 Objective Function Evaluation

Given a trace $\mathbf{x}^s$, a formal specification φ, and a quantitative semantic, the objective function $f^{\varphi}(\mathbf{x}^s)$ is defined such that:

$$f^{\varphi}\left(\mathbf{x}^s\right) = \begin{cases} \theta \text{ if } v = \top \\ -\theta \text{ if } v = \bot\ . \end{cases} \tag{2.18}$$

The evaluation of $f^{\varphi}\left(\mathbf{x}^s\right)$ returns a robustness value, which is a measure of how convincingly the specification is satisfied or how severely it failed. A non-negative robustness value means the specification is satisfied, and a negative value means the specification failed.

2.2.5 Parameter Selector

Either optimization-free or optimization-based methods can be used to search for input signals that falsify specification φ. The objective function f^{φ} is essential in optimization-based falsification since it guides the selection of new input parameters to minimize the objective function. However, no expression of the objective function can be analyzed since the falsification is based on a black-box model. Evaluating the objective function is restricted to simulating the system model. Hence, the optimization is limited to gradient-free methods. In this thesis, the optimization problem consists of minimizing the objective function.

Optimization-free methods select input parameters based on other criteria, e.g., random points or corner points in the domain of the input parameters.

This thesis uses the Hybrid-Corner-Random (HCR) method for optimization-free falsification. Bayesian Optimization (BO), Trust Region Bayesian Optimization (TuRBO), and Line-Search are used for optimization-based falsification. All methods will be described below. In the description below, *point* refers to both input parameters $\mathbf{x}$ and points in space when the exact meaning is clear from the context.

2.2.5.1 Hybrid-Corner-Random

HCR is an optimization-free method that investigates corner points and random points in the search space. An analysis of benchmark problems reveals that this method works well and is, in some cases, more effective than utilizing optimization-based methods [45].

Corner points refer to when all input parameters are at their lower or upper bound in the search space. Thus, for a falsification problem with n input parameters, there are 2^n corner points. Figure 2.6 shows an example with corner points in two dimensions. This figure is inspired by [57]. In this example, the four corner points are (l_1, l_2), (l_1, u_2), (u_1, l_2) and (u_1, u_2).

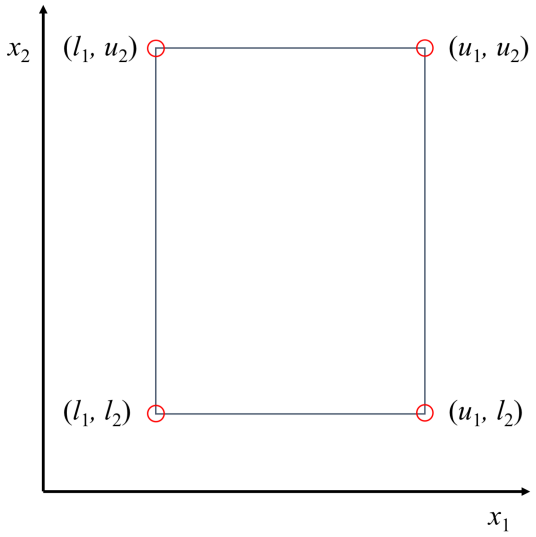


Figure 2.6: An example in two dimensions to show the corner points.

A random generator is used to create random input parameters. This thesis employs a uniform random sampling technique. This means there is an equal probability for all points in the search space to be sampled as the input parameters.

In Algorithm 1 [45], the pseudocode for falsification using HCR is observed. The first step is to generate a set with corner points in the search space (line 1). Then, the algorithm enters a for loop where it starts with selecting a new point $\mathbf{x}$. If the iteration index j is an even integer or the set with corner points is empty, $\mathbf{x}$ is assigned a random point in the search space (line 4). Otherwise, $\mathbf{x}$ is assigned a random element from the corner point set (line 6). This point is removed from the corner point set (line 7). Both sampling processes are performed using uniform probability.

The system model is then simulated with $\mathbf{x}$ as input parameters, and the specification is evaluated with the objective function (line 9). The algorithm terminates if the specification is false, i.e., $f^{\varphi}(\mathbf{x}) < 0$ (line 11). Otherwise, a new random point or corner point will be chosen in the next iteration. Algorithm 1 runs until the specification is falsified or the iteration index reaches max simulations N. The algorithm outputs the tuple $\langle$Falsified / Not Falsified , $\mathbf{x}\rangle$, where the first element describes if the specification is falsified or not, and the second element $\mathbf{x}$ is the last

evaluated input parameter.

Algorithm 1 Hybrid-Corner-Random Falsification

1: $\Upsilon \leftarrow \{\mathbf{x} \in \mathbb{R}^n \mid x_i = l_i \vee x_i = u_i\}$ for $i = 1, \ldots, n$
2: **for** $j = 1, 2, \ldots, N$ **do**
3: **if** j modulo $2 = 0 \vee \Upsilon \subseteq \varnothing$ **then**
4: $\mathbf{x} \sim \mathbb{U}(\mathcal{X})$
5: **else**
6: $\mathbf{x} \sim \mathbb{U}(\Upsilon)$
7: $\Upsilon \leftarrow \Upsilon \backslash \mathbf{x}$
8: **end if**
9: $y \leftarrow f^{\varphi}(\mathbf{x})$
10: **if** $y < 0$ **then**
11: **Return** $\langle$Falsified, $\mathbf{x}\rangle$
12: **end if**
13: **end for**
14: **Return** $\langle$Not Falsified, $\mathbf{x}\rangle$

2.2.5.2 Line-Search Falsification

The optimization method line-search [57] is a gradient-free method that belongs to the family of direct-search methods [59]. These methods involve sequential consideration of trial solutions generated by a particular strategy. Direct-search methods identify new candidate points for future exploration by comparing only objective function values for a given set of points. If an objective function is not continuous or differentiable, direct-search methods can be utilized to find the function optima.

The authors in [45] present LSF, where line-search optimization is adapted for falsification problems. By randomly generating lines in the search space $\mathcal{X}$, LSF combines local search with random exploration. In order to get the minimum value of the objective function along these lines, points on the boundary of the search space are evaluated together with a local search. A new line is generated when no improvements with the previous line are observed after a number of iterations.

LSF is observed in Algorithm 2. The output of LSF is a tuple, where the first element indicates whether or not the specification is falsified. The second element $\mathbf{x}$ is the last evaluated input parameter. The details of LSF will be described in the following.

During the optimization process, LSF requires three points in the search space, and each iteration in LSF generates a new point. At the start of LSF, the first point $\mathbf{x}$ is initialized to the center of the search space (line 1). Also, a counter for the number of completed simulations (line 2) and a dictionary to map the point $\mathbf{x}$ to the corresponding objective function value $f^{\varphi}(\mathbf{x})$ (line 3) are initialized. Then, the function Eval($\mathbf{x}$) is called (line 4). The algorithm for the function Eval is observed in line 17, where the system model is simulated with $\mathbf{x}$ as input parameters, and the corresponding objective function value $f^{\varphi}(\mathbf{x})$ is calculated (line 19). This data is then stored in the dictionary (line 20), and the simulation counter is increased (line

21). If $f^{\varphi}(\mathbf{x}) < 0$, the specification is falsified, and the algorithm terminates since the condition in the while-loop is not satisfied (line 4). If the maximum number of simulations has been reached without falsifying the specification, the point with the lowest objective function value is returned together with Not Falsified (line 6). Otherwise, function H1 (line 38) is called if there has been more than one completed simulation. This function will be introduced later.

Next, the function SelectPoints($\mathbf{x}$) is called (line 11), where the three points, $\mathbf{x}_M$, $\mathbf{x}_L$, and $\mathbf{x}_R$, are generated from a random line that passes through $\mathbf{x}$. The algorithm for the function SelectPoints is observed in line 25. First, a random direction vector is generated (line 26). This direction vector is used with a saturation function g to define a line that goes through $\mathbf{x}$ in the search space. The saturation function is defined as g: $\mathbb{R}^n \to \mathbb{R}^n$ where each dimension i for $1 \leq i \leq n$, of g is defined by g_i as in (2.19).

$$g_i(q) = \begin{cases} u_i, & \text{if } x_i + q\,d_i > u_i \\ x_i + q\,d_i, & \text{if } l_i \leq x_i + q\,d_i \leq u_i \\ l_i, & \text{if } x_i + q\,d_i < l_i \end{cases} \tag{2.19}$$

Let $k \in \{x \in \mathbb{Z} \mid 1 \leq x \leq n\}$ and $q_k^+ \in \mathbb{R}_{>0}$ represent the smallest positive value such that for at least k different dimensions (2.20) is fulfilled.

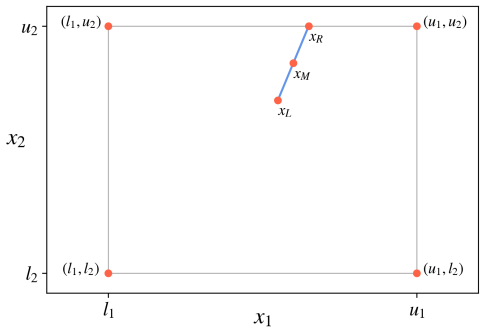
$$g_i\left(q_k^+\right) = u_i \text{ or } g_i\left(q_k^+\right) = l_i. \tag{2.20}$$

Let $j \in \{x \in \mathbb{Z} \mid 0 \leq x \leq n\}$ and $q_j^- \in \mathbb{R}_{<0}$ represent the largest negative value such that for at least j different dimensions (2.21) is fulfilled.

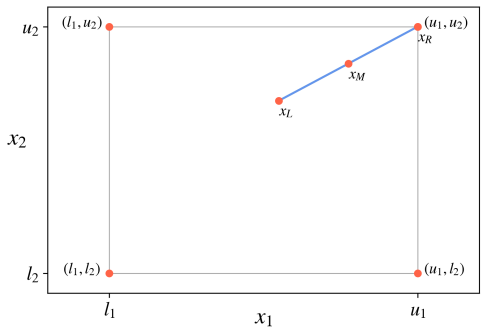
$$g_i\left(q_j^-\right) = u_i \text{ or } g_i\left(q_j^-\right) = l_i. \tag{2.21}$$

Note that the constants k and j are predetermined before Algorithm 2 starts, and the same constants are used during the entire execution. If j is equal to zero, then $\mathbf{x}_L = \mathbf{x}$, $\mathbf{x}_R = g\left(q_k^+\right)$ and $\mathbf{x}_M$ is the midpoint between $\mathbf{x}_L$ and $\mathbf{x}_R$ (line 28-30). If j is greater than zero, then $\mathbf{x}_L = g\left(q_j^-\right)$, $\mathbf{x}_R = g\left(q_k^+\right)$ and $\mathbf{x}_M = \mathbf{x}$ (line 32-34).

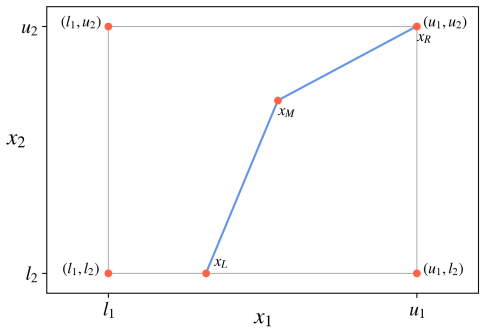
In Figure 2.7, four cases of calling SelectPoints($\mathbf{x}$) for a problem in two dimensions are observed. Note, the cases are established with the same random direction vector and $\mathbf{x}$, but with different values for k and j. In Figure 2.7a, $k = 1$ and $j = 0$. In this graph, the generated line starts from the point $\mathbf{x}_L$ and ends at the point $\mathbf{x}_R$, where it cuts off at the upper bound of the second dimension. In Figure 2.7b, $k = 2$ and $j = 0$. In this graph, the generated line starts from the point $\mathbf{x}_L$ and ends up at the point $\mathbf{x}_R$ where it cuts off at the upper bounds of both the first and second dimension, i.e., the corner point (u_1, u_2). In Figure 2.7c, $k = 2$ and $j = 1$, which generate two line segments. The first line segment starts from the point $\mathbf{x}_M$ and ends at the point $\mathbf{x}_R$, where it cuts off at the upper bounds of the first and second dimensions. The second line segment starts from $\mathbf{x}_M$ and ends at $\mathbf{x}_L$, where it cuts off at the lower bound of the second dimension. In Figure 2.7d, $k = 2$ and $j = 2$, which generate two line segments. The first line segment from $\mathbf{x}_M$ to $\mathbf{x}_R$ is identical to the corresponding line segment in Figure 2.7c since both k and the random direction vector are identical for the two cases. The second line segment



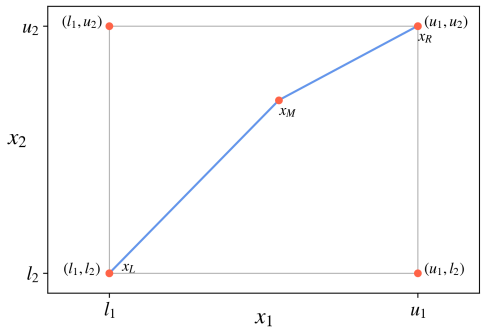
(a) SelectPoints($\mathbf{x}$) with $k = 1$ and $j = 0$.



(b) SelectPoints($\mathbf{x}$) with $k = 2$ and $j = 0$.



(c) SelectPoints($\mathbf{x}$) with $k = 2$ and $j = 1$.



(d) SelectPoints($\mathbf{x}$) with $k = 2$ and $j = 2$.

Figure 2.7: Examples of function SelectPoints($\mathbf{x}$) in two dimensions for four different configurations of k and j.

starts from the point $\mathbf{x}_M$ and ends up at the point $\mathbf{x}_L$ where it cuts off at the lower bounds of both the first and the second dimension, i.e., the corner point (l_1, l_2).

The authors in [45] deem that it is not possible to generalize what values for k and j perform best because it depends on the given system and specification. However, some characteristics of the lines generated can be made. Using $j = 0$ may result in short lines that do not guide the process toward falsification or may cause it to become stuck in a local area. On the other hand, using $j > 0$ generates lines that extend between boundaries in the search space, which increases the likelihood of finding a point with a lower objective function value.

With the three points, $\mathbf{x}_M, \mathbf{x}_L$, and $\mathbf{x}_R$ determined, the function FalsifyLine is called (line 13). The algorithm for the function FalsifyLine is observed in line 43. In this function, the objective function $f^{\varphi}(\mathbf{x})$ is minimized along the line segments with the line-search optimization method. Thus, the local search loop in LSF is represented by this function.

First, a counter for the number of iterations in the local search loop without improvement is initialized (line 44). Then, the system model is simulated and evaluated for the three points $\mathbf{x}_M, \mathbf{x}_L$, and $\mathbf{x}_R$ by calling the function Eval (lines 46-48). FalsifyLine terminates if the specification is falsified for any of these points. Otherwise, the algorithm begins the local search by looking for new points with lower objective function values.

FalsifyLine continues with a conditional statement with three cases depending on which of the three points, $\mathbf{x}_M, \mathbf{x}_L,$ and $\mathbf{x}_R$, that has the lowest objective function value. In each case, a point $\mathbf{x}_{new}$ is generated that is the midpoint between $\mathbf{x}_M$ and $\mathbf{x}_L$, or $\mathbf{x}_M$ and $\mathbf{x}_R$. The point $\mathbf{x}_{new}$ will be used for simulating and evaluating the system. Then, one of the three points, $\mathbf{x}_M, \mathbf{x}_L,$ and $\mathbf{x}_R$ will be assigned the value of $\mathbf{x}_{new}$. For the next iteration in the local search loop, the optimization will be performed on a shorter line segment than the initial line segment, as defined by the selected endpoints.

The first case occurs when the point $\mathbf{x}_L$ has the lowest objective function value of the three points (lines 49-54). If so, $\mathbf{x}_{new}$ is assigned to the midpoint between $\mathbf{x}_L$ and $\mathbf{x}_M$:

$$f^{\varphi}(\mathbf{x}_L) < f^{\varphi}(\mathbf{x}_M) \text{ and } f^{\varphi}(\mathbf{x}_L) < f^{\varphi}(\mathbf{x}_R)$$
$$\mathbf{x}_{new} = \frac{\mathbf{x}_L + \mathbf{x}_M}{2}. \tag{2.22}$$

Then, $\mathbf{x}_{new}$ will be simulated and evaluated, and if it has a lower objective function value than the point $\mathbf{x}_L$, this yields $\mathbf{x} = \mathbf{x}_{new}$. Furthermore, $\mathbf{x}_R$ and $\mathbf{x}_M$ will be assigned to new values as:

$$\mathbf{x}_R = \mathbf{x}_M \text{ and } \mathbf{x}_M = \mathbf{x}_{new}. \tag{2.23}$$

Thus, in the next iteration of the local search loop, line segments connecting the points $\mathbf{x}_L$, $\mathbf{x}_{new}$ and $\mathbf{x}_M$ from the current iteration will be used.

The second case occurs when the point $\mathbf{x}_R$ has the lowest objective function value of the three points (lines 55-60). If so, $\mathbf{x}_{new}$ is assigned to the midpoint between $\mathbf{x}_R$ and $\mathbf{x}_M$:

$$f^{\varphi}(\mathbf{x}_R) < f^{\varphi}(\mathbf{x}_M) \text{ and } f^{\varphi}(\mathbf{x}_R) < f^{\varphi}(\mathbf{x}_L)$$
$$\mathbf{x}_{new} = \frac{\mathbf{x}_R + \mathbf{x}_M}{2}. \tag{2.24}$$

Then, $\mathbf{x}_{new}$ will be simulated and evaluated, and if it has a lower objective function value than the point $\mathbf{x}_R$, this yields $\mathbf{x} = \mathbf{x}_{new}$. Furthermore, $\mathbf{x}_L$ and $\mathbf{x}_M$ will be assigned to new values as:

$$\mathbf{x}_L = \mathbf{x}_M \text{ and } \mathbf{x}_M = \mathbf{x}_{new}. \tag{2.25}$$

Thus, in the next iteration of the local search loop, line segments connecting the points $\mathbf{x}_R$, $\mathbf{x}_{new}$ and $\mathbf{x}_M$ from the current iteration will be used.

The third case occurs when the point $\mathbf{x}_M$ has the lowest objective function value of the three points (lines 61-76):

$$f^{\varphi}(\mathbf{x}_M) \leq f^{\varphi}(\mathbf{x}_L) \text{ and } f^{\varphi}(\mathbf{x}_M) \leq f^{\varphi}(\mathbf{x}_R).$$

In this case, it needs to be decided whether to continue the search on the line segment from $\mathbf{x}_L$ to $\mathbf{x}_M$ or from $\mathbf{x}_R$ to $\mathbf{x}_M$. The line segment with a longer Euclidean distance will be selected for the local search. The authors of [45] propose using the longest

line segment since it has the greatest uncertainty in terms of being least explored. When $\|\mathbf{x}_L - \mathbf{x}_M\|_2 \geq \|\mathbf{x}_M - \mathbf{x}_R\|_2$, the points are updated as:

$$\mathbf{x}_{new} = \frac{\mathbf{x}_L + \mathbf{x}_M}{2}$$
$$\begin{cases} \mathbf{x}_R = \mathbf{x}_M \text{ and } \mathbf{x}_M = \mathbf{x}_{\text{new}} , & \text{if } f^{\varphi}(\mathbf{x}_{\text{new}}) < f^{\varphi}(\mathbf{x}_M) \\ \mathbf{x}_L = \mathbf{x}_{\text{new}} , & \text{otherwise.} \end{cases} \tag{2.26}$$

When $\|\mathbf{x}_L - \mathbf{x}_M\|_2 < \|\mathbf{x}_M - \mathbf{x}_R\|_2$, the points are updated as:

$$\mathbf{x}_{new} = \frac{\mathbf{x}_R + \mathbf{x}_M}{2}$$
$$\begin{cases} \mathbf{x}_L = \mathbf{x}_M \text{ and } \mathbf{x}_M = \mathbf{x}_{\text{new}} , & \text{if } f^{\varphi}(\mathbf{x}_{\text{new}}) < f^{\varphi}(\mathbf{x}_M) \\ \mathbf{x}_R = \mathbf{x}_{\text{new}} , & \text{otherwise.} \end{cases} \tag{2.27}$$

The points that are not explicitly updated in (2.26)-(2.27) will keep their previous values.

The variable m (line 44) is used to count the number of consecutive iterations inside the local search loop without finding a point $\mathbf{x}$ with a lower objective function value than all of the three points $\mathbf{x}_M, \mathbf{x}_L,$ and $\mathbf{x}_R$. If a point $\mathbf{x}$ is found with a lower objective function value than all of the three points, m is set to zero (line 79). The local search is terminated when m has reached M iterations. The function FalsifyLine returns the point $\mathbf{x}$. Next, the LSF method continues to the main script (line 14).

In function H1 (line 38), the objective function value of $\mathbf{x}_{old}$ is compared to $\mathbf{x}$. If $f^{\varphi}(\mathbf{x}_{old}) < f^{\varphi}(\mathbf{x})$, FalsifyLine could not find a point $\mathbf{x}$ with a lower objective function value than $\mathbf{x}_{old}$. In this case, $\mathbf{x}$ is assigned to be the point with the second lowest objective function value of the last call of FalsifyLine in order to force the algorithm to never return to the same $\mathbf{x}$. The function H1 aims to avoid getting stuck in a local minimum [45].

Algorithm 2 Line-Search Falsification

$\mathbf{x} \leftarrow \left\{\mathbf{x} \in \mathbb{R}^n \mid x_i = \frac{l_i+u_i}{2}\right\}$ for $i = 1, \ldots, n$
global $i \leftarrow 0$
global $D \leftarrow$ empty dictionary
while Eval($\mathbf{x}$) ≥ 0 **do**
    if $i \geq N$ **then**
        Return $\langle$Not Falsified, $\mathbf{x}\rangle$
    end if
    if $i > 1$ **then**
        $\mathbf{x} \leftarrow \mathrm{H1}(\mathbf{x}_{old}, \mathbf{x})$
    end if
    $(\mathbf{x}_L, \mathbf{x}_M, \mathbf{x}_R) \leftarrow$ SelectPoints($\mathbf{x}$)
    $\mathbf{x}_{old} \leftarrow \mathbf{x}$
    $\mathbf{x} \leftarrow$ FalsifyLine($\mathbf{x}_L, \mathbf{x}_M, \mathbf{x}_R, \mathbf{x}$)
end while
Return $\langle$Falsified, $\mathbf{x}\rangle$

---

function Eval($\mathbf{x}$)
if $\neg(\mathbf{x} \in keys(D))$ **then**
    $y \leftarrow f^{\varphi}(\mathbf{x})$
    $D[\mathbf{x}] \leftarrow y$
    $i \leftarrow i + 1$
end if
Return $D[\mathbf{x}]$

---

function SelectPoints($\mathbf{x}$)
$\mathbf{d} \leftarrow \mathbb{R}^n \sim \mathcal{U}(\Omega), \quad \Omega = \{\mathbf{x} \in \mathbb{R}^n \mid x_i \neq 0\}$
if $j = 0$ **then**
    $\mathbf{x}_L \leftarrow \mathbf{x}$
    $\mathbf{x}_R \leftarrow g(q_k^+)$
    $\mathbf{x}_M \leftarrow \frac{\mathbf{x}_L+\mathbf{x}_R}{2}$
else if j > 0 **then**
    $\mathbf{x}_L \leftarrow g(q_j^-)$
    $\mathbf{x}_R \leftarrow g(q_k^+)$
    $\mathbf{x}_M \leftarrow \mathbf{x}$
end if
Return $\mathbf{x}_L$, $\mathbf{x}_M$, $\mathbf{x}_R$

---

function H1($\mathbf{x}$)
if $f^{\varphi}(\mathbf{x}_{old}) < f^{\varphi}(\mathbf{x})$ **then**
    $\mathbf{x} \leftarrow$ the second point with lowest objective function value of the last call to
    FalsifyLine
end if

43: **function** FALSIFYLINE($\mathbf{x}_L, \mathbf{x}_M, \mathbf{x}_R, \mathbf{x}$)
44: $m \leftarrow 0$
45: **while** $m < M$ **do**
46: **if** EVAL($\mathbf{x}_L$) < 0 **then Return** $\mathbf{x}_L$ **end if**
47: **if** EVAL($\mathbf{x}_R$) < 0 **then Return** $\mathbf{x}_R$ **end if**
48: **if** EVAL($\mathbf{x}_M$) < 0 **then Return** $\mathbf{x}_M$ **end if**
49: **if** EVAL($\mathbf{x}_L$) $<$ EVAL($\mathbf{x}_R$) $\wedge$ EVAL($\mathbf{x}_L$) $<$ EVAL($\mathbf{x}_L$) **then**
50: $\mathbf{x}_{new} \leftarrow \frac{\mathbf{x}_L + \mathbf{x}_M}{2}$
51: **if** EVAL($\mathbf{x}_{new}$) $<$ EVAL($\mathbf{x}_L$) **then**
52: $\mathbf{x} \leftarrow \mathbf{x}_{new}$
53: **end if**
54: $\mathbf{x}_R \leftarrow \mathbf{x}_M$, $\mathbf{x}_M \leftarrow \mathbf{x}_{new}$
55: **else if** EVAL($\mathbf{x}_R$) $<$ EVAL($\mathbf{x}_M$) $\wedge$ EVAL($\mathbf{x}_R$) $<$ EVAL($\mathbf{x}_L$) **then**
56: $\mathbf{x}_{new} \leftarrow \frac{\mathbf{x}_R + \mathbf{x}_M}{2}$
57: **if** EVAL($\mathbf{x}_{new}$) $<$ EVAL($\mathbf{x}_R$) **then**
58: $\mathbf{x} \leftarrow \mathbf{x}_{new}$
59: **end if**
60: $\mathbf{x}_L \leftarrow \mathbf{x}_M$, $\mathbf{x}_M \leftarrow \mathbf{x}_{new}$
61: **else**
62: **if** $\|\mathbf{x}_L - \mathbf{x}_M\|_2 \geq \|\mathbf{x}_M - \mathbf{x}_R\|_2$ **then**
63: $\mathbf{x}_{new} \leftarrow \frac{\mathbf{x}_L + \mathbf{x}_M}{2}$
64: **if** EVAL($\mathbf{x}_{new}$) $<$ EVAL($\mathbf{x}_L$) **then**
65: $\mathbf{x} \leftarrow \mathbf{x}_{new}$, $\mathbf{x}_R \leftarrow \mathbf{x}_M$, $\mathbf{x}_M \leftarrow \mathbf{x}_{new}$
66: **else**
67: $\mathbf{x}_L \leftarrow \mathbf{x}_{new}$
68: **end if**
69: **else**
70: $\mathbf{x}_{new} \leftarrow \frac{\mathbf{x}_R + \mathbf{x}_M}{2}$
71: **if** EVAL($\mathbf{x}_{new}$) $<$ EVAL($\mathbf{x}_M$) **then**
72: $\mathbf{x} \leftarrow \mathbf{x}_{new}$, $\mathbf{x}_L \leftarrow \mathbf{x}_M$, $\mathbf{x}_M \leftarrow \mathbf{x}_{new}$
73: **else**
74: $\mathbf{x}_R \leftarrow \mathbf{x}_{new}$
75: **end if**
76: **end if**
77: **end if**
78: **if** $\mathbf{x} = \mathbf{x}_{new}$ **then**
79: $m \leftarrow 0$
80: **else**
81: $m \leftarrow m + 1$
82: **end if**
83: **end while**
84: **Return x**

2.2.5.3 Bayesian Optimization

BO [60] is a powerful and efficient technique used for global optimization of objective functions. It is particularly suited for scenarios where direct analytical evaluations of the objective function are either computationally expensive or infeasible, such as tuning hyperparameters in machine learning models or parameter estimation in simulation-based models. In BO, the function can be non-convex and multimodal, i.e., a function with multiple optima. Also, BO tolerates stochastic noise in function evaluations.

BO is based on Bayes' theorem [61], which is observed in (2.28). In this theorem, initial beliefs (prior probabilities) are updated based on new information (likelihood) to obtain updated beliefs (posterior probabilities). $P(A \mid B)$ is the probability of event A occurring, given that event B has occurred. $P(B \mid A)$ is the probability of event B occurring, given that event A has occurred. $P(A)$ and $P(B)$ are the prior probabilities of event A and B, respectively, i.e., initial beliefs in the absence of any evidence.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{2.28}$$

The core idea behind BO is to model the unknown objective function f^{φ}: $\mathcal{X} \to \mathbb{R}$ as a probabilistic surrogate. This surrogate model captures an estimate of the function, including uncertainty in unexplored regions of the search space $\mathcal{X}$.

The optimization process begins with a small number of initial evaluations of the objective function, usually chosen through Latin hypercube sampling [62] or random search. These initial observations are used to build the initial surrogate model.

The BO algorithm iteratively selects the next point in the search space using an acquisition function. This function balances the exploration-exploitation trade-off and provides a mathematical framework for reasoning about the uncertainty and potential benefits of selecting different points. Exploration refers to selecting points in regions with high uncertainty in the surrogate model, allowing the algorithm to gain new information about the objective function. Conversely, exploitation aims to choose points with low predicted objective function values to improve the overall optimization. Commonly used acquisition functions include Probability of Improvement, Expected Improvement , and Lower Confidence Bound (LCB).

Once the next evaluation point is chosen based on the acquisition function, the objective function is evaluated at that point, and the surrogate model is updated with this new information. This process then repeats for a predefined number of iterations or until convergence criteria are met.

In this thesis, Gaussian processes ($\mathcal{GP}$) are used to model the objective function. Given a set of input data points $\mathbf{x}$ and their corresponding objective function values $f^{\varphi}(\mathbf{x})$, the $\mathcal{GP}$ model predicts the function values at any input point $\mathbf{x}^*$. $\mathcal{GP}$ is a stochastic process composed of a finite number of random variables with multivariate Gaussian distributions. The objective function f^{φ} modeled using $\mathcal{GP}$ is defined as:

$$f^{\varphi} \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2.29}$$

where $\mu(\mathbf{x})$ is the mean approximation of $f^{\varphi}(\mathbf{x})$. The covariance function $k(\mathbf{x}, \mathbf{x}')$ captures the uncertainty in the surrogate model. This function determines how ob-

jective function values at the different points $\mathbf{x}$ and $\mathbf{x}'$ are correlated.

There are different kernels used to define the covariance function. The kernels used in this thesis are the squared-exponential kernel and Matérn kernel. The squared-exponential kernel is defined as:

$$k_{sq}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\frac{\left|\mathbf{x}_i - \mathbf{x}_j\right|^2}{2\kappa^2}\right) \tag{2.30}$$

where κ is the length scale of the kernel. The Matérn kernel is defined as:

$$k_{\text{Matérn}}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\left|\mathbf{x}_i - \mathbf{x}_j\right|\sqrt{2\nu}}{\kappa}\right)^{\nu} K_\nu\left(\frac{\left|\mathbf{x}_i - \mathbf{x}_j\right|\sqrt{2\nu}}{\kappa}\right) \tag{2.31}$$

where ν is a smoothness parameter, K_ν is a modified Bessel function, and $\Gamma(\nu)$ is the gamma function.

Let's predict the objective function value at a new point $\mathbf{x}^*$. Given M number of evaluations of the objective function $f^{\varphi}(\mathbf{x})$, we have:

$$\boldsymbol{f} = \begin{bmatrix} f^{\varphi}(\mathbf{x}_1) & f^{\varphi}(\mathbf{x}_2) & \cdots & f^{\varphi}(\mathbf{x}_M) \end{bmatrix}^{\top}$$

$$\boldsymbol{K}(\mathbf{X}, \mathbf{X}) = \begin{bmatrix} k\left(\mathbf{x}_1, \mathbf{x}_1\right) & k\left(\mathbf{x}_1, \mathbf{x}_2\right) & \cdots & k\left(\mathbf{x}_1, \mathbf{x}_M\right) \\ k\left(\mathbf{x}_2, \mathbf{x}_1\right) & k\left(\mathbf{x}_2, \mathbf{x}_2\right) & \cdots & k\left(\mathbf{x}_2, \mathbf{x}_M\right) \\ \vdots & \vdots & \ddots & \vdots \\ k\left(\mathbf{x}_M, \mathbf{x}_1\right) & k\left(\mathbf{x}_M, \mathbf{x}_2\right) & \cdots & k\left(\mathbf{x}_M, \mathbf{x}_M\right) \end{bmatrix}.$$

The mean and variance at any point $\mathbf{x}^*$ are calculated as:

$$\mu(\mathbf{x}^*) = \boldsymbol{K}(\mathbf{x}^*, \mathbf{X})\, \boldsymbol{K}(\mathbf{X}, \mathbf{X})^{-1} \boldsymbol{f}, \tag{2.32}$$

$$\sigma^2(\mathbf{x}^*) = k\left(\mathbf{x}^*, \mathbf{x}^*\right) - \boldsymbol{K}(\mathbf{x}^*, \mathbf{X})\, \boldsymbol{K}(\mathbf{X}, \mathbf{X})^{-1} \boldsymbol{K}(\mathbf{X}, \mathbf{x}^*) \tag{2.33}$$

where,

$$\boldsymbol{K}(\mathbf{X}, \mathbf{x}^*) = \begin{bmatrix} k\left(\mathbf{x}_1, \mathbf{x}^*\right) \\ k\left(\mathbf{x}_2, \mathbf{x}^*\right) \\ \vdots \\ k\left(\mathbf{x}_M, \mathbf{x}^*\right) \end{bmatrix}$$

$$\boldsymbol{K}(\mathbf{x}^*, \mathbf{X}) = \begin{bmatrix} k\left(\mathbf{x}^*, \mathbf{x}_1\right) & k\left(\mathbf{x}^*, \mathbf{x}_2\right) & \cdots & k\left(\mathbf{x}^*, \mathbf{x}_M\right) \end{bmatrix}.$$

The acquisition functions $\alpha\left(\cdot\right)$ used in this thesis are LCB [63] and Thompson sampling (TS) [64], which will be introduced in the following.

LCB uses the mean and covariance from the surrogate model to select the next point $\mathbf{x}^*$ to sample:

$$\alpha_{\text{LCB}}\left(\mathbf{x}^*\right) \in \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmin}}\, \mu(\mathbf{x}) - \gamma \cdot \sigma(\mathbf{x}) \tag{2.34}$$

where the parameter $\gamma \in \mathbb{R}_{\geq 0}$ determines the trade-off between exploration and exploitation. Small values of γ mean more exploitation, while large values mean more exploration.

In TS, the multi-armed bandit problem is addressed, where an agent faces a set of arms (choices) with unknown reward distributions and seeks to maximize the cumulative reward obtained over time. TS employs a Bayesian approach by assigning prior distributions to the unknown reward distributions of each arm. At each time step, the algorithm samples from these priors, and the arm with the highest sampled reward is chosen for exploration. This process updates the priors based on observed rewards, allowing the algorithm to efficiently balance exploration (sampling uncertain arms) and exploitation (selecting arms with potentially high rewards). The basic idea behind TS for BO involves drawing a function f from the surrogate model posterior. It then suggests this function's optimum point $\mathbf{x}$. This process is repeated independently for multiple suggestions. The exploration-exploitation trade-off is naturally handled by the stochasticity in sampling.

An example of BO for the function $f(x) = x \cdot \sin(x)$ is shown in Figure 2.8. In this example, there are three initial points, and the acquisition function is $\alpha_{LCB}(x) = \mu(x) - 2\,\sigma(x)$. In the top graph, a fourth point is identified by the acquisition function. In the middle graph, the surrogate model is updated with the fourth point, and a fifth point is identified. The surrogate model is updated in the bottom graph with the fifth point. The graphs illustrate the mean and covariance (implicit through the standard deviation) of the objective function as estimated by the surrogate model. Although the true objective function is shown as a yellow line, it is unknown in practice.

Pseudocode for falsification using BO is given in Algorithm 3 [46]. The algorithm outputs the tuple $\langle$Falsified / Not Falsified , $\mathbf{x}\rangle$ where the first element describes if the specification is falsified or not, and the second element $\mathbf{x}$ is the last evaluated input parameter.

The falsification algorithm begins by sampling a random point $\mathbf{x}$ from the search space (line 3). The system is simulated using $\mathbf{x}$ as input parameter, and the specification is evaluated (line 4). The sample set $\mathcal{D}$ is augmented with the pair $(\mathbf{x}, f^{\varphi}(\mathbf{x}))$ (line 8). This first for-loop continues until the total number of evaluations M has been reached. Then, the probabilistic surrogate model is fitted on the current sample set (line 11). By optimizing the acquisition function, the next point $\mathbf{x}$ is selected (line 12) and used to simulate and evaluate the system (line 13). The sample set $\mathcal{D}$ is updated with the new data (line 17). This second for-loop continues until the total number of evaluations N has been reached. If a point $\mathbf{x}$ falsifies the specification, i.e., $f^{\varphi}(\mathbf{x}) < 0$, the algorithm terminates early (lines 6 and 15).

2.2.5.4 Trust Region Bayesian Optimization

It is challenging to apply BO to high-dimensional problems with thousands of observations. Additionally, on challenging problems, BO frequently falls short of competing with other optimization methods. According to the authors in [53], this is caused by an inherent homogeneity in global probabilistic models and an overemphasis on exploration that follows from global acquisition. The authors in [53] propose to discard global surrogate modeling to overcome these difficulties. The global optimization is accomplished by maintaining a number of independent local models, each of which is involved in a separate local optimization run. In this method, called TuRBO, global optimization is achieved by maintaining many local

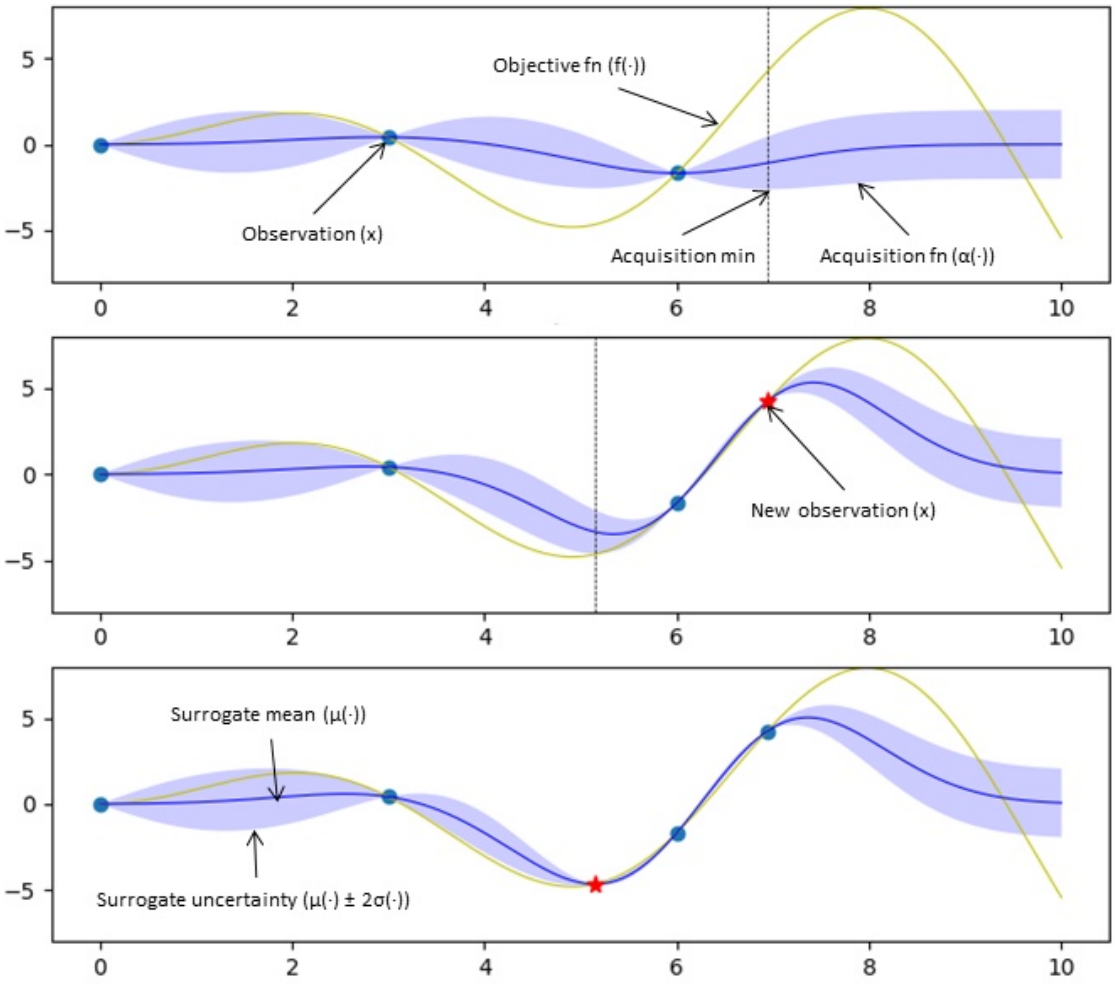


Figure 2.8: Bayesian Optimization for $f(x) = x \cdot \sin(x)$.

models concurrently and allocating samples using a multi-armed bandit approach. This results in an efficient acquisition strategy that directs samples to promising local optimization runs.

The TuRBO algorithm establishes a $\mathcal{GP}$ surrogate model inside a trust region (TR) to achieve principled local optimization. The best solution found so far in a TR is denoted $\mathbf{x}^*$ and serves as the center of the TR, which can be a sphere, polytope, or hyperrectangle. At the beginning of the TuRBO algorithm, the base side length L for the TR is initiated to L_{init}. The actual side length L_i for the ith dimension in search space (where $i = 1, \ldots, n$) is determined from this base side length by rescaling according to its lengthscale ζ_i in the surrogate model. The side length L_i is defined as:

$$L_i = \frac{\zeta_i L}{\left(\prod_{j=1}^{n} \zeta_j\right)^{1/n}}. \tag{2.35}$$

The total volume of a TR is L^n. Running local BO with a large enough L for the TR to encompass the entire search space would be similar to standard global BO.

Algorithm 3 Bayesian Optimization Falsification

$\mathcal{D}_0 \leftarrow \emptyset$
for $i \leftarrow 1$ **to** M **do**
  $\mathbf{x} \sim \mathbb{U}(\mathcal{X})$
  $y \leftarrow f^{\varphi}(\mathbf{x})$
  if $y < 0$ **then**
    Return $\langle$Falsified, $\mathbf{x}\rangle$
  end if
  $\mathcal{D}_0 \leftarrow \{\mathcal{D}_0, (\mathbf{x}, y)\}$
end for
for $i \leftarrow 1$ **to** N **do**
  $p(y \mid \mathcal{D}) \leftarrow \mathcal{GP}\left(y; \mu_{y|\mathcal{D}}, K_{y|\mathcal{D}}\right)$
  $\mathbf{x} \leftarrow \arg\min_{\mathbf{x}\subseteq\mathcal{X}} \alpha\left(\mathbf{x}; \mathcal{D}_{i-1}\right)$
  $y \leftarrow f^{\varphi}(\mathbf{x})$
  if $y < 0$ **then**
    Return $\langle$Falsified, $\mathbf{x}\rangle$
  end if
  $\mathcal{D}_i \leftarrow \{\mathcal{D}_{i-1}, (\mathbf{x}, y)\}$
end for
Return $\langle$Not Falsified, $\mathbf{x}\rangle$

Thus, the TR should be small enough to ensure that the local model within the TR is accurate and large enough to contain good solutions. An example of a spherical TR for the function $f(x_1, x_2) = 0.5\,x_1^2 + 4.5\,x_2^2$ is observed in the lower right of Figure 2.9.

When the optimizer does not find a point $\mathbf{x}$ within a TR with lower objective function value than $\mathbf{x}^*$ after ψ_{fail} consecutive iterations, the size of the TR is halved, i.e., $L \leftarrow L/2$. After ψ_{succ} consecutive iterations with finding a point $\mathbf{x}$ with lower objective function value than $\mathbf{x}^*$ in each iteration, the size of the TR is doubled, i.e., $L \leftarrow \min\{L_{\max}, 2L\}$. If the size of the TR is less than L_{min}, the current TR is discarded, and a new TR with the base side length L_{init} is initialized.

The TuRBO algorithm maintains m TRs simultaneously. Each trust region TR_ℓ with $\ell \in \{1, \ldots, m\}$ utilizes an independent local $\mathcal{GP}$ model. In each iteration k, TS is used to select a batch of q candidates points $\left\{\mathbf{x}_1^{(k)}, \ldots, \mathbf{x}_q^{(k)}\right\}$ drawn from the union of all TRs, and all local optimization problems for which candidates were drawn are updated. The i-th candidate (where $i \in \{1, \ldots, q\}$) is selected as:

$$\mathbf{x}_i^{(k)} \in \operatorname*{argmin}_{\ell} \operatorname*{argmin}_{\mathbf{x}\in \text{TR}_\ell} f_\ell^{(i)} \text{ where } f_\ell^{(i)} \sim \mathcal{GP}_\ell^{(k)}\left(\mu_\ell(\mathbf{x}), k_\ell\left(\mathbf{x}, \mathbf{x}'\right)\right). \tag{2.36}$$

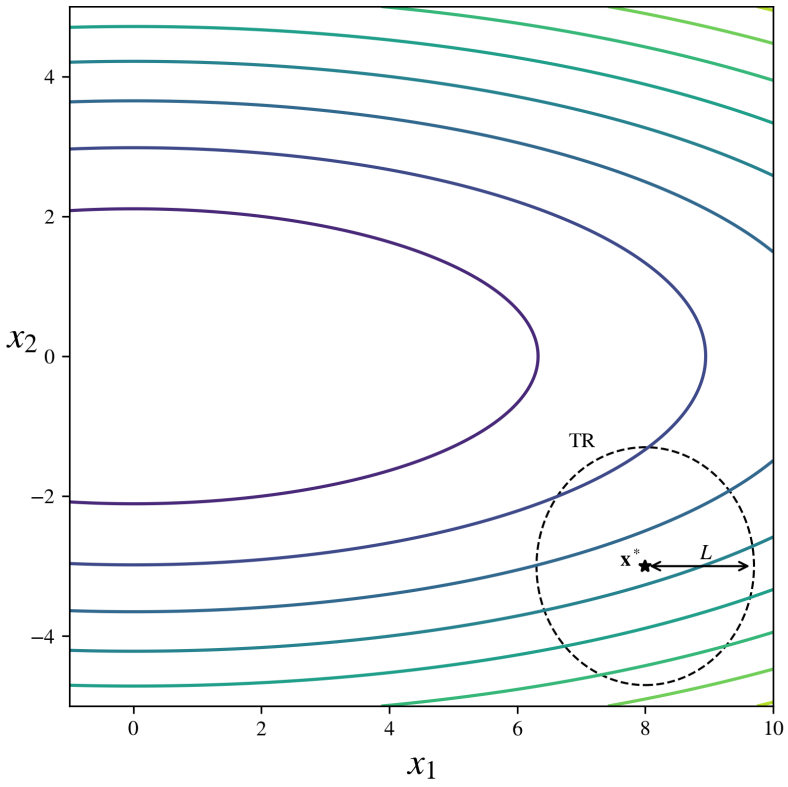


Figure 2.9: Contour plot for $f(x_1, x_2) = 0.5\,x_1^2 + 4.5\,x_2^2$ with a spherical trust region located in the lower right of the graph.

3
Research Approach

This chapter outlines the adopted research approach grounded in the design-science research paradigm. It begins with an overview of the design-science research methodology and a justification for its selection. Subsequently, the chapter proceeds to the research procedure, and based on this, an exploration of potential threats to validity is presented.

3.1 Research Methodology

This thesis reports on a six-month (September 2022 - February 2023) study at Saab Kockums. The thesis was conducted following the framework for design-science research in the information systems discipline proposed by Hevner et al. [65, 66]. Design-science is a problem-solving paradigm that seeks to create and evaluate artifacts to solve identified organizational problems. The artifacts can be methods (algorithms and practices) and instantiations (implemented and prototype systems).

As the intention of this thesis is to develop a system for automated testing, the design-science framework is applicable. An overview of the framework used in this thesis is observed in Figure 3.1.

The *environment* defines the problem space in which resides the phenomena of interest, i.e., test automation for SCMS. The environment comprises the people, the organization, and their existing or planned technologies (detailed in Section 1.2). In the environment are the goals, tasks, problems, and opportunities that define business needs as people within Saab Kockums perceive them. Research relevance is ensured through framing research activities to address business needs. The *knowledge base* includes the scientific theories and engineering methods that provide the foundations for rigorous development of AuTS (detailed in Section 1.3 and Section 2.2). During the *research*, the development of AuTS iterates rapidly between the construction of the system and evaluation with feedback to refine the system further (detailed in Chapter 4). The contributions in this thesis are applied to the business needs at Saab Kockums and add to the content of the knowledge base for further research and practice (detailed in Section 1.7).

3.2 Research Procedure

The thesis was conducted in three consecutive steps: collecting business needs and applicable knowledge; designing, assessing, and refining AuTS; and evaluating its application for SCMS.

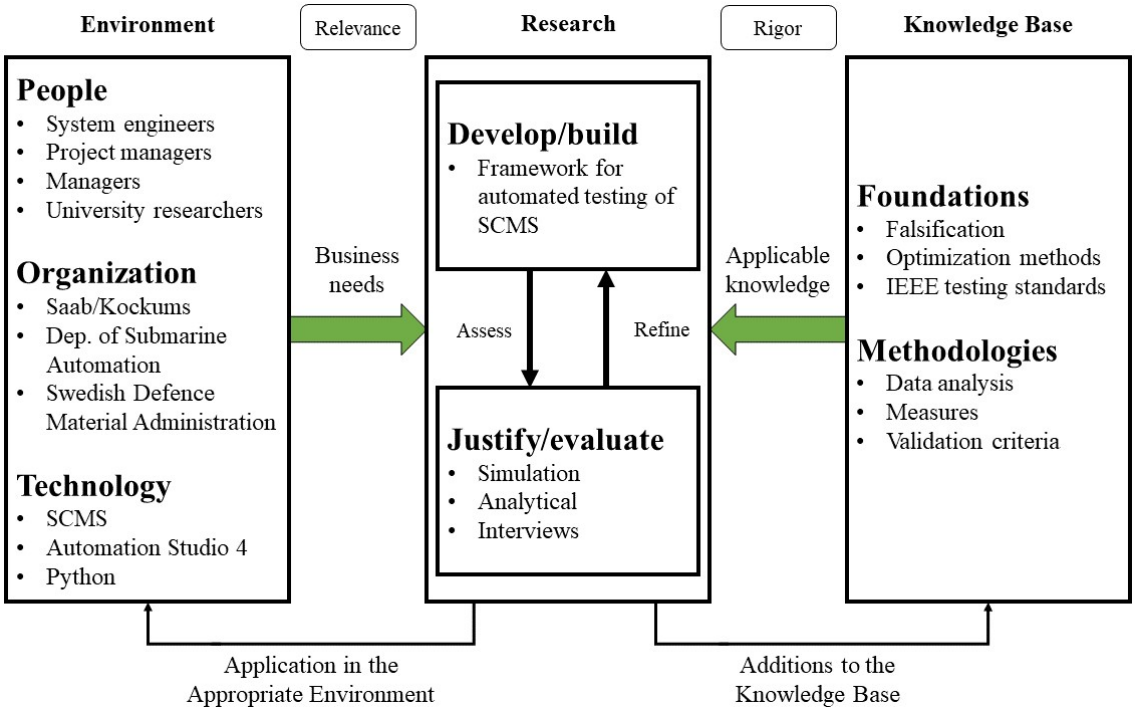


Figure 3.1: Overview of the framework for design-science research.

3.2.1 Business Needs and Applicable Knowledge

To extract concrete business needs and applicable knowledge for AuTS, an investigation of the demands, challenges, and opportunities during testing of the SCMS was performed. Triangulation [67] plays a crucial role in enhancing the accuracy and bolstering the validity of the research. It involves approaching the subject of study from various angles, offering a more comprehensive view of the testing of SCMS. To this end, the following steps were performed: observing systems engineers conducting testing activities; reading source code and related documentation for different ship systems in SCMS; unstructured and semi-structured interviews with various stakeholders; literature review on topics related to testing; data analysis of collected data.

3.2.1.1 Observations

Several observations [68] were conducted when engineers manually tested various ship systems in SCMS. The purpose was to get direct experience of the difficulties they confronted while performing testing. A benefit of observations is that they deal with actual behavior rather than reported behavior. This is important because there may be a significant disparity between what we say and do.

Some observations were combined with interviews as a test demonstration, where systems engineers were asked to explain the system and to demonstrate how to perform specific test cases. During a test demonstration, the systems engineer was told to think aloud about what had to be done and why and how it could be done.

Difficulties identified using observations were related to usability. For instance, there is a need to switch between different software applications to perform specific

test cases. Thus, AuTS needed to be developed in such a way that it could address this challenge. Moreover, it was noted that usability during test case development is fundamental for the long-term success of the automated testing system.

3.2.1.2 Documentary Research

The fact that technology played a considerable role in the development of AuTS made documentary research [68] a vital part. The documentary research facilitated a thorough understanding of the SCMS, the ship systems, and the corresponding testing activities. Furthermore, documentary research enabled clarification and confirmation of the interviews and observations in the documents, i.e., thin description [68]. Two types of documents were primarily studied: the Software Design Description (SwDD) and the Software Test Description (SwTD). This part identified information about how AuTS should be formed and which features it should support.

3.2.1.3 Unstructured and Semi-Structured Interviews

While collecting business needs and applicable knowledge, unstructured and semi-structured interviews [68] were conducted with various stakeholders. Such stakeholders included systems engineers, project managers, line managers, university researchers, and professors. The purpose of the interviews was to generate new ideas for AuTS. Furthermore, interviews were used to validate that different aspects of the identified demands, challenges, and opportunities were correctly addressed.

3.2.1.4 Literature Review

A literature review of scientific theories and engineering methods in relevant topics was performed. This literature review broadened the horizons during development and investigated testing methods that could be incorporated into AuTS.

The literature search was conducted in the databases Chalmers Library, IEEE Xplore, ScienceDirect, arXiv, and Google Scholar. This literature review included several search strings composed of keywords in Table 3.1. Note that the table is not exhaustive for all the keywords used in this thesis. The keywords were updated as the literature review progressed. An initial search of topics led to information that helped create more specific search strings. The keywords of each column in Table 3.1 are considered synonyms and are combined in search strings via the OR operator, e.g., test OR testing. Then, the whole search string is created using the AND operator between the keywords of columns A to D.

Table 3.1: Search strings in the literature review.

A	**B**	**C**	**D**
Falsification	Software	Testing	Automation
	System	Test	Automated
			Automatic

3.2.1.5 Data Analysis

The data collected from observations, documentary research, unstructured and semi-structured interviews, and literature review was compiled in one document. Based on this document, the researcher formulated functional and qualitative requirements for AuTS with the immersion approach [69]. This approach has a low level of structure and is thus more reliant on the intuition and interpretive skills of the researcher.

3.2.2 Development of the Automated Testing System

AuTS corresponds to the artifact described in the framework for design-science research. The process of designing, assessing, and refining AuTS was based on the functional and qualitative requirements. One of the purposes of the automated testing system is to reduce the time needed to conduct regression testing for integrated systems developed during different project phases. Thus, AuTS has to be general and flexible enough for future extensions with more ship systems and simulation models. Being aware of the importance of such attributes, the automated testing system was decided to follow the testing guidelines provided by IEEE [70].

Furthermore, AuTS contains falsification testing. In the source code, each block in the falsification method (see Figure 2.3) has been scripted independently from the other blocks. All blocks except the Simulator are scripted in a general way. The source code for the Simulator contains attributes that are specific to a particular simulation model. Thus, it is mostly a matter of writing the source code for the Simulator in order to extend it with more simulation models.

AuTS was developed incrementally and iteratively, following the essence of agile development [71]. The reason for an incremental and iterative development strategy is to allow for errors to be discovered early and fixed neatly. There will always be room for discrepancy between what Saab Kockums wants and what is perceived they want. To reduce this discrepancy, the idea was to have a working prototype of AuTS as fast as possible. This prototype could be used for continuous feedback to foster an early validation of whether the system meets Saab Kockums's requirements and how it could be further improved to exceed their expectations.

The tools and methods chosen to be incorporated in AuTS were the answers to the research questions RQ1 and RQ3. By performing falsification in AuTS, the research question RQ4 could be answered.

3.2.3 Application of the Automated Testing System at Saab Kockums

To answer the research question RQ2, a qualitative research approach was chosen. This approach was appropriate since it gives rise to understanding the meanings people attribute to their world and facilitates unexpected findings [72]. In the following, the data collection and data analysis during this step will be presented.

3.2.3.1 Data Collection

Semi-structured interviews were used as a method of data collection. To conduct the interviews, an interview protocol with questions and procedures for the demonstration of AuTS was created. The questions were listed according to the funnel model [73], which begins with open questions and moves toward more specific questions. To enable the adaptation of the questions related to the development of the conversation in the interview, it was not necessary to follow the interview protocol strictly.

The interview protocol is observed in Appendix B and was inspired by [67]. The questions were centered around evaluating the applicability, quality, productivity, and usability of the three variants of test automation within AuTS. Applicability is defined as the relevance of the system for testing SCMS. Quality pertains to the system's ability to detect errors in the software. Productivity is measured by the time and costs required for testing activities using the system. Usability is defined as the difficulty of developing and maintaining tests with the system.

The interview protocol was pretested, as suggested by [74], to ensure that it was logical, consistent, and understandable. The pretest was conducted with an experienced researcher (Ph.D.) in case studies. Based on the feedback from the pretest, minor changes were made to the sequence and formulation of questions in the interview protocol.

To find participants, the intention and purpose of the interviews were presented to the Department of Ship Automation Submarine at Saab Kockums during their weekly meeting. Two engineers showed interest in participating. They were formally invited by email to an interview session. A consent information letter [67] and the interview protocol were attached to the email. The purpose of the consent information letter is to make the ethical principles that the researcher commits explicit and transparent to the participant. The purpose of attaching the interview protocol was to make the participants more prepared for the interviews.

The participants were considered suitable for the interviews since they were regularly working with the software in SCMS. Table 3.2 presents details about the participants. The participants have different roles and experiences, which may ensure a better distribution of viewpoints. Both participants were testing software in the SCMS several times a year. It is often just informal testing to verify source code developed by themselves. However, sometimes formal testing is done with the Swedish Defence Materiel Administration of source code developed by someone else. However, only one of the participants was familiar with test automation of software and none with the falsification framework. Thus, a brief introduction to test automation and falsification was given before the interviews. This was to make sure the participants understood the questions more easily. In Table 3.2, Yes and No are abbreviated to Y and N, respectively.

The two interviews were conducted separately at Saab Kockums's testing rig for the SCMS. This rig replicates the SCMS that will be built for the A26 submarine. AuTS was installed on a computer that was connected to the testing rig. The researcher was present throughout the interviews, asking questions and demonstrating the automated testing system. The participants were encouraged to make notes in their copy of the interview protocol during the interview. This was needed

Table 3.2: Demographics of the participants.

Participant	Current role	Industrial experience (years)	Testing frequency	Familiar with falsification/test automation
I_1	Software engineer	4-7	Monthly	N/Y
I_2	Systems engineer	8-11	Quarterly	N/N

because there was only one researcher, and the interviews were neither audio nor video recorded. Saab Kockums is a protection-worthy asset in Sweden, where special permission is needed for such recordings.

At the beginning of the interviews, the participants were informed about the purpose of the study and how it fits Saab Kockums's needs. The interviews ended with wrap-up questions to see if anything was missed in the research or if the participant could contribute with other relevant information. The average time for one interview was about one and a half hours. After the interviews, the participants were encouraged to take some time to compile their interview protocol and, when ready, send this document to the researcher.

3.2.3.2 Data Analysis

The data collected from the interviews were marked by codes [67], i.e., text representing a specific area or theme was identified and highlighted. The coding was performed according to the editing approach [69], where codes were defined based on the findings of the researcher during the analysis. The coded data was inserted into a standard spreadsheet tool, as suggested by [67]. To facilitate the overview of the spreadsheet, rows separated the codes, and columns separated the participants. This approach favored the generalization of findings.

3.3 Research Validity

The validity of a study refers to the trustworthiness of the findings, indicating the degree to which the results are accurate and free from any influence stemming from the researchers' personal perspectives [67]. Various methods exist in the literature for categorizing aspects of validity and potential challenges to validity. This thesis follows the classification scheme outlined in [67]. This scheme makes a distinction between four dimensions of validity: construct validity, internal validity, external validity, and reliability.

3.3.1 Construct Validity

This dimension of validity pertains to the degree to which the measurements used in the study accurately capture the researcher's intended concepts and align with the research questions.

Triangulation was achieved by employing various methods to gather information from various sources, which increased confidence in high construct validity. Furthermore, a pretest for the interview protocol was conducted with an experienced researcher. Notably, the pretester possessed a high knowledge of the SCMS, bolstering the confidence that the measurements accurately captured the intended concepts and aligned with the research questions.

Moreover, the participants were briefed on the study's objectives and essential technical concepts before the interviews to establish a shared understanding. Also, they could review their answers in the interview protocol before submitting it, i.e., member checking [69]. This helped to ensure that the interview data provided a fair representation of the participants' opinions. In order to encourage the participants to express their genuine viewpoints, the author assured complete anonymity for each participant.

3.3.2 Internal Validity

This dimension of validity becomes a concern when exploring causal relationships. When a researcher is investigating whether one factor is influenced by another, there is a potential for the investigated factor also to be influenced by a third factor.

This study aimed to investigate the impact of AuTS on the verification and validation activities for the SCMS. The findings were based on the input of two engineers involved in SCMS. The risk of missing relevant factors in the causal relationships was mitigated because the participants came from diverse backgrounds, held different positions, and regularly worked with the SCMS. Moreover, the automated testing system was reviewed on at least a bi-weekly basis by different stakeholders at Saab Kockums in order to discuss its demands, challenges, and opportunities.

Throughout the interviews, no hints or prompting were given to encourage positive responses regarding the qualities and effects of AuTS. Furthermore, key concepts were discussed during the interviews to ensure no misunderstandings could impact the participants' responses.

3.3.3 External Validity

This validity dimension focuses on the extent to which it is feasible to generalize the findings and whether the results hold significance beyond this study.

The sampling of participants followed the convenience sampling approach [72] since it was simple and cheap to implement. However, the generalizability of findings obtained through convenience sampling can be limited. Furthermore, the study was limited to just two participants because only two volunteered. Hence, data saturation [72] was most likely not achieved. This implies that if the author had continued to collect data, it is probable that further data collection would produce valuable insights.

One of the software applications used in AuTS is a tool developed by Saab Kockums, which is intended for in-house use only. This fact limited the generalizability. Besides that, AuTS is based on tools available to other organizations, e.g., Python and Automation Studio 4. Furthermore, the falsification has been developed generically, and the testing guidelines provided by IEEE have been followed. These aspects increased the generalizability.

3.3.4 Reliability

This dimension refers to the degree to which the data and the analysis rely on the individual researchers. In a hypothetical scenario, if a different researcher were to replicate the same study later, the results should remain consistent.

It was not possible to achieve observer triangulation [75] or peer debriefing [69] since there was only one author. This affected the reliability negatively since reviewing and discussing findings and interpretations with a co-author to identify potential biases, errors, or overlooked aspects of the research could not be performed.

To enhance reliability, various stakeholders reviewed components of the research design, such as the interview protocol and requirements for AuTS. Additionally, the audit trail approach [69] was adopted to maintain a clear chain of evidence [67]. Consequently, data and documents, e.g., interview notes, observation notes, and source code, were systematically stored using version control.

4
Technical Implementation

This chapter introduces the technical implementation of AuTS. The system encompasses three distinct variants of test automation for the SCMS. The first variant, presented in Section 4.1, involves the creation of script-based tests grounded in manual test documentation. The second variant, detailed in Section 4.2, focuses on the falsification of simulation models. The third variant, covered in Section 4.3, explores how the functionality of the ship systems can be tested using falsification.

4.1 Scripting Software Test Descriptions

Currently, the testing of the SCMS is executed solely with a manual approach. The documents describing ship systems' testing procedures in the SCMS are formulated similarly to the test procedure specification presented in the software testing standards by IEEE [76]. These documents will be referred to as SwTD. An example of three test steps in a SwTD with test results is observed in Table 4.1.

Table 4.1: An example of a software test description with test results.

Test Suite 1				
Test #	**Activities**	**Examination of result**	**Actual results**	**Test results**
1	Close valve 1.	Verify that valve 1 is closed.	Valve 1 is closed.	OK
2	Change pressure of tank 1 to 1 *bar*.	Verify that valve 1 is closed.	Valve 1 is closed.	OK
3	Change pressure of tank 1 to 3 *bar*.	Verify that valve 1 is opened.	Valve 1 is closed.	Not OK

Saab Kockums has developed an in-house software application to conduct manual testing of software running in real time in the SCMS. This software application will be called π-*app*. The π-*app* is installed on an external computer, which is connected to the testing rig for the SCMS. From π-*app*, it is possible to manipulate and observe components of ship systems in the SCMS, e.g., open valves, start water pumps, and observe if valves are opened or closed. Before this thesis, π-*app* had been extended with a toolbox to script programs that execute the manipulations and observations of components. Hence, it is possible to script test cases in π-*app*. However, Saab

Kockums had not yet investigated how test automation could be implemented with this tool. In this first variant of AuTS, SwTDs are scripted as source code to switch the test execution from a manual to an automated approach.

The functionality of the toolbox was limited prior to this thesis. During the thesis, the functionality has been extended with more features. Desired functionality was identified as and when the thesis progressed and reported to a software developer of *π-app*. The features became implemented in new software releases of *π-app*, which were continuously delivered to the researcher in this thesis.

The Python language [33] is used in *π-app* to script manipulations and obersvations of components. Two commonly used commands are *addAction* and *addExpectation*, where the former corresponds to executing an action, e.g., opening a valve, and the latter corresponds to executing an expectation, e.g., observing if the valve is opened.

Source code for a script-based test of the example SwTD in Table 4.1 is observed in Listing 4.1. In lines 1-2, the components manipulated and observed during the test are defined, i.e., valve 1 and tank 1. The component IDs for valve 1 and tank 1 are 13 and 37 in the component library, respectively. In line 4, a log file is created, where data from the testing will be stored in chronological order. This log file contains what actions have been executed and if the state of the components has met the expected result. In lines 6-9, Test 1 from Table 4.1 is defined and executed. The test sequence for Test 1 is created in line 6. In lines 7-8, closing valve 1 and verifying the valve is closed are added to the test sequence. In line 9, the test sequence for Test 1 is executed. In lines 11-14 and 16-19, Test 2 and Test 3 are defined and executed, respectively. Lines 12 and 17 correspond to changing the pressure in tank 1. In lines 13 and 18, valve 1 is verified to be closed and opened, respectively. In line 21, the log file containing all the data from the executed test sequences is saved as a txt file. In line 23, a document containing the test results is generated based on the log file. This document is observed in Appendix A for the example in Table 4.1. The test results are logged directly in the SwTD as proposed by IEEE [76].

```
Valve_1 = component(13)
Tank_1  = component(37)

log("Test Suite 1")

Test_1 = createSequence("Test 1")
Test_1.addAction(Valve_1.command("Close"))
Test_1.addExpectation(Valve_1.signal("State").value("Closed"))
Test_1.run()

Test_2 = createSequence("Test 2")
Test_2.addAction(Tank_1.command("Pressure").value(1))
Test_2.addExpectation(Valve_1.signal("State").value("Closed"))
Test_2.run()

Test_3 = createSequence("Test 3")
Test_3.addAction(Tank_1.command("Pressure").value(3))
Test_3.addExpectation(Valve_1.signal("State").value("Opened"))
Test_3.run()

closeLog()

generationReport()
```

Listing 4.1: Script-based test in π-*app* of test suite 1 from Table 4.1.

In this thesis, the SwTD for the LOX system has been scripted in π-*app*. The script file, comprising approximately 8000 rows of code, adheres to conventions of being granulated, sequential, clear, and simple. It represents roughly 80% of the total SwTD for the LOX system, encompassing all functional testing aspects. However, certain test cases in the SwTD are deemed unfeasible for making script-based in π-*app*. For instance, one such case involves verifying the correctness of the image displayed in the HMI. Other cases include verifying that the language in the HMI is Swedish and that the image complies with HMI guidelines. While the automation of these test cases is possible, it does not significantly enhance the overall testing process.

The automated testing of the LOX system identified a previously known defect and uncovered several ambiguities in the test specification, which were subsequently addressed. Once the defect is resolved, Saab Kockums will be able to rerun the complete automated test suite to verify its successful execution. Furthermore, the test scripts can be reused for future updates to the LOX system, though corresponding modifications to the scripts will be required to ensure their continued applicability.

4.2 Falsification of Simulation Models

This test automation variant involves the falsification of simulation models for systems in SCMS, e.g., control algorithms. A simulation model of the submarine dynamics of A26 together with the hovering controller, cf. Section 2.1 has been used as the system model $\mathcal{S}_{HOV}$. The simulation model of the submarine dynamics has been implemented as a separate system in SCMS. Thus, it is running in real time

on the PLCs during falsification. The hovering controller was already implemented on the PLCs prior to this thesis.

The source code for the falsification implementation has been scripted in π-*app*. Thus, no academic tools for falsification have been used, e.g., S-TaLiRo [77] and Breach [78]. In the following, the details of the falsification implementation for $\mathcal{S}_{HOV}$ will be detailed, which follows the framework presented in Section 2.2. Note that the parameter values regarding characteristics of A26 are omitted for confidential reasons.

4.2.1 Simulator

In Figure 4.1, the black-box model for $\mathcal{S}_{HOV}$ is observed. $\mathcal{S}_{HOV}$ is a discrete-time system defined for the time instances $k = 0, \delta t, 2\delta t, \ldots, T$. The input signals $\mathbf{x}_i^s$ consist of the load force F_{load}, sea water density gradient ε, and hover depth set-point $z_{sp,sub}$. From $\mathbf{x}_i^s$, the system model $\mathcal{S}_{HOV}$ generates the corresponding output response $\mathbf{x}_o^s$, which consists of the vertical depth z_{sub}, and vertical speed $\dot{z}_{sub}$.

Figure 4.1: Schematic diagram of the black-box model for $\mathcal{S}_{HOV}$.

4.2.2 Input Signal Generator

In this section, the three input signals F_{load}, ε, and $z_{sp,sub}$ are described.

4.2.2.1 Load Force

The input signal load force F_{load} is generated using the staircase generator. The rationale for using this input generator is to imitate a situation of loading or unloading the submarine. The varying input parameters for the staircase generator are amplitude A_{load} and delay τ_{load}, which are defined as:

$$\begin{aligned} A_{load} &\in \{x \in \mathbb{R} \mid -F_{load,max} \leq x \leq F_{load,max}\}, \\ \tau_{load} &\in \left\{x \in \mathbb{N} \mid \frac{T}{3} \leq x \leq \frac{2\,T}{3}\right\}, \end{aligned} \tag{4.1}$$

where $F_{load,max}$ is the maximum load change magnitude. The parameter base of the staircase generator is set to zero and, thus, is not used as a decision variable in the optimization problem. This lets z_{sub} settle around $z_{sp,sub}$ in the simulation before applying load changes to the submarine. The parameter period λ_{load} in the staircase

generator depends on the amplitude A_{load}. The rationale for deriving λ_{load} this way is a requirement regarding maximum load change magnitude $F_{load,max}$ for a defined time horizon T_{load} from the first load change is applied. Thus, the period λ_{load} is defined as:

$$\lambda_{load} = \frac{|A_{load}|}{F_{load,max}} T_{load}. \tag{4.2}$$

The number of steps in the staircase generator is defined as:

$$m = \left\lceil \frac{T_{load}}{\lambda_{load}} \right\rceil. \tag{4.3}$$

A staircase function f_{sc} is defined as:

$$f_{sc}(k) = A_{load} \sum_{j=0}^{m-1} \sigma(k - \tau_{load} - j \cdot \lambda_{load}) \tag{4.4}$$

where $\sigma(\cdot)$ is the Heaviside step function. The input signal $F_{load}(k)$ is observed in (4.5) and saturates according to the boundaries in the domain of A_{load}.

$$F_{load}(k) = \begin{cases} f_{sc}(k), & \text{if } -F_{load,max} \leq f_{sc}(k) \leq F_{load,max} \\ F_{load,max}, & \text{if } f_{sc}(k) > F_{load,max} \\ -F_{load,max}, & \text{if } f_{sc}(k) < -F_{load,max} \end{cases} \tag{4.5}$$

4.2.2.2 Sea Water Density Gradient

The sinusoidal input generator has been used to generate the sea water density gradient ε input signal. The rationale for using this generator is to imitate the local fluctuating density gradient in the sea due to changes in salinity, temperature, and pressure. The varying input parameters for the sinusoidal generator are period λ_ρ, delay τ_ρ, and amplitudes $A_{\rho,1}$, and $A_{\rho,2}$, which are defined as:

$$\begin{aligned} \lambda_\rho &\in \left\{ x \in \mathbb{R} \mid \frac{T}{30} \leq x \leq T \right\} \\ \tau_\rho &\in \left\{ x \in \mathbb{R} \mid 0 \leq x \leq \frac{T}{4} \right\} \\ A_{\rho,1},\, A_{\rho,2} &\in \left\{ x \in \mathbb{R} \mid 0 \leq x \leq 2 \cdot 10^{-5} \right\}. \end{aligned} \tag{4.6}$$

To describe a sinusoidal wave in the conventional way, the amplitude A_ρ and base D_ρ are defined as:

$$\begin{aligned} A_\rho &= \frac{|A_{\rho,1} - A_{\rho,2}|}{2} \\ D_\rho &= \frac{A_{\rho,1} + A_{\rho,2}}{2}. \end{aligned} \tag{4.7}$$

The input signal $\varepsilon(k)$ is defined as:

$$\varepsilon(k) = D_\rho + \sigma(k - \tau_\rho) \cdot A_\rho \cdot \sin\left(2\pi \frac{k - \tau_\rho}{\lambda_\rho}\right). \tag{4.8}$$

4.2.2.3 Hover Depth Setpoint

The input signal hover depth setpoint $z_{sp,sub}$ is generated using the constant input generator with base $D_{sp,sub}$ as parameter. The rationale for using this input generator is that a constant hover depth setpoint is considered the most reasonable use case for a hovering controller. The parameter $D_{sp,sub}$ is defined as:

$$D_{sp,sub} \in \{x \in \mathbb{R} \mid z_{sp,sub,min} \leq x \leq z_{sp,sub,max}\}, \tag{4.9}$$

where $z_{sp,sub,min}$ and $z_{sp,sub,max}$ are minimum and maximum depth for hover depth setpoint. The input signal $z_{sp,sub}(k)$ is defined as:

$$z_{sp,sub}(k) = D_{sp,sub}. \tag{4.10}$$

In summary, to generate the input signals $\mathbf{x}_i^s$, there are seven input parameters needed: A_{load}, τ_{load}, λ_ρ, τ_ρ, $A_{\rho,1}$, $A_{\rho,2}$, and $D_{sp,sub}$.

4.2.3 Quantitative Semantics and Specification

The specification φ_{HOV} is used to assess the behavior of $\mathcal{S}_{HOV}$. This specification contains two requirements. The first requirement concerns the vertical deviation of the submarine from the hover depth setpoint, which is not allowed to be more than ξ_1 m. The second requirement restricts the vertical speed of the submarine not to exceed ξ_2 m/s. The requirements are defined using the VBools comparison operator as:

$$\begin{aligned} |z_{sub} - z_{sp,sub}| &\leq_v \xi_1 \\ |\dot{z}_{sub}| &\leq_v \xi_2 . \end{aligned} \tag{4.11}$$

At time instance k, φ_{HOV} is defined with Max semantics as:

$$\varphi_{HOV}(k): \quad (|z_{sub}(k) - z_{sp,sub}(k)| \leq_v \xi_1) \wedge_{\text{Max}} (|\dot{z}_{sub}(k)| \leq_v \xi_2). \tag{4.12}$$

However, the evaluation of φ_{HOV} can be solely dependent on one of the requirements in (4.11). This derives from one of the requirements operating in a narrower domain compared to the other. This behavior can be fixed with normalization.

Let us explain this behavior with an example. Assume $\xi_1 = 50$ m, $\xi_2 = 10$ m/s, $z_{sub}(k) = 960$ m, $z_{sp,sub}(k) = 1000$ m, and $\dot{z}_{sub}(k) = 1$ m/s. This means the submarine can deviate 10 meters further from $z_{sp,sub}$ before falsifying the vertical deviation requirement. Also, the submarine can increase the vertical speed by 9 m/s before falsifying the speed requirement. In this example, the specification φ_{HOV} is evaluated to:

$$(\top, 10) \wedge_{\text{Max}} (\top, 9) = (\top, 9).$$

This means that the speed requirement is closer to becoming falsified than the vertical deviation requirement, which is in absolute terms but not in relative terms. This can be an issue when guiding the choice of the next set of input parameters to

falsify φ_{HOV} using optimization. To remedy this issue, the requirements in φ_{HOV} are normalized based on their limits ξ_1 and ξ_2 as:

$$\varphi_{HOV}(k) = \left(\frac{|z_{sub}(k) - z_{sp,sub}(k)|}{\xi_1} \leq_v 1 \right) \wedge_{\text{Max}} \left(\frac{|\dot{z}_{sub}(k)|}{\xi_2} \leq_v 1 \right). \tag{4.13}$$

With the same example as above, the normalized φ_{HOV} is evaluated to:

$$(\top, 0.2) \wedge_{\text{Max}} (\top, 0.9) = (\top, 0.2).$$

This evaluation reflects that the vertical deviation requirement is closer to being falsified than the speed requirement.

The normalized specification φ_{HOV} was used in the falsification implementation for $\mathcal{S}_{HOV}$ and must hold for all time instances over the interval $[0, T]$:

$$\square_{\text{Max},[0,T]}\, \varphi_{HOV} = \bigwedge_{k=0}^{T} {}_{\text{Max}}\ \varphi_{HOV}(k) = \langle v, \theta \rangle. \tag{4.14}$$

The objective function f^{φ}_{HOV} is defined as:

$$f^{\varphi}_{HOV} = \begin{cases} \theta & \text{if } v = \top \\ -\theta & \text{if } v = \bot. \end{cases} \tag{4.15}$$

4.2.4 Parameter Selector

The falsification of $\mathcal{S}_{HOV}$ has been implemented with four different methods to search for input parameters: HCR, LSF, vanilla BO, and TuRBO. For all methods, the maximum number of simulations is 256.

4.2.4.1 Hybrid-Corner-Random

The implementation of HCR has followed the pseudocode in Algorithm 1. The HCR method starts with a corner point in the search space $\mathcal{X}$, and the next point is a random uniformly sampled point from $\mathcal{X}$. It switches between the corners and random points until the maximum number of simulations, 256 here, is reached or a falsified point is found. The number of corners is $2^7 = 128$ since there are seven input parameters. Thus, all corner points will be evaluated if no falsified point is found.

4.2.4.2 Line-Search Falsification

The implementation of LSF has largely followed the pseudocode from Algorithm 2. Three algorithm adjustments have been made to better suit the falsification problem in this thesis.

The function H1 in Algorithm 2 has been adjusted. The updated H1 is observed in Algorithm 4. The first adjustment concerns the comparison operator for the if-statement in line 2, where the operator is $\leq$, instead of $<$ as in Algorithm 2. Let us motivate this adjustment with an example in 2D that considers the line-search

Algorithm 4 Updated H1

1: **function** H1($\mathbf{x}, \mathbf{x}_{old}$)
2: **if** $f^{\varphi}(\mathbf{x}_{old}) \leq f^{\varphi}(\mathbf{x})$ **then**
3: $\mathbf{x} \leftarrow$ the point $\mathbf{x}_{new}$ with lowest objective function value of the last call to
4: FalsifyLine
5: **end if**

of the function in (4.16). One characteristic of this function is that the objective function value continuously decreases in all directions from the origin (except for the trivial case when either x_1 or x_2 is zero).

$$\begin{aligned} f(x_1, x_2) &= -|x_1 \cdot x_2| + 0.2 \\ x_1, x_2 &\in \{x \in \mathbb{R} \mid -1 \leq x \leq 1\} \end{aligned} \tag{4.16}$$

Lets assume $\mathbf{x}_L$, $\mathbf{x}_M$, and $\mathbf{x}_R$ have been generated according to the line in Figure 4.2. The points $\mathbf{x}_L$, $\mathbf{x}_M$, and $\mathbf{x}_R$ are located at $(-1, -1)$, $(-0.2, -0.2)$, and $(1, 1)$, respectively. After two iterations of the function FalsifyLine, the points $\mathbf{x}_{new,1}$ and $\mathbf{x}_{new,2}$ have been generated and evaluated. Since the objective function f continuously decreases in all directions from the origin, evaluating points $\mathbf{x}_{new}$ along the line will never yield a lower objective function value than the corner points of the search space. Thus, $\mathbf{x}$ will never be updated in the function FalsifyLine. As a result, $\mathbf{x}$ will get stuck at the current location since the condition for the if-statement in H1 from Algorithm 2 will never be satisfied. To remedy this behavior, the comparison operator for the if-statement was changed.

The second adjustment is in line 3 in Algorithm 4. Here, $\mathbf{x}$ is assigned the $\mathbf{x}_{new}$ that had the lowest objective function value from the last call of FalsifyLine. Hence, $\mathbf{x}$ is not considered to be assigned any of the points $\mathbf{x}_L$, $\mathbf{x}_M$, and $\mathbf{x}_R$. This is to avoid getting stuck in local minima, particularly if those are located at the boundaries of the search space.

The third adjustment of the LSF method concerns the generation of the random direction vector in line 26 of Algorithm 2. In this thesis, the random direction vector is defined as:

$$\mathbf{d} = \mathbf{r} \circ \mathbf{v} \tag{4.17}$$

where:

$$\begin{aligned} \mathbf{r} &\sim \mathcal{U}(\Omega) \\ \Omega &= \{\mathbf{x} \in \mathbb{R}^n \mid -1 \leq x_i \leq 1, x_i \neq 0\} \\ \mathbf{v} &\in \{\mathbf{x} \in \mathbb{R}^n \mid x_i = u_i - l_i\}, \text{ for } i = 1, \ldots, n. \end{aligned} \tag{4.18}$$

Each element of the random direction vector is a non-zero random number between -1 and 1 multiplied by the width of the corresponding input parameters' search space. Scaling the random direction vector prevents the bias of being more likely to generate lines almost parallel to the dimensions with the smallest width in the search space. Figure 4.3 shows an example of this bias. In Figure 4.3a, the random direction vector is generated as [1, 1], according to Algorithm 2. In Figure 4.3b, the random direction vector is [1, 1] multiplied element-wise with $\mathbf{v}$, according to

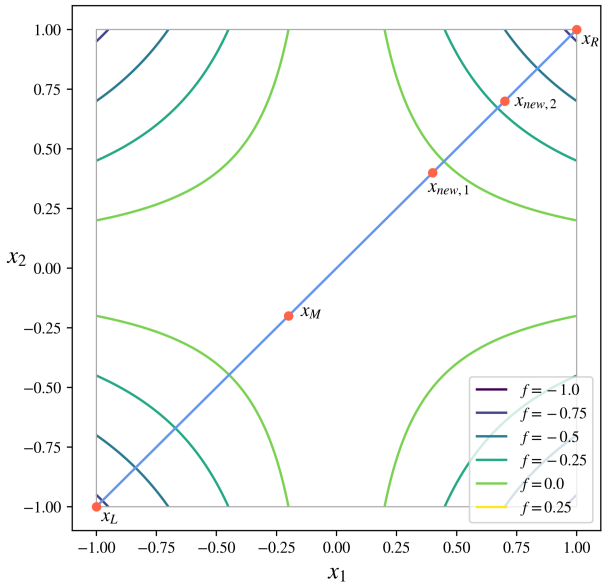


Figure 4.2: An example of line-search for $f(x_1, x_2) = -|x_1 \cdot x_2| + 0.2$.

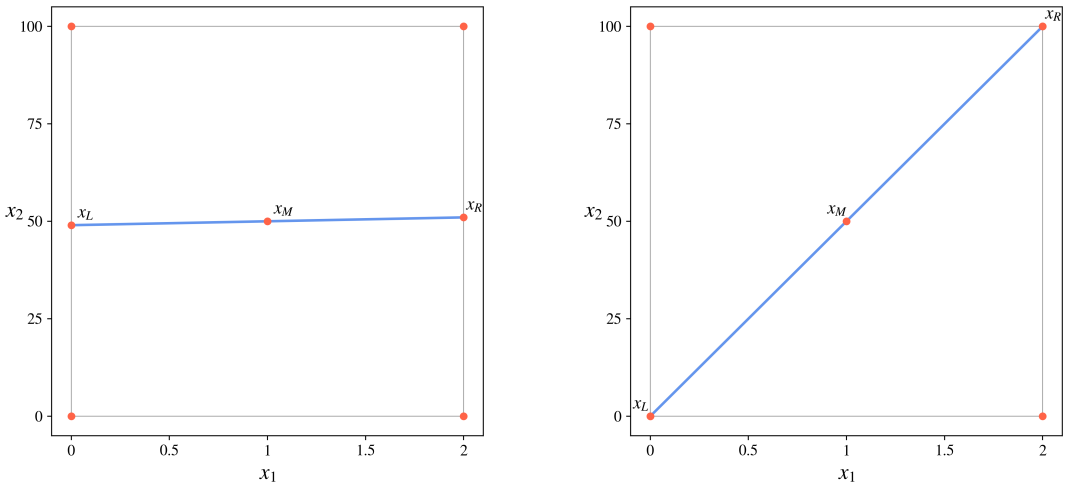
(4.17).

The maximum number of iterations without improvement for a single line in FalsifyLine is set to 5. Let us motivate this choice with an example. Assume $\mathbf{x}_M$ and $\mathbf{x}_R$ are arbitrary corner points in the search space. Furthermore, in each iteration of FalsifyLine, the algorithm generates a new point closer to $\mathbf{x}_R$. The objective function values for the new points are higher than for $\mathbf{x}_R$. When the function FalsifyLine terminates, the line-search algorithm has traveled $100 - 100/2^5 = 96.875\,\%$ of the distance of the original line segment between $\mathbf{x}_M$ and $\mathbf{x}_R$. This reach was considered sufficient in the trade-off between exploring a line sufficiently but not exploring too much on lines that do not generate points with low objective function values.

In the function SelectPoints($\mathbf{x}$), the parameters k and j were assigned the values 7 and 4, respectively. Hence, the end of one line segment is a corner point, and the end of the other line segment is on the boundary for at least half of the dimensions.

4.2.4.3 Vanilla Bayesian Optimization

The implementation of BO adheres to the pseudocode outlined in Algorithm 3. For clarity and distinction from TuRBO, this specific implementation is termed 'vanilla BO,' representing the conventional and widely-used version of BO. The squared-exponential kernel is used to define the covariance function. The acquisition function is defined as LCB. The parameter γ has been defined with the well-known formulation used in practice [79], $\gamma = \sqrt{0.125\, log(2N + 1)}$, where N is the maximum number of simulations. To build the initial surrogate model, $2 \cdot n$ number of samples is used, cf. [46], where n is the number of input parameters.



(a) Generating random direction vector without scaling.

(b) Generating random direction vector with scaling.

Figure 4.3: Examples of generating the random direction vector using different algorithms.

4.2.4.4 Trust Region Bayesian Optimization

The implementation of TuRBO has primarily followed the pseudocode in Algorithm 3, except for substituting vanilla BO for TuRBO. The implementation is based on the code release for the TuRBO method published in [53]. The Matérn kernel is used to define the covariance function. The acquisition function is defined as TS. To build the initial surrogate model, $2 \cdot n$ number of samples is used, as recommended by [53].

4.3 Falsification of Functionality in Ship Systems

This test automation variant involves falsification of the functionality for ship systems in the SCMS. This variant has been applied to perform automated testing of the software in the LOX system, which runs in real time on the PLCs. The source code for the falsification implementation has been scripted in *π-app*.

Saab Kockums has documentation (SwDD) for the ship systems that corresponds to a software design description presented in the IEEE standard [80]. The SwDD for the LOX system was used to establish the falsification problem, such as defining system model $\mathcal{S}_{LOX}$, search space $\mathcal{X}_{LOX}$, and specification φ_{LOX}. However, since the LOX system developed for the A26 submarine is confidential, all details regarding the actual LOX system are omitted from the report. Instead, a simplified fictive system model $\mathcal{S}_{SYS}$, with its search space $\mathcal{X}_{SYS}$ and specification φ_{SYS} will be used to explain the implemented falsification of the functionality. Nevertheless, the essence of the implemented falsification for the actual LOX system will be clear.

4.3.1 Fictive System Model

The schematics of the fictive system model $\mathcal{S}_{SYS}$ is observed in Figure 4.4. The system model includes three valves (V_1, V_2, V_3), one heater H, one LOX tank L, one pump P, and one pressure gauge G. Each component can be considered as a black box with inputs and outputs. An operator can manipulate some components by sending commands to SCMS through the HMI. In the following, the functionality of the components in $\mathcal{S}_{SYS}$ will be detailed.

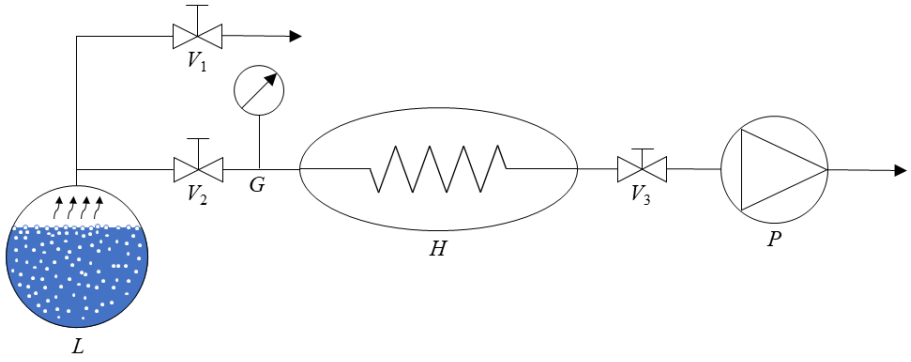


Figure 4.4: Schematic diagram of fictive system model $\mathcal{S}_{SYS}$.

The valve V_1 can either be opened or closed. An operator cannot open or close the valve manually since its state is controlled automatically by the SCMS. Thus, the state of V_1 solely depends on internal logic in the SCMS. The valve V_1 has no operator input but output defined as:

$$V_{1,\text{out,state}} \in \{\text{Opened, Closed}\} \tag{4.19}$$

where Opened and Closed correspond to whether the valve is opened or closed.

The valves V_2 and V_3 can either be opened or closed, and their states are controlled automatically by the SCMS. These valves can also be opened or closed through manual commands sent by an operator. Thus, the inputs to the valves V_2 and V_3 are defined as:

$$V_{2,\text{in,mode}},\ V_{3,\text{in,mode}} \in \{\text{Open, Close}\} \tag{4.20}$$

where Open and Close correspond to the manual commands an operator sends to open or close the valves. The outputs from the valves V_2 and V_3 are defined as:

$$V_{2,\text{out,state}},\ V_{3,\text{out,state}} \in \{\text{Opened, Closed}\}. \tag{4.21}$$

However, due to interlocks in the SCMS, the commands from the SCMS override the commands from the operator under certain circumstances. For example, if the SCMS sends the command Close to the valves, the valves are closed, even if the command Open is sent from the operator.

The LOX tank L contains a liquid form of molecular oxygen. The pressure in the LOX tank can be changed in the SCMS testing environment. Note that an operator does not change this pressure; it is a simulated pressure change for testing purposes. The pressure is changed to see that the functions in the SCMS that depend on the

pressure in the LOX tank work properly. Thus, the input to L is the tank pressure defined as:

$$L_{\text{in,p}} \in \{x \in \mathbb{R} \mid 0 < x \leq p_{max}\} \tag{4.22}$$

where p_{max} is the maximum simulated pressure in L. The output from L is the tank pressure defined as:

$$L_{\text{out,p}} \in \{x \in \mathbb{R} \mid 0 < x \leq p_{max}\} \tag{4.23}$$

i.e., $L_{\text{out,p}}$ copies the value of $L_{\text{in,p}}$.

The pressure gauge G measures the pressure after valve V_2 and does not take any inputs from the operator. The output from G is the pressure defined as:

$$G_{\text{out},p} \in \{x \in \mathbb{R} \mid 0 < x \leq p_{max}\}\,. \tag{4.24}$$

The heater H and pump P are controlled automatically by the SCMS and have the states On and Off. An operator can also start and stop H and P. Due to interlocks, the commands from SCMS override the commands from the operator. The inputs to H and P are defined as:

$$H_{\text{in,mode}},\ P_{\text{in,mode}} \in \{\text{Start, Stop}\} \tag{4.25}$$

where Start and Stop correspond to the manual commands an operator sends to start or stop the heater and pump. The outputs from H and P are defined as:

$$H_{\text{out,state}},\ P_{\text{out,state}} \in \{\text{On, Off}\} \tag{4.26}$$

where On and Off correspond to H and P being turned on or off.

The input and output sets for the components in $\mathcal{S}_{SYS}$ are observed in (4.27). The components in $\mathcal{S}_{SYS}$ and their inputs and outputs are summarized in Table 4.2.

$$\begin{aligned}
\mathcal{A} &= \{\text{Open}, \text{Close}\} \\
\mathcal{B} &= \{\text{Opened, Closed}\} \\
\mathcal{C} &= \{x \in \mathbb{R} \mid 0 < x \leq p_{max}\} \\
\mathcal{E} &= \{\text{Start}, \text{Stop}\} \\
\mathcal{F} &= \{\text{On}, \text{Off}\}
\end{aligned} \tag{4.27}$$

Table 4.2: Inputs and outputs for the components in $\mathcal{S}_{SYS}$.

Component	Input	Output
V_1		$V_{1,\text{out,state}} \in \mathcal{B}$
V_2	$V_{2,\text{in,mode}} \in \mathcal{A}$	$V_{2,\text{out,state}} \in \mathcal{B}$
V_3	$V_{3,\text{in,mode}} \in \mathcal{A}$	$V_{3,\text{out,state}} \in \mathcal{B}$
L	$L_{\text{in},p} \in \mathcal{C}$	$L_{\text{out},p} \in \mathcal{C}$
G		$G_{\text{out},p} \in \mathcal{C}$
H	$H_{\text{in,mode}} \in \mathcal{E}$	$H_{\text{out,state}} \in \mathcal{F}$
P	$P_{\text{in,mode}} \in \mathcal{E}$	$P_{\text{out,state}} \in \mathcal{F}$

The search space $\mathcal{X}_{SYS}$ is defined as:

$$\mathcal{X}_{SYS} = \{\mathbf{x} \in \textit{a space with 5 dims} \mid x_1 \in \mathcal{A},\, x_2 \in \mathcal{A},\, x_3 \in \mathcal{C},\, x_4 \in \mathcal{E},\, x_5 \in \mathcal{E}\}\,. \tag{4.28}$$

4.3.2 Simulator

In Figure 4.5, the black-box model for $\mathcal{S}_{SYS}$ is observed. $\mathcal{S}_{SYS}$ is a discrete-time system defined for the time instances $k = 0, \delta t, 2\delta t, \ldots, T$. The input $\mathbf{x}_i^s$ consists of the five signals: $V_{2,\text{in,mode}}$, $V_{3,\text{in,mode}}$, $L_{\text{in},p}$, $H_{\text{in,mode}}$, and $P_{\text{in,mode}}$. From $\mathbf{x}_i^s$, the system model $\mathcal{S}_{SYS}$ generates the corresponding output response $\mathbf{x}_o^s$, which consists of the seven outputs: $V_{1,\text{out,state}}$, $V_{2,\text{out,state}}$, $V_{3,\text{out,state}}$, $L_{\text{out},p}$, $G_{\text{out},p}$, $H_{\text{out,state}}$, and $P_{\text{out,state}}$.

Figure 4.5: Schematic diagram of the black-box model for $\mathcal{S}_{SYS}$.

4.3.3 Input Signal Generator

All the input signals for $\mathcal{S}_{SYS}$ are generated using a variant of the staircase generator. An example of this input signal generator adapted for categorical variables is observed in Figure 4.6. The input signal x_i^s can be parameterized by *delay*, $state_0$ and $state_1$, which for example can be defined as:

$$state_0,\ state_1 \in \mathcal{A}.$$

The rationale for using this input generator is a simple implementation where the input signal switches from one distinct value to another.

4.3.4 Quantitative Semantics and Specification

To assess the behavior of $\mathcal{S}_{SYS}$, the specification φ_{SYS} is used, which is observed in Table 4.3. Note, β_i for $i = 1, \ldots, 7$ corresponds to an arbitrary real number. Furthermore, $\wedge$ will be considered as shorthand for $\wedge_+$, i.e., additive semantic is used.

The specification φ_{SYS} is fulfilled if the specifications φ_{SYS}^j for $j = 1, \ldots, 7$ all are fulfilled. The specification φ_{SYS}^1 is fulfilled if the following is true: if the pressure in the LOX tank $L_{\text{out},p}$ is above a threshold β_1, then the valve V_1 is opened, and if $L_{\text{out},p}$ is below β_2, then V_1 is closed. The specification φ_{SYS}^2 is fulfilled if the following

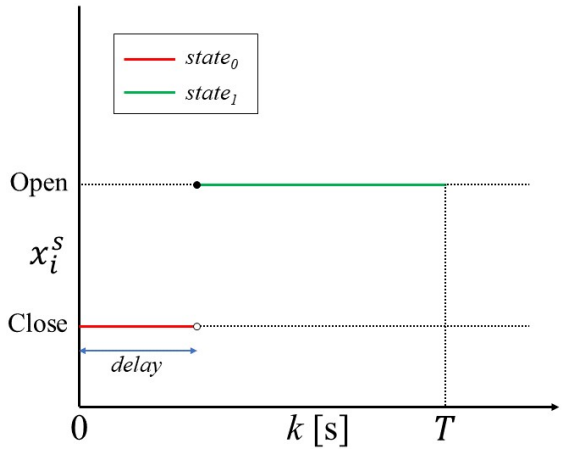


Figure 4.6: The parameterization technique used to generate input signals with categorical variables for $\mathcal{S}_{SYS}$.

is true: if $L_{\text{out},p}$ is between β_3 and β_4, then V_2 can be opened and closed through the HMI. The specification φ^3_{SYS} has similar logic as in φ^1_{SYS}, but for the valve V_2. The specification φ^4_{SYS} is fulfilled if the following is true: if V_2 is opened, then the heater H is running, and if V_2 is closed, then H is not running. The specification φ^5_{SYS} has similar logic as in φ^2_{SYS} but for the valve V_3 and pressure gauge G. The specification φ^6_{SYS} has similar logic as in φ^1_{SYS}, but for V_3 and pressure gauge G. The specification φ^7_{SYS} is fulfilled if the following is true: if $G_{\text{out},p}$ is below β_7 and V_3 is opened, then the pump P is running, and if V_3 is closed, then P is not running.

Note, the specifications φ^j_{SYS} for $j = 1, \ldots, 7$ have variables that are defined using categorical variables. Also, the specifications have logical operators not defined using VBools, e.g., material conditional. Thus, calculating the distance to the specifications being falsified according to (2.12) is impossible. Hence, standard values were used to estimate the distance for falsified specifications. For example, if specification φ^1_{SYS} is fulfilled at a certain time instance, its robustness value is 1. On the other hand, if the specification is not fulfilled at a time instance, its robustness value is -1.

The specification φ_{SYS} was used in the falsification implementation for $\mathcal{S}_{SYS}$ and must hold for all time instances over the interval $[0, T]$:

$$\square_{+,[0,T]}\, \varphi_{SYS} = \bigwedge_{k=0}^{T}{}_{+}\ \varphi_{SYS}(k) \#' \delta t = \langle v, \theta \rangle\,. \tag{4.29}$$

The objective function f^{φ}_{SYS} is defined as:

$$f^{\varphi}_{SYS} = \begin{cases} \theta & \text{if } v = \top \\ -\theta & \text{if } v = \bot. \end{cases} \tag{4.30}$$

4.3.5 Parameter Selector

Due to the dynamics of the system model $\mathcal{S}_{LOX}$ with most of the inputs and outputs defined with categorical variables, optimization-based methods were con-

Table 4.3: Specification φ_{SYS} for the fictive system model $\mathcal{S}_{SYS}$.

Specification	VBools Formula
φ_{SYS}	$\varphi^1_{SYS} \wedge \varphi^2_{SYS} \wedge \varphi^3_{SYS} \wedge \varphi^4_{SYS} \wedge \varphi^5_{SYS} \wedge \varphi^6_{SYS} \wedge \varphi^7_{SYS}$
φ^1_{SYS}	$((L_{\text{out},p} > \beta_1) \rightarrow (V_{1,\text{out,state}} = \text{Opened})) \wedge ((L_{\text{out},p} < \beta_2) \rightarrow (V_{1,\text{out,state}} = \text{Closed}))$
φ^2_{SYS}	$((\beta_3 < L_{\text{out},p} < \beta_4) \wedge (V_{2,\text{in,mode}} = \text{Open}) \rightarrow (V_{2,\text{out,state}} = \text{Opened})) \wedge ((\beta_3 < L_{\text{out},p} < \beta_4) \wedge (V_{2,\text{in,mode}} = \text{Close}) \rightarrow (V_{2,\text{out,state}} = \text{Closed}))$
φ^3_{SYS}	$((L_{\text{out},p} \leq \beta_3) \rightarrow (V_{2,\text{out,state}} = \text{Opened})) \wedge ((L_{\text{out},p} \geq \beta_4) \rightarrow (V_{2,\text{out,state}} = \text{Closed}))$
φ^4_{SYS}	$((V_{2,\text{out,state}} = \text{Opened}) \rightarrow (H_{\text{out,state}} = \text{On})) \wedge ((V_{2,\text{out,state}} = \text{Closed}) \rightarrow (H_{\text{out,state}} = \text{Off}))$
φ^5_{SYS}	$((\beta_5 < G_{\text{out},p} < \beta_6) \wedge (V_{3,\text{in,mode}} = \text{Open}) \rightarrow (V_{3,\text{out,state}} = \text{Opened})) \wedge ((\beta_5 < G_{\text{out},p} < \beta_6) \wedge (V_{3,\text{in,mode}} = \text{Close}) \rightarrow (V_{3,\text{out,state}} = \text{Closed}))$
φ^6_{SYS}	$((G_{\text{out},p} \leq \beta_5) \rightarrow (V_{3,\text{out,state}} = \text{Closed})) \wedge ((G_{\text{out},p} \geq \beta_6) \rightarrow (V_{3,\text{out,state}} = \text{Opened}))$
φ^7_{SYS}	$(((G_{\text{out},p} < \beta_7) \wedge (V_{3,\text{out,state}} = \text{Opened})) \rightarrow (P_{\text{out,state}} = \text{On})) \wedge ((V_{3,\text{out,state}} = \text{Closed} \rightarrow (P_{\text{out,state}} = \text{Off}))$

sidered inappropriate for this falsification problem. It makes no sense to guide the falsification process when the objective function evaluates to a number that does not correctly estimate the distance to the specification being falsified. Thus, the optimization-free method HCR was chosen. The implementation of HCR has followed the pseudocode in Algorithm 1.

The implementation of falsification for the LOX system has been performed with 1000 simulation runs. Let us assume that 50 components in the LOX system have binary inputs, i.e., accepting signals with two distinct values. Then, 2^{51} simulation runs are needed to explore all corner points using HCR. Thus, a small fraction ($<$ 0.1%) of all corner points have been tested with this implementation.

5
Results

This chapter presents the outcomes addressing the research questions. Section 5.1 details the developed automated testing system for the SCMS, serving as the response to RQ1. The impact of this system on testing activities at Saab Kockums is explored in Section 5.2 by the study of RQ2. Section 5.3 presents the response to RQ3, outlining suggested enhancements to the falsification framework. The comparison of various test-generation methods for falsification of the hovering controller is presented in Section 5.4, corresponding to the findings for RQ4.

5.1 RQ1 - Tools and Methods for Automated Testing of Ship Control and Monitoring System

The outcome of this research question is the automated testing system for the SCMS. AuTS encompasses three distinct forms of test automation for the SCMS. The first variant entails the conversion of manual test procedures into script-based testing. The second variant involves the falsification of simulation models. The third variant focuses on testing the functionality of ship systems through the use of falsification. A comprehensive description of the technical solution is provided in Chapter 4.

5.2 RQ2 - Effects of Automated Testing System for Ship Control and Monitoring System

This section explores the impact of AuTS on testing activities at Saab Kockums, drawing insights from the semi-structured interviews conducted with the participants listed in Table 3.2. The results are classified into two groups: the first encompasses outcomes linked to the first variant of AuTS (Section 5.2.1), while the second addresses the second and third variants (Section 5.2.2).

5.2.1 Scripting Software Test Descriptions

This section presents results regarding the attributes of the first AuTS variant: applicability, quality, productivity, and usability.

5.2.1.1 Applicability

Converting manual test procedures into script-based testing received recognition from both participants, who agreed that this variant is suitable for testing the SCMS.

Both participants are working with systems in the SCMS that frequently need to be regression tested, where automated tests are lacking. Thus, the approach to automate the execution of today's SwTDs is highly appreciated. The systems engineer concluded about the applicability of this variant of AuTS:

> *I think scripting SwTDs would fit all systems and many but not all tests. The way the LOX system's SwTD has been implemented is a suitable level. This should be implemented for all systems (with some exceptions) so that regression testing can be performed when changes have been implemented to ensure SUT still passes testing.*
>
> Systems engineer

On the contrary, both participants were skeptical about entirely replacing the manual testing of today with this test automation variant. During manual testing, functionality that behaves correctly according to SwTD but is not working as intended can be identified. For example, the Swedish Defence Materiel Administration (customer of A26) has given feedback during manual testing concerning the behavior of a system where some functionality is missing or implemented incorrectly. These findings were never identified when reviewing the documents. Likewise, it is difficult to identify these findings when performing script-based testing of SwTDs. Thus, both participants believe Saab Kockums still needs to perform some testing of the SCMS manually.

5.2.1.2 Quality and Productivity

Both participants agreed that script-based execution of SwTDs may boost the productivity and quality of the testing activities. It takes time to write the test scripts, but most systems are tested frequently, so using a script will save effort and, thus, costs in the long run. Furthermore, when script-based testing of a system exists, the testing can be executed more frequently than manual approaches. As a result, it is more likely to find errors in the code early in the development process when the error is straightforward to fix. The software engineer elaborated on the productivity and quality advantages of this variant of AuTS:

> *Usually, when we regression test systems manually, we analyze the entire SwTD to figure out which test steps are affected and need to be regression tested. It would be less time-consuming just to run the entire script automatically. Also, for the manual testing, some parts of the test specification might be affected that were missed and never regression tested.*
>
> Software engineer

As long as the testing script is reviewed and works as intended, there is not much room for errors in the test execution. This can be compared to the manual testing approach, where the participants point out that the testing can be tedious, which increases the risk of human errors. However, the software engineer emphasizes the importance of reviewing testing scripts so they behave as intended:

> *The scripts need to be reviewed thoroughly ... it is important to make sure the check mark for a passed test is indeed correct.*
>
> Software engineer

5.2.1.3 Usability

Writing a script-based test has to be straightforward and unambiguous. Furthermore, a test step must be easily added to a script without breaking the whole structure. The script-based tests can be written incrementally, which is suitable since Saab Kockums possesses many SwTDs that can be updated with new tests. The software engineer concluded the following of scripting SwTDs for more systems:

> *Now that we have a script for the LOX system implemented already, we have a base, and from what I have seen, it is not that difficult to create new scripts for other systems.*
>
> Software engineer

AuTS generates documents that present the test results, where it is clear what test cases failed. It is also possible to investigate what failed in the test cases. The software engineer summarized how test results are presented:

> *To get the results from the executed test is simple and straightforward.*
>
> Software engineer

5.2.2 Falsification

This section presents results regarding the attributes of the second and third variants of AuTS: applicability, quality, productivity, and usability.

5.2.2.1 Applicability

Both participants deemed the falsification framework useful for functions in SCMS where simulation models exist. The software engineer agreed that optimization

might help find difficult inputs for their simulation models since the worst-case inputs might be difficult to know precisely. The systems engineer elaborated on the applicability of falsification:

> *For our more complex functions, e.g., controllers and autopilots, falsification could be helpful to identify the situations (where in the input space) that should be focused to test during set to work or verification period to ensure that the controller can handle the most difficult situations.*
>
> Systems engineer

The software engineer considered the falsification framework applicable for testing functionality of ship systems, but in another way compared to the proposed implementation in this thesis. First and foremost, the software engineer explains that the state explosion problem that arises when using the proposed implementation of falsification for ship systems might pose difficulties. This is because there are a lot of different components in a ship system, where each component has a lot of different states. However, the software engineer proposes to test each component type, e.g., a standardized valve, using falsification with HCR.

5.2.2.2 Quality and Productivity

The participants held opposing viewpoints on the productivity gains of using the falsification method for the simulation models. The software engineer deemed that falsification can save time and effort in testing simulation models since optimization speeds up searching for difficult inputs. The systems engineer discussed falsification for their simulation models as:

> *If we use it, it would be because it would be interesting for us engineers, i.e., not save time or cost compared to not using it.*
>
> Systems engineer

5.2.2.3 Usability

According to both participants, establishing a falsification environment for a complex simulation model is difficult, which may restrict the usage of falsification in the testing of the SCMS.

5.3 RQ3 - Improvements for Falsification Methods

The outcome of this research question is based on the implementation of LSF for the hovering controller. The LSF framework has been adjusted to improve the

testing performance in this thesis, but the adjustments are considered applicable in the general case of LSF. Section 4.2.4.2 comprehensively describes the technical solution.

5.4 RQ4 - Performance of Test-Case Generation Methods in Falsification

In this section, we present simulation results obtained through the falsification of the system model $\mathcal{S}_{HOV}$ as detailed in Section 4.2. The falsification process employs various test-case generation methods, including HCR, LSF, vanilla BO, and TuRBO.

To summarize the numerical results from the falsification, the testing of A26's hovering controller demonstrated consistent and reliable performance across all test scenarios. The system effectively maintained hovering within the predefined limit values in every instance, ensuring operational stability. These results confirm the reliability and effectiveness of the hovering controller in meeting the operational requirements for submarine stability.

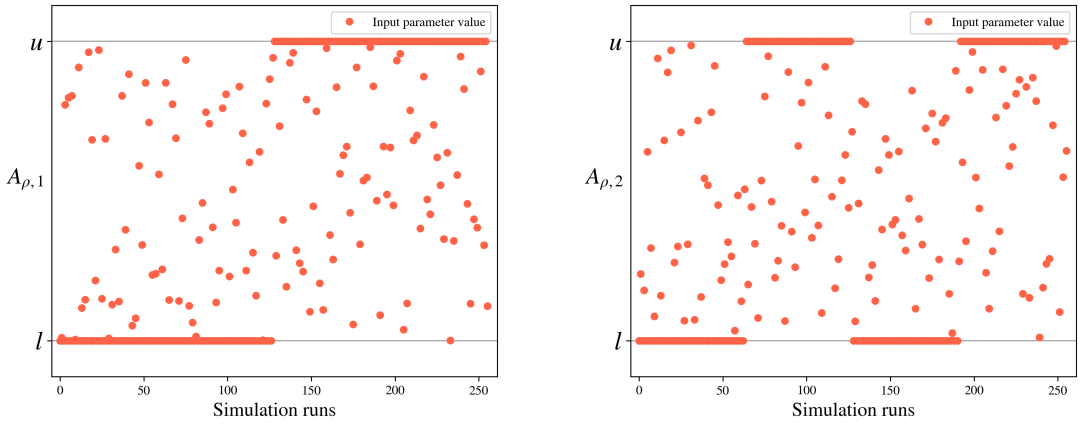
5.4.1 Hybrid-Corner-Random Falsification

In Figure 5.1, the simulation results for the HCR falsification are observed. The seven input parameters are observed in Figures 5.1a-5.1g. The robustness values are observed in Figure 5.1h. The x-axis represents the simulation runs in chronological order, i.e., from the first to the last (256th) simulation run. The y-axis represents the value for the input parameters and robustness value. The domain boundaries for the input parameters are observed in the figures. The input parameters are sampled as expected for the HCR method, with half of the points on the boundaries and half uniformly sampled from their domain. The robustness values are distributed relatively uniformly.

The mean $\overline{f^{\varphi}(\mathbf{x}^s)}$, median $\text{Med}(f^{\varphi}(\mathbf{x}^s))$, and standard deviation $\sigma(f^{\varphi}(\mathbf{x}^s))$ for the robustness values are observed in Table 5.1. The median and mean of the robustness values are almost equal, which can indicate that the robustness values are close to uniformly distributed. This is confirmed in Figure 5.1h. Furthermore, the minimum robustness value and the corresponding input parameters generating this value are observed in Table 5.1. The minimum robustness value is found at a corner point with the following characteristics: the sine wave for the sea water density gradient has maximum amplitude, minimum delay, and minimum period; the hover depth setpoint is the minimum value, i.e., closest to the water surface; the load force is maximum amplitude with minimum delay, i.e., a single step of maximum load as early as possible.

Table 5.1: Statistical measures for 256 simulations with HCR.

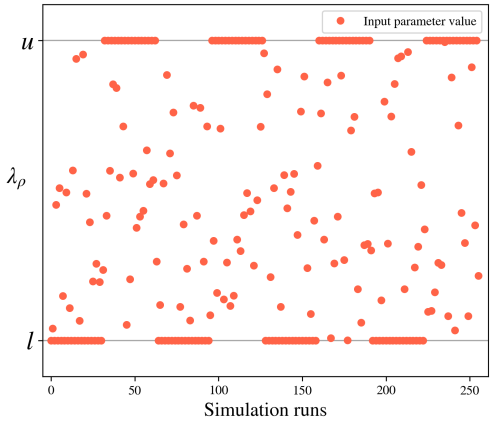
Type	Value
$\overline{f^{\varphi}(\mathbf{x}^s)}$	0.4892
$\text{Med}(f^{\varphi}(\mathbf{x}^s))$	0.4869
$\sigma(f^{\varphi}(\mathbf{x}^s))$	0.1477
$\min\ f^{\varphi}(\mathbf{x}^s)$	0.0802
$\text{argmin}\ f^{\varphi}(\mathbf{x}^s)$	$\begin{cases} A_{\rho,1} = 0 \\ A_{\rho,2} = 2 \cdot 10^{-5} \\ \lambda_{\rho} = \frac{T}{30} \\ \tau_{\rho} = 0 \\ D_{sp,sub} = z_{sp,sub,min} \\ \tau_{load} = \frac{T}{3} \\ A_{load} = F_{load,max} \end{cases}$



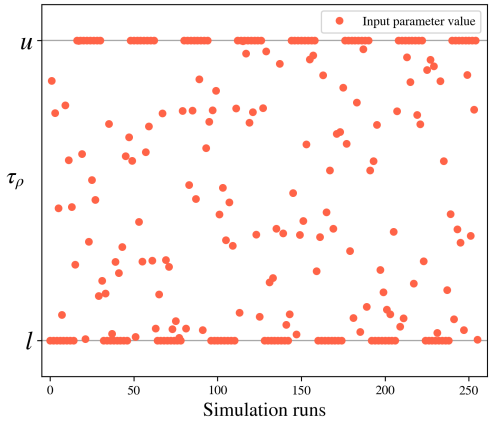
(a) Amplitude for a sinusoidal signal representing sea water density gradient.

(b) Amplitude for a sinusoidal signal representing sea water density gradient.

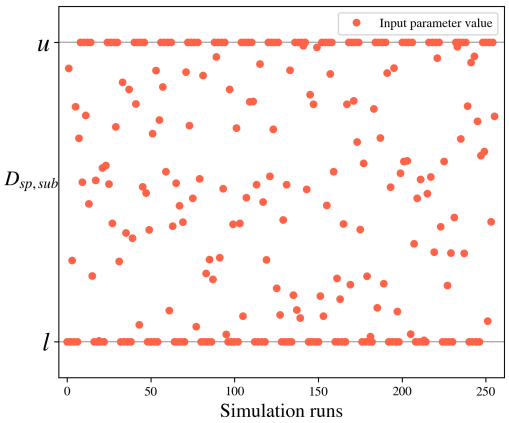
Figure 5.1: Falsification results for 256 simulations with HCR.



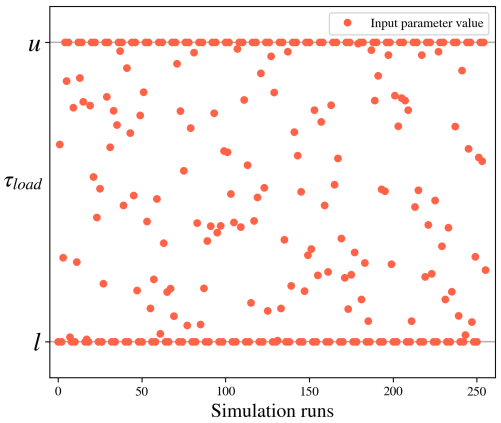
(c) Period for a sinusoidal signal representing sea water density gradient.



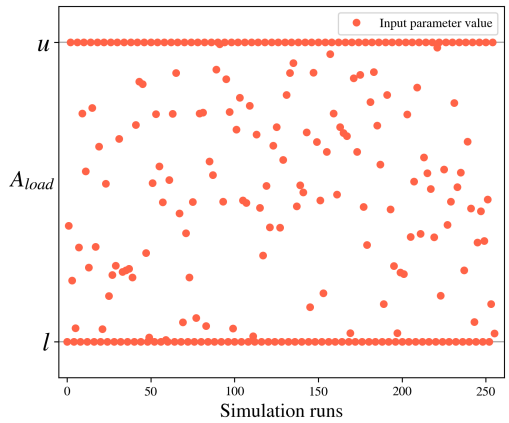
(d) Delay for a sinusoidal signal representing sea water density gradient.



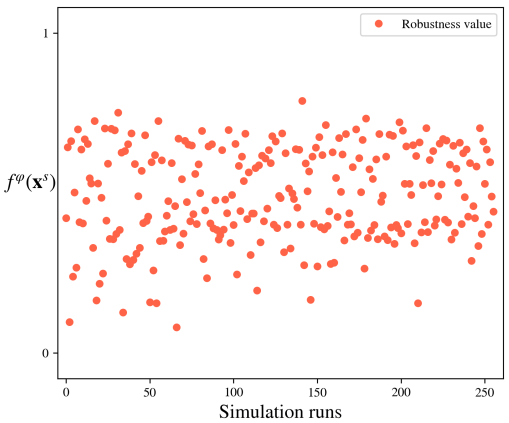
(e) Base for a constant signal representing hover depth setpoint.



(f) Delay for staircase signal representing load force.



(g) Amplitude for staircase signal representing load force.



(h) Robustness values for HCR.

Figure 5.1: Falsification results for 256 simulations with HCR.

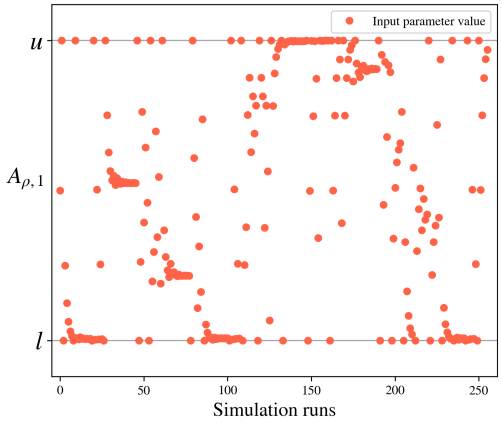
5.4.2 Line-Search Falsification

Figure 5.2 shows the simulation results for the LSF. The seven input parameters are observed in Figures 5.2a-5.2g. The LSF method frequently explores points with low values for $D_{sp,sub}$ and high values for A_{load} and τ_{load}. There is no distinct trend for the other input parameters. The robustness values are observed in Figure 5.2h, which have a skew towards higher values, and lower values are found over time. Thus, LSF may learn to find lower robustness values.

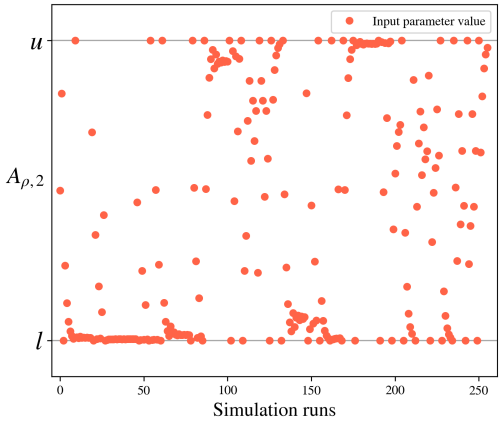
The mean $\overline{f^{\varphi}(\mathbf{x}^s)}$, median $\text{Med}(f^{\varphi}(\mathbf{x}^s))$, and standard deviation $\sigma(f^{\varphi}(\mathbf{x}^s))$ for the robustness values are observed in Table 5.2. The median is lower than the mean, which can indicate that the robustness values are skewed toward higher values, as confirmed above. Furthermore, the minimum robustness value and the corresponding input parameters generating this value are observed in Table 5.2. The minimum robustness value is found at a point with the following characteristics: the sine wave for the sea water density gradient oscillates around low parameter values, has high delay, and medium period; the hover depth setpoint is close to the minimum parameter value; the load force is close to maximum amplitude with high delay, i.e., almost a single step of maximum load as late as possible.

Table 5.2: Statistical measures for 256 simulations with LSF.

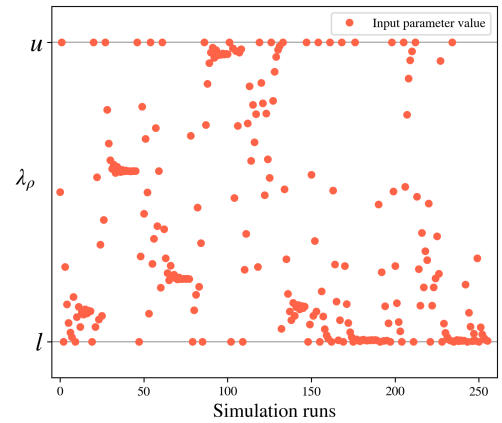
Type	Value
$\overline{f^{\varphi}(\mathbf{x}^s)}$	0.4318
$\text{Med}(f^{\varphi}(\mathbf{x}^s))$	0.4109
$\sigma(f^{\varphi}(\mathbf{x}^s))$	0.1214
$\min\ f^{\varphi}(\mathbf{x}^s)$	0.1872
$\text{argmin}\ f^{\varphi}(\mathbf{x}^s)$	$\begin{cases} A_{\rho,1} = 4.3241 \cdot 10^{-6} \\ A_{\rho,2} = 3.7488 \cdot 10^{-7} \\ \lambda_{\rho} = 0.2362\,T \\ \tau_{\rho} = 0.2270\,T \\ D_{sp,sub} = 1.0002\,z_{sp,sub,min} \\ \tau_{load} = 0.6614\,T \\ A_{load} = 0.9960\,F_{load,max} \end{cases}$



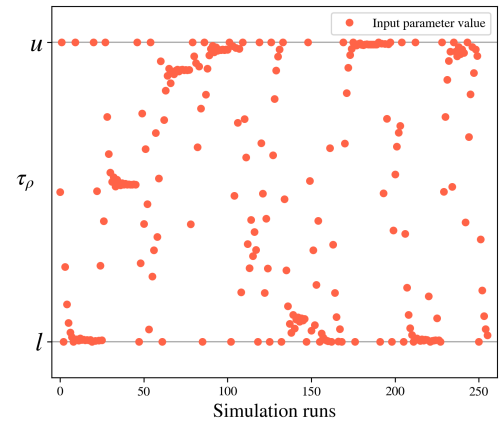
(a) Amplitude for a sinusoidal signal representing sea water density gradient.



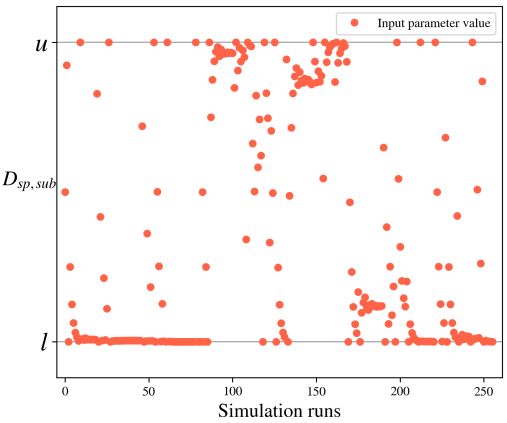
(b) Amplitude for a sinusoidal signal representing sea water density gradient.



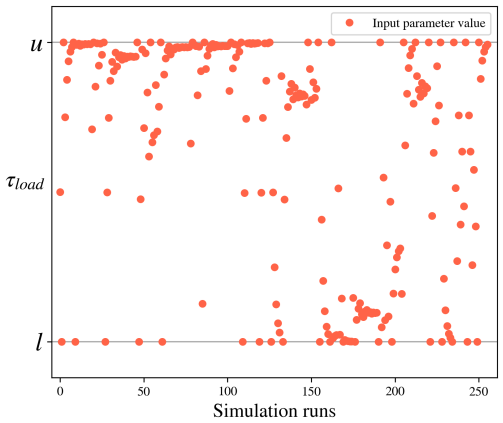
(c) Period for a sinusoidal signal representing sea water density gradient.



(d) Delay for a sinusoidal signal representing sea water density gradient.

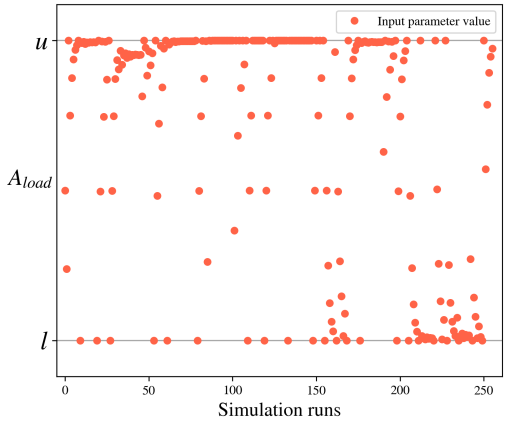


(e) Base for a constant signal representing hover depth setpoint.

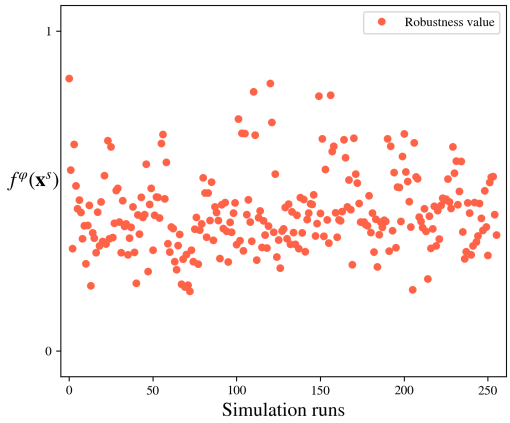


(f) Delay for staircase signal representing load force.

Figure 5.2: Falsification results for 256 simulations with LSF.



(g) Amplitude for staircase signal representing load force.



(h) Robustness values for LSF.

Figure 5.2: Falsification results for 256 simulations with LSF.

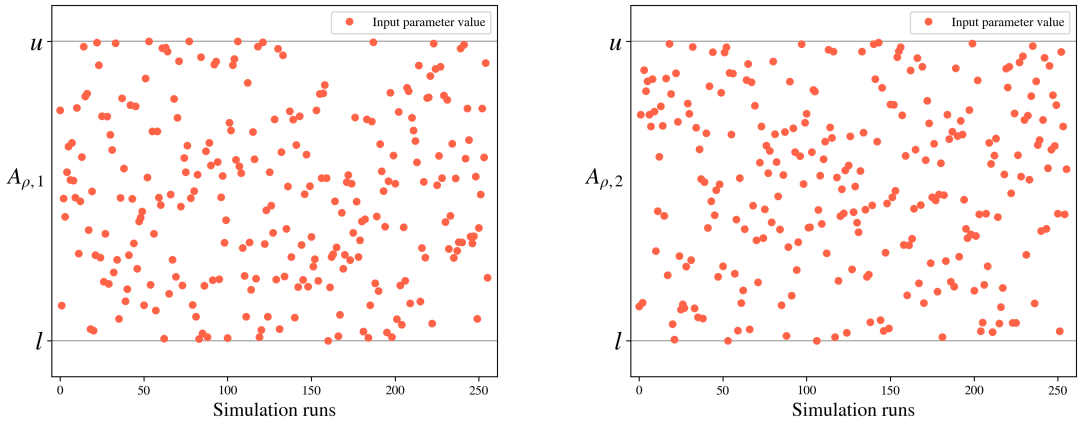
5.4.3 Vanilla Bayesian Optimization Falsification

In Figure 5.3, the simulation results for the vanilla BO falsification are observed. The seven input parameters are observed in Figures 5.3a-5.3g. The BO method is not able to generalize optimal parameter values for $A_{\rho,1}$, $A_{\rho,2}$, and $D_{sp,sub}$. For τ_ρ and τ_{load}, BO generalizes optimal values to be located in the lower and upper half of the domains, respectively. Optimal values for λ_ρ and A_{load} are identified to be located close to the lower and upper boundaries of the domains, respectively. The robustness values are observed in Figure 5.3h, where lower robustness values are found over time.

The mean $\overline{f^{\varphi}(\mathbf{x}^s)}$, median $\text{Med}(f^{\varphi}(\mathbf{x}^s))$, and standard deviation $\sigma(f^{\varphi}(\mathbf{x}^s))$ for the robustness values are observed in Table 5.3. The median is lower than the mean, which can indicate that the robustness values are skewed towards higher values. This is confirmed in Figure 5.3h. Furthermore, the minimum robustness value and the corresponding input parameters generating this value are observed in Table 5.3. The minimum robustness value is found at a point with the following characteristics: the sine wave for the sea water density gradient oscillates around low parameter values, has low delay, and low period; the hover depth setpoint is around 20% deeper compared to the minimum parameter value; the load force is close to maximum amplitude with relatively high delay, i.e., almost a single step of maximum load applied late in the simulation.

Table 5.3: Statistical measures for 256 simulations with vanilla BO.

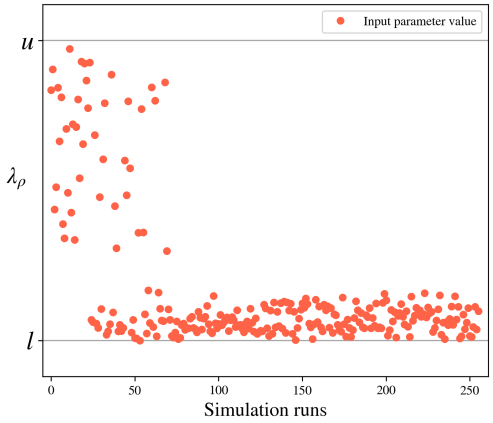
Type	Value
$\overline{f^{\varphi}(\mathbf{x}^s)}$	0.4431
$\text{Med}(f^{\varphi}(\mathbf{x}^s))$	0.4249
$\sigma(f^{\varphi}(\mathbf{x}^s))$	0.0997
min $f^{\varphi}(\mathbf{x}^s)$	0.2285
argmin $f^{\varphi}(\mathbf{x}^s)$	$\begin{cases} A_{\rho,1} = 4.9647 \cdot 10^{-6} \\ A_{\rho,2} = 9.5838 \cdot 10^{-6} \\ \lambda_{\rho} = 0.1393\,T \\ \tau_{\rho} = 0.0799\,T \\ D_{sp,sub} = 1.2143\,z_{sp,sub,min} \\ \tau_{load} = 0.5693\,T \\ A_{load} = 0.9524\,F_{load,max} \end{cases}$



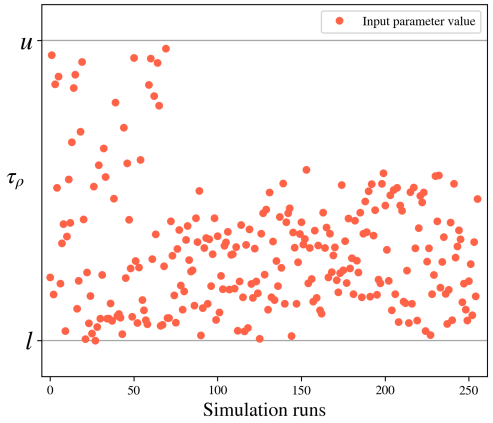
(a) Amplitude for a sinusoidal signal representing sea water density gradient.

(b) Amplitude for a sinusoidal signal representing sea water density gradient.

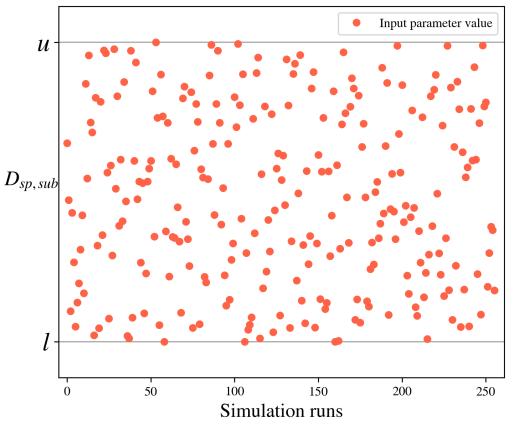
Figure 5.3: Falsification results for 256 simulations with vanilla BO.



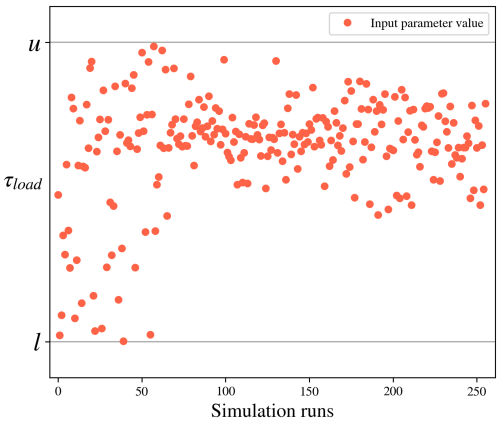
(c) Period for a sinusoidal signal representing sea water density gradient.



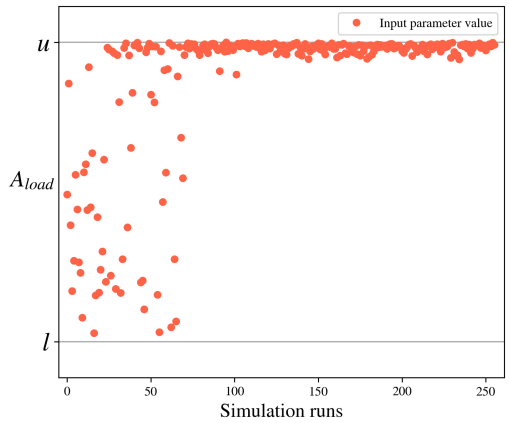
(d) Delay for a sinusoidal signal representing sea water density gradient.



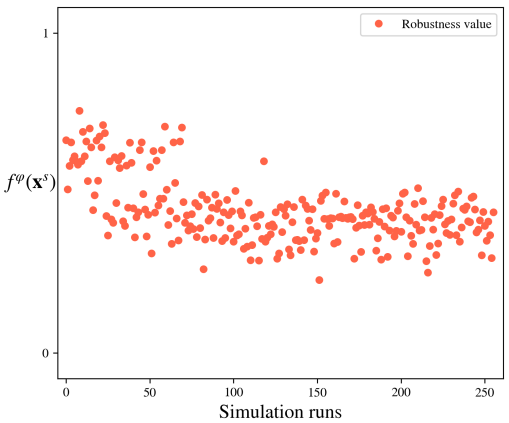
(e) Base for a constant signal representing hover depth setpoint.



(f) Delay for staircase signal representing load force.



(g) Amplitude for staircase signal representing load force.



(h) Robustness values for vanilla BO.

Figure 5.3: Falsification results for 256 simulations with vanilla BO.

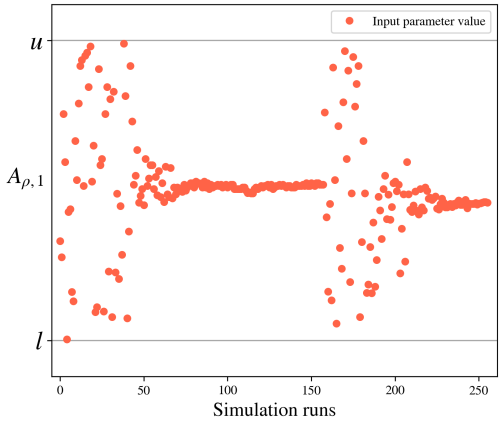
5.4.4 Trust Region Bayesian Optimization Falsification

In Figure 5.4, the simulation results for the TuRBO falsification are observed. The seven input parameters are observed in Figures 5.4a-5.4g. It can be seen that the algorithm has a significant event at around 150 simulation runs when current TRs are discarded, and new TRs are initialized. Before and after this significant event, the TuRBO method focuses on parameter values in an interval of 1/4 of the span of the domain for $A_{\rho,1}$, $A_{\rho,2}$, τ_ρ, $D_{sp,sub}$, and A_{load}. For τ_{load} and λ_ρ, the TuRBO method focuses on parameter values that are separated by a distance greater than 1/4 of the span of the domain. The TuRBO method seems to find optimal parameter values for A_{load} after 20 simulation runs. This can be compared to all other input parameters, where around 50 simulation runs are necessary. The robustness values are observed in Figure 5.4h, where lower robustness values are found over time, both before and after the significant event.

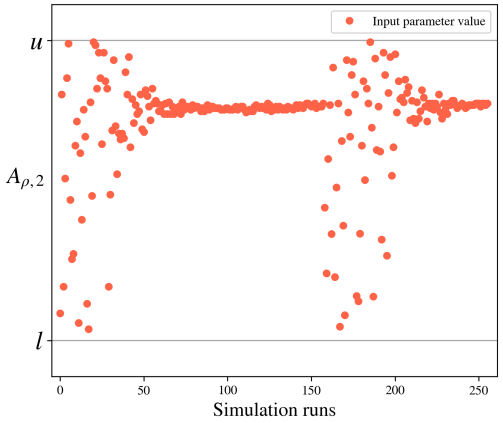
The mean $\overline{f^\varphi(\mathbf{x}^s)}$, median $\text{Med}(f^\varphi(\mathbf{x}^s))$, and standard deviation $\sigma(f^\varphi(\mathbf{x}^s))$ for the robustness values are observed in Table 5.4. The median is higher than the mean, which can indicate that the robustness values are skewed towards lower values. However, by investigating the robustness values in Figure 5.4h before and after the significant event separately, it can be observed that both segments are skewed toward higher values. Furthermore, the minimum robustness value and the corresponding input parameters generating this value are observed in Table 5.4. The minimum robustness value is found at a point with the following characteristics: the sine wave for the sea water density gradient oscillates in the upper half of the domain, has low delay, and low period; the hover depth setpoint is around 108% deeper compared to minimum parameter value; the load force is close to maximum amplitude with medium delay, i.e., almost a single step of maximum load applied at about half time of the simulation.

Table 5.4: Statistical measures for 256 simulations with TuRBO.

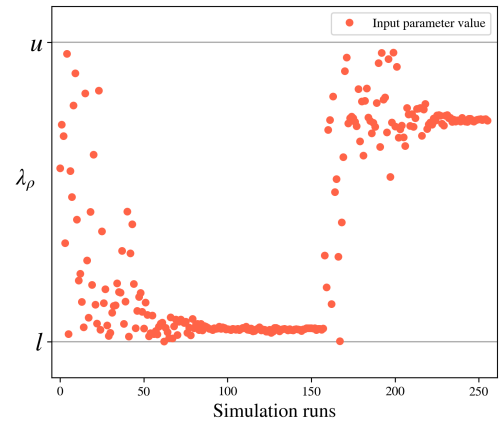
Type	Value
$\overline{f^\varphi(\mathbf{x}^s)}$	0.3745
$\text{Med}(f^\varphi(\mathbf{x}^s))$	0.3849
$\sigma(f^\varphi(\mathbf{x}^s))$	0.1312
$\min\ f^\varphi(\mathbf{x}^s)$	0.1089
$\text{argmin}\ f^\varphi(\mathbf{x}^s)$	$\begin{cases} A_{\rho,1} = 1.04 \cdot 10^{-5} \\ A_{\rho,2} = 1.57 \cdot 10^{-5} \\ \lambda_\rho = 0.0738\,T \\ \tau_\rho = 0.0561\,T \\ D_{sp,sub} = 2.0836\,z_{sp,sub,min} \\ \tau_{load} = 0.4314\,T \\ A_{load} = 0.9991\,F_{load,max} \end{cases}$



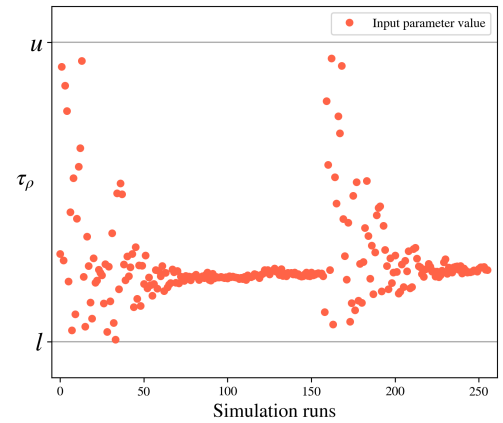
(a) Amplitude for a sinusoidal signal representing sea water density gradient.



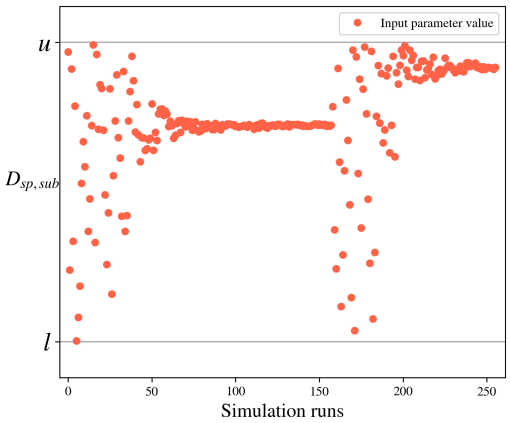
(b) Amplitude for a sinusoidal signal representing sea water density gradient.



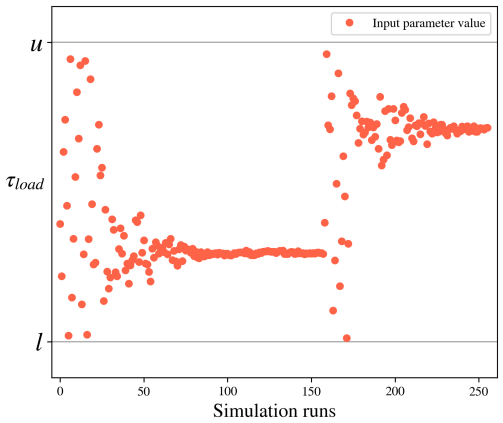
(c) Period for a sinusoidal signal representing sea water density gradient.



(d) Delay for a sinusoidal signal representing sea water density gradient.

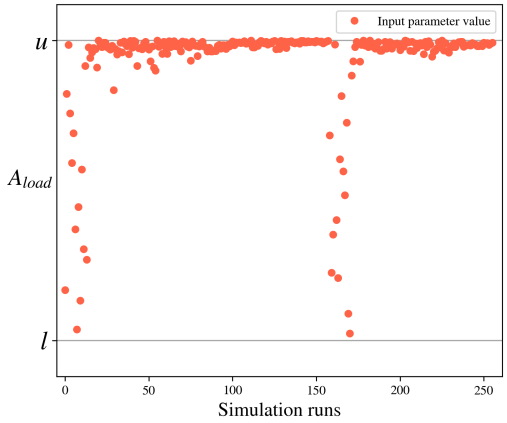


(e) Base for a constant signal representing hover depth setpoint.

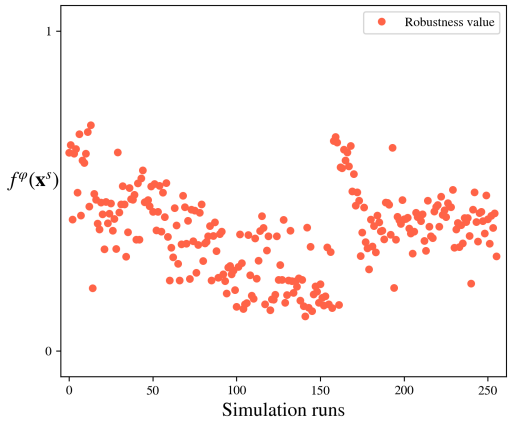


(f) Delay for staircase signal representing load force.

Figure 5.4: Falsification results for 256 simulations with TuRBO.



(g) Amplitude for staircase signal representing load force.



(h) Robustness values for TuRBO.

Figure 5.4: Falsification results for 256 simulations with TuRBO.

6
Discussion

This chapter delves into an analysis of the findings presented in Chapter 5, reflecting on how the findings of the thesis are either in keeping with the research of others or how these results differ from those of others. Following this, the thesis explores the social and ethical dimensions associated with conducting research in the military industry.

6.1 RQ1 - Tools and Methods for Automated Testing of Ship Control and Monitoring System

An automated testing system has employed tools and methods suitable for the SCMS. AuTS was created by adhering to functional and qualitative requirements derived from recognized demands, challenges, and opportunities. This development yielded an automated testing system encompassing three distinct variants of test automation.

While figuring out how to bring together various tools and methods for the automated testing system, some patterns were noticed along the way.

- *Length of test steps.* The number of actions (e.g., open valve) and expectations (e.g., observing the state of the valve) that are executed in a test step should be kept to a minimum, i.e., promote granularity of the test scripts. This principle prevents that race condition [81] takes place. It also facilitates test scripts' reusability, as numerous test cases often replicate similar patterns or share common subsets of test steps that can be efficiently reused across test cases.
- *Simplicity of the structure.* The test scripts should be composed of test cases with clarity and simplicity, utilizing sequential steps. This principle facilitates the coexistence of test scripts and SwTDs, e.g., increased clarity as they follow the same structure.
- *Test automation for repetitive tasks.* Test automation achieves its best value when performed on repetitive tasks as it accelerates manual work. Automation is possible for non-repetitive tests, e.g., verifying that an HMI image has been reviewed for compliance with HMI guidelines, but it does not add value to the process. Executing such non-repetitive tests independently of the automated tests is advisable to reduce overall test complexity.
- *Centralized testing system.* Unification amongst different types of testing and applications is one of the core ideas of AuTS. Previously, the manual testing

of the SCMS was performed in several applications. However, unifying the testing in a centralized testing system offers effective means to systemize and organize all types of testing within π-*app*.

In the transition to automated testing, it was decided to centralize the testing around π-*app* since a toolbox required for test automation was already implemented for this application. Thus, it was necessary to adapt the testing environment to enable the execution of all desired tests in π-*app* that were previously not performed in this application. Centralizing AuTS around π-*app* presents a limitation regarding extensibility with other applications. The diminished extensibility is attributed to the insufficient emphasis on designing interfaces that facilitate the integration of the automated testing system with other applications.

The proposed solution in this thesis is established on a Python-based toolbox. This is similar to what has been found in previous research, e.g., [31] and [34], where the Python-based Robot Framework was used. On the contrary, a difference between this thesis and previous research regards the dependence on in-house tools. AuTS is primarily based on in-house tools at Saab Kockums, e.g., π-*app*. In contrast, the researchers in [31] and [38] use open-source or commercial tools for implementing test automation. In-house tools restrict the generalizability of the solution outside the case company. However, the need for in-house tools at Saab Kockums may partly be explained by the long product life cycles in the naval defense industry. The longer the product life cycles are, the more exposed the manufacturer is to depend on end-of-life tools that stop being supported by suppliers.

In previous research on automated testing for PLC-based systems, the focus was to implement test automation directly in IDEs dedicated to PLC development. For instance, in [42], the authors use test automation tools compatible with the IDE Codesys. On the contrary, in this thesis, PLC-based systems are tested using tools not dedicated to IDEs in PLC development. However, one fundamental difference is the focus of implementation-based testing in [42], whereas this thesis focuses on specification-based testing. It is reasonable that dedicated PLC tools are needed for implementation-based testing, and more general tools can be used for specification-based testing.

Simulation models can replicate a lot of the dynamics in real systems. However, the simulation models rarely tell you everything. Thus, falsification was implemented for a HIL environment, with hardware that will be part of the launched A26 submarine. There is a preference for substituting simulation models with hardware whenever feasible to ensure comprehensive verification and validation. However, a drawback of falsification with HIL is the combination of long duration for test runs (since tests run in real time) and the number of test runs needed for optimization-based methods. These aspects also prove why falsification is primarily applicable to simulation models, where test runs are cheaper and faster. As established in [57], falsification is suitable in the design phase of the V-model [82] in model-based development design. However, a reasonable compromise stated by one participant in the interviews, see Section 5.2, is that falsification of simulation models could be helpful to identify the situations that should be focused to test during set to work or verification period for the full-size ship. Then, prior knowledge of the falsification

problem can be used with πBO by defining where there is a higher chance of the system being falsified.

While this report introduces a tool designed for automated specification-based testing, the capabilities of this tool unveil new opportunities for testing systems in situations where human intervention is impractical. For instance, tests involving continuous input to the SUT for extended periods, e.g., 24 hours straight.

In this thesis, formal verification was not implemented. However, the third variant in AuTS can correspond to model checking by defining all states and actions for a ship system. Then, all possible system scenarios could be examined systematically with the brute-force approach, i.e., similar to model checking.

The researcher formulated the functional and qualitative requirements for AuTS with the immersion approach. It is hard to present and obtain a clear chain of evidence in informal immersion approaches [67]. However, this approach was used due to its simplicity. It was deemed feasible under the requirement analysis due to its brief duration and the involvement of only a single researcher.

6.2 RQ2 - Effects of Automated Testing System for Ship Control and Monitoring System

Ensuring satisfactory software performance often involves testing, a widely adopted technique encompassing various activities. However, several challenges are encountered using manual testing, such as its costliness, time-intensive nature, repetitiveness, and susceptibility to errors. The first variant of AuTS significantly impacts the testing process by addressing these challenges. According to unanimous participant feedback, the first variant is user-friendly and applicable for testing the SCMS. Furthermore, they deem that this variant can increase the productivity and quality of the testing activities for SCMS.

As concluded by both participants, there are some cases where manual testing is inevitable for the SCMS and should be seen as complementary to automated testing. The authors in [17] also reported this conclusion. The authors even demonstrate that integrating manual and automated testing yields favorable outcomes. This outcome demands that manual and automated testing coexist in a unified fashion. The challenge to unify manual and automated testing is addressed in this thesis. AuTS has strengths in compatibility and collaboration with the conventional manual testing of SCMS. As the script-based tests for SCMS are based on SwTDs and are scripted in a toolbox on top of π-*app*, the manual and automated testing approaches primarily require the same testing environment.

As found in previous research, manual testing can be tedious and thus may lead to errors. One participant also identified these drawbacks. Furthermore, the participant stated the importance of thoroughly reviewing the testing scripts to increase reliability. The verification process has transitioned from manual system testing to a process where scripts are verified manually, subsequently performing the system testing. Verifying testing scripts is also tedious; thus, human errors may appear. However, in the long run, it is considered less demanding to verify that the script-based tests are reliable than that the test results from manual testing are reliable.

Prior research has indicated that employing test automation allows faster, more reliable, cost-effective, and frequent testing with shorter feedback cycles than manual testing. The participants also identified all these advantages of test automation regarding the first variant of AuTS. In particular, the participants considered AuTS beneficial for regression testing. Automated regression testing is advocated as a beneficial practice and has been integrated into agile development processes to facilitate continuous integration [83]. Furthermore, regression testing was identified in [35] as particularly suited for test automation. The authors in [35] were operating in the naval defense industry, i.e., the same industry as Saab Kockums, although this is not a prominent reason for similar conclusions.

Applying falsification in different settings for the SCMS was considered applicable, but the participants were not unanimously convinced regarding the gains in productivity and quality. Moreover, falsification was considered difficult according to the participants, which could hamper the usage of it.

One deviation between the results in this thesis and previous research is the opinions of falsification. In [57], falsification is presented as a simple method compared to other verification alternatives. Conversely, the participants in this thesis perceived falsification as challenging to the point that it might limit its practical application. However, the introduction the participants received on falsification might have been too brief for them to grasp the concept. In that case, a more profound introduction of falsification might have resulted in different opinions of its usability.

Furthermore, the participants' perception may depend on the fact that the falsification implementation was developed from scratch. By implementing falsification with dedicated falsification applications, e.g., S-TaLiRo, the falsification may have been perceived as simpler and more usable by the participants. On the contrary, each block in the falsification method (see Figure 2.3) has been scripted independently from the other blocks in the implementation. It is mostly a matter of writing the Simulator block's source code to extend with more simulation models. The Simulator block is based on the simulation model and is not related to the domain knowledge of falsification. Thus, extending Saab Kockums's falsification environment with more simulation models should not be overly complicated since they already know their models.

In general, the results from the interviews regarding the effects of test automation for SCMS can be questioned regarding validity. Only two participants were participating in the study, where both were working for the case company. Hence, no interviews were conducted with people from academia to assess AuTS. The researcher would likely extract more information if the data collection were extended to include more participants with different backgrounds.

6.3 RQ3 - Improvements for Falsification Methods

The proposed modifications to the LSF method stem from insights gained while implementing falsification for the system model $\mathcal{S}_{HOV}$. Addressing challenges encountered with the standard LSF method, the proposed adjustments, detailed in

Section 4.2.4.2, serve as effective heuristics. Experimental evaluations demonstrate that the first and second adjustments prevent the algorithm from becoming trapped in local maxima and minima. While identified within the context of this thesis, these heuristics hold potential applicability to other instances of LSF. The third adjustment, aimed at reducing directional bias when generating the random direction vector, is inherently valuable for minimizing bias in various falsification problems and can be extended for use in broader LSF contexts.

While this study has explored adjustments of LSF with compelling results, it is essential to acknowledge the inherent complexity of the subject matter. Consequently, it is prudent to recognize that the standard LSF may offer enhanced insights or yield more robust outcomes compared to the LSF proposed in this.

6.4 RQ4 - Performance of Test-Case Generation Methods in Falsification

Optimization-based and non-optimization-based falsification are explored for the problem with the hovering controller. Employing an optimization-based approach necessitates the definition of quantitative semantics. Additionally, optimization-based methods entail an extra cost for solving the optimization problem. When dealing with falsifying CPS, gradients are generally unavailable, as simulations are the only viable option for non-trivial systems. Consequently, gradient-free optimization methods are required. It should offer significantly greater efficiency than non-optimization-based approaches to justify the complexity and cost of employing an optimization-based falsification approach. However, the non-optimization-based HCR method identifies the lowest robustness value, cf. Tables 5.1-5.4. Consequently, the specification was closest to being falsified by assessing extreme combinations of values, such as the minimum and maximum values within the permissible parameter ranges.

The HCR method finds the lowest robustness value at a corner point. This highlights the strength of the HCR method, i.e., to evaluate corner points where the toughest inputs exist in some problems. In this falsification problem, some corner points correspond to the highest load force amplitude and a hover depth setpoint closest to the water surface. This load force amplitude entails it to be applied in one step, compared to smaller amplitudes where more load force steps are applied but spread out over time. With the load force spread over time, the hovering controller has more time to reduce the depth tracking error e due to the accumulated load force. Hence, the submarine will deviate less from the hover depth setpoint, ceteris paribus. This dynamics reasoning is supported in Figures 5.2g, 5.3g, and 5.4g, where all optimization-based methods focus on high load force amplitudes. Regarding the hover depth setpoint, the force from surface suction increases the closer the submarine is to the water surface [55]. Thus, the maximum influence of surface suction in this problem should be with the hover depth setpoint closest to the water surface. However, the generalization of explored values for hover depth setpoint that minimizes the robustness value can not be identified, cf. Figures 5.2e, 5.3e, and 5.4e.

In [57], the author proposes the HCR method as a baseline method suitable for comparing different test-case generation methods in falsification. Furthermore, the author describes that assessments of standard benchmark problems indicate that HCR performs well on many problems, except for the most difficult ones. There is no exception in this thesis, where HCR finds the lowest robustness value of all methods. This may be partially explained by the characteristics of the falsification problem, where the most challenging inputs can be derived to be located on the boundaries of the parameter space, i.e., easy problems. However, the HCR method found the highest mean and median robustness value among all four test-case generation methods. This is a plausible outcome since HCR does not choose input parameters based on optimization, which is the case for the other three methods.

The author in [57] states that a carefully chosen formulation based on BO is the new state-of-the-art optimization method for simulation-based falsification. This is based on the result from [46], where the TuRBO method is superior to other optimization methods, such as vanilla BO. Similar observations are identified in this thesis, where the TuRBO method finds the lowest mean and median robustness values among all four test-case generation methods. Furthermore, it finds a minimum robustness value close to the lowest robustness value found among all four methods. On the contrary, the vanilla BO yields the highest minimum robustness value among all four methods. Also, it finds the highest mean and median robustness values among the three optimization-based methods. By observing the Figures in 5.3a-5.3g, the vanilla BO method cannot generalize narrow areas of optimal values for most input parameters. This indicates a shortcoming in this method's capability to find lower robustness values, given that all input parameters influence the robustness value. However, as mentioned in Section 1.6, the time allocated to find optimal features for the optimization methods was limited. Hence, the vanilla BO might have performed better with more effort put into optimizing its features.

In [46], the performance of LSF is between the performance of vanilla BO and TuRBO for hard problems. This thesis presents comparable findings. The minimum, median, and mean robustness values using LSF are between the values obtained from vanilla BO and TuRBO, cf. Section 5.4. However, the LSF method cannot elicit the advantage of emphasizing extreme parameter values since it did not find the same corner point that yielded the lowest robustness value for HCR. The restricted amount of simulation runs is one explanation. Another explanation is the values assigned to the parameters k and j in SelectPoints($\mathbf{x}$), where only one end of the line segments is a corner point. Hence, more simulation runs would be needed to find the corner point that generated the minimum robustness value using the HCR method.

No test-case generation method succeeded in falsifying φ_{HOV}. Thus, no calculations can be made regarding the average success rate for falsifying specifications. This metric compares the performance of test-case generation methods in [46]. However, the minimum robustness value can be used as a proxy for comparing the performance of test-case generation methods. Using this proxy to rank the four methods used in this thesis in order of performance, from best to worst, yields the following sequence: HCR, TuRBO, LSF, and vanilla BO. This outcome is similar to what is reported for optimization-based methods on hard problems in [46], where the fol-

lowing ranking is reported from best to worst: TuRBO, LSF, vanilla BO.

Performing 256 simulations per test-case generation method can be considered too few simulations for this falsification problem. In [46] and [51], 1000 simulations are performed for each test-case generation method in benchmark problems. However, most of the benchmark problems have more dimensions in parameter space compared to $\mathcal{S}_{HOV}$, which partly motivates the difference in the number of simulation runs. Nevertheless, it can be argued that insufficient simulation runs were performed to falsify the specification. However, with HIL, the simulations are performed in real time, and the motions of submarines are relatively slow. Thus, the total number of simulation runs could not be increased further.

Moreover, all corner points were examined in the HCR method, and the robustness values for TuRBO and vanilla BO seem to converge to an interval over time, see Figures 5.4h and 5.3h. There is no trend for reduced robustness values over time for LSF in Figure 5.2h. Thus, the saturation of data can be considered reached, meaning that it is unlikely that the specification φ_{HOV} would be falsified even if more simulations had been performed.

The results of the falsification of A26's hovering controller highlight its consistent and reliable performance across all test scenarios. The system successfully maintained hovering within the predefined limit values in every case, ensuring operational stability. These findings suggest that the system is well-suited to support the stability needs of the submarine under varying conditions. However, one should remember the quote by Edsger W. Dijkstra, "*Program testing can be used to show the presence of bugs, but never to show their absence!*"

6.5 Social and Ethical Aspects

The military industry is a unique and heavily criticized economic sector [84]. The objective of military products is to defend states and attack when necessary. Thus, it is questionable whether working with defense material is morally justifiable. The research conducted in this thesis may improve the effectiveness of developing military products, but it does not raise any new ethical issues that must be addressed. Developing and selling military products can be considered two fundamentally different ethical dilemmas. They can contribute to peace or conflict depending on who has the military products. In Sweden, the Inspectorate of Strategic Products controls the exports of military equipment and dual-use products. Thus, the Inspectorate of Strategic Products can be considered responsible for the products not falling into the wrong hands.

United Nation's sustainability goal 16.1 states *significantly reduce conflict-related deaths*. Saab AB's products may counteract this goal. However, Saab AB strives to keep society and people safe [85]. With Saab AB's systems and solutions that increase security, creating an environment where humans feel safe becomes possible. As a result, Saab AB's products may enhance the work towards the United Nation's sustainability goals of peace and justice, i.e., goal 16. Furthermore, for the above reasons, Saab AB's purpose is in line with Article 3 in the Universal Declaration of Human Rights, *everyone has the right to life, liberty and security of person.*

The implementation of test automation may contribute to United Nation's goal

8.2, which aims to *achieve higher levels of economic productivity*. This metric is measured in the annual real GDP growth rate per employed person. The results presented in this thesis support this goal, as AuTS has been shown to enhance testing productivity, thereby enabling greater economic productivity. On the other hand, the automated testing system may counteract United Nation's goal 8.5, which aims for *lower unemployment rate*, given that the system will replace test engineers, ceteris paribus.

7
Conclusion

This thesis can be seen as a three-stage rocket. First, it identifies a set of demands, challenges, and opportunities associated with automating the testing of an SCMS. Second, it proposes a solution to implement test automation of the SCMS. Third, it investigates the solution's impact on the verification and validation activities within Saab Kockums.

By employing a design-science research methodology, functional and qualitative requirements are identified. These requirements were the foundation when developing the IT artifact, an automated testing system for an SCMS. AuTS consists of three different variants of test automation. The first variant involves writing script-based tests based on documents that describe the manual test procedure. The second variant involves the falsification of simulation models. The third variant focuses on testing the functionality of ship systems through the use of falsification. The automated testing system has been applied to in-house testing problems.

Combined, AuTS has proved that it confronts the business needs at Saab Kockums by enhancing the productivity and quality of the testing activities. Although writing test scripts initially takes time, the investment pays off in the long run as most systems undergo frequent testing, resulting in saved effort and costs. Additionally, script-based testing allows for more frequent test executions compared to manual methods. This increases the likelihood of early error detection in the development process when issues are easier to fix. Also, the experiments conducted have improved our understanding of the LSF. The increased understanding has led to the proposal of several adjustments to the LSF, ultimately enhancing the effectiveness of falsification for investigated problems.

This study set out to gain a better understanding of the test automation for PLC-based systems. The findings contribute to the specification-based testing of the PLC-based system, where general implementations are applicable. The study also aimed to deepen understanding of the field of falsification for HIL testing. The research indicates that, in contrast to MIL testing, the falsification approach is less suitable for HIL testing, particularly for systems requiring long durations in real-time testing.

This thesis presents a workflow that can serve as a guide for organizations that want to move from manual to automated testing methods. The methods and tools used for test automation can inspire practitioners to develop similar frameworks, and it helps them understand the fundamental concepts of developing such frameworks. While AuTS holds promise for enhancement of the testing activities in this thesis, it may necessitate certain additions and modifications for practical application on a larger scale.

7.1 Future Work

For future work, it would be interesting to do a case study at an organization where test automation has already been implemented. Then, the challenges during test automation maintenance could be investigated. For instance, what advantages does test automation offer throughout the product life cycle? This includes modifying and updating existing test scripts to ensure their ongoing functionality and alignment with the evolving verification and validation requirements.

Another interesting problem is to analyze how artificial intelligence can be exploited during the testing process. One possibility would be to use large language models [86] to verify that the language used in the HMI of the SCMS is Swedish. Another possibility would be to use artificial intelligence to write the script-based tests, cf. [87].

Continuing the study of falsification with the HIL environment would be an interesting topic for future work. The scope should cover systems where HIL tests can be performed relatively fast compared to the systems tested in this thesis. Furthermore, BO is a powerful and efficient technique for global optimization of objective functions. However, current approaches experience a decline in performance when the noise level is high, which limits their applicability. In response to this challenge, extensions of BO have been derived to handle noisy observations [88]. For future work, these extensions could be incorporated into the optimization-based falsification with the HIL environment.

Bibliography

[1] L. J. Moukahal, M. Zulkernine, and M. Soukup, "Vulnerability-Oriented Fuzz Testing for Connected Autonomous Vehicle Systems," *IEEE Transactions on Reliability*, vol. 70, no. 4, pp. 1422–1437, 2021.

[2] A. Homaifar, A. Karimoddini, B. A. Erol, M. A. Khan, E. Tunstel, R. L. Roberts, R. F. Young, K. Snyder, R. S. Swanson, M. Jamshidi, Y. Seong, and E. A. Doucette, "Operationalizing Autonomy: A Transition From the Innovation Space to Real-World Operations," *IEEE Systems, Man, and Cybernetics Magazine*, vol. 5, no. 4, pp. 23–32, 2019.

[3] M. Berk, O. Schubert, H.-M. Kroll, B. Buschardt, and D. Straub, "Reliability Assessment of Safety-Critical Sensor Information: Does One Need a Reference Truth?," *IEEE Transactions on Reliability*, vol. 68, no. 4, pp. 1227–1241, 2019.

[4] S. Ulewicz and B. Vogel-Heuser, "Industrially Applicable System Regression Test Prioritization in Production Automation," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1839–1851, 2018.

[5] J. Herkert, J. Borenstein, and K. Miller, "The Boeing 737 MAX: Lessons for Engineering Ethics," *Science and Engineering Ethics*, vol. 26, pp. 2957–2974, Dec. 2020.

[6] N. Leveson and C. Turner, "An investigation of the Therac-25 accidents," *Computer*, vol. 26, no. 7, pp. 18–41, 1993.

[7] M. Kassab, J. F. DeFranco, and P. A. Laplante, "Software Testing: The State of the Practice," *IEEE Software*, vol. 34, no. 5, pp. 46–52, 2017.

[8] A. Bezbaruah, B. Pratap, and S. B. Hake, "Automation of Tests and Comparative Analysis between Manual and Automated testing," in *2020 IEEE Students Conference on Engineering & Systems (SCES)*, pp. 1–5, 2020.

[9] H. Mohanty, J. Mohanty, and A. Balakrishnan, *Trends in Software Testing*. Springer Singapore, 2017.

[10] G. J. Myers, T. Badgett, and C. Sandler, *The Art of Software Testing*. Wiley, 2012.

[11] J. Boby, *Test Automation : A Manager's Guide*. BCS, The Chartered Institute for IT, 2021.

[12] B. Hailpern and P. Santhanam, "Software debugging, testing, and verification," *IBM Systems Journal*, vol. 41, no. 1, pp. 4–12, 2002.

[13] A. Axelrod, *Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects*. Apress Berkeley, CA, 2018.

[14] V. Garousi and E. Yildirim, "Introducing Automated GUI Testing and Observing Its Benefits: An Industrial Case Study in the Context of Law-Practice

Management Software," in *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 138–145, 2018.

[15] E. Borjesson and R. Feldt, "Automated System Testing Using Visual GUI Testing Tools: A Comparative Study in Industry," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, pp. 350–359, 2012.

[16] A. Cervantes, "Exploring the use of a test automation framework," in *2009 IEEE Aerospace conference*, pp. 1–9, 2009.

[17] A. Leitner, I. Ciupa, B. Meyer, and M. Howard, "Reconciling Manual and Automated Testing: The AutoTest Experience," in *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, pp. 261a–261a, 2007.

[18] C. Hobbs, *Embedded Software Development for Safety-Critical Systems*. Taylor & Francis, 2016.

[19] P. Rümmer and M. A. Shah, "Proving Programs Incorrect Using a Sequent Calculus for Java Dynamic Logic," in *Tests and Proofs*, pp. 41–60, Springer Berlin Heidelberg, 2007.

[20] E. W. Dijkstra, *Chapter I: Notes on Structured Programming*, p. 1–82. GBR: Academic Press Ltd., 1972.

[21] S. Ray, *Scalable Techniques for Formal Verification*. Springer US, 2010.

[22] W. Ahrendt, B. Beckert, R. Bubel, R. Hähnle, P. H. Schmitt, and M. Ulbrich, *Deductive Software Verification – The KeY Book: From Theory to Practice*. Programming and Software Engineering: 10001, Springer Cham, 2016.

[23] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, *Handbook of Model Checking*. Springer International Publishing, 2018.

[24] C. Baier and J. P. Katoen, *Principles of Model Checking*. The MIT Press, 2008.

[25] T. Gedell, *Combining and Strengthening Program Analysis and Verification*. PhD thesis, Dept. Comput. Sci. and Eng., Chalmers Univ., Gothenburg, Sweden, 2008.

[26] Saab AB, "Saab får tilläggsbeställning för ubåt A26 till Sverige," 2021. Accessed: 2022-09-14. Available: `https://www.saab.com/sv/newsroom/press-releases/2021/saab-far-tillaggsbestallning-for-ubat-a26-till-sverige`.

[27] The Swedish Defence Materiel Administration, "Ubåt A26," 2022. Accessed: 2022-09-14. Available: `https://www.fmv.se/projekt/ubat-a26/`.

[28] Saab AB, "First Steel Cut For Saab Kockums A26 Submarines," 2015. Accessed: 2022-09-14. Available: `https://www.saab.com/newsroom/press-releases/2015/first-steel-cut-for-saab-kockums-a26-submarine`.

[29] B&R, "Automation Studio," 2023. Accessed: 2023-02-10. Available: `https://www.br-automation.com/sv/produkter/software/automation-software/automation-studio/`.

[30] "ISO/IEC/IEEE International Standard - Software and systems engineering - Software testing – Part 2: Test processes," *ISO/IEC/IEEE 29119-2:2021(E)*, pp. 1–64, 2021.

[31] S. Wang, Q. Shang, Q. Fang, and F. Fang, "Research on Automated Testing Method of Railway Signaling System," in *2020 IEEE 5th International Conference on Intelligent Transportation Engineering (ICITE)*, pp. 165–169, 2020.

[32] S. Bisht, *Robot Framework Test Automation*. Packt Publishing, Limited, 2013.

[33] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[34] C. C. Dias, C. S. H. Soares, L. T. Da Silva, C. C. de Carvalho, J. P. M. Matos, Y. L. M. Silva, D. W. Albuquerque, and D. F. S. Santos, "On-Target Test Automation for an Embedded Digital Signal Processing Device," in *2022 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1–2, 2022.

[35] X. Han, N. Zhang, W. He, K. Zhang, and L. Tang, "Automated Warship Software Testing System Based on LoadRunner Automation API," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 51–55, 2018.

[36] J. Cao, Z. Zhang, and Y. Liu, "Design and Implementation of the Automatic Testing System in the Multi-mode Terminal Tester," in *2012 Fourth International Conference on Computational and Information Sciences*, pp. 997–1000, 2012.

[37] M. Mane, J. S. Kulkarni, and V. Gupta, "Automation of FMET Testing," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–3, 2018.

[38] N. M. Raul, N. M. Wagdarikar, and C. R. Bhukya, "Development of Hardware-In-Loop Automated Test Bench for Liquid-Assisted After-Treatment Controls System's Regression Tests," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–6, 2018.

[39] National Instruments Corp, "Components of LabWindows/CVI," 2022. Accessed: 2022-09-29. Available: `https://www.ni.com/docs/en-US/bundle/labwindows-cvi/page/cvi/usermanual/cvioverview.htm`.

[40] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*. Prentice Hall Professional Technical Reference, 2nd ed., 1988.

[41] National Instruments Corp, "TestStand," 2022. Accessed: 2022-09-29. Available: `https://www.ni.com/sv-se/shop/software/products/teststand.html`.

[42] M. E. Salari, E. Paul Enoiu, W. Afzal, and C. Seceleanu, "Choosing a Test Automation Framework for Programmable Logic Controllers in CODESYS Development Environment," in *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 277–284, 2022.

[43] D. H. Hanssen, *Programmable Logic Controllers: A Practical Approach to IEC 61131-3 using CODESYS*. John Wiley & Sons, 2015.

[44] A. Karimoddini, M. A. Khan, S. Gebreyohannes, M. Heiges, E. Trewhitt, and A. Homaifar, "Automatic Test and Evaluation of Autonomous Systems," *IEEE Access*, vol. 10, pp. 72227–72238, 2022.

[45] Z. Ramezani, K. Claessen, N. Smallbone, M. Fabian, and K. Åkesson, "Testing Cyber–Physical Systems Using a Line-Search Falsification Method," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 8, pp. 2393–2406, 2022.

[46] Z. Ramezani, K. Šehic, L. Nardi, and K. Åkesson, "Falsification of Cyber-Physical Systems using Bayesian Optimization," 2022.

[47] R. Alur, *Principles of Cyber-Physical Systems*. MIT Press, 2015.

[48] R. Rhinehart R., *Nelder-Mead (NM) Simplex: Spendley, Hext, and Himsworth.*, ch. 11.5. John Wiley & Sons, 2018.

[49] W. Huyer and A. Neumaier, "SNOBFIT - Stable noisy optimization by branch and fit," *ACM Trans. Math. Softw.*, vol. 35, 07 2008.

[50] Z. Ramezani, N. Smallbone, M. Fabian, and K. Åkesson, "Evaluating Two Semantics for Falsification using an Autonomous Driving Example," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, pp. 386–391, 2019.

[51] Z. Ramezani, J. L. Eddeland, K. Claessen, M. Fabian, and K. Åkesson, "Multiple Objective Functions for Falsification of Cyber-Physical Systems," *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 417–422, 2020. 15th IFAC Workshop on Discrete Event Systems WODES 2020 — Rio de Janeiro, Brazil, 11-13 November 2020.

[52] Z. Ramezani, A. Donzé, M. Fabian, and K. Åkesson, "On Input Generators for Cyber-Physical Systems Falsification," 2022.

[53] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable Global Optimization via Local Bayesian Optimization," in *Advances in Neural Information Processing Systems*, pp. 5496–5507, 2019. The code is available at `https://github.com/uber-research/TuRBO`.

[54] C. Hvarfner, D. Stoll, A. Souza, M. Lindauer, F. Hutter, and L. Nardi, "πBO: Augmenting Acquisition Functions with User Beliefs for Bayesian Optimization," 2022.

[55] M. Renilson, *Submarine Hydrodynamics*. Springer Cham, 2018.

[56] V. Gladkikh and R. Tenzer, "A Mathematical Model of the Global Ocean Saltwater Density Distribution," *Pure and Applied Geophysics*, vol. 169, pp. 249–257, 02 2012.

[57] Z. Ramezani, *On Optimization-Based Falsification of Cyber-Physical Systems*. PhD thesis, Dept. Elect. Eng., Chalmers Univ., Gothenburg, Sweden, 2022.

[58] K. Claessen, N. Smallbone, J. Eddeland, Z. Ramezani, and K. Åkesson, "Using Valued Booleans to Find Simpler Counterexamples in Random Testing of Cyber-Physical Systems," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 408–415, 2018. 14th IFAC Workshop on Discrete Event Systems WODES 2018.

[59] R. Hooke and T. A. Jeeves, ""Direct Search" Solution of Numerical and Statistical Problems," *J. ACM*, vol. 8, p. 212–229, apr 1961.

[60] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[61] J. Joyce, "Bayes' Theorem," in *The Stanford Encyclopedia of Philosophy* (E. N. Zalta, ed.), Metaphysics Research Lab, Stanford University, Fall 2021 ed., 2021.

[62] S. Ross, "Chapter 10 - Additional Variance Reduction Techniques," in *Simulation (Fifth Edition)* (S. Ross, ed.), pp. 233–246, Academic Press, 2013.

[63] W. Gan, Z. Ji, and Y. Liang, "Acquisition Functions in Bayesian Optimization," in *2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*, pp. 129–135, 2021.

[64] W. R. Thompson, "On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[65] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[66] A. Hevner and S. Chatterjee, *Design Research in Information Systems: Theory and Practice*. Integrated Series in Information Systems: 22, Springer New York, NY, 2010.

[67] M. Höst, A. Rainer, P. Runeson, and B. Regnell, *Case Study Research in Software Engineering : Guidelines and Examples.* John Wiley & Sons, Incorporated, 2012.

[68] M. Hammond and J. J. Wellington, *Research Methods: The Key Concepts*. Routledge Key Guides, Routledge, 2021.

[69] C. Robson, *Real world research*. 2nd edition. Blackwell, 2002.

[70] "ISO/IEC/IEEE International Standard - Software and systems engineering – Software testing –Part 1:General concepts," *ISO/IEC/IEEE 29119-1:2022(E)*, pp. 1–60, 2022.

[71] T. Stober and U. Hansmann, *Agile Software Development: Best Practices for Large Software Development Projects*. Springer Berlin Heidelberg, 2010.

[72] E. Bell, A. Bryman, and B. Harley, *Business Research Methods*. Oxford University Press, 2022.

[73] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, pp. 131–164, Apr. 2009.

[74] C. Buschle, H. Reiter, and A. Bethmann, "The qualitative pretest interview for questionnaire development: outline of programme and practice.," *Quality & Quantity*, vol. 56, no. 2, pp. 823 – 842, 2022.

[75] R. E. Stake, *The Art of Case Study Research*. SAGE Publications, Inc, 1995.

[76] "IEEE/ISO/IEC International Standard for Software and systems engineering–Software testing–Part 3:Test documentation," *ISO/IEC/IEEE 29119-3:2021(E)*, pp. 1–98, 2021.

[77] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems," in *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 254–257, Springer Berlin Heidelberg, 2011.

[78] A. Donzé, "Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems," in *Computer Aided Verification* (T. Touili, B. Cook, and P. Jackson, eds.), (Berlin, Heidelberg), pp. 167–170, Springer Berlin Heidelberg, 2010.

[79] L. Nardi, D. Koeplinger, and K. Olukotun, "Practical Design Space Exploration," in *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 347–358, 2019.

[80] "IEEE Recommended Practice for Software Design Descriptions," *IEEE Std 1016-1998*, pp. 1–23, 1998.

[81] S. He, S. Li, Y. Chen, and D. Guo, "Uncertainty Analysis of Race Conditions in Real-Time Systems," in *2015 IEEE International Conference on Software Quality, Reliability and Security*, pp. 227–232, 2015.

[82] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts, "Simulation-Based Approaches for Verification of Embedded Control Systems: An Overview of Traditional and Advanced Modeling, Testing, and Verification Techniques," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 45–64, 2016.

[83] E. Alégroth, M. Nass, and H. H. Olsson, "JAutomate: A Tool for System- and Acceptance-test Automation," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, pp. 439–446, 2013.

[84] W. Hartung, "Should Arms Makers Be Held Responsible for How Their Weapons Are Used?," *Forbes*, September 2019. Accessed: 2022-09-29. Available: `https://www.forbes.com/sites/williamhartung/2019/09/09/should-arms-makers-be-held-responsible-for-how-their-weapons-are-used/?sh=62a390c17093`.

[85] Saab AB, "Annual and Sustainability Report 2020," 2021. Accessed: 2022-09-14. Available: `https://www.saab.com/globalassets/cision/documents/2021/20210304-saab-publishes-its-annual-and-sustainability-report-2020\-en-0-3908922.pdf`.

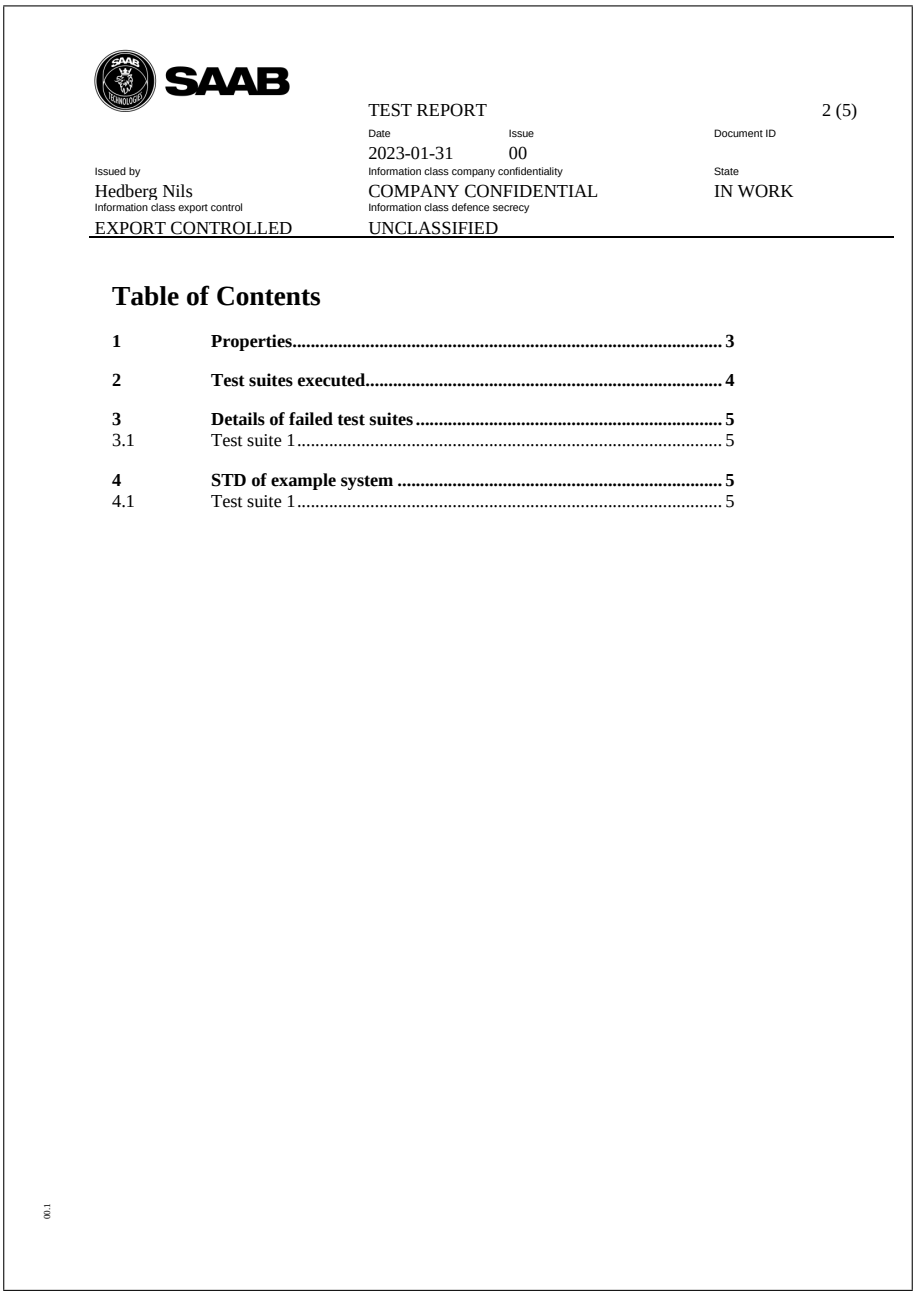
[86] R. Gozalo-Brizuela and E. C. Garrido-Merchan, "ChatGPT is not all you need. A State of the Art Review of large Generative AI models," 2023.

[87] M. Chen, H. Zhang, C. Wan, Z. Wei, Y. Xu, J. Wang, and X. Gu, "On the Effectiveness of Large Language Models in Domain-Specific Code Generation," 2023.

[88] B. Letham, B. Karrer, G. Ottoni, and E. Bakshy, "Constrained Bayesian Optimization with Noisy Experiments," 2018.

A

Software Test Document



SAAB

TEST REPORT 2 (5)

Date	Issue	Document ID
2023-01-31	00	

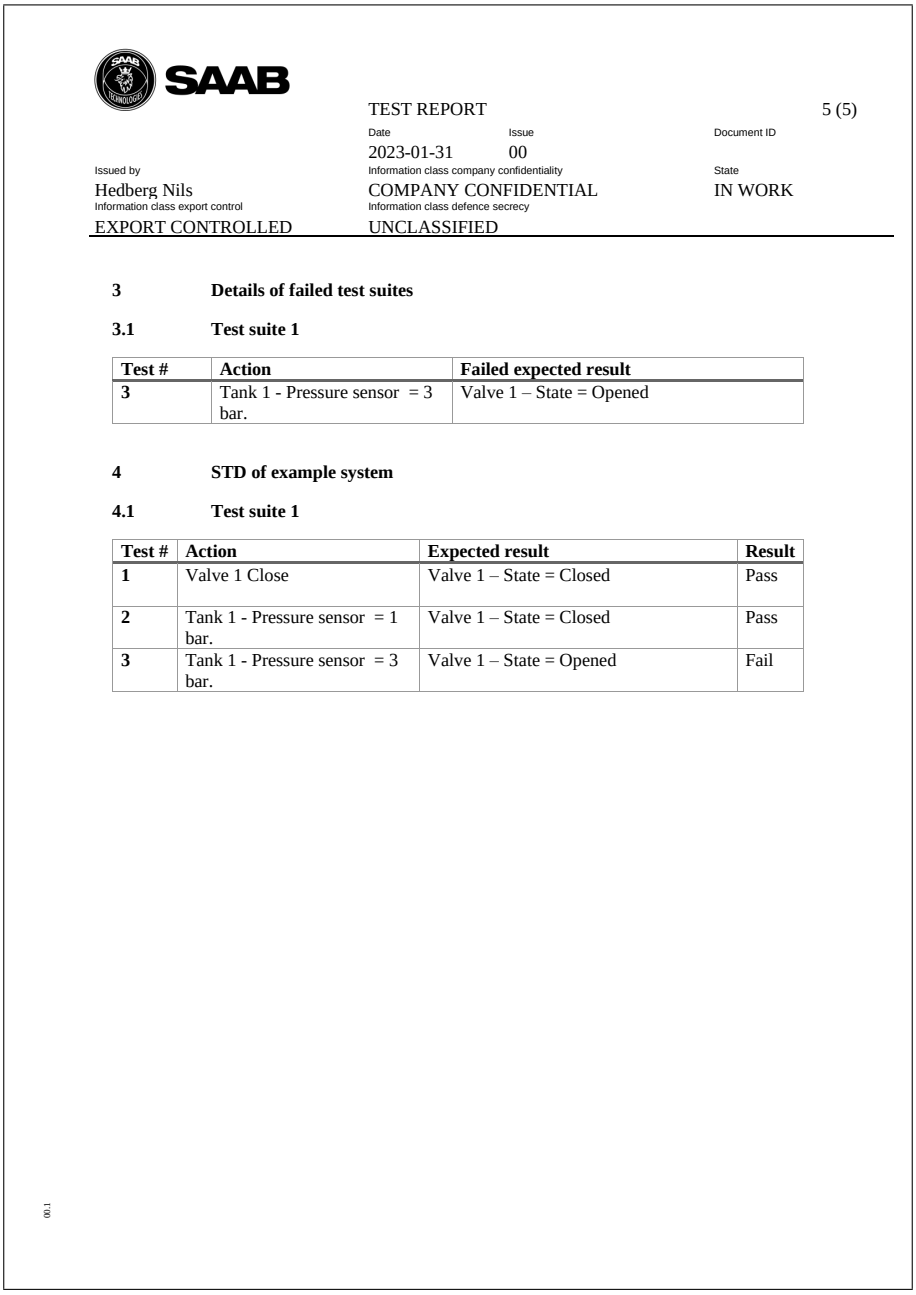
Issued by	Information class company confidentiality	State
Hedberg Nils	COMPANY CONFIDENTIAL	IN WORK

Information class export control	Information class defence secrecy
EXPORT CONTROLLED	UNCLASSIFIED

Table of Contents

00.1

Figure A.1: Table of contents from document containing test results from script-based testing of the test suite in Table 4.1.



SAAB

TEST REPORT 5 (5)

Date	Issue	Document ID
2023-01-31	00	

Issued by	Information class company confidentiality	State
Hedberg Nils	COMPANY CONFIDENTIAL	IN WORK

Information class export control	Information class defence secrecy
EXPORT CONTROLLED	UNCLASSIFIED

3 Details of failed test suites

3.1 Test suite 1

Test #	Action	Failed expected result
3	Tank 1 - Pressure sensor = 3 bar.	Valve 1 – State = Opened

4 STD of example system

4.1 Test suite 1

Test #	Action	Expected result	Result
1	Valve 1 Close	Valve 1 – State = Closed	Pass
2	Tank 1 - Pressure sensor = 1 bar.	Valve 1 – State = Closed	Pass
3	Tank 1 - Pressure sensor = 3 bar.	Valve 1 – State = Opened	Fail

00.1

Figure A.2: Document containing test results from script-based testing of the test suite in Table 4.1.

B

Interview Protocol

Step	Interview question	Category
1.	1. What is your role in the organization? 2. How many years of industrial experience do you have?	Participant's background
2.	**The purpose and objectives of the thesis are presented to participant. The key concepts in the thesis are discussed.**	
3.	3. How much do you come across testing in your work? If so, what type of testing? 4. Are you familiar with test automation? 5. Are you familiar with the falsification framework for cyber-physical systems?	Participant's background
4.	**The researcher gives an interactive tutorial in the first variant of AuTS. Based on a SwTD, script-based tests are created together with the participant.**	
5.	6. Do you think that this variant of script-based testing could fit into the testing process for the SCMS? Where? (a) Certain systems and functions? (b) Certain type of tests? (c) Other?	Applicability

6.	7. Do you think that this variant of script-based testing could bring benefits to the testing of the SCMS, compared to today's testing? Elaborate. (a) Time and/or effort? (b) Increased quality? (c) Other?	Quality and productivity
7.	8. Can you describe how difficult it is to script test cases in AuTS? 9. Can you describe how difficult it is to get the test results in AuTS?	Usability
8.	**The researcher gives a tutorial in falsification and how it is applied to test simulation models and functionality in ship systems, i.e., the second and third test automation variants.**	
9.	10. Do you think that the falsification framework can fit into the testing of SCMS? Elaborate. (a) To test functionality of ship systems? (b) To test simulation models? (c) Other?	Applicability
10.	11. Do you think that the falsification framework could bring benefits to the testing of the SCMS, compared to today's testing? Elaborate. (a) Time and/or effort? (b) Find inputs that falsify specifications? (c) Other?	Quality and productivity
11.	12. Can you describe how difficult it is to establish a falsification environment for a new simulation model in AuTS?	Usability
12.	13. How do you think AuTS could be improved? 14. Other observations? Anything relevant that is not covered?	Wrap-up

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY